Modélisation implicite par squelette et Applications

Cédric Zanni

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE
Spécialité : **Mathématiques-Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Cédric Zanni

Thèse dirigée par **Marie-Paule Cani**

préparée au sein **du Laboratoire Jean Kuntzmann (LJK)**
et de **l'école doctorale EDMSTII**

Skeleton-based Implicit Modeling & Applications

Thèse soutenue publiquement le **6 Décembre 2013**,
devant le jury composé de :

George-Pierre Bonneau
Professeur, Grenoble Université, Président
Karan Singh
Professeur, University of Toronto, Rapporteur
Loïc Barthe
Maitre de Conférence, Université Paul Sabatier, Rapporteur
Brian Wyvill
Professeur, University of Bath, Examinateur
Evelyne Hubert
Chargée de Recherche, INRIA, Examinatrice
Marie-Paule Cani
Professeur, Grenoble Université, Directeur de thèse

RÉSUMÉ

Modélisation implicite par squelette et Applications

Modéliser avec des squelettes est une alternative très séduisante aux "points de contrôle" souvent placés à l'extérieur des formes : cette approche, analogue à un fil de fer dans une forme modelée, permet de créer des modèles de toutes géométries et topologies. Pour cela, il faut que les formes définies par chacun des squelettes soient capable de se mélanger de manière lisse.

Introduites en informatique graphique dans les années 90, les surfaces implicites sont la principale solution à ce problème. Elles constituent un modèle puissant à la fois pour la modélisation d'objets tridimensionnels et pour leur animation : leur construction par squelette et leurs capacités de mélange par sommation des champs potentiels qui les définissent permettent en effet la conception progressive et le stockage compact d'objets volumiques, ainsi que l'animation de déformations pouvant comprendre des changements de topologie.

Les surfaces implicites, et plus particulièrement les surfaces de convolution, forment donc un modèle particulièrement adapté à la modélisation par squelette. Toutefois, elles présentent un certain nombre de défaut qui les ont rendu inutilisables en pratique.

Cette thèse propose de nouveaux modèles implicites à squelettes, s'inspirant de la convolution mais basés aussi sur des déformations de l'espace. Ils permettent :

– une génération plus aisée de forme le long de squelettes formés de courbes (des arc d'hélices),
– un meilleur contrôle des formes tant au niveau de leur épaisseur que de leur mélange, notamment nos modèles sont invariant par homothétie ce qui les rend plus intuitif,
– la génération de surfaces ayant une topologie plus proche de celle des squelettes,
– la génération de détails fins engendrés par un bruit procédural, les détails se comportant de manière cohérente avec la surface (et les squelettes) sous-jacente.

ABSTRACT

Implicit modeling with skeleton and Applications

Modeling with skeleton is an attractive alternative to "control points" usually placed outside a shape in order to model it : this paradigm, similar to a wire inside the modeled shape, enables model of arbitrary geometry and topology. In order to do so, shapes defined by skeletons should be able to smoothly blend together.

Introduced in computer graphics in the 70's, implicit surfaces are one of the main solution to this problem. They are powerful both for the modeling of 3D models and their animations : their construction from a skeleton and their blending capacity by simply summing their scalar field provide an easy way to incrementally create shapes and store them in a compact way, it also facilitates animation containing changes in topology.

Implicit surfaces, and more specifically Convolution surfaces, are therefore particularly well adapted to skeleton-based modeling. However, they present a number of drawback that make them difficult to use in practice.

This thesis propose new skeleton-based implicit models, inspired not only by convolution but also from space deformations. They enable :

- an easier generation of shape along curve skeletons (arcs of helix),
- a better control of generated shape both in term of thickness and blending, in particular our model are scale-invariant that make them more intuitive,
- the generation of shape which topology better reflects the topology of its skeleton,
- the generation of small details from a procedural texture, the details behave in a coherent way with the underlying surface (and its skeleton).

REMERCIEMENTS

Je remercie tout d'abord Marie-Paule Cani pour m'avoir encadré durant ces trois années de thèse et pour avoir pris tant de temps pour la relecture de ma thèse.

Je tiens aussi à remercier les différentes personnes avec qui j'ai principalement eu l'occasion de collaborer : Evelyne Hubert, Ares Lagae et Adrien Bernhardt. Je remercie également Quentin Merigot pour la discussion qui a entrainé la réalisation d'une partie de mes travaux et Bloomenthal qui a trouvé un peu de temps pour lire ma thèse et apporter quelques conseils bienvenues pour l'écriture de mon introduction. Je remercie tout particulièrement Maxime Quiblier avec qui j'ai initialement travaillé sur la librairie Convol et sans qui mes travaux serait probablement moins avancé.

Mes remerciement vont également à tout les gens ayant travaillé sur Convol : Galel, Amaury et Rémi, et a tout les gens de Toulouse travaillant sur les surfaces implicites : Loic Barthe et ses étudiants Rodolphe, Florian et Even. Je remercie également les stagiaires que j'ai encadré : Yannick, Cleobulo et Antoine.

Un grand merci aux anciens d'Evasion, notamment Adrien, Michael, Francois et Damien, pour leur accueil et la bonne ambiance de l'équipe. Un merci spécial à Lucian sans qui je me serai senti bien seul lors de ma première conférence, et à George-Pierre Bonneau sans qui les pauses café serai bien moins animé.

Un grand merci à Thomas Oberlin et Mathieu Huard m'ayant supporté à l'ENSIMAG puis pendant leur thèse. Je remercie également tout les doctorants d'Evasion ayant débuté leur thèse en même temps que moi, notamment Manuel, Ali, Simon et l'année d'après, Arnaud, Benjamin et l'année encore d'après ...

Enfin, j'adresse mes remerciement à tout les membres de l'équipe Imagine qui font perdurer la bonne ambiance de l'équipe, qu'ils soient doctorant, ingénieur, assistant administratif, artiste ou permanent.

Finalement, merci à tout mes amis qui m'ont supporté pendant ces trois années (et bien plus...). Enfin, un grand merci à ma famille.

TABLE DES MATIÈRES

INTRODUCTION

It is important to understand the inner structure of a shape. For instance, skeletons can represent the underlying structure of vertebrate animals and trees. Therefore, modeling with skeletons is an attractive alternative to "control points" usually placed outside a shape in order to model it : this paradigm, similar to a wire inside the modeled shape, enables the creation of models of arbitrary geometry and topology while remaining intuitive to manipulate (in order to do so, shapes defined by skeletons should be able to smoothly blend together). This is the reason why skeleton-based modeling should provide a way to make modeling within the reach of children.

Implicit surfaces are a powerful tool for creating shapes surrounding skeletons and animating them. This is mainly due to their nice smooth blending properties enabling to easily create shape of arbitrary topological genus and animation containing change in topology. Among them, convolution surfaces, introduced in computer graphics in the 90's, provide an easy way to incrementally create shapes from a skeleton which is close to a medial axis, thanks to their capacity to blend in a natural way by simply summing their scalar field when new skeleton parts are added. This model has proved instrumental in sketch-based modeling applications where shapes are created by extracting a skeleton from a 2D picture.

Although implicit surfaces and more specifically convolution surfaces were introduced in the 80's or early 90's, skeleton-based modeling did not spread in standard modeling systems so far. This is due to a number of limitations, detailed next. The general goal of this work was to make skeleton-based implicit modeling usable, and therefore to propose new models solving these long lasting issues.

OPEN PROBLEMS IN SKELETON-BASED MODELING

Despite their interesting properties, convolution surfaces suffer from several drawbacks (most of them common to all implicit surfaces) that have limited their development.

Firstly, implicit surfaces are said to create only *blobby shapes* : this is due to two main difficulties. Shapes that contains both smooth parts and sharp edges are difficult to create. Furthermore, partly due the lack of parametrization, creating repetitive surface details that nicely follow the parent shape is difficult. In addition, blending usually smoothes out details.

Secondly, in addition to smoothing out details, blending often leads to no respect of the *topology* of the skeleton and is therefore not much predictable. This can become particularly non intuitive for the user, since the intended shape features can be lost during blending (for instance the hand of an animated character can be blended with the rest of the body during animation if they come close to each other). Recent solutions to the blending problem were developed for general implicit surfaces. However, they are based on binary composition operator, so they are not well adapted to convolution surfaces (usually blended with a sum). Indeed, they would break their nice properties, such as independence to skeleton subdivision.

Lastly, a problem specific to convolution surfaces is the difficulty to find a close-form expression of the scalar field for all types of geometric skeletons. Not having such an expression raises efficiency issues. For this reason, convolution surfaces do not provide a lot of choice in primitive shapes : only planar skeletons (segments and arcs of circle) are available so far, and more complex shapes have to be split into many of these primitives.

CONTRIBUTIONS

The aim of this work is to make skeleton-based modeling usable, and therefore to solve the open issues we just listed. We will mainly focus on the ones that lead to a lack of intuitiveness during modeling, but also on the ones that limit the range of shapes that can be represented. The contributions of this work are as follows :

First, we present a new kind of implicit primitive : the goal is to create a convolution-like primitive along an helical skeleton. In order to do so, we use a space warping technique that creates the primitive by deforming a simpler one based on a segment or a circle. We focus on reducing the unwanted deformation of the cross-section of the simpler shape. The new primitive should help the creation of implicit shapes along free-form curves.

Secondly, we present new skeleton-based implicit primitives that address all the issues we identified in the blending behavior of convolution surfaces while retaining their advantages (independence from skeleton subdivision). This is the main contribution of this work. This is done in two step : we first introduce scale-invariant integral surfaces, an alternative way of representing surfaces of varying thickness along a skeleton : by giving scale-invariance to the model, we obtain a more intuitive behavior, ease the setting of desired radius along the skeleton and avoid blurring of small details. Furthermore, these integral surfaces make compact support kernels really usable in practice which was not the case so far. This is a prerequisite for working with large skeletons. Based on the intuitive behavior of this new model, we introduce two new blending methods in order to improve blending behavior with respect to topology. The first one is a blending-graph based method that selects the most interesting part of the skeleton depending on the query point. The second one is a gradient-based method that post-corrects a classic blending by summation when unwanted blending is detected.

Lastly, we present a method to generate small scale surface details on skeleton-based implicit surfaces that behave in a coherent way with the underlying shape. The method is based on the use of a procedural surface noise which does not require parametrization but still behaves in a coherent way with the surface if a frame field is provided. We define the latter thanks to the skeleton used as a compact shape descriptor.

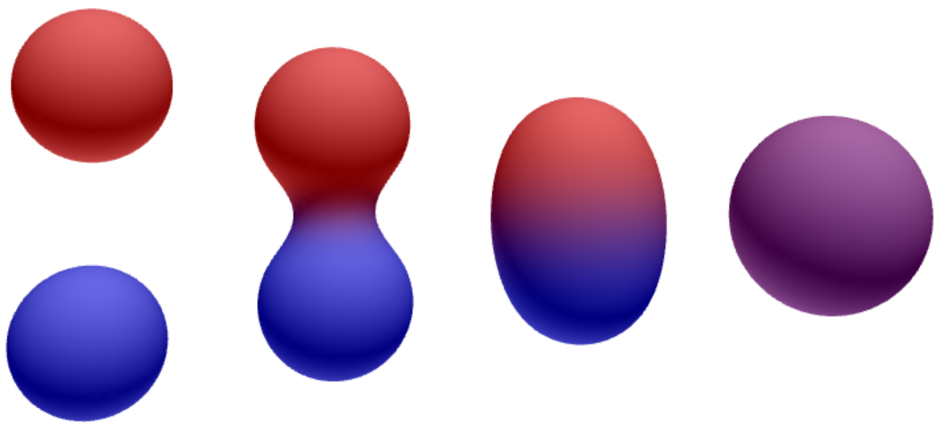
The work carried out in this thesis has been developed within a new C++ library devoted to implicit surfaces, with a focus on convolution surfaces (and more generally integral surfaces). The associated applications include both a skeleton-based modeling framework and a sketch-based modeling tool.

Structure of the document

This document is divided into 5 main parts. First, we present in chapter 1, a state-of-the-art of implicit modeling. We mainly focus on skeleton-based implicit modeling and blending methods but we will also review some work on generation of details, implicit surface visualization, as well as different applications of implicit surfaces. In chapter 2, we present our new helical implicit primitive, created from space warping. In chapter 3, we detail the main drawbacks of convolution surfaces and our new Scale-invariant Integral Surfaces. In chapter 4, we study two way to further improve blending behavior of our new model so that its topology better reflects the topology of its skeleton. Lastly, in chapter 5, we present our new method to generate small scale details over an skeleton-based implicit surfaces. Several appendices are included at the end of the document to give additional details on the different models we introduced.

CHAPTER

1 IMPLICIT MODELING : STATE OF THE ART



Teaser Figure : *Two blobs merging.*

IMPLICIT surfaces $\mathcal{S}$ are defined as the set of points where a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$ is equal to a given iso-value $c \in \mathbb{R}$:

$$\mathcal{S} = \left\{ P \in \mathbb{R}^3 / f(P) = c \right\}.$$

In the same way implicit volumes $\mathcal{V}$ can be defined by :

$$\mathcal{V} = \left\{ P \in \mathbb{R}^3 / f(P) \geq c \right\}.$$

Implicit surfaces have been studied from a long time in mathematics, mainly through the study of algebraic surfaces for which f is a polynomial function of Cartesian coordinates (some examples are shown in Figure 1.1).

Note that we use at least C^1 scalar fields, leading to continuously varying normal vectors :

$$\mathbf{N}(P) = -\frac{\nabla f(P)}{\|\nabla f(P)\|}.$$

More generally, the degree of continuity of an iso-surface is the one of the scalar field that defines it outside singular points ($\nabla f(P) = 0$).

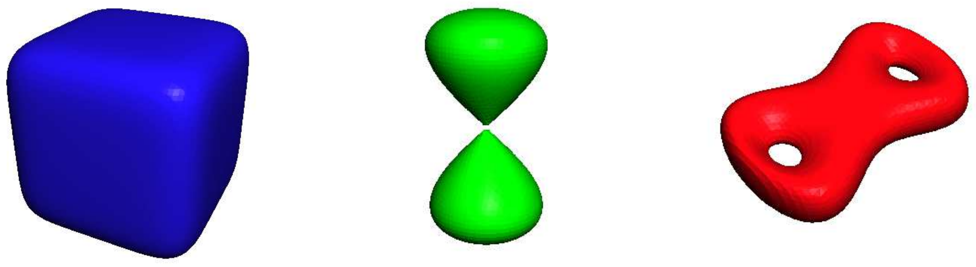


FIGURE 1.1 – *Some algebraic surfaces of degree lower or equal to* 6.

In order to use implicit surfaces as a modeling tool, they should be intuitive to manipulate, which is not the case for algebraic surfaces. A lot of different implicit surfaces (with different terminologies) have been used for the purpose of modeling. Among them we can find Function Representations (FRep), level-sets, grid-based representations (voxel) and skeleton-based implicit surfaces. In this thesis, we focus on the last one, which is one of the most common kind of implicit surfaces used in Computer Graphics.

In the remainder of this chapter, we will first review the main models used in skeleton-based implicit modeling. In a second part we will present the main methods for combining and deforming implicit surfaces : hence we will talk about blending and warping. The third part will talk about visualization of implicit surfaces (meshing and ray-tracing). Then we will present several methods to add details on both implicit surfaces and other kinds of surfaces. Lastly we will present a serie of applications of implicit modeling.

1.1 Skeleton-based Implicit Surfaces

Constructive implicit modeling makes use of skeletons (such as points, segments or triangles) for generating the field function that defines an implicit primitive. This representation has two advantages : its compactness and its intuitiveness.

We first present some notations that are re-used all along this document.

1.1.1 Definitions & Notations

Weighted skeletons : general formulation

The most general way to define a skeleton is to define it through a set of points. A *skeleton* $\mathcal{Sk}$ can therefore be defined as :

$$\mathcal{Sk} = (i_{\mathcal{Sk}}, \mathbf{S}_{\mathcal{Sk}}, \tau_{\mathcal{Sk}}). \tag{1.1}$$

The set $\mathbf{S}_{\mathcal{Sk}}$ is the set of points in $\mathbb{R}^3$ that belong to the skeleton (thus $\mathbb{1}_{\mathbf{S}_{\mathcal{Sk}}}$ is equal to 1 over the skeleton and 0 outside) while $i_{\mathcal{Sk}} \in [0;3]$ is the dimension of the skeleton (i.e. the dimension of the manifold associated with the skeleton : 0 for points, 1 for curves, 2 for surfaces and 3 for volumes). The function $\tau_{\mathcal{Sk}} : \mathbf{S}_{\mathcal{Sk}} \rightarrow \mathbb{R}^+$ defines a weight for each point of the skeleton. It is possible to define additional properties such as color on each skeleton point.

Union of skeletons :

This formulation is well adapted to express operations on the skeletons such as union. For instance, the union of two skeletons with the same dimension i is :

$$\mathcal{Sk}_1 \cup \mathcal{Sk}_2 = \left(i, \mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}, \max(\tau_{\mathcal{Sk}_1}, \tau_{\mathcal{Sk}_2})\right).$$

It is also possible to define the union of skeletons of different dimensions, but in this case $i_{\mathcal{Sk}}$ becomes a function from $\mathbb{R}^3$ to $\mathbb{N}$.

Weighted skeleton : parametric formulation

It is also possible to define a skeleton in a parametrical way. This formulation that depends on the skeleton dimension is less general but usually more useful in practice. In the case of curves, the skeleton $\mathcal{Sk}$ is defined as :

$$\mathcal{Sk} = (\Omega, \Gamma, \tau).$$

where $\Omega \subset \mathbb{R}$ is a domain and $\Gamma : \Omega \rightarrow \mathbb{R}^3$ is a parametric curve. We assume that Γ is a regular curve, hence it is continuously differentiable and Γ' never vanishes. We use s as curve parameter when using arclength parametrization and t when using another parametrization. In the first case the infinitesimal arc-length is $\mathrm{d}s$ and in the second case it is $|\Gamma'(t)\ |\mathrm{d}t$. The function $\tau : \Omega \subset \mathbb{R} \rightarrow \mathbb{R}^+$ defines the weight on the skeleton. The link between the two formulations is :

$$\mathbf{S}_{\mathcal{Sk}} = \Gamma(\Omega)$$

and

$$\tau(t) = \tau_{\mathcal{Sk}}(\Gamma(t))$$

For surface skeletons, Γ is a parametric surface of parameters $(u, v) \in \Omega \subset \mathbb{R}^2$, and the infinitesimal area is given by $|\Gamma_u \times \Gamma_v| \mathrm{d}u\mathrm{d}v$.

Kernel

Most of skeleton-based implicit surfaces are defined thanks to scalar field functions that we will denote K (also called kernel when talking about convolution surfaces). In our framework, those functions, from $\mathbb{R}^+$ to $\mathbb{R}^+$, are decreasing functions of the distance to points on the skeleton :

$$K: \begin{array}{ccc} \mathbb{R}^+ & \rightarrow & \mathbb{R}^+ \\ d & \mapsto & K(d) \end{array} \tag{1.2}$$

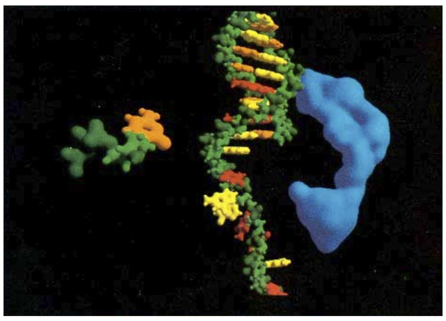
We will call $K_{\|.\|}$ the isotropic extension of K to $\mathbb{R}^3$:

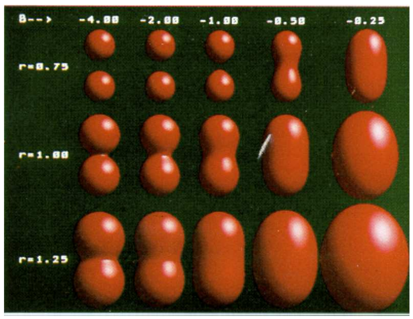
$$K_{\|.\|}: \begin{array}{ccc} \mathbb{R}^3 & \rightarrow & \mathbb{R}^+ \\ \overrightarrow{u} & \mapsto & K(\|\overrightarrow{u}\|) \end{array} \tag{1.3}$$

1.1.2 Point skeletons : "Blobby Molecules"

The first skeleton-based implicit surfaces to be used in Computer Graphics were point-based surfaces : the Blobby Molecules introduced by Blinn in [Bli82] in order to display molecular models (see Figure 1.2(a)). The method was inspired by electron density maps of hydrogen atoms which are represented thanks to a *Gaussian* Kernel (1.4). Density maps for a set of atoms are defined by summing the density map of each atom. In the same way Blinn defines surfaces through a density function, the *Gaussian* Kernel which is parametrized by a scale parameter and an amplitude. For multiple points the densities are summed. The surface is defined as an iso-value of the density/scalar field. Blinn advises to use a more intuitive parametrization of the kernel used, the parameter being the radius in isolation (when the point primitive is isolated) and the blobiness (i.e. the kind of blending). The effect of these two parameters can be seen in figure 1.2(b).

The main attractive feature of this shape representation, compared to parametric surfaces or meshes, is their nice blending property. Implicit primitives can be used in a constructive


(a)



(b)

FIGURE 1.2 – *(a) First blobby objects : molecules and DNA. (b) Blend between two blobs for varying blobiness and radius in isolation. Figures from [Bli82]*

modeling framework where smooth shapes of arbitrary topological genus are progressively blended into more complex ones by simply summing their field functions. The blended surface is then defined as :

$$\mathcal{S} = \left\{ P \in \mathbb{R}^3 / \sum_i f_i(P) = c \right\}.$$

This property made blobby molecules well suited to design both organic-looking shapes and animated objects that change of topology over time such as liquids and modeling clay. However, they also suffers from various drawbacks such as the difficulty to precisely control blending and the high computational time needed to display them. These drawbacks are reviewed in more details in sections 1.2 and 1.5.

Several alternatives to the *Gaussian* Kernel were proposed to create implicit primitives with local influence, which is more convenient when performing modeling tasks. This was done by using compact kernel functions such as those of *Metaballs* [NHK*85], *Soft Objects* [WMW86] and *Wyvill* function [Blo97] which are all piece-wise polynomial functions with finite support. In addition to easing user control, using such scalar field with local support increases computational efficiency when a large number of primitives are used thanks to their local influence (the generated field is null above a given threshold).

Here are the formula of this main kernels (with σ a resizing constant) :

– *Blobby Molecules* (*Gaussian* kernel) :

$$K(d) = \exp\left(-\left(\frac{d}{\sigma}\right)^2\right) \tag{1.4}$$

– *Metaballs* :

$$K(d) = \begin{cases} 1 - 3\left(\frac{d}{\sigma}\right)^2 & if\ 0 \leq d \leq \frac{\sigma}{3}\,, \\ \frac{3}{2}\left(1 - \frac{d}{\sigma}\right)^2 & if\ \frac{\sigma}{3} < d < \sigma\,, \\ 0 & otherwise. \end{cases} \tag{1.5}$$

– *Soft Objects* :

$$K(d) = \begin{cases} 1 - \frac{4}{9}\left(\frac{d}{\sigma}\right)^6 + \frac{17}{9}\left(\frac{d}{\sigma}\right)^4 - \frac{22}{9}\left(\frac{d}{\sigma}\right)^2 & if\ d < \sigma\,, \\ 0 & otherwise. \end{cases} \tag{1.6}$$

– *Wyvill* function :

$$K(d) = \begin{cases} \left(1 - \left(\frac{d}{\sigma}\right)^2\right)^3 & if\ d < \sigma\,, \\ 0 & otherwise. \end{cases} \tag{1.7}$$

1.1.3 Distance surfaces

In order to generate more complex shapes, point-based surfaces can be generalized to more complex skeletons. *Distance surfaces* use scalar fields defined as decreasing functions K of the distance to some geometric skeleton, typically points, line-segments or triangles :

$$f_{\mathcal{Sk}}(P) = K\left(d(\mathbf{S}_{\mathcal{Sk}}, P)\right) = K\left(\min_{G \in \mathbf{S}_{\mathcal{Sk}}} \|\overrightarrow{GP}\|\right) \tag{1.8}$$

where $\mathcal{Sk}$ is a geometric skeleton and $d(\mathbf{S}_{\mathcal{Sk}}, P)$ is the minimal Euclidian distance between the skeleton and the query point P. One of the advantages of these surfaces is the ease to prescribe

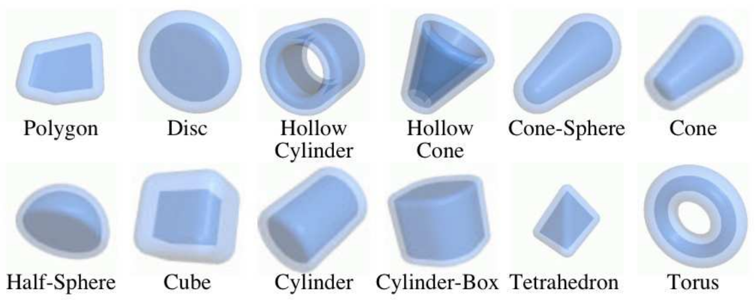


FIGURE 1.3 – *Some basic distance surfaces (also called offset surfaces) generated from a large variety of geometric skeletons. Figure from [BGA05]*

radii around isolated skeletons. However, such surfaces suffer from two main drawbacks that we present later.

In order to generate more complex surfaces, a lot of kinds of primitives have been used in addition to points, line-segments and triangles. Among them we can find portions of cylinders, cones, torus, cubes... Some distance surfaces are depicted on Figure 1.3. In order to generate a larger varieties of shapes [Bli82, TCW99] use an anisotropic metric around the skeletons (for instance using a quadric to compute the distance to a point) and also distance metrics that are different from the Euclidian one (typically L_p metrics).

Lack of regularity

Depending on the kind of skeleton, the regularity of distance-based implicit primitives heavily depend on the nature of their skeletons : Whereas point-based scalar field are usually C^∞ inside their support (except on the skeleton itself), this is not the case anymore for more complex skeleton. For instance segment-based surfaces are defined in two main parts : the region between the end point is a cylindrical scalar field whereas at both end points the scalar field is spherical. The gradient is thus continuous but not differentiable hence the surface is C^1 but not C^2 which creates a not so aesthetics shape (see figure 1.4(a)). A second problem occurs when the kernel support is larger than the skeleton's radius of curvature : in this case gradient discontinuities can arise. This becomes problematic in two cases : the discontinuity of

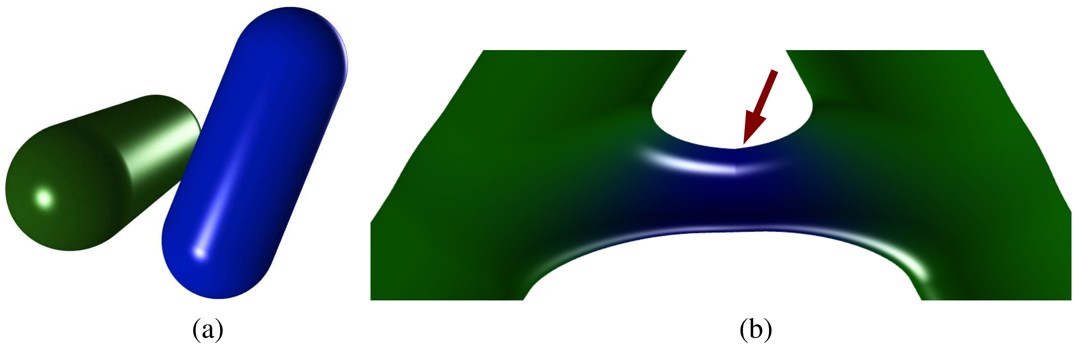


FIGURE 1.4 – *(a) Aesthetic of lighting impaired by a C^1 only scalar field. (b) Gradient discontinuity can break the smoothness of a blend : here a small blue segment has been blended into the concave part of a bigger green skeleton with a V shape.*

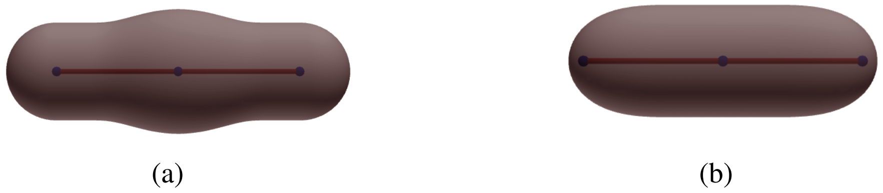


FIGURE 1.5 – *Blending by summation of two primitives : distance primitives (a) create a bulge while convolution surfaces (b) seamlessly blend.*

the scalar field is located on the surface (the generated shape is not smooth) ; or it is located in a blending area. In the latter case the blend smoothness is impacted (see figure 1.4(b)).

Bulging effet

Distance surfaces present a major drawback : the bulging effect. It was formally defined by Bloomenthal as :

"A 'surface bulge' has a cross-section that exhibits negative, then positive, then negative curvature with respect to an underlying skeleton." [Blo95a]

Such bulge occurs whenever two skeletons are set in continuity and blended with a simple sum (see Figure 1.5(a)). This is due to the fact that :

$$K\left(\min_{G\in \mathbf{S}_1\cup\mathbf{S}_2}\|\overrightarrow{GP}\|\right) \neq K\left(\min_{G\in \mathbf{S}_1}\|\overrightarrow{GP}\|\right) + K\left(\min_{G\in \mathbf{S}_2}\|\overrightarrow{GP}\|\right)$$

This behavior is really problematic when one wants to create a complex object by using several primitives (which is usually needed since distance to complex primitives is not always known and complex primitives can introduce discontinuities of gradient). For example, modeling the tail of an animal would require several segments so that the tail can be animated from straight to curved, and therefore would be subject to bulges.

A first attempt was made to avoid this drawback, not for the sum but for an algebraic blend : the super-elliptic blend [RO85]. This blend is defined as

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

for $t = 2$ the graph of B is elliptical. In order to prevent bulges, this blend has been enhanced with a parametrization based on the angle θ between gradients of the scalar field to be blended [Roc89] :

$$f(P) = B(f_1(P), f_2(P)) = 1 - \left[1 - \frac{f_1(P)}{r_1(1-\cos(\theta))}\right]_+^t - \left[1 - \frac{f_2(P)}{r_2(1-\cos(\theta))}\right]_+^t.$$

However, this blend is not cumbersome in practice. First, on contrary to the super-elliptic blend, it cannot be extended to a N-ary operator, this prevent independence from subdivision of the skeleton. Furthermore, it is only a C^0 operator.

1.1.4 Convolution Surfaces

Distance surfaces are not the only way to generalize blobs to skeletons of higher dimension. Convolution surfaces [BS91, Blo95b] were developed to avoid unwanted bulges when implicit primitives generated by neighboring skeletons blend. They define f as the integral of field contributions from all the individual points Q belonging to the skeleton (the integral being an infinite sum, this is actually the natural extension of blobs to skeleton of higher dimensions) :

$$f_{\mathcal{S}k}(P) = (g_{\mathcal{S}k} * K_{\|.\|})(P) = \int_{R^3} g_{\mathcal{S}k}(Q)\ K\left(\|\overrightarrow{PQ}\|\right)\ \mathrm{d}Q \tag{1.9}$$

where $g_{\mathcal{S}k}$ is a distribution describing the geometry of the skeleton and $K_{\|.\|}$ is the kernel function that gives the contribution of a skeleton point. The definition of $g_{\mathcal{S}k}$ depends on the dimension $i_{\mathcal{S}k}$ of the skeleton $\mathcal{S}k$. For volumetric skeleton, $g_{\mathcal{S}k}$ is defined by :

$$g_{\mathcal{S}k}(P) = \mathbb{1}_{\mathbf{S}}(P) = \begin{cases} 1 & \text{if } P \in \mathbf{S}_{\mathcal{S}k} \\ 0 & \text{otherwise} \end{cases}$$

For skeleton of lower dimension, the function $g_{\mathcal{S}k}$ is defined by Dirac delta functions. In dimension 0 (for point skeletons), convolution surfaces are equivalent to blobs. Note that convolution surfaces are usually presented in a simpler (but less accurate) way :

$$f_{\mathcal{S}k}(P) = \int_{Q\in\mathcal{S}k} K\left(\|\overrightarrow{PQ}\|\right)\ \mathrm{d}Q.$$

The bulge avoidance property comes from the additivity of the integration on intervals (which is a generalization of Chasles' relationship) : the integral over a support is equal to the sum of the integrals over the different parts of the support under the condition that the intersections between these parts is null. This gives to the model an additivity property in respect to the skeleton :

$$f_{\mathcal{S}k_1\cup\mathcal{S}k_2}(P) = f_{\mathcal{S}k_1}(P) + f_{\mathcal{S}k_2}(P),$$

under the condition that $\mathcal{S}k_1 \cap \mathcal{S}k_2$ is a measure-zero set. The generated shape is thus independent of the subdivision of the skeleton into components, and is therefore bulge-free. We can note that this property also derives from the linearity of the convolution operator :

$$g_{\mathcal{S}k_1} * K_{\|.\|} + g_{\mathcal{S}k_2} * K_{\|.\|} = (g_{\mathcal{S}k_1} + g_{\mathcal{S}k_2}) * K_{\|.\|},$$

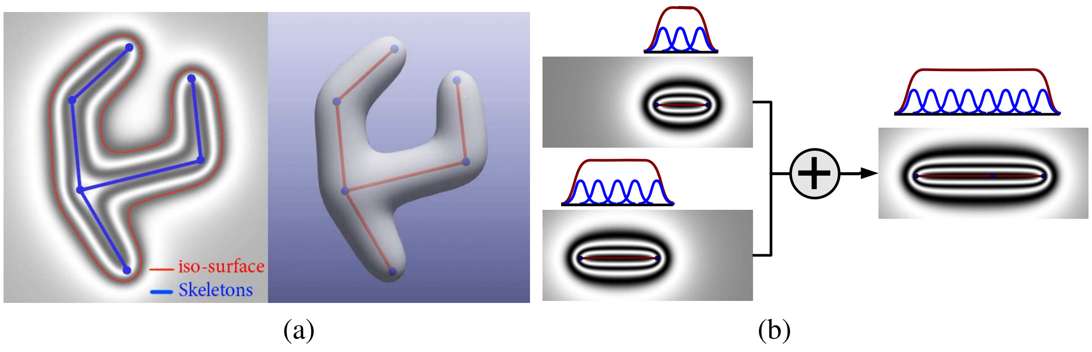


FIGURE 1.6 – *(a) a skeleton with its scalar field and an associated iso-surface. (b) field generated by integration give to the model additivity with respect to the skeleton : summing the contribution of two skeletons or seeing the two skeletons as one give the same scalar field.*

which is a more general property (the linearity of the convolution operator implies the additivity on intervals but not the contrary).

Independence from subdivision enables us to use complex networks of curves and surfaces as skeleton, the latter being split into adequate sets of line-segment and triangle skeletons for evaluation.

For practical purposes, a formulation that depends on the skeleton parametrization is useful :

$$f_{\mathcal{S}k}(P) = \int_{s\in\Omega} K\left(\|\overrightarrow{P\Gamma(s)}\|\right)\,\mathrm{d}s, \tag{1.10}$$

where K is a smoothing kernel, and $\mathcal{S}k$ a geometric skeleton of with arclength parameterization Γ. If the parametrization is not an arclength one, by substitution, the formula becomes :

$$f_{\mathcal{S}k}(P) = \int_{t\in\tilde{\Omega}} K(\|\overrightarrow{P\Gamma(t)}\|)\,|\Gamma'(t)|\,\mathrm{d}t.$$

While the first convolution surfaces used *Gaussian* kernels, subsequent work provided kernels with closed-form expressions [MS98, She99d] for both convolution along line segment and triangle skeletons. Kernels were organized into families in [HC12], they can be expressed as :

– *Cauchy* of order i :

$$K(d) = \frac{1}{\left(1+\left(\frac{d}{\sigma}\right)^2\right)^{\frac{i}{2}}} \tag{1.11}$$

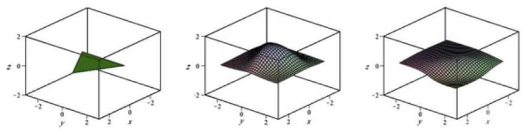
– *Inverse* of order i :

$$K(d) = \frac{1}{\left(\frac{d}{\sigma}\right)^i} \tag{1.12}$$

– *Compact Polynomial* of order i :

$$K(d) = \begin{cases} \left(1-\left(\frac{d}{\sigma}\right)^2\right)^{\frac{i}{2}} & \text{if } d<\sigma\ , \\ 0 \text{ otherwise.} \end{cases} \tag{1.13}$$

where σ is a given resizing constant (note that in [MS98, She99d], formulas were only provided for *Cauchy* of degree 4, *Inverse* of degree 1 and 2 and *Compact Polynomial* of degree 4). Note that *Compact Polynomial* generalizes *Wyvill*'s function of equation 1.7. Each kernel has its assets : *Cauchy* and *Inverse* are C^∞ but with a global support. *Inverse* also guarantees that the


(a)

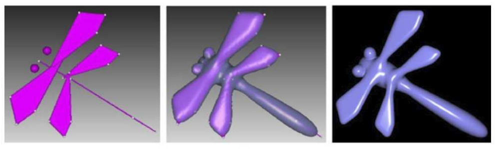

(b)

FIGURE 1.7 – *(a) A triangle skeleton, the associated convolution field and the step function resulting from the application of the Green theorem (Figure from [Hub12]), (b) example (Figure from [ZJLZ12]).*

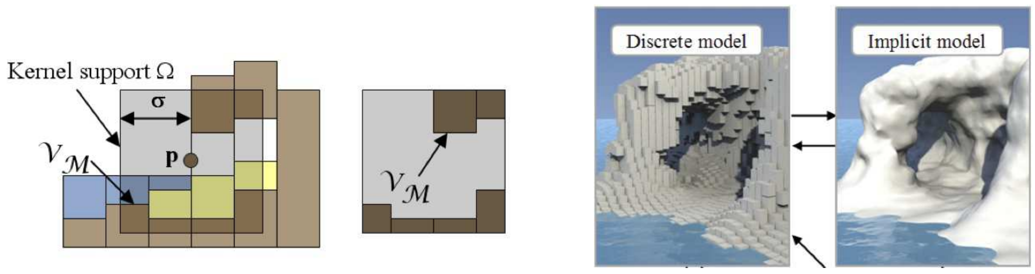


FIGURE 1.8 – *Fast convolution on volume. (a) use of cube skeletons and infinite norm, (b) modeled terrain. Picture from [PGMG09].*

skeleton is totally included inside the generated surface (thanks to infinite field values over the skeleton). This comes with a drawback : this kernel is less convenient when digging operators such as Barthe's difference [BWdG04] are used, since these operations will have little effect near the skeleton. *Compact Polynomial* is "only" $C^{\lfloor \frac{i-1}{2} \rfloor}$, but it has the advantage of having a local support. This eases local shape control and enables efficient field queries.

One of the main difficulties to compute convolution on triangle with *Compact Polynomial* kernel is the clipping of the triangle. Indeed the kernel being defined by parts, the domain of integration should be subdivided [JTZ09]. In order to provide better efficiency, recent work [ZJLZ12, Hub12] use the Green theorem in order to transform the integral on the triangle into an integral on its boundary, hence line integrals, for which clipping is much simpler (see Figure 1.7).

One of the only use of convolution surfaces on volume skeletons (cubes) was performed in [PGMG09], in order to model terrains of arbitrary topology. In order to obtain more efficient computation they used compact constant kernels, the convolution thus being the computation of the volume of skeleton inside the kernel support. To further increase computational efficiency they used an infinite norm for which the unit sphere is a cube (see figure 1.8(a)). Due to these simplifications the resulting scalar field is only C^0. This is not that problematic for their application since it is used to model stone (see Figure 1.8(b)).

1.1.5 Extension to non-constant radius

Being able to vary the radius of a primitive along its skeletons is important in order to obtain more freedom in the created shape. In order to achieve radius variation, Shertsyuk [She99c] uses profiling functions to scale the value of output convolution field based on the projection of the computation point on the skeleton. While this provides increased freedom for modeling, this method breaks the additivity in function of the skeleton (and independence to subdivision).

Bloomenthal, in [BS91], made two important remarks about radius control :
- radius variations can be obtained by setting weights along the skeleton,
- the width (σ parameter) of the kernel should be set to lower values when creating thin features in order to limit their blurring.

Two different models were developed, based on these remarks, both using a weighting function τ along the skeleton.

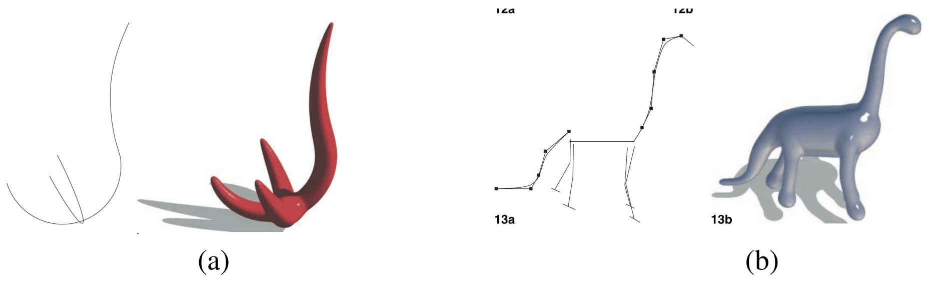


FIGURE 1.9 – *Weighted convolution on all primitives generated from 1D skeleton with known closed-form solution. (a) segments and arcs of circle, (b) segments and quadrics. Figure from [JT02b].*

Weighted convolution surfaces : Standard formulation

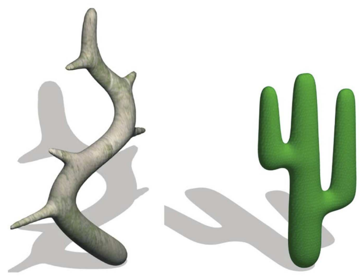


FIGURE 1.10 – *Weighted convolution on segment skeleton from [JT02a].*

The *standard formulation* for weighted convolution defines the field function as :

$$f_{\mathcal{S}k}(P) = \int_{s\in\Omega} \tau_{\mathcal{S}k}(s)\, K\left(\|\overrightarrow{P\Gamma(s)}\|\right) \mathrm{d}s \tag{1.14}$$

It was first studied by Jin [JTFP01, JT02b, JT02a] for polynomial weight up to degree 3 (for *Cauchy* 4 kernel, *Inverse* of degree 2 and all the odd degree $2k+1$ and *Compact Polynomial* of degree 4) and later re-used in [AC02, lTZkF04, BPCB08]. Extensions to other kernels were later developed [HC12] for segment skeletons by providing recurrence formula. In the latter paper, behavior of convolution surfaces was extensively studied (proof of direction of minimal and maximal thickness around a segment skeleton in function of it length and weight, as well as the relationship between them). Closed-form expressions for convolution on segments with arbitrary polynomial weights were derived for classical kernels of arbitrary degree. Circles where also studied for *Compact Polynomial* kernels. The more complex curves studied up-to-date [JT02b] were Bezier curves of degree 4 for *Compact Polynomial* kernels. However, we noted a problem in the formula given in the paper, due to wrong formulation of the integrals caused by a parametrization problem.

Efficiency comparison were performed both by comparing the numbers of required operations and timings for the different kernels. However comparisons were not performed on complex skeletons where compact polynomial kernels would have really benefited from their compact support.

Hornus integral surfaces

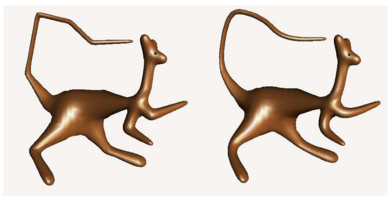


FIGURE 1.11 – *Hornus integral surfaces used with subdivision curves as skeleton.*

In contrast to the standard formulation, Hornus [HAC03] introduced the following formulation, by taking into account the second advice of Bloomenthal :

$$f_{\mathcal{S}k}(P) = \int_{s\in\Omega} K\left(\frac{\|\overrightarrow{P\Gamma(s)}\|}{\tau_{\mathcal{S}k}(s)}\right) \mathrm{d}s. \tag{1.15}$$

This is equivalent to having the kernel width vary along the skeleton according to the local weight, and thus to the local radius. A closed-form solution on segment-skeletons was only provided for the *Inverse* kernel of degree 2. One should note that Equation (1.15) isn't a convolution anymore, mathematically speaking, because of the division by the weight within the kernel function. For this reason we will use the term *Integral surfaces* instead of *Convolution surfaces* when speaking of both Hornus integral surfaces and Convolution surfaces (either weighted or not).

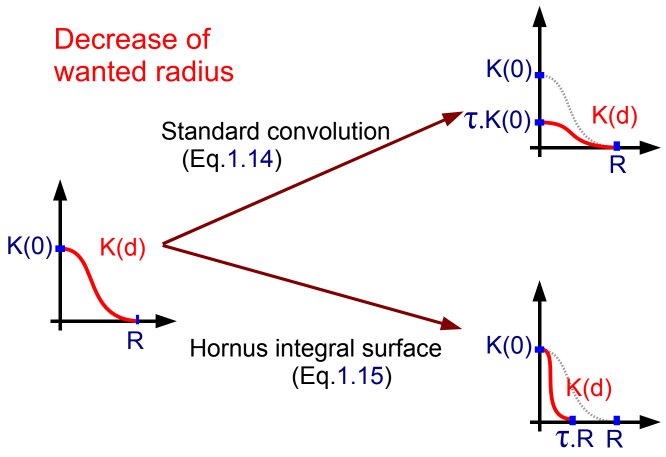


FIGURE 1.12 – *Weighted models : kernel variations when the user decreases τ to get a thinner shape along a segment-skeleton.*

Remarks

- Figure 1.12 compares the different strategies used by these two models to generate radius variations : while Equation (1.14) scales the height of the kernel, keeping the width of the support unchanged, Equation (1.15) generates a kernel of constant height, but of varying support-size.
- Generally speaking, Integral surfaces, defined thanks to a skeleton along which we can assign a fluctuating weight, provide an intuitive control of a shape of varying thickness.
- Thanks to the additivity of the integral operator, different partitions of the skeleton leave the field unchanged. Integral surfaces are thus an excellent model for defining levels of details (LOD) on implicit surfaces, through the recursive subdivision of their skeleton [AC02, HAC03].

As all implicit surfaces, Convolution surfaces and Hornus integral surfaces suffer from some blending artifacts which will be presented in next section. Specific problems of Weighted convolution and Hornus surfaces are studied in more detail in chapter 3.

1.2 Combining implicit surfaces

One of the main interests of implicit surfaces is the way they can be combined in order to create complex shapes. Their greatest strength is their smooth blending ability but it is not the only way to combine them, as detailed next.

1.2.1 Constructive Solid Geometry

The first modeling systems used boolean operations in order to combine shapes together [Blo97] : this approach is called Constructive Solid Geometry or CSG. The basic boolean operations are the union, the intersection and the difference. These operations can be performed hierarchically thanks to a Construction Tree. They have been introduced to implicit modeling by Ricci in [Ric73] where simple operations are performed on the scalar field, for instance the union of two scalar field is :

$$(f_1 \cup f_2)(P) = \max(f_1(P), f_2(P))$$

The main drawback of this approach is that it creates a C^0 only scalar field, which can lead to the creation of artifacts in the case of subsequent blendings.

In order to overcome this problem, several *clean union* operators have been introduced both for global support [PASS95] and compact support primitives [BWdG04, BBCW10] : such union has a discontinuity of gradient only where $f_1 = f_2$. The shape is therefore the same as for union, but the resulting surface can be more easily reused in subsequent combinations.

1.2.2 Blending

We already introduced the most classical way of smoothly combining implicit surfaces : namely the summation blend. Summing fields provides an easy way to create shapes of arbitrary topology. However, this blending is both the greatest strength and the major drawback of implicit surfaces, due to a number of blending artifacts classified in four sub-problem in [GBC*13].

Blending drawbacks :

1. Bulging problem,
2. Absorption problem (or blurring of details),
3. Locality problem (or blending at distance),
4. Topology problem

to which we can of course add their consequences : the lack of controllability. Visual depictions of these problems is given in Figure 1.14.

The **bulging problem** as already been discussed in previous sections. While reduced by convolution surfaces, it still arises when more than two skeletons join, for instance in a T-junction.

The first thing to note about the **absorption problem** is that it depends on the kind of scalar field. For classic scalar field with compact support (created thanks to polynomial based kernel), this problem is well described in [WW00] : when there is a large difference of radius (and of kernel support) between two objects to be blended, the thinner one will see its volume inflated and the shape of the blend will be sharper than one would expect (see Figure 1.13, note that the volume inflation is bounded by the kernel support). This problem increases as the difference of radius increase. In the case of global support, the only problem is usually the one of the volume inflation. In both cases, the volume inflation problem comes from the fact that the field of the larger shape varies very slowly in the region where the thiner shape is.

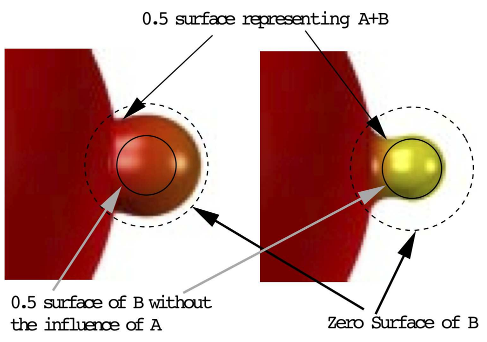

0.5 surface representing A+B
0.5 surface of B without the influence of A
Zero Surface of B

FIGURE 1.13 – *Unwanted inflation of the volume of the thin primitive during blending by summation and sharpening of the blending for large difference of radius (left) compared to a corrected behavior from [WW00] (right). Figure from [WW00].*

The two remaining problems, which are related, are probably the two most annoying when it comes to modeling, partly due to the fact that they can become really unpredictable. The **blending at distance problem** is the fact that proximity is the only factor that make implicit surfaces blend. This makes progressive modeling difficult since users do not know at which distance the different elements should be placed to reach their modeling goals.

This can become even more problematic in some cases, for instance if the hand of a character comes close to its head during an animation, they will blend, which is more than probably unwanted. This **unwanted change of topology** is the last problem : the shape does not neces-

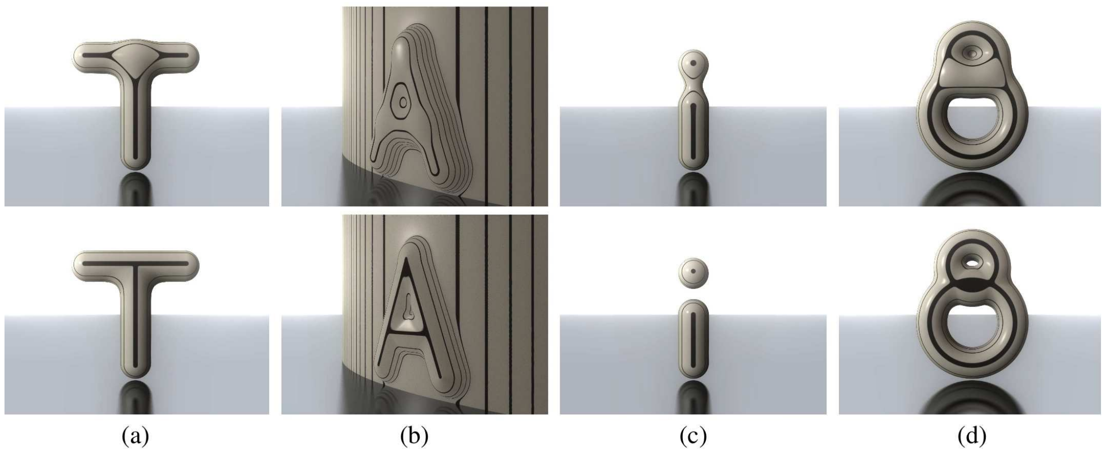


FIGURE 1.14 – *In [GBC*13], the unwanted behavior of the blending are classified into four sub-problems : (a) Bulging problem, (b) Absorption problem, (c) Locality problem, (d) Topology problem. The first row correspond to the unwanted behavior while the second correspond to the desired behavior (obtained through the gradient blending method presented in section 1.2.2). Figure from the paper.*

sarily preserve the topology expected from its skeleton. There is another example depicted in Figure 1.14(d) : if a shape with a hole is created, the hole can be filled if the shape is blended with other one that has some influence in this region of space. To overcome these, blending should be able to preserve the topology of the union of the objects to be blended.

Remarks :

It is important to recall that the blending behavior depends a lot on the scalar field properties. Furthermore, the same blending function cannot always be applied to both *Real-function* [PASS95] which use 0 iso-value and that have global support, and to blob-like scalar field which are positive. We will mainly focus on the second kind.

There are two main ways of combining implicit surfaces, either N-ary blending (as the summation that can combine any number of primitives) or binary blending that will combine only two primitives (as the clean union) and usually provide more freedom.

The remainder of this section studies the existing solutions to the four problems developed in terms of N-ary and then binary operators. We will see that while all blending problems have now been solved for binary operators, most of them are still open for N-ary operators such as the sum. Note that the sum is up to now the only combination operation that preserves independence from subdivision of convolution surfaces.

N-ary operators

Ricci's blending : In order to provide more control on blending, Ricci introduced the following operator [Ric73] :

$$f(\mathbf{P}) = \sqrt[n]{\sum_i f_i(\mathbf{P})^n}.$$

It describes a family of blending that spans from the blending by sum ($n = 1$) to the union by maxima ($n = +\infty$), see Figure 1.15. However, when n increases the gradient direction of the resulting field quickly changes in the region where there is at least two values f_i that are equal and greater than the other field values to be blended. This will lead to poor quality for future blending in this region.

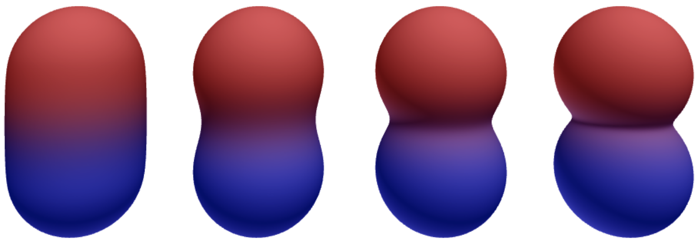


FIGURE 1.15 – *Two blobs blended thanks to Ricci's blending operator [Ric73], from left to right, the parameter n is equal to 1, 2, 4 and 8. Note that when $n = 1$ we have a blending by summation, while when n increases, we tend toward an union by maxima.*

Locally restricted blending : This blending method, introduced in [dGWvdW09], is based on the analysis of the blurring of details problem presented in [WW00] and a proposed solution : the reduction of the blending range of the larger primitive in function of the smaller one.

While this is done by using a modified definition of the base field function in [WW00], this is performed in [dGWvdW09], for more generality, by using a mapping on the base field. The mapping is parametrized by a single parameter that gives the new blending range. The control of the blending range is obtained through the introduction of a complex controller which is parametrized from the one to one wanted blending. This gives the possibility to finely tune the blend (it does not only allow the improvement of blending between primitives with large difference of radii, it also allow users to reduce blending between primitives of the same radius if wanted, and it enables the creation of asymmetric blending), but in return this gives a lot of parameters to be set (ie - the square of the number of primitives to be blended). Results of this blending are shown in Figure 1.16. Colors are usually blended through a barycentric blending whose coefficients are the field contribution of each primitives. This, combined to the mapping of field values used in this method, naturally limits the bleeding of colors.

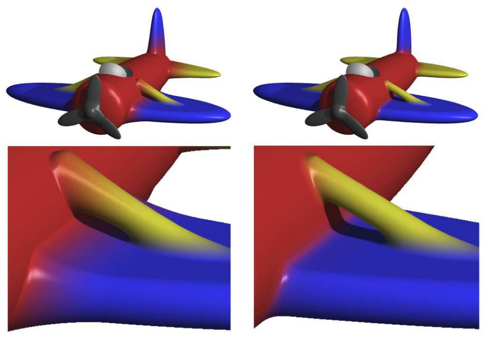


FIGURE 1.16 – *Comparison between blending by summation (left) and locally restricted blending (right). Figure from [dGWvdW09].*

The main drawback of this method is that it may become difficult to set blending when multiple primitives require different blendings in the same area. Furthermore, the use of primitives with varying radius would make the choice of blending range even more complex.

Graph-based blending for convolution : To our knowledge, the two works presented in this paragraph are the only blending methods that are designed especially for convolution surfaces. Both are based on the skeleton seen as a graph, an idea already explored for distance surfaces in [GW95].

In [AC02], the idea is to only use the element of the skeleton which has the highest influence at the computation point and its direct neighbors in the graph in order to compute the field function generated by the full skeleton. The kernel of the neighbor skeleton is multiplied by a weighting function which depends on the distance along the skeleton. This prevents elements

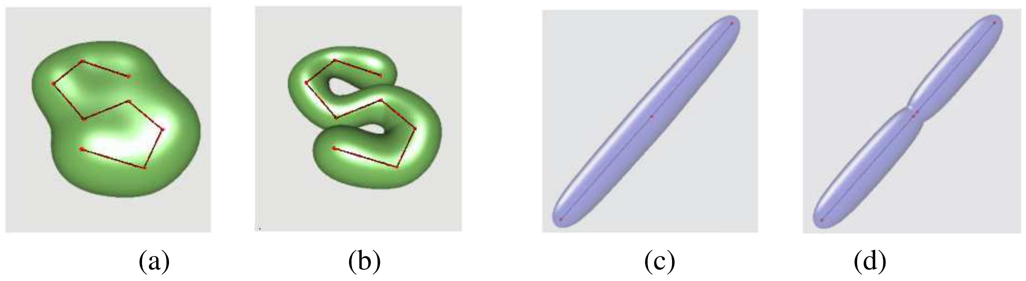


FIGURE 1.17 – *Graph-based blending method of [AC02] : (a,b) blending by summation versus graph-based method, (c,d) artefact creating by small segment insertion. Figure from the paper.*

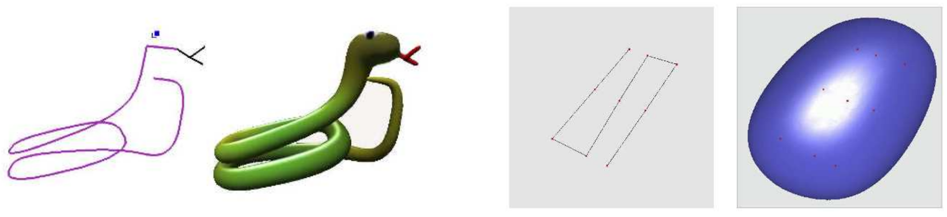


FIGURE 1.18 – *Graph-based blending method of [HAC03] : (a) absence of blending between different part of the snake body, (b) if the folding is too fast the unwanted blending is not avoided. Figure from the paper.*

of the skeleton that are far from one another in the skeleton graph to have mutual influence onto each other (see Figure 1.17(a,b)). However, this comes with a drawbacks : whenever a small skeleton is inserted between two larger ones then a crease appears : the independence to skeleton subdivision is lost (see Figure 1.17(c,d)). Furthermore, the choice of skeleton of main influence can introduce discontinuities.

The approach used in [HAC03] has the same spirit as the previous one, but instead of just using direct neighbors, fields are added until their contributions become negligible. Thus, if the skeleton is a loop, when it comes back close to itself again, it will not blend. However, if the shape folds back directly on itself the desired shape is lost such as in Figure 1.18(b).

Binary operators

Extension to CSG and highly parametrizable blending : In addition to introducing clean union, [PASS95] also introduced a blending function :

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

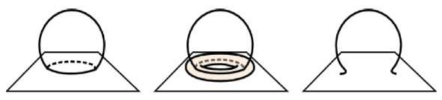
the latter has three parameters, the first one is the global strength of the blending while the two others enable asymmetric blending. If the first parameter is null, we fall back on a clean union. This blending function is only designed for *Real-functions* (global support, 0 iso-value).

In [BDS*03, BWdG04], an alternative method is proposed that gives more control on the shape of the blend, first for global support functions and then for compactly supported scalar fields. In order to explain this blending, the representation of binary blending operator introduced in [HH85] is used : blending operators are represented in 2D, the abscissa being one of the two field value and the ordinate the other, iso-line of the resulting blend are drawn in this plane. One of the main characteristic of the blending function introduced by Barthe is to limit the region where blending occurs in this space : outside of the blending region, the behavior is a classic maximum while inside the region the blend is defined through its iso-line. The simplest solution is to use iso-lines defined by arc of an ellipses but more complex iso-lines can be defined using polynomial functions (through a polygon of control) in order to obtain local shape control in the blending region. The region is defined through an opening angle defined by two boundary lines going through $(0, 0)$. It is equivalent to saying that if the ratio between f_1 and f_2 is greater than a given threshold, then no blending occurs (it works the same way for the ratio f_2 over f_1).

Lately, most of the blending problems were overcome in the case of binary blending operators based on two distinct ideas : bounded blending and gradient-based blending.

Bounded blending : Bounded blending, first introduced in [PPIK02, PPK05], enables the localization of the blend, either by using two control points on the surfaces or by setting an arbitrary bounding volume defined by an implicit surface. This method enables a variety of blending styles and users can tune it in order to control the smoothing of small details. While providing great control, this blending also requires a lot of user input in order to obtain good results.

By introducing a complex binary operator based on extra primitive generated at the *intersection* of the input surfaces, [BBCW10] was able to overcome the blurring details problem automatically. These properties come from the fact that the size of the bounding primitives is chosen accordingly to local surfaces properties such as gradients and curvatures. More importantly, this method also solves the locality problem since blending will only occur near intersection of surfaces and partly solves the loss of topology problem. A possibly remaining problem would be if a small torus is blended into a large object. Then there is no guarantee that the hole would be kept. Result can be seen in Figure 1.19. However, this operator can become quite costly due to the need of finding surface intersection curves which can become quite expensive when there is multiple intersections. The method of [BBCW10] was used in a sketch-based modeling framework, where the created shape is directly blended into an existing object. Therefore, it uses the meshes of the new and old objects in order to compute the intersection curves in an efficient way. The meshes in a local neighborhood (the blending being localized) are deleted and re-generated according to the new local scalar field.


(a)

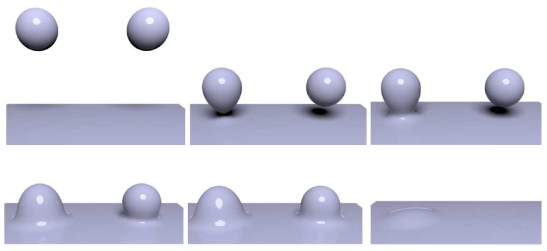

(b)

FIGURE 1.19 – *Blend restricted to contact between surfaces : (a) extraction of the intersection curve and creation of a bounding volume according to surface features, (b) comparison to the summation blending, we can note the absence of blend at distance. Figures from [BBCW10].*

Gradient-based blending : By building on [Roc89] and [BWdG04], Gourmel et al [GBC*13] recently proposed a method that overcomes most of the blending problem : from the first method, it re-uses the angle between gradients (of the scalar fields to be composed) to parametrize the blend, while from the second, it re-uses the idea of an opening function to define where the blending should occur.

The authors introduce a continuous family of quasi-C^∞ blending operators ranging from clean-union like blend to a super-elliptic like blend (see Figure 1.20). The blend to be used in one point in space is chosen thanks to a controller function whose parameter is the angle between gradients of the two fields to be combined.

The main controller which solves the main blending problems uses the fact that if the gradients are collinear then any blending that will occur will create bulging, so the controlleris set to ask for clean-union like blending. On the other hand if the gradients are opposite then any blending have chance to cause loss of topology, so the controller once again asks for clean-union like blending. On the contrary orthogonal gradients means that blending is desired. Blending with this controller are depicted in the second row of Figure 1.14.

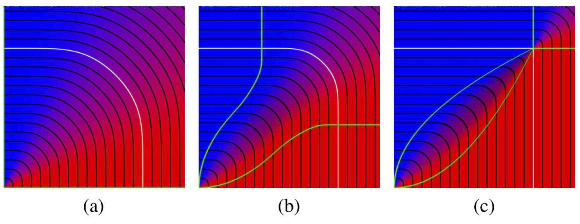


FIGURE 1.20 – *Three blends of the family introduced in [GBC*13] depicted in the representation of [HH85] : (a) is the maximal blending, (b) the mid-range blending and (c) is the clean union or minimal blending. Figure from [GBC*13].*

In addition to solving the four long lasting blending problems, this method also provides a high freedom on the kind of blending that can be obtained due to the possible parametrization of the controller.

In order to obtain an efficient implementation, the authors rely on some pre-computation. The main drawback of the method is the fact that the controller should be adjusted with respect to the field variation of the input field function. Since most of the time a unique kernel is used for all the primitives this limits the required work.

When this blending is used with convolution surfaces, it is important to note that the maximal obtainable amount of blending is somewhere between a Ricci 2 and Ricci 4 blending, which is much less than a summation.

1.2.3 Other combinations of scalar field

Blending primitives is not the only way to combine them. In addition to blending, which can be seen as a smooth extension of the boolean union, it is also possible to introduce digging operators which are a smooth extension of the boolean difference. A more original combination is the squashing combination of implicit surfaces.

Subtractive blending :

In [Bli82], negative implicit volume are introduced in order to dig into existing shapes. This is done by using negative coefficients for blobs and then performing classic blending by summation. For this reason, the basic way of digging can be seen as the subtraction. However this operation presents a major drawback : it can create negative "field wells" which will make subsequent blending completely unpredictable.

The work of [BWdG04] can also be used to produce smooth boolean difference difference with a high control on the level of smoothing. In order to do so it uses an inversion of the field proposed by [Ric73] : $inv(f) = 2c - f$ (which is not well adapted for scalar fields whose values are not bounded between 0 and $2c$).

Squashing :

Such an effect, known as Precise Contact Modeling (PCM), has been introduced in [Can93, CGD97, OC97] : the idea is to detect collision and modify each scalar fields both in the inter-

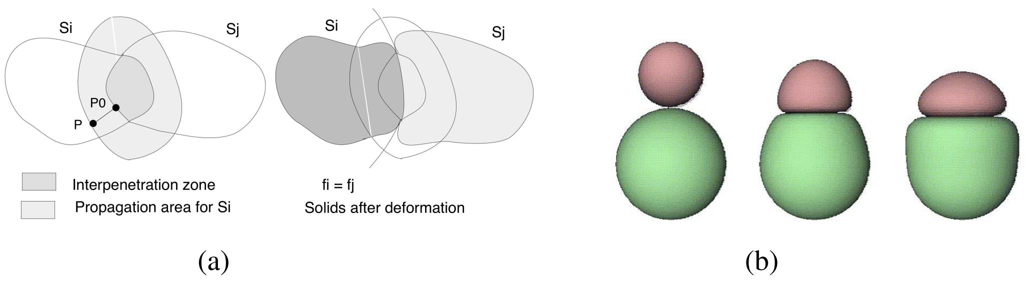


FIGURE 1.21 – *Deformation in the case of contact : (a) scalar field are deformed in a local neighborhood of the overlapping area, (b) result of the method. Figures from [OC97].*

penetration zone and in a local neighborhood of the interpenetration zone (see Figure 1.21(a)). In the intersection region the field f_1 is replaced by $f_1 + (c - f_2)$ while in the neighborhood it is replaced by $f_1 + b_1(f_2)$ with b_1 a function guaranteeing the continuous junction with the remainder of the shape. The second scalar field is modified in the same way, the definition in the intersection area guaranties that the iso-surface is going through points where $f_1 = f_2$ in this region.

In the case of animation, it also enables some parametrization in order to take into account velocities of the objects (see Figure 1.22 for results).

In [AJC02], some additional work has been done in order to apply PCM to convolution surfaces, both on the use of skeletons to improve collision detection and on the choice of the function b_i which determines the shape of the deformation around the collision area (one of the proposed function enables the creation of ripples).

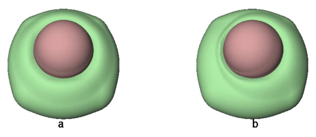


FIGURE 1.22 – *Precise contact modeling can be parametrized in order to take into account velocities of the objects. Figure from [OC97].*

Gradient-based blending [GBC*13] provides a way to create this kind of effect by using alternative completion curves (and a different controller) in the area where the blending should occur.

1.2.4 Adequate inner bound for implicit surfaces

When composition operators such as boolean difference (whether it is smooth or not) are used, some artifacts can be created. They are mainly due to the fact that the term $(1 - f)$ used to perform the required inversion of the field function (in the case of 0.5 iso-value) can become negative. In [CGB13], a solution to this problem is proposed : note only the initial scalar field should be bounded in amplitude, but the bending operator should be defined in order to preserve this property. In the case of 0.5 iso-value, all field values should stay in the range $[0; 1]$ which means that the inverse $(1 - f)$ will also verify this property. The authors proposed a modified version of the gradient-based blending that meets this new constraint : this is done by introducing a new family of boundary curves in the 2D representation of blending operator : these new

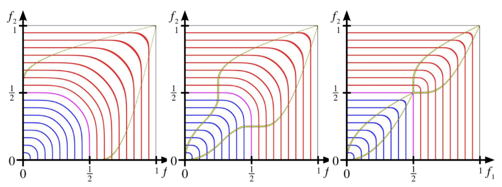


FIGURE 1.23 – *Modification of the gradient blending in order to preserve inner bound. Figure from [CGB13].*

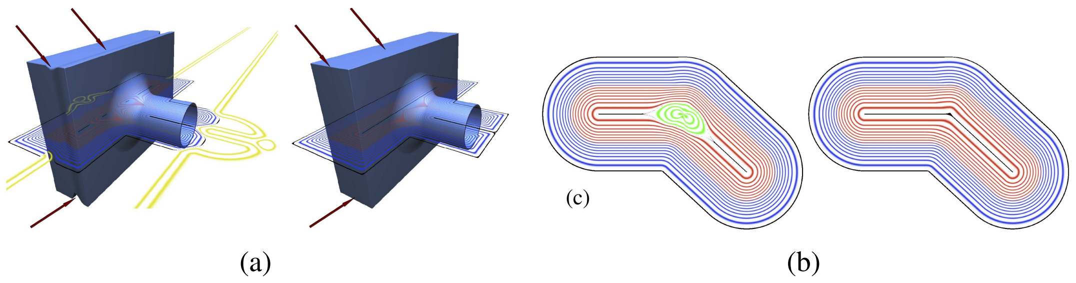


FIGURE 1.24 – *Result of blending with inner bound : (a) it prevents negative field values in boolean difference, (b) it improves the inner scalar field in adapted gradient-based blending (which could have an impact on subsequent compositions). Figures from [CGB13].*

curves go through the point $(1, 1)$ which guarantee to have $g(1, f_2) = 1$ and $g(f_1, 1) = 1$ (see Figure 1.23). It does not only improve the result of difference but also the one of blending (see Figure 1.24).

In addition to these new constraints, the authors also propose a modified blending that improves the field quality when boolean operations (smooth or not) are applied in order to add small details on a larger shape. The aim is to remove the ghost shape of the details that remains after the operation (see Figure 1.25). Not doing so would impair subsequent blending. This is done by modifying the gradient-based operator by progressively ignoring the field of the details when it becomes larger than the iso-value.

In conclusion, the methods presented in [CGB13] highly improve the intuitiveness of the composition behavior in the case of the multiple successive compositions that are required for modeling complex objects.

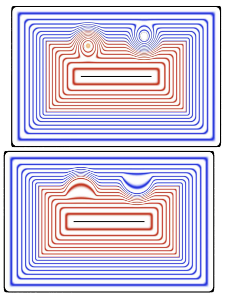


FIGURE 1.25 – *Undesired ghost shape of details remain in the field with standard boolean operations (up), the method proposed in [CGB13] removes this artifact (bottom). Figures from the paper.*

1.2.5 The Blobtree

In order to give more freedom to the user, a hierarchical modeling framework was introduced in [WGG97, WGG99]. The combination of blending and warping (which is presented in Section 1.3) are organized as a tree where each node represents either a composition operator or a warping. The leaves of the tree are the implicit primitives.

In addition to the previously described techniques, optimization methods were also introduced in the Blobtree to accelerate field queries : One of the first acceleration method is the Blobtree reduction and pruning introduced by [FGW01] in order to limit the cost of a field query. Reduction is the simplification of the affine transformation nodes by pushing them down in the tree, while the pruning is the construction of simplified trees depending on regions of space (indeed, in large Blobtree, only a small subset of the primitives are used in a given region of space). Later, a caching scheme was introduced by [SWG05] in order to avoid expensive computational costs for objects that aren't modified by the user .

Main limitations : While all the existing blending and composition operations give a lot of freedom to the user, most of them are binary operations. This implies an explosion of the height of the Blobtree when creating complex objects : this can have an impact on the computational time. Furthermore, the main drawback is probably that it makes the handling of this structure more cumbersome, making it less practical for interactive manipulations.

1.3 Deformation

Combination of implicit surfaces through blending, digging and squashing operators are not the only tools to create complex shapes with implicit surfaces : deformation of space - *warping* - has been used to greatly increase the kind of shapes that can be created.

1.3.1 Warping : basic principle

Deformation of space (or space warps) such as Barr's bending, twisting and tapering [Bar84] have been used for a long time for defining complex objects from simpler ones.

The idea was extended to the skeleton based implicit modeling framework in [WO97], where Barr's deformations were used to easily define complex primitives. The idea was to define the scalar field $\tilde{f}$ of the new primitive from a warping $w^{-1} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ and a simple primitive of field f, in the following way :

$$\tilde{f}(\mathbf{p}) = f(w^{-1}(\mathbf{p}))$$

In other words, query points are transported through w^{-1} to another space, where the new primitive warps into a simpler one (see figure 1.26).

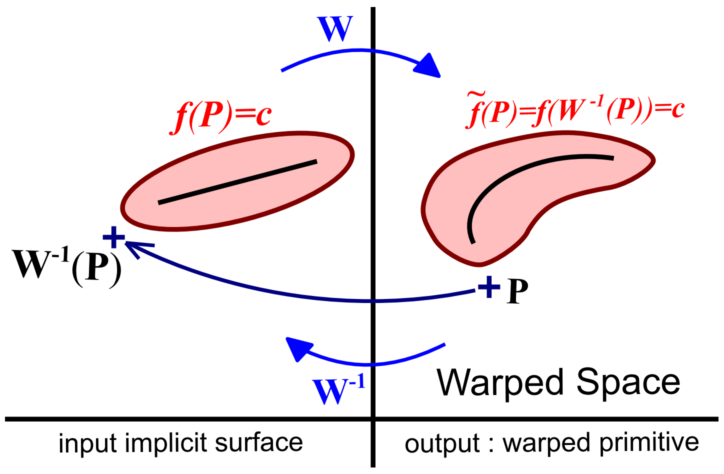
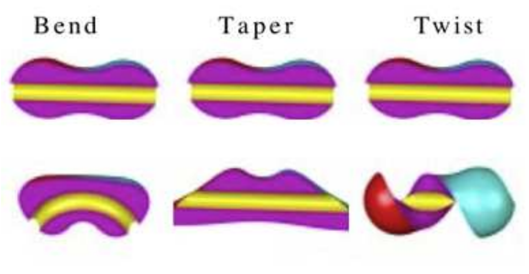


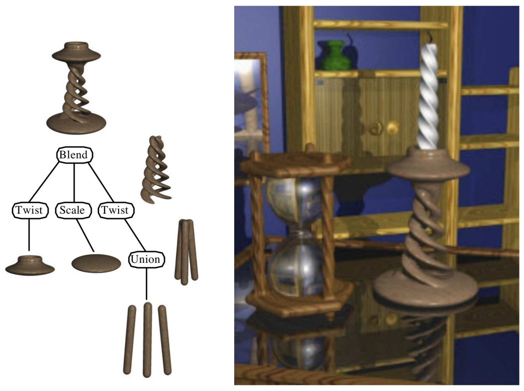
Figure 1.26 – *Principle of warp-based implicit primitives.*

It is important to note that w^{-1} is an inverse warp, since it is the function that maps the scalar field embedding a complex object after the deformation to a simpler scalar field seen as an input. This is due to the implicit nature of the surface. Indeed, if we want to perform blending after the deformation, then we should have access to all the scalar field values in space. We cannot just apply the deformation w to the surface itself as it would have been done for explicit surface representation such as parametric one.

While basic warps are defined on all the space, it is possible to restrict their use to a bounded domain by defining a deformation field of compact support.



(a)



(b)

FIGURE 1.27 – *(a) Classic warps applied to implicit surfaces, Figure from [WO97]. (b) Twisting used inside the Blobtree in order to create a complex shape , Figure from [WGG99].*

A problem that should be noted when using classical warps to create complex primitives from simpler ones is that the surface usually undergoes larger deformation than the skeleton : for instance, circular cross-sections are usually not transformed into circles, which can be seen as a drawback in some cases.

1.3.2 Intuitive deformation :

Classic warping are transformations that affect the whole space, and do not provide much control on the shape deformation. Sugihara introduced two different methods to make deformations more intuitive for the user. Both methods use curves drawn on the surface as a deformer manipulated by the user.

The first method [SdGWS08] is based on free-form deformations (FFD), a method used to deform an object embedded inside a grid. Because manipulating the grid itself is cumbersome, the authors compute the deformation of the grid thanks to a curve drawn on the surface and deformed by the user. However, FFD does not provide an inverse mapping, which is required to apply a warping to implicit surfaces. In order to do so, the authors compute a deformation field through variational warping (a method that guarantees to interpolate a set of constraints through a C^2 deformation). Yet, such a method is expensive due to the inversion of a large matrix, for this reason an approximation scheme is developed in order to provide interactive feed-back.

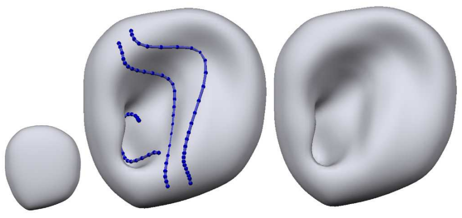


FIGURE 1.28 – *An ear created from a single flat primitive and three Warp Curves. Figure from [SWS10].*

In [SWS10], Warp Curve, a deformation method also based on a deformation curve is presented. In contrast with the previous method, the goal is to obtain a local deformation that precisely meets the curve constraints. In order to do so, it combines a variational warp with a bounding volume. The variational warp is computed from the curve constraint (each vertex of the curve sets a displacement corresponding to the difference between old and new position) and additional off-

curve constraints with null displacement in order
to guarantee that the displacement is decreasing as we go away from the curve. Because the variational warp has a global influence, it is necessary to cancel it outside of a local neighborhood, which can be done by multiplying it by a cancelation factor since it is defined as a displacement. This cancelation field is computed through a convolution field generated by the constraint curve. The authors also present a method to take into account successive deformations while still meeting all the constraints. This is a powerful tool since it enables the creation of complex shapes that cannot be easily created with other implicit modeling methods (see the ear in Figure 1.28). The main drawback of the method is probably the fact that it deforms the field at distance, and not in a geodesic way.

1.3.3 Conclusion

This thesis heavily relies on space warping, not only to create convolution-like primitives along 3D curves, but also to improve blending property of convolution surfaces.

1.4 Generating Details

There are several way to enhance surfaces with details. The two main kinds of details are textures (that represent the material of an object : its color, shininess,...) and the actual geometric details. The first one can be used to represent the second one in *trompe-l'oeil* using the bump-mapping technique which consists in modifying normals of an object using a gradient texture. Note that if the surface is represented by a mesh, geometric details can be obtained by using displacement-mapping which consists in moving the vertices along the normal of the surface.

Although many methods are available to add details to mesh-based surfaces (e.g., displacement mapping [Coo84, ADBA09] or shell mapping [PBFJ]), methods to add details (or textures) to implicit surfaces are scarce due to the difficulty to add coherent surface details when no parameterization is available. We only present the methods that can be applied without loosing the implicit representation of the shape.

1.4.1 Solid texture & Hypertexture

Solid textures, introduced simultaneously by Perlin and Peachey [Per85, Pea85] are procedural method to define noise in 3D : they were introduced in order to texture arbitrary surfaces. In order to texture a surface, the noise value just have to be computed for points belonging to the surface. This does not require any parametrization since the noise is defined in the global coordinate system.

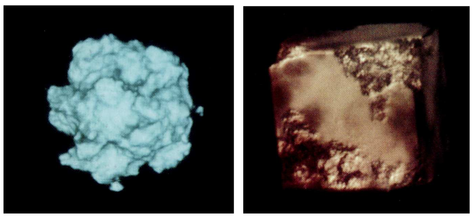


FIGURE 1.29 – *Small scale detail obtained by modulating a scalar field with a volume noise function. Figure from [PH89].*

However, the noise function is anchored in the 3D space which is problematic for animated implicit surfaces : the texture would slide onto the surface as the latter moves. In [WWM87], the problem is overcame by using local coordinates in order to compute the noise value. These coordinates are defined as the weighted sum of local space coordinates associated to each individual primitives.

Hypertextures were introduced in [PH89] in order to modify volume model (including implicit surfaces). The scalar field is directly modified by the noise function. While this method enables the creation surfaces with a lot of details (see Figure 1.29), it cannot generate *surface* details : it can generate small disconnected elements around the main shape as well as holes inside the object, furthermore the noise appearance is not preserved (which is mainly problematic for anisotropic noise).

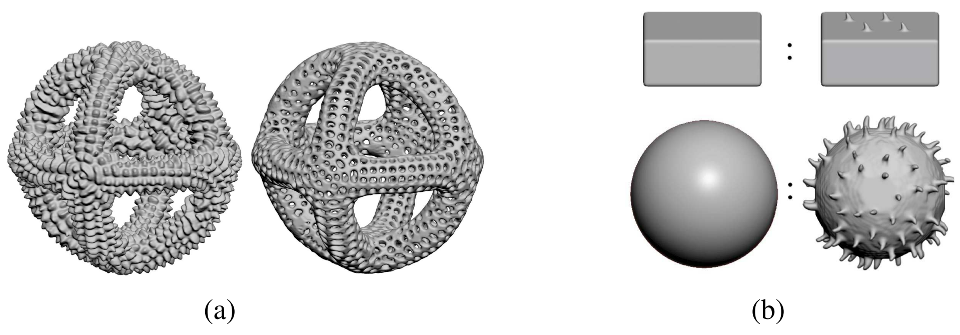
1.4.2 Cellular Texture Generation

Cellular systems, introduced in [FLCB95], can be used to add details over a surface (either implicit or not). In order to do so, a particle system is set to spread on the surface. Each particle can then generates a small scale detail. One of the main advantages of this method is that details can have complex shape. However the cost of the simulation to obtain the repartition of details can be high.

Other methods use particles in order to define details, for instance [SAC*99] use particles inside an implicit surface (point skeletons are used as particles) to locally define a Perlin noise that follow the surface during animation : this method is used to simulate lava flow.

1.4.3 Geometric Texture Synthesis by Example :

The only method that requires no parametrization in order to create coherent surface detail is presented by [BIT04], which extends example-based texture synthesis to volumetric objects defined by iso-surfaces of a discrete scalar field, stored in a voxel grid. One of the advantage of this method is that it can create details that are not mere height map, including details that completely change the topology of the shape. Unfortunately, this method is compute-intensive, the complexity of details is limited by the resolution of the grid, and the method does not provide any mechanism enabling geometric texture to be smoothly interpolated when two textured surfaces blend. Furthermore, when small details with large distance between them are to be created, the method also creates small noise over the surfaces even if it is perfectly smooth (see Figure 1.30).



(a) (b)

FIGURE 1.30 – *Geometric texture synthesis by exampes : (a) created shape can change of topology if wanted, (b) base example, we can note that despite the detail example is smooth between the spike, it is not the case anymore for the generated sphere. Figure from [BIT04].*

1.4.4 Mapping 2D textures on implicit surfaces

Particle system mapping

A solution presented in [ZGVdF97, ZGV*98] for mapping a 2D texture onto an implicit surface is to introduce a physical particle system driven by the gradient field to match points of the implicit surface with those of a textured support surface. This method provides a global mapping of an implicit surface, as in Figure 1.31 where a cylinder is used as support surface. However, it can easily produce texture distortion in curved areas of the surface.

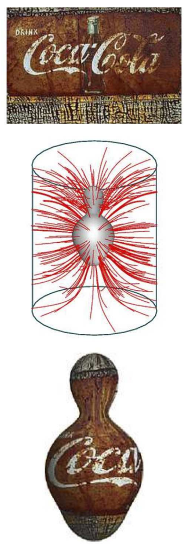
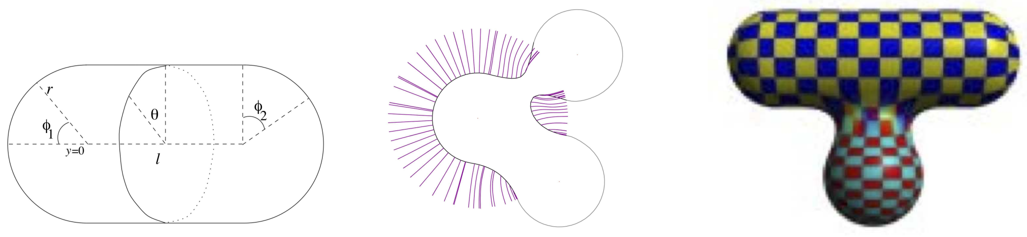


FIGURE 1.31 – *Texture mapping between an implicit surface and a textured surface (a cylinder) driven by a particle system. Figure from [ZGVdF98].*

Some extensions of this work have been presented : animated articulated shapes are handled in [ZGVdF98] and texture mapping without discontinuity in presence of CSG operations are provided in [TW98]. Lastly, Tigges [TW99] uses the fact that skeletons (associated to their 0 iso-value) are surfaces that are easily parametrizable in order to remove the need to define a support surface. In area unaffected by blending, the mapping is directly obtained without needing the expensive migration process, while a special migration scheme is introduced in blending area, each skeleton obtains its own texture which is blended to the other using a weighted sum.

The main drawbacks of this family of methods are probably its high computational cost and lack in user controllability (either distortion of texture or ghosting effect are present).



(a) (b) (c)

FIGURE 1.32 – *(a) support surface parametrization for a segment skeleton, (b) migration taking into account blending area, (c) result for a blending between two skeletons. Figure from [TW99].*

Decals

Decals to texture implicit surfaces were introduced by [Ped95]. In this method, patches should be designed by the user on the implicit surface which is a time consumming task.

In order to overcome this problem, [SGW06] introduce a decal method based on exponential map. This method is designed for any type of sampled surfaces including implicit surfaces. It uses a modified Dijkstra's graph-distance algorithm to compute an approximation of the exponential map which is a local parametrization ; the user just have to provide a center, orientation and radius in order to apply a texture patch onto the implicit surface. The texture behaves

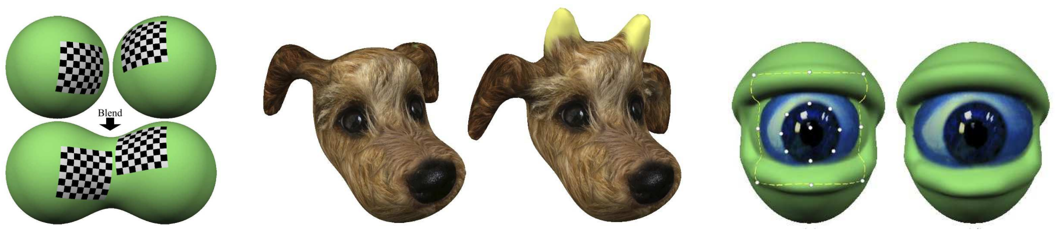


FIGURE 1.33 – *(a) Behavior of decals in blending area, (b) texture are preserved during shape modifications, (c) parametrization can be deformed to provide more control to the user.. Figure from [SGW06].*

nicely in blending regions, thus the user does not have to bother about topology changes (see Figure 1.33(a,b)). Furthermore, the texture parametrization can be deformed to provide more control to the user (see Figure 1.33(c)). This method provides dynamic texture placement and can be applied to animated implicit surfaces (provided that the decals are attached to the skeletons).

1.5 VISUALIZATION OF IMPLICIT SURFACES

In this section, we provide a short review of methods used to visualize implicit surfaces. Indeed, if not the topic of this document, it is still a point to be discussed since their display is one of their main drawbacks due to its high computational cost. There are two main ways of visualizing objects in Computer Graphics, projective rendering of meshes (for which the hardware is optimized) and ray-tracing. Therefore, there are two main ways of visualizing implicit surfaces, either by meshing them or ray-tracing them, an additional solution being point-based rendering.

1.5.1 Meshing

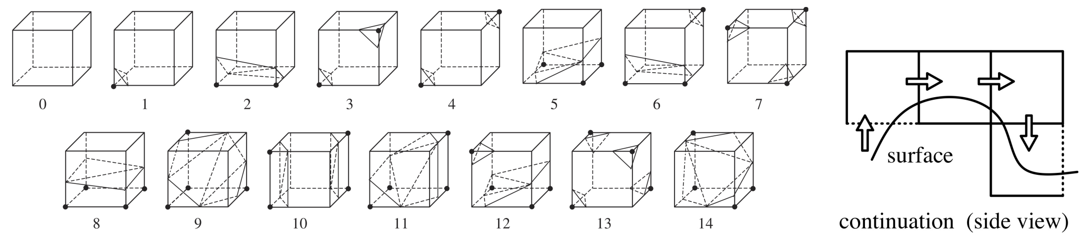
Methods for meshing implicit surfaces can be classified in four categories : grid-based methods such as marching cubes, propagation methods such as marching triangles, cage-based methods and particle systems.

Grid-based methods

Grid-based methods are probably the most common methods to be used (this type of tessellation method is implemented in common software such as Maya, 3DS Max and Blender for the display of metaballs).

Exhaustive enumeration : marching cubes

Marching cubes [WMW86, LC87] is the most common method to mesh an implicit surface. It works by subdividing space into a regular grid, each cell of the space being treated independently in order to create triangles representing the surface. In order to do so, field values are computed at each corner of a cell. Then, a precomputed table is used to create the required triangles (there are 15 base cases that could arise, corresponding to 256 effective cases, see Figure 1.34(a)). Each edge that crosses the iso-value creates a vertex for the triangulation (the



(a) (b)

FIGURE 1.34 – *(a) The 15 base cases of marching cubes, Figure from [NY06]. (b) Propagation process to reduce computational time, Figure from [Blo94].*

exact position being computed either through linear interpolation or some optimization methods).

This method is simple but it suffers from a high computational cost and it results in triangles of poor quality (see left torus in Figure 1.35(a)). Furthermore, it may lead to inconsistency for the resulting mesh in some cases.

In [WMW86], and later [Blo94], in order to avoid a lot of processing (and so save a lot of computational time), the algorithm can be used to only process cells that are near the iso-surface by using a propagation method. In order to do so, a first point on the surface should be known. The other cells to be processed are then the one adjacent to a face of the initial cell that cross the iso-surface (see Figure 1.34(b)). This propagation method drastically decreases the computational cost. However a point on the surface must be known for each connected part of the surface (which is not really a problem when using skeleton-based implicit surfaces).

Improvements :

A lot of improvements over marching cubes have been proposed over the initial method to tackle four problems :
- manifold and topology,
- triangle quality,
- multi-resolution,
- sharp edges.

Let us quickly describe some of the main improvements over marching cubes.

Refinement of the grid, leading to an octree subdivision of space, can be used to adaptively refine the surface [Blo88] (criteria such as divergence of normal at vertices or distance between the center of polygon and the surface are used). Improved criteria of subdivision, such as interval arithmetics [Sny92] or Lipschitz constant [KB89], can be used to obtain better topological guarantee. Adjacent cells with different depth in the octree can lead to cracks in the resulting mesh if no correction step is performed.

There is a range of dual methods, [JLSW02, Nie04, SW04], that create vertices inside cells instead of on the edges, and create quads for the edges of the grid : these methods provide meshes of higher quality. In one of the paper on the topic, [JLSW02], it is shown that this kind of method can naturally be extended to octree instead of regular grid (removing the problem of cracks). Furthermore, special placement of vertex inside a cell can be used to reconstruct sharp edges.

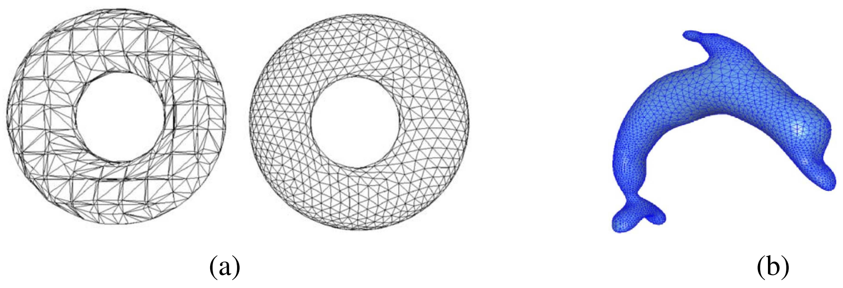


FIGURE 1.35 – *Marching triangles : (a) comparison with marching cubes shows triangles with better quality, (b) multiresolution triangulation. Figure from [dAJ05].*

Grid-free propagation methods

One of the main problem of grid-based method is the poor quality of triangles they produce. In order to provide mesh with near equilateral triangle, marching-triangle algorithm were introduced [HIIT97], see Figure 1.35(a). A first triangle should be created, then each edge of this triangle will create a new point which is projected onto the iso-surface. Then, Delaunay constraints are checked, if they are met, a new triangle is created. Such a method can create seams where another method should be used to obtain a final watertight mesh. This algorithm has a lot of variants, mainly to obtain multi-resolution [AG01, dAJ05, CS07].

Such a method tends to be slower than grid-based method when the field function is expensive to evaluate due to the required projection step ; but it results in a higher mesh quality.

Cage-based methods

Such methods are based on the fact that the skeleton used to generate implicit primitives are generally a good descriptor of the shape. Cages are created around each skeletons, then points are projected onto the iso-surface. Among them, the territory method [DTpG95] creates an independent mesh for each skeleton polylines, this is particularly adapted to the meshing of tubular shapes due to initial cage created form quads aligned with the base skeletons. Note that vertices are restricted to stay in a given neighborhood of the associated skeleton in order to reduce the overlapping area between meshes. Furthermore, it is particularly well adapted to animation due to small update of vertices position between two frames.

On the other hand, B-Mesh [JLW10] try to create a high quality quad-mesh. In order to do so, the initial cages have only square sections. They are connected around the skeleton connections thanks to a convex hull. From this initial rough surface, a quad mesh of arbitrary resolution is obtained through Catmull-Clarck subdivision and projection of vertices on the iso-value. The main problem of this kind of method is the total absence of topological guarantee, if the initial "guess" is wrong then whatever the number of subdivision the result will be of poor quality.

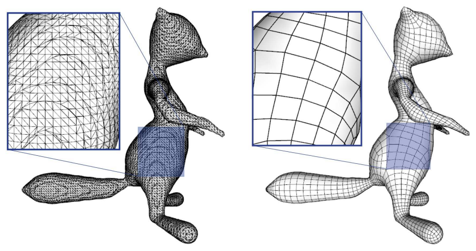


FIGURE 1.36 – *Comparison between marching cube (left) and B-Mesh (right). Figure from [JLW10].*

Dynamic particles system and tracking

Dynamic system of particles can be used to track an evolving implicit surface, of course such methods need an initial sampling of the surface. In [BN07], a method based on finite element is used to provide a "mechanical" mesh with near equilateral triangles that tracks the implicit surface. This mechanical mesh is subdivided at each frame in function of the curvature to obtain more details in the zones of high curvature. In [YT13], a new definition of the velocity of an implicit surface is introduced that helps surface tracking.

In [SH97], Morse theory is used to detect changes in topology by tracking critical points of the field function for time evolving implicit surfaces. This is used to provide guarantee on topology during both initial meshing of an implicit surface and update of the mesh over time.

1.5.2 Ray-Tracing

Ray-tracing that consist in casting ray from each pixel of the camera provide better quality pictures than meshing with rasterization, usually at the cost of higher computational time. The brute force method that consists of marching in the direction of the ray with a fixed step size and then performing dichotomy as soon as a surface is crossed is particularly slow. Several methods have been used in order to provide more efficient algorithms either limited to polynomial scalar field or to scalar field with known properties (such as bounds on the derivatives).

Polynomial functions

Analytical root finding : Some point-based surfaces are described by part thanks to polynomial of low degree. In this case, provided that a ray is subdivided according to the different part of the definition, it is possible to analytically find the root of the polynomial (hence the iso-surface) on a sub-interval provided that the degree of the polynomial is lower or equal to five. This method was introduced by [NHK*85] to ray-trace metaballs.

In [GPP*10], an acceleration structure is introduced in order to render larger number of metaballs in real time (see Figure 1.37).

Numerical methods : Several numerical methods exist to find all the roots of a polynomial of arbitrary degree. For instance, [WT90] used Laguerre's method in order to ray-trace Soft Objects. In [NN94], properties of polygon of control in Bernstein basis (Bezier curves are inside the convex envelop of its control polygon) provide fast way to discard non crossing interval and fast rate of convergence to the iso-surfaces in the other cases (by using DeCasteljau algorithm). This method is known as Bézier clipping.

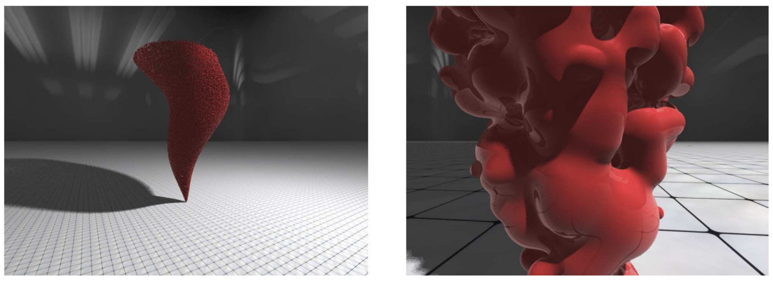


FIGURE 1.37 – *Combination of analytical polynomial root-finding, optimization structure and GPU can provide real-time rendering for large set of metaballs. Figure from [GPP*10].*

Restricted scalar field behavior

For more general functions, some efficient algorithms can be used to provide efficient iso-surface detection along a ray. This is for instance the case of both LG-surfaces [KB89] and interval analysis [Mit90] which aim at isolating the first crossing of the iso-value along a ray. When it is found, more simple optimization technic can be used to find the root (such as a combination of Newton method and regula-falsi). LG-surfaces uses bound on variation of the field and its gradient while interval analysis performs computations on intervals instead of reals.

Instead of trying to isolate the first root along the ray, sphere tracing [Har94] find the size of a step that guaranties to not cross the iso-surface. Similarly to LG-surfaces, this method requires the knowledge of a bound on the Lipschitz constant of the field function, however it does not need knowledge on the variation of the gradient (nor does it require its computation). The main drawback of this method is that it performs poorly when a ray is tangent to the iso-surface and passes close to it (see Figure 1.38).

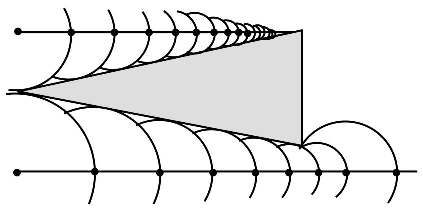


FIGURE 1.38 – *Sphere tracing with a Lipschitz bound of 1. Numerous steps are required when the ray follow the surface at a small distance. Figure from [Har94].*

Optimization for convolution

Since convolution surfaces are quite expensive to compute, Sherstyuk [She99b] proposed the idea of using low degree polynomial approximation in order to approximate them in the direction of the rays. The idea is to sample the support of a primitive in the ray direction and use cubic interpolation by part of the scalar field values and derivatives. This provides two main benefits : it reduces the number of convolution evaluation needed and it provides analytical root finding (provided that a blending by summation is used).

1.6 APPLICATIONS

Implicit surfaces have been used in a lot of different kind of applications from fluid simulation to sculpting method, we present some of them in this part.

1.6.1 Soft & Fluid-like material

Implicit surfaces are particularly adapted to model object with topology that can vary freely along time. In fluid simulations various kinds of blobs (isotropic and anisotropic one to improve appearance) have been used in order to render water from simulated particles. Blobs can also be used to simulate lava flow. Other kinds of soft substance can be simulated using particle system and blobs such as in [DG95] where precise contact modeling and volume preservation are performed. Some examples are visible in Figure 1.39.

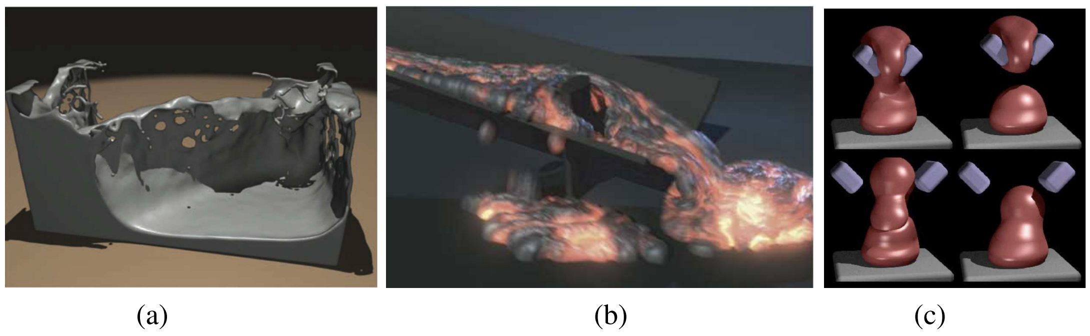


FIGURE 1.39 – *Examples of soft & fluid like material generated from particle system and blob-like surface. (a) In [SS11] anisotropic kernel are used to improve the quality of water simulation, (b) in [YT13] a new method to compute speed of implicit surfaces is introduced, it is used to improve quality of motion blur such as in lava simulation, (c) in [DG95] a soft substance undergoes changes in topology with complex behavior. Figures from the papers.*

1.6.2 Skeleton-based modeling

Skeletons provide a simple way to manipulate surfaces. A lot of the modeling framework using implicit surfaces directly manipulate a skeleton. This is for instance the case of the work of Bloomenthal [Blo95b] (which used skeleton-based implicit surfaces to model organic form such as a hand) and Sherstuyk [She99a] (which modeled seafood, see Figure 1.40). Blobtree structure was initially used in skeleton-modeling framework, recent work introducing improved blending and new method of texturing were presented in [dG08].

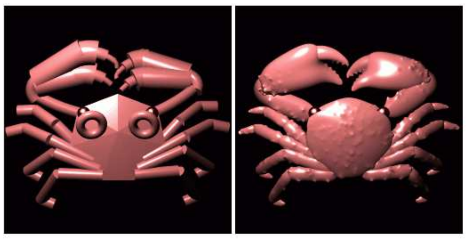


FIGURE 1.40 – *A crab modeled thanks to point, circle, line and triangle skeletons. Figure from [She99a].*

1.6.3 Sketch-based modeling

Various kinds of sketch-based modeling framework have been used in order to provide intuitive modeling techniques. Among them we can find ConvMo [lTZkF04], ShapeShop [SWSJ07] and Matiss [BPCB08]. Both ConvMo and Matiss use convolution surfaces in order to generate shape from an extracted medial-axis. On the contrary, ShapeShop introduces a new kind of primitive based on variational interpolation in order to ease the approximation of the sketch. These primitives are used inside the Blobtree structure to enable the application of various CSG operations and blendings.

Two main sketching paradigms are used : either the user only draws lines corresponding to the silhouettes of the object which makes the analysis of the

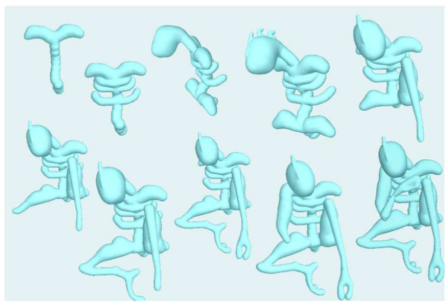


FIGURE 1.41 – *A complex shape is created from simpler one in the sketch-based modeling application Matiss. Figure from [BBCW10].*

drawing difficult, or the user draws a plain object which ease the analysis of the shape.

In [PCP10], a painting metaphor is also used but with additional annotation used to improve the reconstruction of non-planar branching shapes with possible occlusion (the aim being the representation of vascular system). This work also uses convolution surfaces for the effective reconstruction of the geometry.

1.6.4 Animation

There have been several interesting applications of implicit surfaces to animation.

Wrinkling in cloth simulation :

Cloth simulation can become really expensive if fine details are to be obtained. In order to overcome this drawback, in [RPC*10], a coarse cloth simulation is augmented with wrinkling, the latter are generated when compression is detected in the coarse simulation. In order to generate the actual geometry, convolution surfaces are used to deform the base mesh. The placement of convolution skeleton uses a space time approach that ensures temporal coherence. The implicit formulation enables smooth appearance and disappearance of wrinkles as well as fusion and separation. Results of this technique are shown in Figure 1.42.

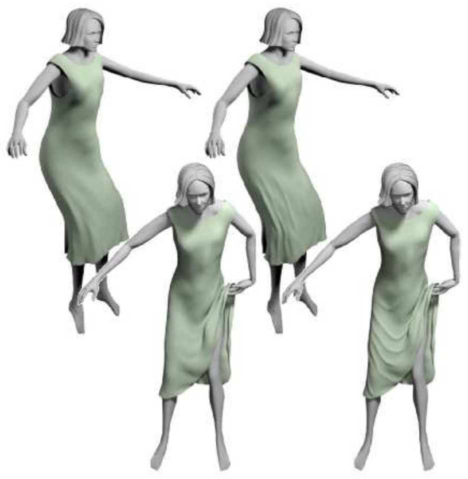


FIGURE 1.42 – *Wrinkling are generated on a coarse simulation in order to enhance its appearance without impacting to much the computational time. The geometry of wrinkling is generated thanks to convolution surfaces Figure from [RPC*10].*

Skinning :

Skinning of character for animation can present a lot of artifacts near skeleton joints. In [Blo02a], in order to reduce this artifact, a first skinning is used to animate the medial axis of the shape which in turn is used to animate the surface itself. All the skinning weights are automatically computed using convolution.

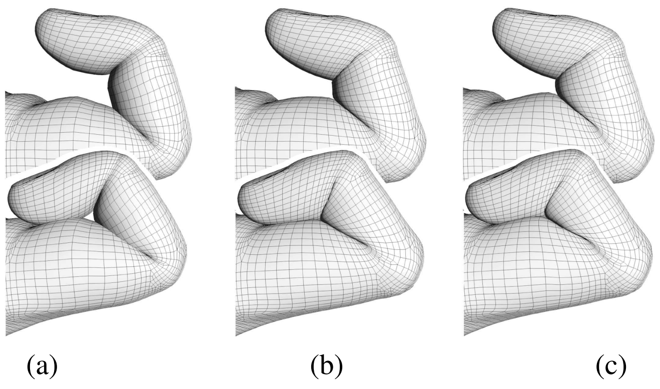


FIGURE 1.43 – *Comparison between : (a) linear blending, (b) base implicit skinning, (c) implicit skinning with bulge in contact. Figure from [VBG*13].*

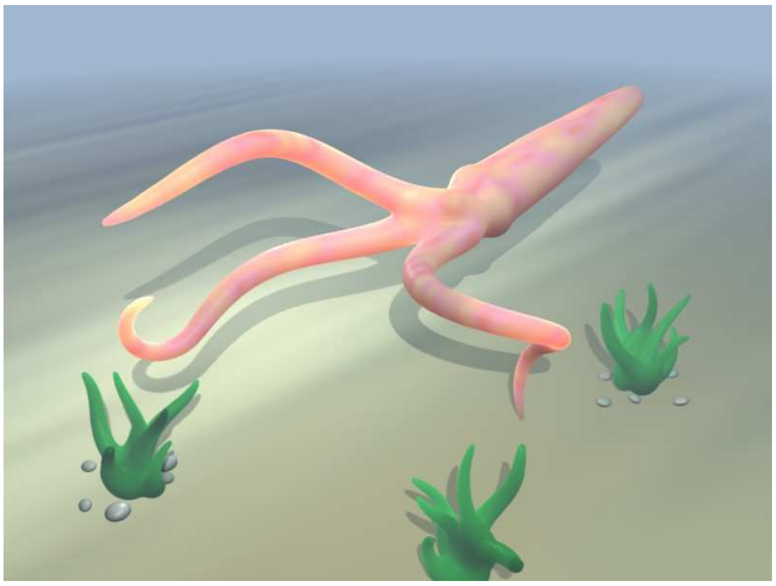
In [VBG*13], skinning is improved by using implicit surfaces to handle skin contact and other effects such as bulges in contact, providing better result that classical skinning techniques and still performing in real-time. First the mesh is subdivided into parts that correspond to each skinning bone. Each part is approximated by a Hermite RBF. These base primitives represent an implicit version of the initial mesh after combination by union. The idea is to re-project the vertex obtained from classic dual quaternion skinning onto the new implicit representation of the shape. In order to keep details, each vertex of the initial mesh stores its initial iso-value and is reprojected onto it. Furthermore, in order to prevent sliding of vertices, a tangential relaxation is introduced. In order to introduce a bulge in contact special versions of the composition operator can be used. Results are shown in figure 1.43.

1.7 Conclusion

To summarize, here are some of the main remaining drawbacks of convolution surfaces : First, they lack complex skeletons that would enable shapes to be created along 3D curves. Secondly, and more importantly, while blending problems have been handled with binary operators, the latter broke the main properties of convolution surfaces which is independence to skeleton subdivisions. Furthermore, using too many binary operators makes the Blobtree more complex, leading to more cumbersome manipulations for the user. Lastly, there is only few methods to add structured details on implicit surfaces.

CHAPTER

2 COMPLEX PRIMITIVES BY WARPING WARP-BASED HELICAL IMPLICIT PRIMITIVES



Teaser Figure : *Implicit modeling of a squid. Each tentacle is made of two helical primitives.*

Publication :

ZANNI C., HUBERT E., CANI M.-P. : Warp-based Helical Implicit Primitives. In *Shape Modeling International (SMI)* (Herzliya, Israël, June 2011), vol. 35, Issue 3 of Computers & Graphics, pp. 517–523.

CONVOLUTION surfaces require a closed form expression of their field functions to achieve reasonable computational time. So far, this has been found for planar skeletons such as line segments, arcs of circle, quadratic curves or triangles. For this reason, 3D skeleton curves are typically approximated by a serie of planar curve segments. Depending on the number of those planar primitives, either the computational efficiency or the visual quality of the shape is impaired.

In this chapter, we introduce for the first time an analytical implicit primitive around the simplest non planar space curve, the circular helix ; i.e. a curve with constant curvatures and constant torsions. It is a promising high level primitive for tessellating arbitrary 3D curves and can be used by itself for modeling realistically a wide range of organic shapes, from hair locks [BAC*06] to natural branching structures such as trees or antlers, see Figures 2.16 and 2.17.

Since deriving closed-form solution for integral surfaces along helices is difficult, the solution we explore is based on space warping : we show that warping techniques can be adapted to efficiently generate convolution-like implicit primitives of varying radius along helices. As we shall see, initial warping transformations such as *twist* do not preserve the circular shape of the normal cross-section (i.e. the section orthogonal to the local helix tangent) of the primitive (see Figure 2.11, first row), and thus cannot be applied as is for our purpose. Depending on a single parameter of the helix, we warp it onto an arc of circle or onto a line segment, for which closed form convolutions are known for entire families of kernels. The new warps introduced preserve the circular shape of the cross section of the primitive.

Overview

Our work builds on both analytical convolution primitives and on space warps : we introduce specific space warps to get artefact-free analytical implicit primitives along circular helices.

Our first contribution, described in Section 2, is the introduction of convolution primitives for arcs of circles with polynomial weights and families of infinite support kernel, a part of the work mostly done by our collaborator Evelyne Hubert. A new piecewise defined warp then provides analytical expressions for helical implicit primitives.

Our second contribution, presented in Section 3, is the introduction of artefact-free space warps to generate helical primitives from line-segment ones.

We show in Section 4 that these two alternatives are complementary and can be combined to provide helical convolution primitives covering the whole range of helical skeleton shapes. We additionally show that, for a lower computational cost, the shapes based on our helical primitives are visually more appealing than those obtained with a standard tessellation into line segments.

Notations : Without loss of generality we consider that the helical skeleton to be processed is given by :

$$\begin{cases} x = R\cos(u) \\ y = \epsilon\, R\sin(u) \\ z = S\,u \end{cases} \tag{2.1}$$

where $\epsilon = \pm 1$ and $u \in I \subset \mathbb{R}$. I is the interval defining the part of the helix used as skeleton. The length of the latter is thus $\sqrt{R^2 + S^2}|I|$. The twist is defined as the ratio $\frac{S}{R}$; we shall see that it dictates the warp to be used. When this ratio tends to 0 (respectively to ∞) the helix tends to a circle (respectively to a line segment). Figure 2.1 gives some examples of the shape

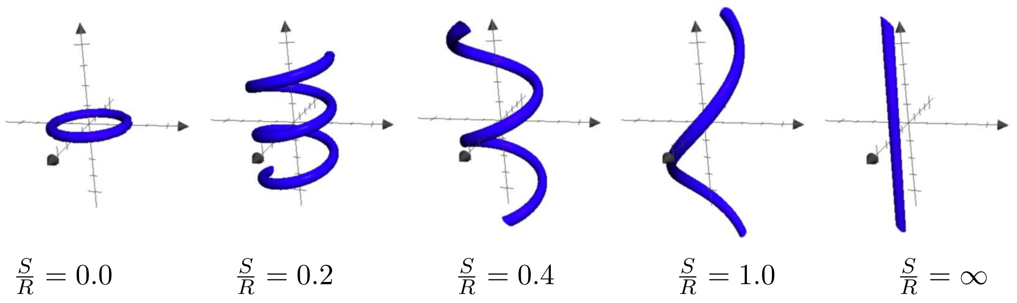


FIGURE 2.1 – *Shape of the helix for increasing value of* $\frac{S}{R}$.

of the helix according to the twist. The thickness of the primitive is assumed to be smaller than the radius of curvature of the helix, as otherwise the warps run into singularities.

The Cartesian coordinates of a point P in 3D space are noted (x, y, z). Its cylindrical coordinates are (r, θ, z) where θ is the planar angle with the x-$axis$ (see Fig. 2.2(a)).

Our further derivations are applicable to common superset of Cauchy and power inverse kernels of arbitrary orders.

The warps we make use of are space transformation $W : \mathbb{R}^3 \to \mathbb{R}^3$ that maps the circular helix to a simpler skeleton curve (see Fig. 2.2(b)). The helical primitive function $\tilde{f}$ we look for is then defined from the simpler skeleton primitive f by :

$$\tilde{f}(P) = f(W(P))$$

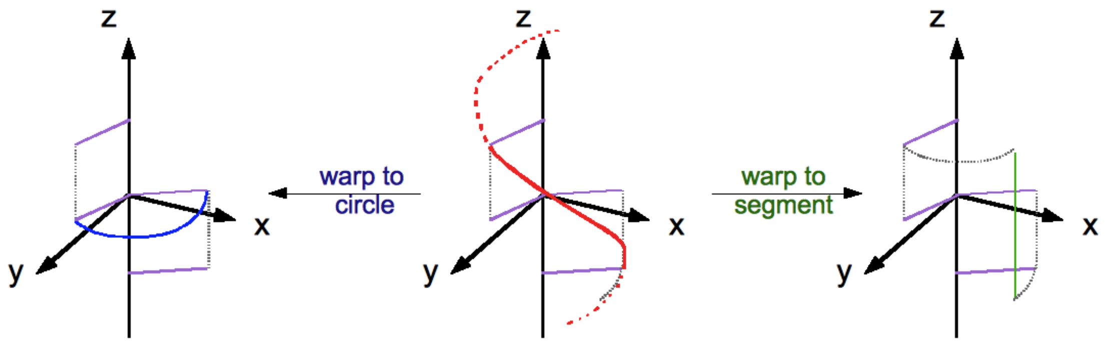


FIGURE 2.2 – *The z-$axis$ helix in red can be warped either to the virtual segment in green (right) or to the virtual circle in blue (left).*

2.1 FROM CIRCLE TO HELICAL PRIMITIVES

In this section we first introduce closed form convolution primitives for arcs of circle with a polynomial weight enabling to define varying radius, a case never solved so far. Our approach applies uniformly to both Cauchy and power inverse kernels of even order ; we include as an example the closed form expression for a linear weight in the case of the classical (i.e. order 4) Cauchy kernel. We then present a warp that maps the helix onto a circle. This provides a first method for computing an analytical helical primitive.

2.1.1 Closed-form primitives for arcs of circles

When it comes to integration, rational functions are the dependable class [Bro05]. The main ingredient in obtaining closed-form convolution primitives for arcs of circle is to introduce an appropriate rational parameterization.

Let $\overset{\frown}{AB}$ define the arc of circle to be used as skeleton. We assume that the points O, A and B (see Fig. 2.3) are not aligned, and such that $|OA| = |OB| = r$. They define a plane in space and two arcs of the same circle, one of angle α the other of angle $\pi + \alpha$, where $0 < \alpha < \pi$ is given by $\alpha = \arccos\left(\frac{\overrightarrow{OA}\cdot\overrightarrow{OB}}{r^2}\right)$. According to which of the two arcs we deal with $T = \tan\left(\frac{\alpha}{4}\right)$ or $T = \tan\left(\frac{\pi+\alpha}{4}\right)$.
Momentarily we consider the coordinate system (x, y, z) where the origin is the center of the circle, the x-axis is the bisector of the chosen angle defined by O, A and B and the (x, y) plane is the plane of the circle. A parametrization of the arc of circle is then given by

$$\begin{array}{rccc} \Gamma : & [-T, T] & \longrightarrow & \mathbb{R}^3 \\ & t & \mapsto & \left(r\,\dfrac{2t}{t^2+1}, r\,\dfrac{t^2-1}{t^2+1}, 0\right). \end{array}$$

When $t \to \pm\infty$, $\Gamma(t)$ tends to the point diametrically opposite to the mid-point of the arc. The other points of the circle are obtained for a parameter $t \in \mathbb{R}$. In terms of this parameter t the infinitesimal arc length is $|\Gamma'(t)| = \frac{2\,r}{t^2+1}$.

Consider a point $P(x, y, z)$ in space. We have

$$|P\Gamma(t)|^2 = \frac{\alpha t^2 - 2\,\beta\,t + \gamma}{t^2+1}$$

where $\alpha = x^2 + (y-r)^2 + z^2$, $\beta = 2\,r\,x$, $\gamma = x^2 + (y+r)^2 + z^2$. Note that

$$\begin{aligned} \gamma + \alpha &= 2\,(|OP|^2 + r^2), && (2.2)\\ \alpha\,T^2 + 2\,\beta\,T + \gamma &= (T^2+1)\,|AP|^2, && (2.3)\\ \gamma\,T^2 - 2\,\beta\,T + \gamma &= (T^2+1)\,|BP|^2, && (2.4) \end{aligned}$$

so that (α, β, γ) is actually the solution of a linear system that depends on T and the squares of the distances of P to O, A and B. There is a unique solution provided that A, O and B are

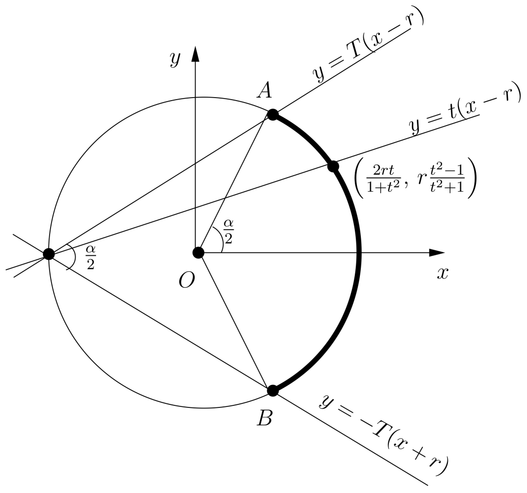


FIGURE 2.3 – *Parameterization of the arc of circle*

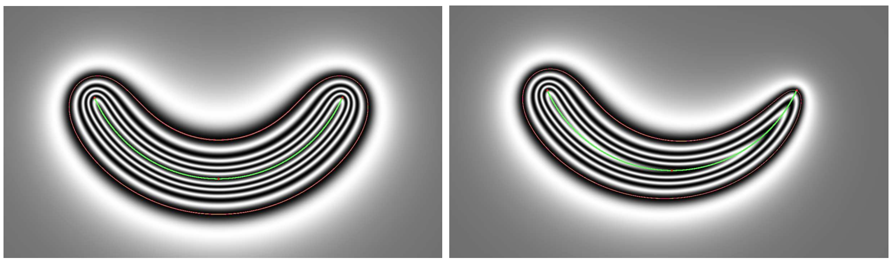


FIGURE 2.4 – *The level sets of the convolution primitive for an arc of circle for constant (right) and linear (left) weights.*

not aligned, i.e. $T(T^2-1) \neq 0$. An explicit expression for (α, β, γ) is obtained by Cramer's rules.

Hence the weighted convolution primitive (eq. (1.14), which is the method usually used to obtain varying radius) for the arc of circle with weight $\tau : [-T, T] \rightarrow \mathbb{R}$ is given by :

$$f_{\mathcal{S}k(OAB)}(P) = 2\,r \int_{-T}^{T} \tau(t)\, \frac{(t^2+1)^{\frac{i}{2}-1}}{(a\,t^2 - 2\,b\,t + c)^{\frac{i}{2}}} dt \tag{2.5}$$

where $a = 1 + s\,\alpha$, $b = s\,\beta$, $c = 1 + s\,\gamma$ for Cauchy kernels and $a = \alpha$, $b = \beta$, $c = \gamma$ for power inverse kernels (of order i). Closed form expressions for the following integrals are given in [HC12] :

$$I_{k,i} = \int \frac{t^k}{(a\,t^2 - 2\,b\,t + c)^{\frac{i}{2}}} dt$$

which appear in line-segment primitives. When using even order Cauchy or power inverse kernels those expressions also provide closed form primitives for arcs of circle with polynomial weight function τ. It is just a matter of expanding the numerator. For odd order kernels, the closed form primitives are expressed thanks to elliptic functions.

Effect of rational parametrization on weight control

If we look at equation 2.5, we can note that the weight τ is defined as a function of the same parameter as the rational parametrization Γ which is not the arc-length parametrization. Since the function τ defines the volume around the shape, it is important to see the difference between the weight defined on the skeleton if we use a linear function of t and a linear function of the arc-length s. We have :

$$s(t) = \int_0^t \frac{2r}{u^2+1} du$$
$$s(t) = 2r\ atan(t)$$

which is defined on $[-T, T]$ and take value in $[-\frac{r\alpha}{2}, \frac{r\alpha}{2}]$. So, in order to have a weight function $\tilde{\tau}$ that depend on the arc-length s, we should have used :

$$\tau(t) = \tilde{\tau}(s(t))$$

which is equivalent to :

$$\tau(t) = \tilde{\tau}(2r\ atan(t)).$$

For instance, if we want to use a weight function that varies linearly on the skeleton, that would make appear a $atan$ in the function to integrate which would not be cumbersome. In order to simplify integration, we can approximate $s(t)$ by :

$$\hat{s}(t) = \frac{r\alpha}{2T}t$$

We can easily check that the error of approximation is zero in $-T$, 0 and T. The maximal error is obtained for $t = \pm\sqrt{1 - \frac{\tan(\frac{\alpha}{4})}{\frac{\alpha}{4}}}$. The error relative to the size of parameter's interval is 0.03 for $\alpha = \frac{\pi}{2}$, 0.09 for π and 1.0 for 2π. Because the error is not too important for an arc of circle of opening lower than π (on should take into account that the error is partly smoothed out due to integration), it is possible to use this approximation to compute the convolution, thus we obtain

$$\tau(t) \simeq \tilde{\tau}(\frac{r\alpha}{2T}t)$$

and so if we want to set a polynomial function on the skeleton we can easily approximate it by a polynomial function of parameter t of same degree.

Consider the particular case of an arc of circle with a linear weight defined by the two endpoints weights τ_A and τ_B, that is : $\tau(t) = ut + v$ where $u = \frac{\tau_A - \tau_B}{2T}$, $v = \frac{\tau_B + \tau_A}{2}$ so $\mathcal{C}_{AOB}(P)$. For the classical (order 4) Cauchy kernel the primitive (with weight approximation) is given by

$$2r\left[u(I_{4,3}(a,b,c) + I_{4,1}(a,b,c) + v(I_{4,2}(a,b,c) + I_{4,0}(a,b,c))\right]_{-T}^{T}.$$

The explicit expressions for $I_{4,0}, \dots, I_{4,3}$ are given in [HC12]. The left hand sides of Equations 2.2-2.4 arise as subexpressions and it is worth using those latter for a faster evaluation. Figure 2.4 illustrates the level sets of this primitive in the case where $\tau_A = \tau_B$ and in the case where $\tau_A \neq \tau_B$.

2.1.2 Warp based extension to helical primitives

A first way to generate an analytical, convolution-like field along a helical skeleton is to use a warping method, based on a piece-wise map of the helix onto arcs of a circle. More precisely, we use a translation along the z-$axis$, where the amount of translation depends on

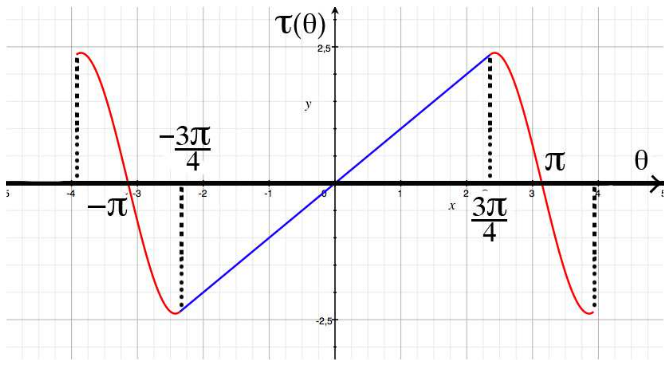


FIGURE 2.5 – *Amount of translation along the z-$axis$ according to the angle θ in cylindrical coordinates*

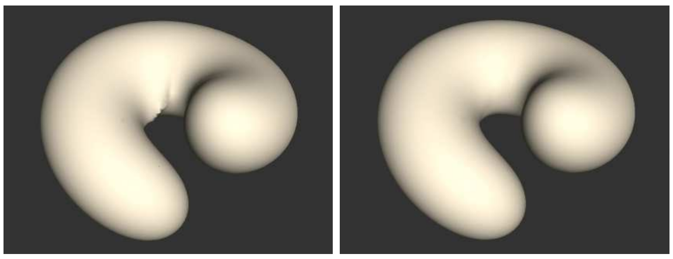


FIGURE 2.6 – *Left : for a thicker volume around the helix, a pinched blend occurs in the vicinity of the axis if we use the warp of Section 2.1.2. Right : the lift of Section 2.1.3 erases this unwanted singularity.*

the θ coordinate of the query point P :

$$W(r,\theta,z) = \left\{ \begin{array}{c} r \\ \theta \\ z - \tau(\theta) \end{array} \right\} \text{ with } \tau(\theta) = \epsilon\, S\, \theta$$

For τ to be continuous, we use more than two arcs of circle for mapping a turn of the helix (using a different warp per arc of circle) : this way, the warp is injective and O, A, B are not aligned (see Section 2.1.1). More precisely, we define a warp of an arc of helix using $\tau(\theta) = \epsilon.S\,\theta$ on $[-\frac{3\pi}{4}, \frac{3\pi}{4}]$ which includes all the points of the helix arc. We use a cubic spline junction to have a continuous warp defined everywhere in space (see Fig. 2.5). Results are depicted on Figure 2.7.

2.1.3 Handling of thick helical primitives

The warp introduced in previous section is not well defined along the helix axis and bears no continuous extension. As we go across the axis, from θ to $\theta + \pi$, τ takes a non zero step unless $\theta = 0, \pm\pi$. As a result the scalar field for the primitive is singular along the axis. Plotting its level sets in the vicinity of the axis is thus error prone. The situation is remedied by increasing the dimension of the problem : a fourth coordinate t is introduced. This latter is made to tend to ∞ when r tends to 0 and equals 0 when $r \geq R$. The initial circle primitive used for the definition of a convolution surface depends on three parameters, which are the distances between a point in space and the points O, A and B. By introducing the parameter t, the circle used for the convolution is actually embedded in a 4 dimensional space. As shown in Figure 2.6, this allows us to remove the pinched area in the vicinity of the axis of the helix.

The improved warp allows us to generate implicit primitives for highly twisted helices. This is illustrated in Figure 2.7. For the top row we used a constant weight function on the circle. The helical primitive has then a constant radius as is expected. For the bottom row we used a linear weight function. That reflects in a varying radius along the helix that we can wish for.

The deformation introduced by the above circle based warp on the normal cross-section to the helix becomes noticeable as $\frac{S}{R}$ increases, i.e. when the helix becomes closer to a line-segment. Typically, when $\frac{S}{R}$ is greater than 0.6 the ratio between the incircle and excircle of the section becomes smaller than 0.8. We thus offer next a helical primitive using a warp onto a line-segment. Section 4 shows how to combine them to adequately span the full range of circular helices.

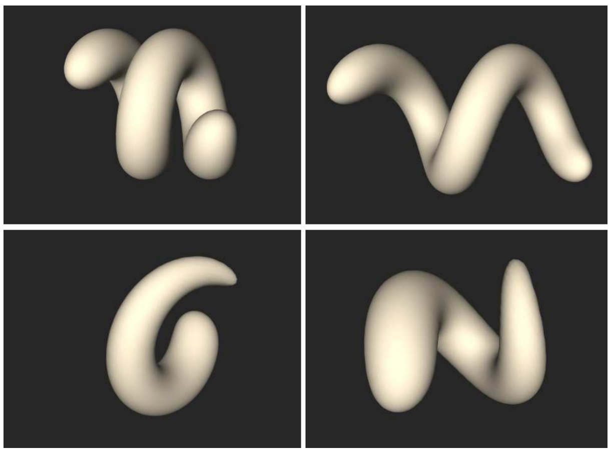


FIGURE 2.7 – *Helical primitives based on warps to the circle. Top : constant weight. Bottom : varying weight.*

2.2 HELICAL PRIMITIVES BY WARPING TO A LINE SEGMENT

In this section we shall introduce warps of a helix to a line segment in order to provide better quality when the twist $\frac{R}{S}$ increases. Our starting point is the inverse of Barr's twist, which was already used in [WO97] to create implicit primitives. We exhibit the imperfections of this warp for our purpose and offer new transformations to compose it with, in order to obtain the desired results. Those new warps form an essential part of our contribution in this chapter.

As a convention for the rest of this section we use the line segment parallel to the axis of the helix and shown in green in Figure 2.2 (right). In contrast to the method of previous section, any number of turns in the helix can be handled in a single warp.

2.2.1 Naive twist-based warps

Twisting back a helix of parameter S to a line-segment can be done by setting v to $-\frac{\epsilon}{S}$ in the equation of a twist :

$$twist(v,(x,y,z)) = \left\{ \begin{array}{c} x\cos(v.z) - y\sin(v.z) \\ x\sin(v.z) + y\cos(v.z) \\ z \end{array} \right\}$$

We therefore define our basic warp as :

$$W_{twist}(x,y,z) = twist(-\frac{\epsilon}{S},(x,y,z))$$

As shown in the top row of Figure 2.11, the normal sections of the resulting primitives are not circular. To get some insight we consider a helix with $R = 1$, $\epsilon = 1$ and examine the linear part, W_{simple}, of the map W_{twist} around $(1,0,0)$:

$$W_{simple}(x,y,z) = \left\{ \begin{array}{c} x \\ y - \frac{z}{S} \\ z \end{array} \right\}.$$

The preimage of the vector $(0,0,1)$ is the tangent vector to the helix at the point $(1,0,0)$, that is $(0,1/S,1)$. Figure 2.8(a) depicts a circular cylinder, the axis of which is given by this tangent

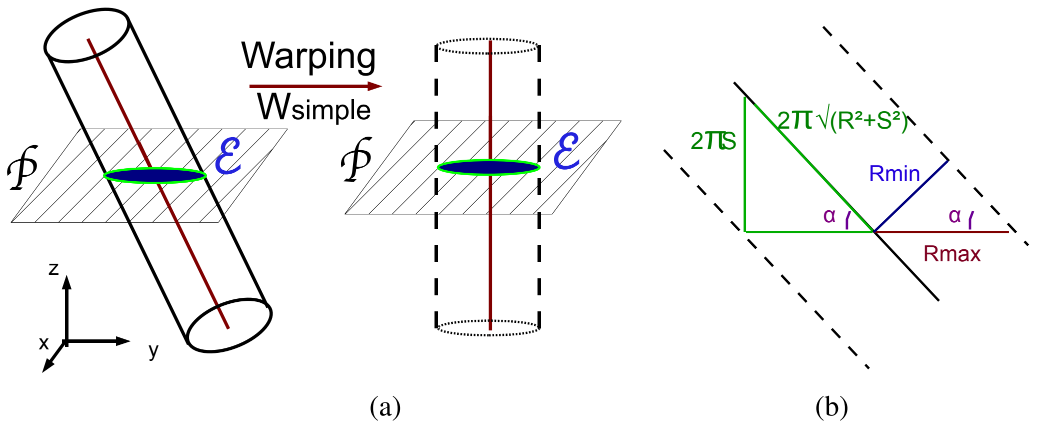


FIGURE 2.8 – *(a) Transformation W_{simple} applied to an inclined cylinder, (b) Lateral view of that cylinder*

vector, and its image under W_{simple}. This latter is a cylinder with an elliptic section. Indeed the intersection of the left hand side cylinder with a plane $z = c$ is an ellipse $\mathcal{E}$ and the restriction of W_{simple} to that plane is a translation within this plane. Therefore the section of the image cylinder is the same ellipse $\mathcal{E}$. As explained on Figure 2.8(b) the ratio of the major and minor radius of this ellipse is :

$$\frac{r_{min}}{r_{max}} = \sin(\alpha) = \frac{S}{\sqrt{R^2 + S^2}} \tag{2.6}$$

Scaling the warp along the y-$axis$ will bring the horizontal section closer to a circle. The resulting transformation is :

$$\tilde{W}(x, y, z) = \left\{ \begin{array}{c} x\cos(\frac{\epsilon}{S}\,z) - y\sin(\frac{\epsilon}{S}\,z) \\ \frac{S}{\sqrt{R^2+S^2}}\,(x\sin(\frac{\epsilon}{S}\,z) + y\cos(\frac{\epsilon}{S}\,z)) \\ z \end{array} \right\}$$

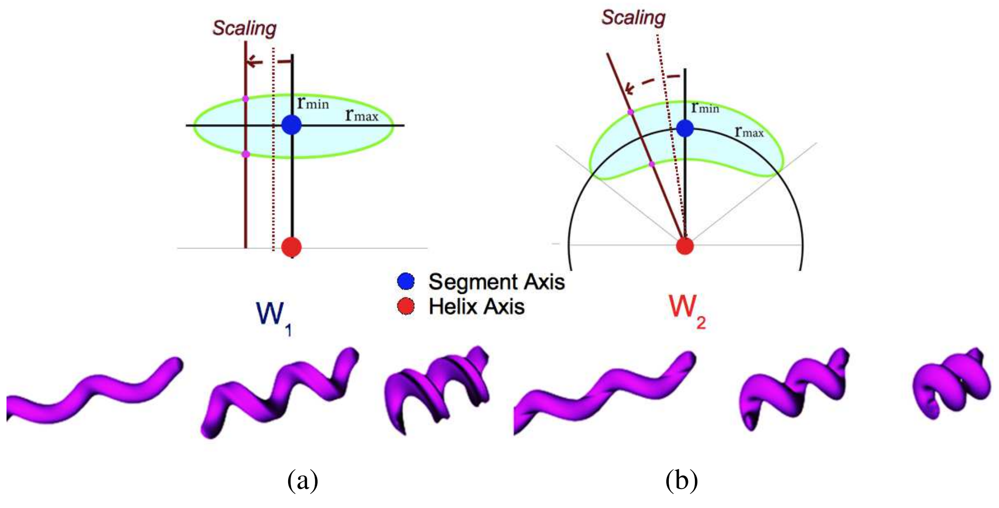


FIGURE 2.9 – *The principle behind the two naive warps and their results.*

However, this transformation still does not work for highly twisted helices (see figure 2.9(a)) : our additional scaling component neglected the effect of the rotation (if we take again the analogy with an oblique cylinder, the latter should rotate along the helix). It is more appropriate to apply the scaling correction to the angle in the cylindrical coordinates : $\bar{W}(r,\theta,z) = \left\{r, \frac{S}{\sqrt{R^2+S^2}}(\theta - \frac{\epsilon}{S}z), z\right\}$. This is equivalent to applying a rotation around the helix axis after the twisting (see figure 2.9(b)). The angle of rotation depends on the polar coordinate after twisting. However the rotation to be considered for a helix is a rotation around its center of curvature.

2.2.2 Correctly handling the central part

The warps we described so far gave us some insight on the corrections required to get a reasonably circular cross-section : a scaling should be applied after a rotation around the local center of curvature of the helix. The center of curvature at a parameter t along the helix is given by :

$$\left(-\frac{S^2}{R}\cos(t),\ -\frac{S^2}{R}\sin(t),\ S\,t\right).$$

When the helix tends to a line segment the centers of curvature wander off to infinity whereas, when the helix tends to a circle, the centers of curvature are attracted toward the axis of the helix. The correction we define thus varies smoothly from the value of $\tilde{W}$ to that of $\bar{W}$ according to the amount of twist of the helix.

Our new transformation can be written as :

$$W(x,y,z) = g(twist(-\frac{\epsilon}{S},(x,y,z)))$$

where

$$g(x,y,z) = \left\{ \begin{array}{c} (x+\frac{S^2}{R})\cos(\mu\,\psi(x,y,z)) - y\sin(\mu\,\psi(x,y,z)) - \frac{S^2}{R} \\ (x+\frac{S^2}{R})\sin(\mu\,\psi(x,y,z)) + y\cos(\mu\,\psi(x,y,z)) \\ \frac{\sqrt{R^2+S^2}}{S}\,z \end{array} \right\}, \tag{2.7}$$

$$\mu = \frac{S}{\sqrt{R^2+S^2}} - 1,$$

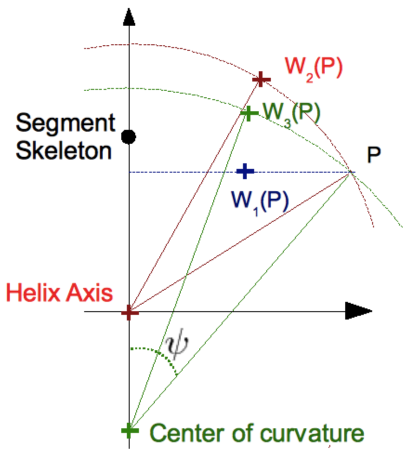


FIGURE 2.10 – *Difference between the naive warps and our improved solution.*

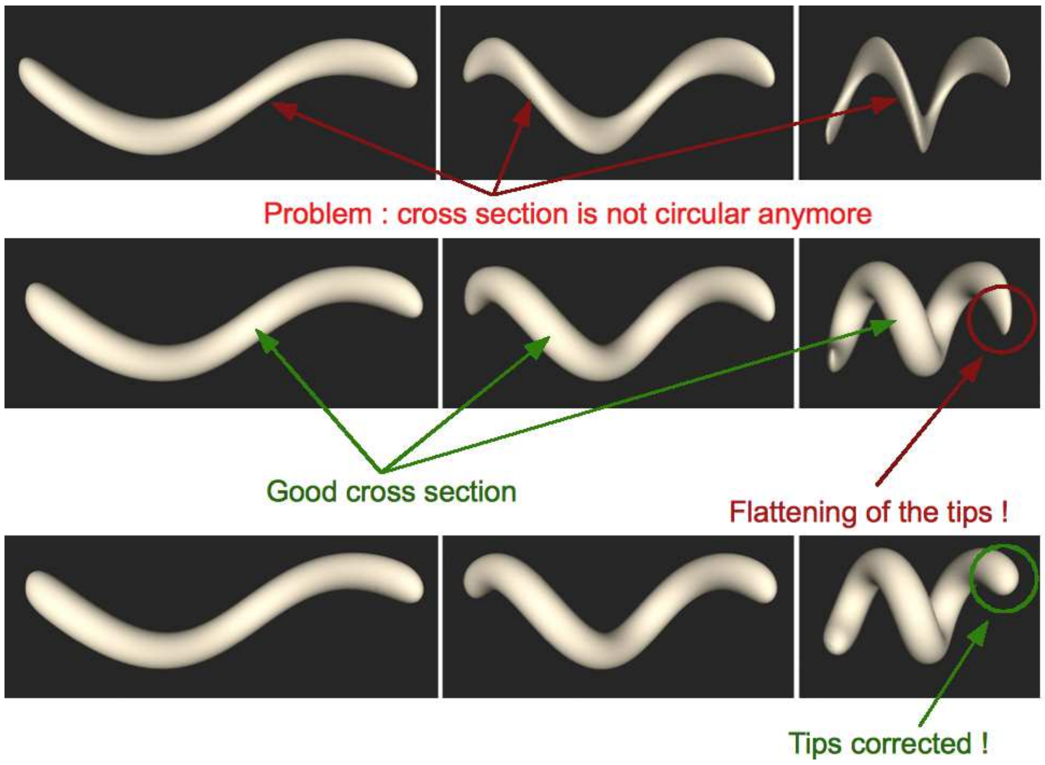


FIGURE 2.11 – *Comparison, for different values of $\frac{S}{R}$, between the classic twist (first row), the solution with the end point problem (second row) and our final solution (third row).*

and

$$\psi(x, y, z) = \text{atan2}(y, x + \frac{S^2}{R}).$$

The restriction of g to a plane orthogonal at some point to the helix axis is a rotation around the center of curvature of the helix at this point (see Figure 2.10). This is equivalent to multiplying the polar coordinates, with respect to the center of curvature, of the point by the ratio $\frac{S}{\sqrt{R^2+S^2}}$ (Equation 2.6). Another important feature is to ensure that the helical primitive has the same thickness as the associated line segment primitive. The length of the skeleton needs to be preserved to ensure this. As the last coordinate of g in Equation 2.7 discloses it, this is obtained by scaling along the $z\text{-}axis$.

The primitives obtained with this new warp are illustrated in the second row of Figure 2.11.

2.2.3 Final solution for segment-based primitives

Figure 2.11 provides a comparison between the results obtained with the original twist and those obtained with the presented new warp. A closer look at the leftmost image in the second row reveals an unwanted artefact toward the end points of the shape. This artefact furthermore prevents the seamless blends that are sought when the helix is part of a more complex skeleton.

The artefacts observed around the tips of the helical primitive can be explained through the linearized warp W_{simple} already studied in Section 2.2.1. Figure 2.12 illustrates the cause of the problem at the end points and how to measure the unwanted deformation in order to introduce an appropriate correction. A translation along the $z\text{-}axis$, the norm of which depends linearly on y in the deformed space, is enough to correct the tips in this linearized version. As before,

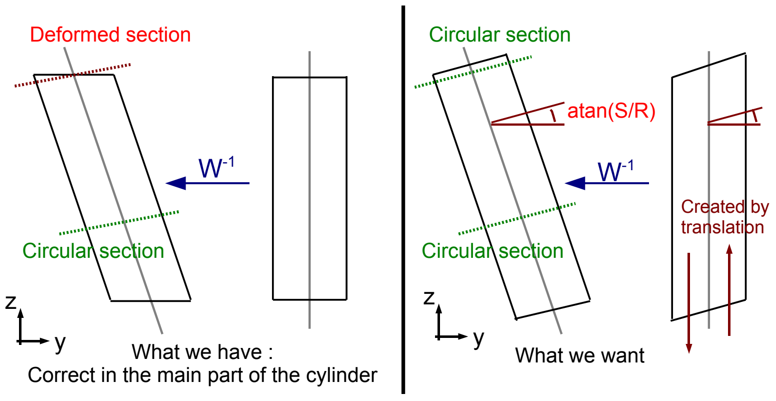


FIGURE 2.12 – *Idea justifying the correction of both ends of the primitive.*

this correction is adapted to our main transformation through a translation that depends on the polar coordinate with respect to the center of curvature. This is given explicitly by :

$$W_{correct}(x,y,z) = \tau(W(x,y,z))$$

where

$$\tau(x,y,z) = \left\{ \begin{array}{c} x \\ y \\ z - \epsilon\,\frac{R^2+S^2}{S}\,\psi(x,y,z) \end{array} \right\}$$

The map τ introduces a discontinuity of the field function when $\psi = \pm\pi$. The latter can be easily overcome through a linear or spline junction as was done for warps onto arcs of circles (Figure 2.5).

The improvement introduced by this last adjustment can be easily appreciated by comparing the second and third row in Figure 2.11.

To obtain a helical primitive with varying radius, we can simply assign a polynomial weight function on the line-segment skeleton the helix is warped onto and use the results of [HC12]. Results are shown in Figure 2.13.

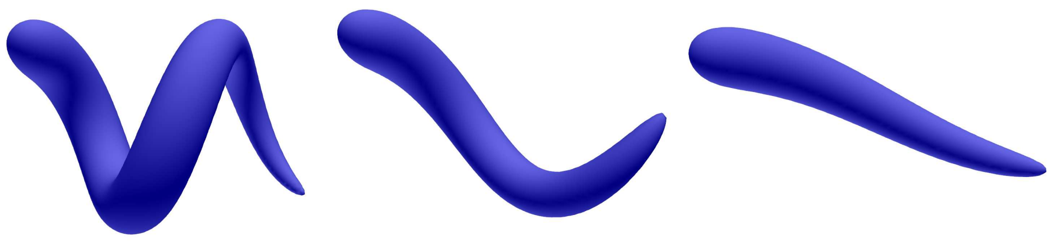


FIGURE 2.13 – *Helical primitives with varying radius created by warping a single segment.*

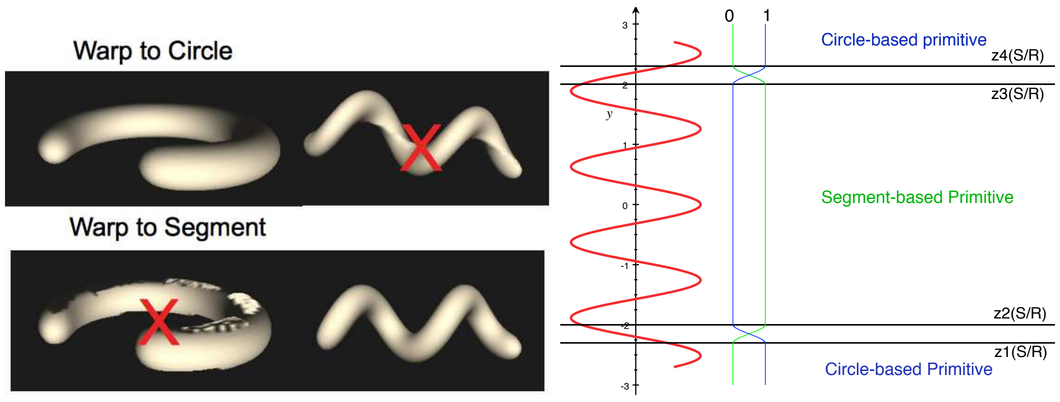


FIGURE 2.14 – *Transition between the two kind of primitives in function of* $\frac{S}{R}$

2.3 Evaluation and combination of our solutions

We presented two ways of obtaining a helical primitive by warping the helix either on arcs of a circle or onto a line segment. We lead an experimental evaluation of the results obtained through those two warps. On one hand, the warp to circle solution presented in Section 2 leads to good results when $\frac{S}{R} \in [0, \approx 0.5]$ (see Figure 2.1 to have an idea of the shape of the helix). Beyond that the shape of the cross-section is noticeably deformed. We may think of improving the situation by introducing corrections as was done for line segments in Section 3. On the other hand, the solution based on warp to line segment developed in Section 3 gives good results for all kinds of helices, except when the helix becomes highly twisted : for a helix close to a circle, our solution still locally flattens the primitive around the end points of the helix. In our experiments, this is negligible until $\frac{S}{R}$ becomes lower than ≈ 0.35 (instead of ≈ 0.7 without the final correction for the tips). Thus, the whole range of helices can be handled nicely by at least one of the two methods we proposed, with good results.

Combination of the two methods

Since the quality of the result depends on the same ratio $\frac{S}{R}$, automatically choosing the method to apply is very easy.

A first solution is to use a segment based primitive, which is computationally cheaper, when $\frac{S}{R} > 0.4$, a circle based primitive when $\frac{S}{R} < 0.3$ and an intermediate solution in-between. This intermediate solution is constructed by interpolation, for instance a cubic interpolation. In this first approach, the evaluation of the helical primitive when $0.3 < \frac{S}{R} < 0.4$ requires both the evaluation of the circle based primitive and the line segment based primitive. A more efficient

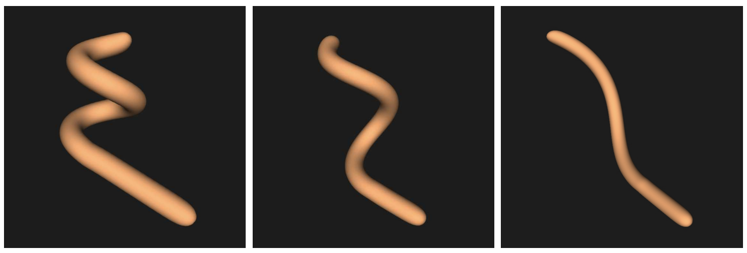


FIGURE 2.15 – *Helical primitive prolonged by a segment primitive for different values of* $\frac{S}{R}$

	segment primitive	helical from circle	helical from segment
Time for 200^3 field queries	$1.45s$	$2.63s$	$3.27s$

TABLE 2.1 – *Computation time for a single primitive. CPU : 2.4 GHz Intel Core 2 Duo (only one core used).*

way to handle the transition zone of twisting, i.e. when $0.3 < \frac{S}{R} < 0.4$, is to work with a warp of a line segment for the middle part of the helix and only two warps onto arcs of circles for the tips of the shape. Then interpolation between the two different primitives occurs only within a limited range of the z coordinate (see figure 2.14).

In figure 2.15, the helix is prolonged by a line segment with continuous tangency. The resulting shape remains smooth around the transition between the helical primitive and the line segment primitive. Figure 2.17 emphasizes the quality of our global helical primitive in the context of a more complex skeleton (each tentacles are created from two helices).

Computational efficiency

Table 2.1 compares the timings for the evaluations of our two analytical helical primitives and of a line-segment primitive. This shows that when 3 segments or more are needed to approximate a helical skeleton, our new helical primitive is much more efficient. The fact that the warp-to-circle solution used for highly twisted helices requires two primitives per turn does not change this result, since more than 6 segments per turn would be needed then. To make this comparison more visual, Figure 2.16 depicts the quality we would get, at constant computational cost, by tessellating a skeleton into line-segments or into helices. For this level of zoom, getting the same visual quality with a tessellation into line segments would respectively multiply computational time by factors 2.71 for the octopus, which includes non helical parts, and 5.84 for the antlers, which is only made of helices.

Manipulation of primitives

Having base primitives that are efficient both in term of shape representation and computational efficiency is not sufficient. The shape should also be easy to manipulate in order to make the primitive useable. In our modeling framework, helices can be manipulated through their end-points : when one end point is moved, we compute a deformation of the helix that

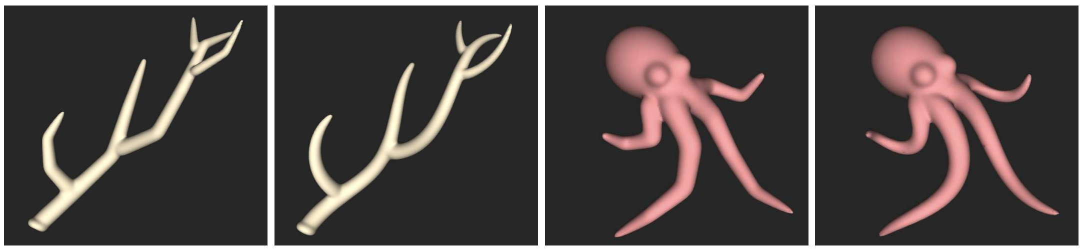

(a) 14 segments (b) 5 helix (c) 13 segments (d) 4 helices

FIGURE 2.16 – *Comparison of model quality using a tessellation of 3D skeleton curves into line-segments (left) compared to our helical primitives (right) when a given computational time is allowed.*

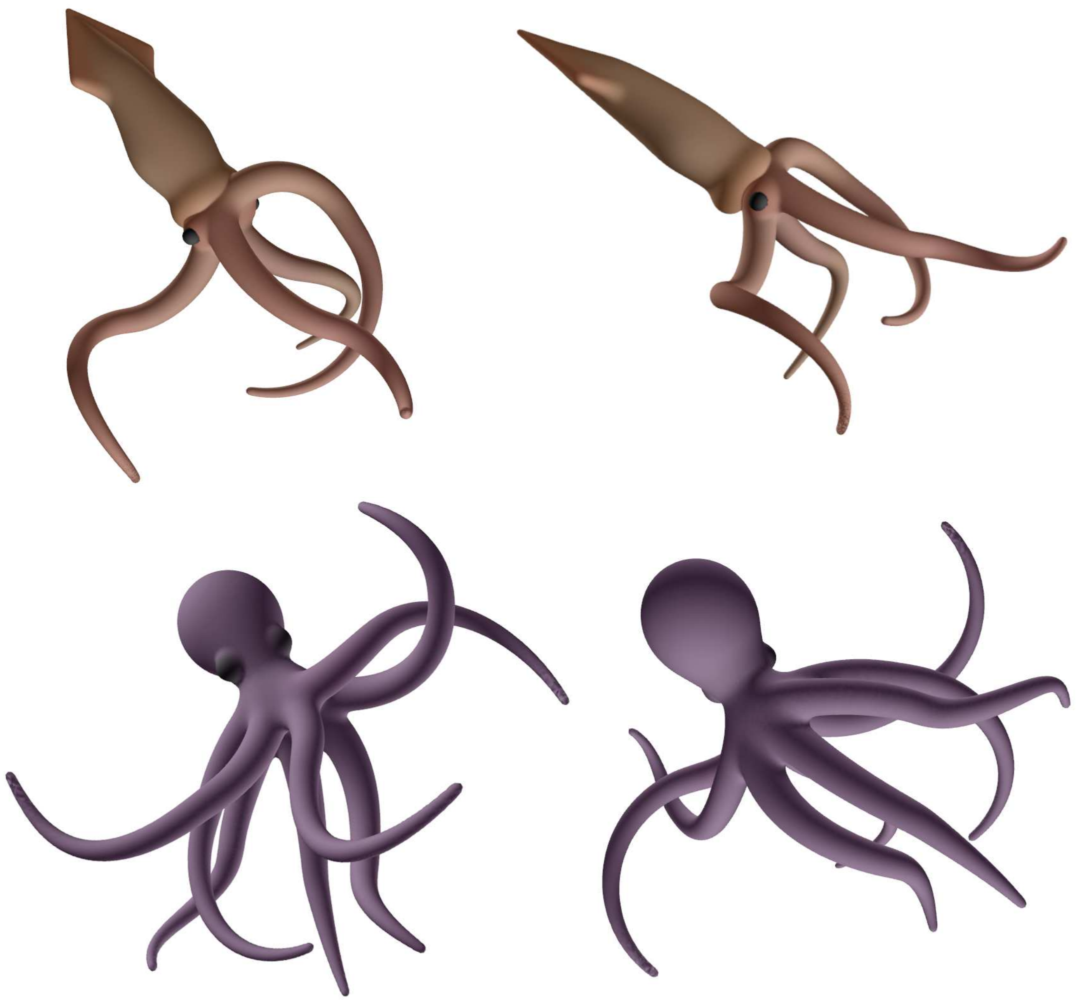


FIGURE 2.17 – *Two views of the same squid and octopus, all tentacles are created from 2 helices. Note that even without C^1 transition between helix skeletons the resulting surface look smooth.*

has the property to pass through the two end point while minimizing the deformation of the helix shape. In order to do so we decompose this transformation in two base one : the first one is a rotation (the shape of the helix is unchanged) that aligns the two end points of the helix and the aim. The second transformation is a global scaling of the helix which will ensure the new end-point constraint (since it is a global scaling, it does not deform the helix shape, $\frac{S}{R}$ is constant, it only change its scale). The helix can be modified by keeping its end point at a given position. The first modification left the helix shape and scale unchanged, it is a rotation which axis is the vector between the two end points. The two other deformations are the change of radius and the change of stretch. Manipulating helices this way is quite easy, however there are plenty of ways to improve how we create and manipulate our skeletons. For creation we could use [CCM13] that compute a 3D curves from a sketch curve, the 3D curve being a helix by part. An alternative solution could be to use [DJBDT13] that create a G^1 helix by part from a spline. While for manipulation, physical simulation [BAC*06] would be an interesting alternative.

2.4 Conclusion and Future work

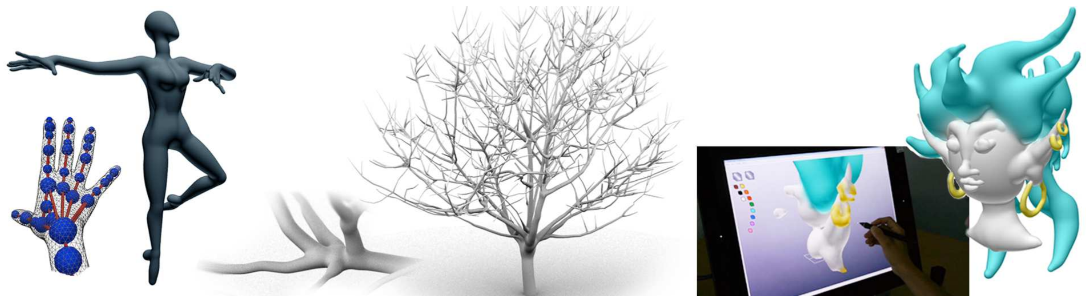
We have introduced the first analytical formulation for helical implicit primitives. The method handles primitives of varying radius and can be used with arbitrary convolution kernels such as power inverse and Cauchy kernels of arbitrary order. Indeed, a side, yet important, contribution is a new general formulation for arc of circle convolution primitives, which allows for varying radius.

Our solution for helices can be considered as a *pseudo-convolution* since it built on arcs of circle and line-segment convolution primitives by warping. Similarly to convolution, the resulting helical primitives can be used in complex configurations where the skeleton is tessellated into several helices : as with direct closed-form convolution, the resulting curve is bulge-free.

We furthermore have set up the bases of a methodology to design advanced space warps that progressively build complex primitives from simpler ones. That opens the door to further extensions : the same methodology could be applied to get implicit primitives along $3D$ spiral skeletons, which were shown to be very useful to represent a larger variety of natural shapes such as seashells and horns [HT11]. Modifying our deformation to the sweeping of more general cross-sections around the helix skeleton would also be a useful extension of this work, made possible by our section preserving warp.

CHAPTER

3 SCALE-INVARIANT INTEGRAL SURFACES



Teaser Figure : *SCAle-invariant Integral Surfaces (SCALIS) makes interactive, skeleton-based modeling intuitive (a), provides a direct reconstruction of complex shapes from their medial axes (b) and is very convenient for sketch-based modeling (c).*

Publication :

ZANNI C., BERNHARDT A., QUIBLIER M., CANI M.-P. : SCALe-invariant Integral Surfaces. *Computer Graphics Forum* (to appear).
ZANNI C., BERNHARDT A., QUIBLIER M., CANI M.-P. : Surfaces Implicites Homothétiques. *Revue Electronique Francophone d'Informatique Graphique - REFIG* (to appear).

GEOMETRIC skeletons, namely lower dimensional structures centered inside a shape, have been instrumental in shape analysis, shape matching or as a tool for shape deformation for many years [Lon98, BL99]. A typical example is the medial axis of a solid shape [SPB96], a non-manifold graph of curve and surface components, defined as the locus of the centers of maximal spheres included into the shape. Other examples are Reeb graphs used for extracting topological information, or centered graphs of curves used for automatically fitting animation skeletons [BGSF08, AHLD07]. With RigMesh, [BJD*12] introduced a sketch-based modeling method that unify modeling and rigging stages of the character animation pipeline, thus enabling the easy re-posing and animation of the created shape.

Easing the reverse operation, i.e. the generation of smooth 3D shapes from skeletons, would bring many benefits : firstly, it would enable to store solid models in skeletal form, a highly compact representation compared with B-reps due to their lower topological dimension ; secondly, using skeletons as modeling primitives would ease conceptual shape design as well as subsequent deformation and animation, due to the high-level vision they provide on geometry and topology ; Lastly, being able to generate shapes from skeletons would ease the reconstruction of smooth solids from raw data such as point-sets or voxels, for which skeleton extraction techniques are already available.

Theoretically, skeletal structures are reversible shape representations when associated with radius information. However, generating smooth solid models from skeletal information is intricate. Convolution surfaces, namely iso-surfaces of scalar fields generated by convolving a skeleton with a smoothing kernel, are probably the most attractive model to do so : individual primitives generated by each skeleton component seamlessly blends when their field contributions are summed, which makes modeling by parts straightforward. This model was successfully used in both animation and sketch-based modeling applications [Blo02b, BPCB08]. Unfortunately, convolution surfaces also bring severe drawbacks, which restricted their use : firstly, although the smoothing kernel can be weighted along the skeleton, there is no mechanism for prescribing desired radii ; secondly, modeling large shapes with fine details is difficult, as small size components are typically smoothed out when blended into larger ones ; even more surprisingly, convolution surfaces are not scale invariant : depending of the scale of the work space, blending produces quite different effects for the same input shapes. These drawbacks make shape control very difficult and may be the reason why skeletons never spread as a modeling primitive.

We therefore introduce *SCAle-invariant Integral Surfaces* (SCALIS), a new implicit model aimed at making skeleton-based modeling usable. Based on a combination of convolution and space warps, SCALIS retains the advantages of convolution surfaces, namely making the resulting shape independent from the degree of subdivision of the skeleton and producing seamless blends (using a simple sum of fields) when new skeletal branches are added. However, in contrast with convolution surfaces, SCALIS avoids the three problems we listed : the radii of the generated shapes can be explicitly tuned ; the level of smoothing, still controlled through the choice of a smoothing kernel, is self-similar at different scales and brings scale-invariant blending effects ; lastly, small shape components are not smoothed out anymore when blended with larger shapes, which makes detailed modeling possible.

3.1 Drawbacks of weighted convolution

As we have seen, Convolution surfaces are attractive because they create smooth shapes around skeletons independently from the subdivision of the latter. Moreover, their extensions to weighted convolution, standard formulation (1.14) and Hornus (1.15), enable to create shapes with varying radii. However, they raised a number of issues :

1. **Lack of radius control :** There is no mechanism for prescribing a desired radius. This comes from the non-linear dependency between the weighting function τ and the radius of the iso-surface of interest (see figure 3.1(c)). In addition to preventing precise shape control, this makes modeling non-intuitive (see figure 3.2(a) and 3.8(a,b)).
2. **Blurring and vanishing details :** Small scale details are smoothed out when blended to a shape of larger size, especially using eq. (1.14) (see figure 3.3(a)). In addition, thin surfaces tend to collapse (figure 3.1(a)). As noted in [JT02a], the *Inverse* kernel is the only one that does not raise this problem due to the infinite field value on the skeleton.
3. **Scale-dependent blending :** The way two convolution surfaces blend when the associated fields are added does not depend only on their shapes, but also on the scale at which they were defined : as τ decreases, blending becomes smoother and smoother (see figure 3.4).

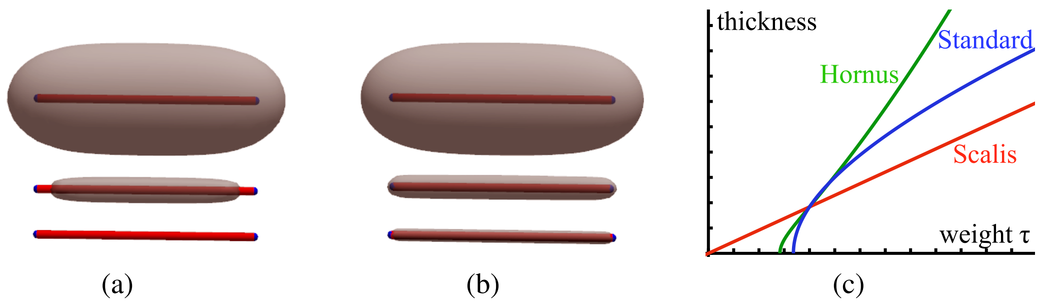


Figure 3.1 – *Primitives of decreasing radius (Cauchy kernel) : The standard formulation (a) causes the surface to collapse at the center of the skeleton and then vanish, contrary to SCALIS (b). (c) plots the thickness of the generated shape in function of τ for an infinite segment skeleton.*

Recent work in implicit modeling has addressed some of these issues. Two sketch-based modeling systems based on convolution had to cope with radius control [ITZkF04, BPCB08]. They either used an optimization process to compute τ at skeleton vertices from the 2D skeletons and from the radii extracted from the sketch, or relied on a sketching space of pre-defined size, combined with pre-set correspondence between weight and radius.

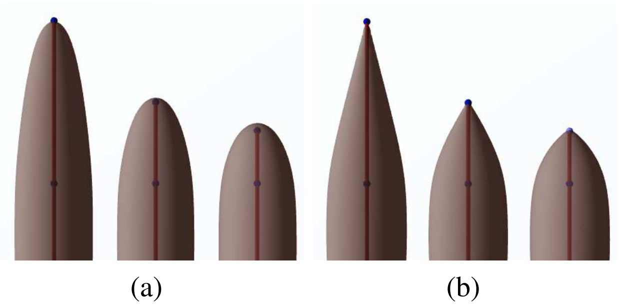


Figure 3.2 – *Shapes generated when the weight linearly falls to 0 : (a) standard formulation, (b) SCALIS. Note the unwanted rounded extremities in (a).*

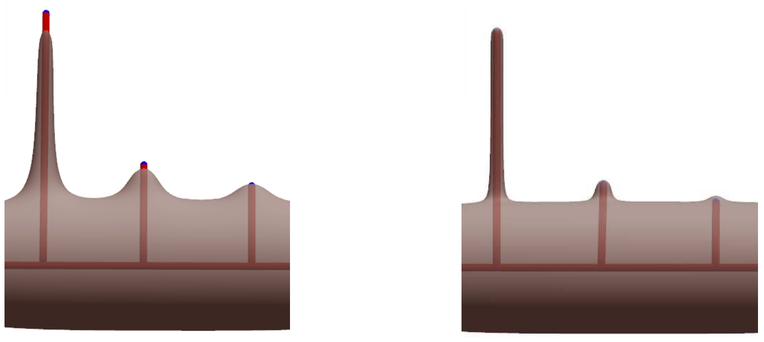


FIGURE 3.3 – *Blending of thin primitives into a thick one. The details blur when the standard formulation is used (a), but not with SCALIS (b).*

The blurring details problem was recently overcome in an automatic way by introducing a complex binary operator, based on an extra influence primitive generated at the intersection of the input surfaces [BBCW10]. However, this operator is costly compared to the sum, does not handle n-ary combinations, and as the previous ones, would not insure shape independence from skeleton subdivision. We therefore develop a new model below, which tries to overcome these limitations.

3.2 SCALIS : SCALE-INVARIANT INTEGRAL SURFACES

Since standard weighted convolutions do not generate a field that is a linear function of τ, they are not scale-invariant (see figure 3.4 (a)) : both the relative radii of individual primitives and the way they blend change when models are scaled. In practice, this prevents the use of convolution primitives in general modeling systems, since shape resizing operations cannot be applied.

In this work, we revise convolution surfaces in order to solve the three major problems we have identified while maintaining their most attractive features, namely n-ary blending using a simple sum and shape independence from skeleton subdivision. Our insight is that making convolution scale invariant is sufficient to solve the three issues listed in section 2.4.

Figure 3.5 depicts the different strategies used by the two previous models to generate radius variations : while equation (1.14) changes the height of the kernel while keeping the width of the support unchanged, equation (1.15) generates a kernel of constant height, but of varying support-size. Let us give some intuition on why the lack of scale-invariance is the main problem. Let us look at skeleton as a density of charge that creates a potential field. A skeleton with a small weight will create less energy than a larger one (but in a domain of the same size). The problem is that the iso-surface corresponds to a given energy level. For this reason small skeletons will have more difficulty to create a surface by themselves. If the energy was scale-invariant, skeletons with small weights would generate the same energy in a given point in space (although less energy in all the space), thus creating the expected surface. In comparison to previous method, our new solution SCALIS, scales both the width of the support and its height while keeping the area below the curve constant (which can be seen has a preservation of energy). This brings scale invariance.

This section develops our solution in the case of a primitives of constant weight τ. The extension of SCALIS to primitives of varying radii is presented in Section 4.

3.2.1 Combining convolution with warping

As seen in the state-of-the-art and in the previous section, space warps can be used to define complex primitives from simpler ones. We rely on this warping principle to define scale-invariant integral surfaces.

To get two identical implicit primitives at different scales, not only in terms of shapes but also in terms of blending behavior, their scalar fields should be scaled versions of each other. To get this invariance property, the idea is to compute field values in a warped space, where all skeletons have a unit weight, i.e. where the field is computed using the initial convolution formulation of equation (1.9). This is done by using a warped primitive, where the warping W is a simple scaling, applied to both the query point and the primitive parameters (skeleton $\mathcal{S}k$ and weight τ_c). See figure 3.6.

In summary, the new scalar field is defined by :

$$f_{\tau_c,\mathcal{S}k}(P) = \tilde{f}_{W_{\tau_c}(\mathcal{S}k)}(W_{\tau_c}(P)) \tag{3.1}$$

where $\tilde{f}_{W_{\tau_c}(\mathcal{S}k)}$ is the application of the basic convolution (eq. (1.9)) along the warped skeleton $W_{\tau_c}(\mathcal{S}k)$. The scaling W_{τ_c} applied to a point is defined by :

$$W_{\tau_c}(P) = O + \frac{1}{\tau_c}\overrightarrow{OP}, \tag{3.2}$$

where O is the origin of the coordinate system. In a consistent manner, the scaling applied to a skeleton (a set of points) is defined by $W_{\tau_c}(\mathcal{S}k) = \{W_{\tau_c}(P)/P \in \mathcal{S}k\}$.

If we look at the formula used to compute $\tilde{f}_{W_{\tau_c}(\mathcal{S}k)}(W_{\tau_c}(P))$, we can see that the distance to the skeleton points is the only way the query point is used. This means that the center of the scaling has no effect on the result. Indeed the same scaling is applied to both the query point

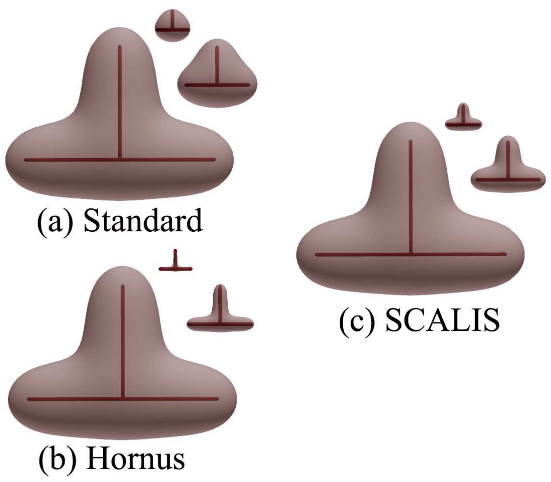


FIGURE 3.4 – *Scaling applied to both convolution skeletons and τ : (a) Standard formulation, (b) Hornus formulation, (c) SCALIS. The first scaling factor is $\frac{1}{3}$ and the second is $\frac{1}{6}$, for a factor of $\frac{1}{10}$ the surface for Hornus formulation would only exist on a small vicinity of the intersection of the two skeletons. Presented results use* Cauchy *kernel.*

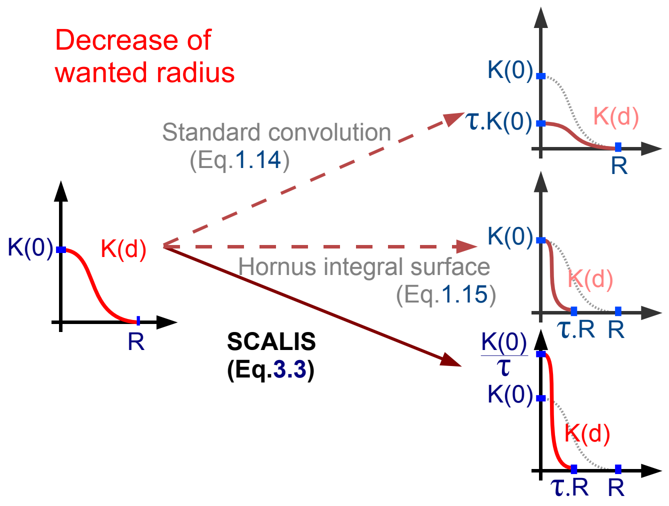

Decrease of wanted radius
Standard convolution (Eq.1.14)
Hornus integral surface (Eq.1.15)
SCALIS (Eq.3.3)
K(0)
K(d)
R
τ.K(0)
τ.R
K(0)/τ

FIGURE 3.5 – *Weighted models : kernel variations when the user decreases τ to get a thinner shape along a segment-skeleton.*

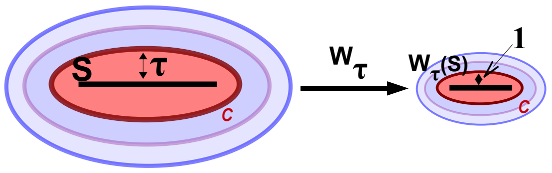


FIGURE 3.6 – *Space warp used to get scale-invariant implicit primitives.*

and to the skeleton and during a scaling all distances are multiplied by the ratio of the scaling (due to Thalès' theorem). Thereafter, since the center of the scaling has no effect, we will use the following abuse of notation : αP and $\alpha \mathcal{S}k$ to represent the scaling of ratio α and center O respectively applied to a point P and to a skeleton $\mathcal{S}k$.

We can easily check that this new field is scale-invariant :

$$f_{\alpha\tau_c,\alpha\mathcal{S}k}(\alpha P) = \tilde{f}_{\frac{\alpha}{\alpha\tau_c}\mathcal{S}k}(\frac{\alpha}{\alpha\tau_c}P) = \tilde{f}_{\frac{1}{\tau_c}\mathcal{S}k}(\frac{1}{\tau_c}P) = f_{\tau_c,\mathcal{S}k}(P)$$

We also note that the good properties of convolution surfaces are maintained : if a segment-skeleton splits into pieces, the two parts, transformed through the warp, are still two halves of a segment, on which a standard convolution is computed. Therefore, SCALIS models are independent of the degree of subdivision of the skeleton, and smoothly blend when fields are summed (the proof is given in appendix B.1). As for convolution surfaces, this enables arbitrary graphs of curve-segments and of surface-patches to be used as skeletons, the later being split into adequate sets of line-segments or of triangles for computing f. However, as for convolution surfaces, modeling general shapes will require SCALIS primitives of non constant radius, which we present next.

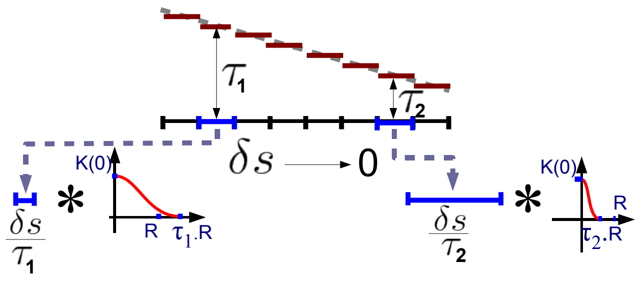


FIGURE 3.7 – *Passing the space warping method to the limit.*

3.3 ENABLING RADIUS VARIATIONS

For sake of simplicity, we first extend SCALIS to non-constant radii in the case of curve-skeletons. Then, the solution is generalized to higher-order skeletons. Finally, we discuss the way to compute closed-form expressions in order to accelerate field queries.

3.3.1 Case of curve-skeletons

Suppose that a varying weight $\tau(s)$ has been defined along a curve-skeleton. The simple idea developed in the previous section, namely using τ as the control parameter of a space-warp (equation (3.1)) to get scale-invariance, cannot be applied anymore. However, we can extend the model by passing it to the limit, i.e. by reasoning about different local warps (each depending on the local weight value) applied to each infinitesimal arc-length of the skeleton.

Let us take a look at the field contribution of an infinitesimal element ds of a curve-skeleton, as depicted in figure 3.7. The local space warp around s would modify distances by a factor $\frac{1}{\tau(s)}$. Thus the influence of the element ds is given by $k(\frac{1}{\tau(s)}\|\overrightarrow{P\Gamma(s)}\|)$. Nevertheless, we should remember that the warp is also applied to the skeleton. Thus, the infinitesimal arc length becomes $\frac{ds}{\tau(s)}$. Replacing the discrete sum over infinitesimal segments by an integral, this yields :

$$f_{\mathcal{S}\mathcal{K}}(P) = \int_{s\in\Omega} k\left(\frac{\|\overrightarrow{P\Gamma(s)}\|}{\tau_{\mathcal{S}\mathcal{K}}(s)}\right)\frac{\mathrm{d}s}{\tau_{\mathcal{S}\mathcal{K}}(s)}. \tag{3.3}$$

If the integral is not parametrized by arc-length, the formula becomes :

$$f_{\mathcal{S}\mathcal{K}}(P) = \int_{t\in\Omega} k\left(\frac{\|\overrightarrow{P\Gamma(t)}\|}{\tau_{\mathcal{S}\mathcal{K}}(t)}\right)|\Gamma'(t)|\,\frac{\mathrm{d}t}{\tau_{\mathcal{S}\mathcal{K}}(t)}. \tag{3.4}$$

Compared with the expression used by Hornus (equation (1.15)), our formulation consists in giving a density to the skeleton, this latter being inversely proportional to the local weight. Moreover, since integrating increases by one the degree of regularity of a function, our new field function is C^{∞} for *Inverse* and *Cauchy* kernels and $C^{\lfloor\frac{i+1}{2}\rfloor}$ for *Compact Polynomial* kernel.

3.3.2 Extension to skeletons of higher dimension

In the case of surface skeleton, the same method can be used. However, the resulting formulation is slightly different, since scaling an infinitesimal surface element $dudv$ requires scaling it in both directions, by dividing it by $\tau(u,v)^2$. Therefore, the expression of SCALIS of non-constant radius on triangle-skeletons uses a factor $\frac{1}{\tau(u,v)^2}$ instead of $\frac{1}{\tau(s)}$ in eq. (3.3) :

$$f_{\mathcal{S}k}(P) = \int_{(u,v)\in\Omega} k\left(\frac{\|\overrightarrow{P\Gamma(u,v)}\|}{\tau_{\mathcal{S}k}(u,v)}\right) |\Gamma'(u,v)| \frac{\mathrm{d}u\mathrm{d}v}{\tau_{\mathcal{S}k}(u,v)^2} \tag{3.5}$$

Similarly, the formulation would be easy to extend to elementary volume skeletons, as the ones used in [PGMG09], but with a division of the elementary volume by the third power of τ.

3.3.3 Non-parametric formulation of Scale-invariant Integral Surfaces

It is possible to introduce a formulation of SCALIS that is closer to the initial formulation of convolution in $\mathbb{R}^3$ and thus that is independent from any parametrization. For this we will re-use the description of the skeleton $\mathcal{S}k$ as a set of point (with additional information) as it is described in the state-of-the-art 1.1.1 :

$$f_{\mathcal{S}k}(P) = \int_{\mathbb{R}^3} \frac{g_{\mathcal{S}k}(Q)}{\tau_{\mathcal{S}k}(Q)^d} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{S}k}(Q)}\right) \mathrm{d}Q \tag{3.6}$$

with $g_{\mathcal{S}k}$ the distribution that depend of the dimension of the skeleton described in 1.1.4.

This new formulation, which is equivalent to the parametrical one, although less useful for computation purpose, is helpful to demonstrate some properties. For instance, we use it to demonstrate that the field function $f_{\mathcal{S}k}$ is additive in relation to the skeleton $\mathcal{S}k$ (which gives the independence to skeleton subdivision : the main property of convolution surfaces). See Appendix B.1.

Skeletons of 0-dimension : the case of points

In the case of 0-dimension skeleton, the distribution representing the skeleton is

$$g_{\mathcal{S}k}(P) = \sum_{Q\in\mathbf{S}} \delta(P-Q).$$

Injecting this equality in equation (3.6) leads to the definition of classic point primitives. Indeed, for 0-dimension skeleton, the normalization of the arc-length is $\frac{1}{\tau^0} = 1$. Thus in the case of 0 dimension skeleton, SCALIS formulation and Hornus formulation are the same and correspond to standard point definition, which is not the case of weighted convolution (the difference appears with *Compact Polynomial* kernel : the maximal radius does not depend on the weight but only on the kernel and the sharpness of the primitive increase with the weight).

3.3.4 Closed-form solutions for SCALIS

Let us first stress that as for the convolution integral, providing closed-form solutions for the SCALIS integral (equations (3.3) and (3.5) is essential to get efficient field queries. The only alternative, when such expressions cannot be found, is numerical integration which implies choosing a trade-of between efficiency and accuracy.

Segment-primitive with linear radius

The case of segment-skeletons is the easiest. Indeed, general, closed-form solutions have been developed for weighted convolution [HC12]. The integral used in SCALIS being very similar, we extend these solutions below in the case of linear weight variations along segment-skeletons : this is the most useful case in practice, since weights are generally assigned at vertices along poly-line skeletons and linearly interpolated in-between. If needed, subdivision can be used to get smoother variations [AC02].

Assume we have a segment of line $[AB]$, with the following parametrization defined on $[0;1]$

$$\Gamma(t) = A + t\,\overrightarrow{AB}.$$

Two weights are defined at both ends of the segment (τ_0 in A and τ_1 in B), in order to define weight all along the skeleton the simplest way is to use a linear interpolation of the radius at end points. Thus we introduce a weighting function τ defined on $[0;1]$ (which is the interval used for the parametric definition of the segment)

$$\tau(t) = \tau_0 + \Delta\tau\, t$$

with $\Delta\tau = \tau_1 - \tau_0$. Then, we have (with P the query point) :

$$\|\overrightarrow{P\Gamma(t)}\|^2 = \|\overrightarrow{AB}\|^2\, t^2 - 2\overrightarrow{AB}.\overrightarrow{AP}\, t + \|\overrightarrow{AP}\|^2$$

Injecting this relation into equation (3.3) yields to the formulae given in table 3.1. The antiderivatives we get are either of the form :

$$\int w(t)(at^2 - 2bt + c)^{\frac{i}{2}}\, dt \text{ with } w(t) \text{ polynomial and } i \in \mathbb{Z}$$

or

$$\int \frac{(at^2 - 2bt + c)^{\frac{i}{2}}}{(dt + e)^k}\, dt \text{ with } k \in \mathbb{N}^* \text{ and } i \in \mathbb{N}$$

Thus, we can obtain closed-form expressions of the scalar field by following the same principle as in [HC12, Hub12] : we find the closed-form associated with the first formula by directly applying the recurrence formula given in [HC12, Hub12]. If i is even, closed-form of the second integral can be easily found by expanding the numerator and then applying k integrations by part. The recurrence formulae are given in table 3.3 and 3.4.

Note that applying the differentiation rule under the integral sign (Leibniz' rule) yields closed-form expressions for the gradient of the scalar field as well. Indeed, it is sufficient to know the differential of the following functions in order to obtain the expression of the gradient as an integral :

* $N : P \mapsto \|\overrightarrow{AP}\|^2 \;\Rightarrow\; \vec{\nabla} N : P \mapsto 2\overrightarrow{AP}$
* $S : P \mapsto \overrightarrow{AB}.\overrightarrow{AP} \;\Rightarrow\; \vec{\nabla} S : P \mapsto \overrightarrow{AB}$
* $R : P \mapsto \dfrac{1}{h(P)^\alpha} \;\Rightarrow\; \vec{\nabla} R : P \mapsto -\alpha\dfrac{\vec{\nabla} h(P)}{h(P)^{\alpha+1}}$
* $M : P \mapsto (h(P))^\alpha \;\Rightarrow\; \vec{\nabla} M : P \mapsto \alpha\vec{\nabla} h(P).(h(P))^{\alpha-1}$

These integral formula are given in table 3.1. The same recurrence formula as for scalar fields can be used to find closed-form solutions of the gradients, which was never done before with previous models. In practice, as suggested in [HC12], we use Maple to unroll the recurrence and create optimized code for the evaluation of our functions. The number of required operations for each field or gradient query is given in table 3.2. Note that when both f and its gradient are needed, the number of operations can be optimized since many terms appear in both expressions.

Kernel	Formula
Cauchy	$\mathcal{C}_{g,i}(P) = \Vert\overrightarrow{AB}\Vert \displaystyle\int_0^1 \frac{(\Delta\tau\, t + \tau_0)^{i-1}}{((l^2+\Delta\tau^2)t^2 - 2(uv(P) - \Delta\tau\tau_0)t + d(P)^2 + \tau_0^2)^{\frac{i}{2}}}\,\mathrm{d}t$
	$\vec{\nabla}\mathcal{C}_{g,i}(P) = \frac{i\Vert\overrightarrow{AB}\Vert}{\sigma^2} \displaystyle\int_0^1 \frac{(\Delta\tau\, t + \tau_0)^{i-1}}{((l^2+\Delta\tau^2)t^2 - 2(uv(P) - \Delta\tau\tau_0)t + d(P)^2 + \tau_0^2)^{\frac{i+2}{2}}}\,\overrightarrow{h}(t,P)\,\mathrm{d}t$
Inverse	$\mathcal{P}_{g,i}(P) = \Vert\overrightarrow{AB}\Vert \displaystyle\int_0^1 \frac{(\Delta\tau\, t + \tau_0)^{i-1}}{(l^2t^2 - 2\,uv(P)\,t + d(P)^2)^{\frac{i}{2}}}\,\mathrm{d}t$
	$\vec{\nabla}\mathcal{P}_{g,i}(P) = \frac{i\Vert\overrightarrow{AB}\Vert}{\sigma^2} \displaystyle\int_0^1 \frac{(\Delta\tau\, t + \tau_0)^{i-1}}{(l^2t^2 - 2\,uv(P)\,t + d(P)^2)^{\frac{i+2}{2}}}\,\overrightarrow{h}(t,P)\,\mathrm{d}t$
Comp. Poly.	$\mathcal{R}_{g,i}(P) = \Vert\overrightarrow{AB}\Vert \displaystyle\int_{l1}^{l2} \frac{((\Delta\tau^2 - l^2)t^2 - 2(-\Delta\tau\tau_0 - uv(P))t + \tau_0^2 - d(P)^2)^{\frac{i}{2}}}{(\Delta\tau\, t + \tau_0)^{i+1}}\,\mathrm{d}t$
	$\vec{\nabla}\mathcal{R}_{g,i}(P) = \frac{i\Vert\overrightarrow{AB}\Vert}{\sigma^2} \displaystyle\int_{l1}^{l2} \frac{((\Delta\tau^2 - l^2)t^2 - 2(-\Delta\tau\tau_0 - uv(P))t + \tau_0^2 - d(P)^2)^{\frac{i-2}{2}}}{(\Delta\tau\, t + \tau_0)^{i+1}}\,\overrightarrow{h}(t,P)\,\mathrm{d}t$

TABLE 3.1 – ***Integrals for SCALIS segments with linear variation of radius, we use simplified notations :*** $l^2 = \Vert\overrightarrow{AB}\Vert^2/\sigma^2$, $uv = (\overrightarrow{AB}.\overrightarrow{AP})/\sigma^2$, $d(P)^2 = \Vert\overrightarrow{AP}\Vert^2/\sigma^2$ ***and*** $ovh(t,P) = \overrightarrow{AB}\ t - \overrightarrow{AP}$. ***For*** **Compact Polynomial** ***kernel,*** l_1 ***and*** l_2 ***are the parametric coordinates of point of the segment at the limit of the kernel support.***

	f				∇f				$f, \nabla f$			
	$+$	$*$	$/$	$\sqrt{}, \arctan, \ln$	$+$	$*$	$/$	$\sqrt{}, \arctan, \ln$	$+$	$*$	$/$	$\sqrt{}, \arctan, \ln$
SCALIS *Cauchy 4*	29	38	4	(1,2,1)	38	62	6	(1,2,0)	47	81	6	(1,2,1)
SCALIS *Inverse 3*	25	33	5	(5,0,1)	32	58	5	(2,0,0)	40	71	5	(5,0,1)
SCALIS *Comp. Polynom. 4*	43	82	3	(0,0,1)	37	67	3	(0,0,0)	50	102	3	(0,0,1)
SCALIS *Comp. Polynom. 6*	71	144	3	(0,0,1)	62	125	3	(0,0,0)	83	172	3	(0,0,1)
Standard *Inverse 3* Linear	18	20	4	(2,0,0)	26	46	4	(2,0,0)	29	50	4	(2,0,0)
Standard *Cauchy 4* Linear	21	26	4	(1,2,0)	29	40	6	(1,2,0)	32	58	6	(1,2,0)
Standard *Cauchy 4* Cubical	27	34	4	(1,2,1)	36	59	6	(1,2,0)	45	77	6	(1,2,1)

TABLE 3.2 – ***Number of required operations for the evaluation of the closed-form solutions associated to standard convolution and SCALIS, for different kernels.***

$I_{k,i}(a,b,c) = \int \frac{t^k}{(at^2-2bt+c)^{\frac{i}{2}}}\, dt$ with $k \in \mathbb{N}$ and $i \in \mathbb{Z}$
Base cases
$I_{0,1}(a,b,c) = \frac{1}{\sqrt{a}} \ln\left(\frac{at-b}{\sqrt{a}} + (at^2-2bt+c)^{\frac{1}{2}}\right)$ $I_{0,2}(a,b,c) = \frac{1}{\sqrt{ac-b^2}} \arctan\left(\frac{at-b}{\sqrt{ac-b^2}}\right)$
Recurrence for $k = 0$
$(i-2)(ca-b^2)I_{0,i} + a(3-i)I_{0,i-2} = \frac{at-b}{(at^2-2bt+c)^{\frac{i-2}{2}}}$
Recurrence for $k = 1$
$I_{1,i} - \frac{b}{a}I_{0,i} = \begin{cases} \frac{1}{2a}\ln(at^2-2bt+c) & \text{if } i = 2 \\ \frac{1}{a(2-i)}\frac{1}{(at^2-2bt+c)^{\frac{i-2}{2}}} & \text{otherwise} \end{cases}$
Recurrence for $k \geq 2$
$aI_{i-1,i} - bI_{i-2,i} - I_{i-3,i-2} = \frac{1}{2-i}\frac{t^{i-2}}{(at^2-2bt+c)^{\frac{i-2}{2}}}$ if $k = i-1$ $a(i-k-1)I_{k,i} + b(2k-i)I_{k-1,i} - c(k-1)I_{k-2,i} = -\frac{t^{k-1}}{(at^2-2bt+c)^{\frac{i-2}{2}}}$ otherwise

TABLE 3.3 – *Recurrence formula for the computation of closed-form solution of the first type of integral encountered. Note that the first recurrence for $k = 0$ is only valid if $at^2 - 2bt + c > 0$, for more detail refer to [HC12, Hub12].*

$G_{k,i}(d,q) = \int \frac{t^k}{(dt+q)^i}\, dt$ with $k \in \mathbb{N}$ and $i \in \mathbb{N}^*$
Base cases
$G_{0,i}(d,q) = \begin{cases} \frac{1}{d}\ln(dt+q) & \text{if } i = 1 \\ \frac{-1}{(i-1)d}\frac{1}{(dt+q)^{i-1}} & \text{otherwise} \end{cases}$ $G_{k,1}(d,q) = (-1)^k \frac{q^k}{d^{k+1}} \ln(dt+q) + \sum_{l=1}^{k}\left(\frac{(-1)^{k+l}}{l}\frac{q^{k-l}}{d^{k-l+1}}t^l\right)$
Recurrence
$G_{k,i}(d,q) = \frac{1}{(i-1)d}\left(k\, G_{k-1,i-1}(d,q) - \frac{t^k}{(dt+q)^{i-1}}\right)$

TABLE 3.4 – *Recurrence formula for computation of closed-form solution of the second type of integral encountered.*

Case of* Inverse *kernels :

Inverse kernel has a special property in comparison to other kernel. We note that the scale-invariant formulation associated to *Inverse* kernel of order i corresponds to the use of the weight $(\Delta\tau.t+\tau_0)^{i-1}$ in the standard formulation (eq. 1.14). This is due to the fact that $K(\frac{d}{\tau}) = \frac{K(d)}{K(\tau)}$. The other property will be seen in next section.

Case of* Cauchy *and* Inverse *kernels :

For even degree *Cauchy* and *Inverse* kernels, there is an important property from a computational efficiency point of view. The evaluation of these scalar fields uses the sum of two arctangent functions (see table 3.3) which amount for a non negligible part of the computational time. In order to improve efficiency when using those kernels, we can take advantage of the fact that $atan(x) + atan(y) = atan(\frac{x+y}{1-xy}) + k\pi$ which in turn can be evaluated in a numerical stable way by computing $atan2(x + y, 1 + xy)$. For *Cauchy* kernel of degree 4, it reduces computational time by 25 percent.

Case of* Compact Polynomial *kernels :

In order to compute the integral associated to *Compact Polynomial* kernels, one should first select the part of the skeleton that is inside the finite support of the kernel. In order to do so, we search t such as :

$$K\left(\frac{\|\overrightarrow{P\Gamma(t)}\|}{\tau(t)}\right) > 0,$$

with K a decreasing function. This is equivalent to studying :

$$1 - \frac{1}{\sigma^2}\left(\frac{\|\overrightarrow{P\Gamma(t)}\|}{\tau(t)}\right)^2 > 0,$$

or

$$\|\overrightarrow{P\Gamma(t)}\|^2 < \sigma^2\ \tau^2(t)$$

with $\sigma^2\ \tau^2(t) = \sigma^2\Delta\tau^2\ t^2 + 2\sigma^2\Delta\tau\tau_0\ t + \sigma^2\tau_0^2$.

This yields a second degree relation :

$$\begin{array}{rcc}
 & (\|\overrightarrow{AB}\|^2 - \sigma^2\Delta\tau^2) & t^2 \\
- & 2(\overrightarrow{AB}.\overrightarrow{AP}) + \sigma^2\Delta\tau\tau_0) & t \quad < 0 \\
+ & \|\overrightarrow{AP}\|^2 - \sigma^2\tau_0^2 &
\end{array}$$

We solve it by studying the sign of its discriminant. Note that from geometric considerations, we are ensured that the solution is always a connected part of $[0, 1]$ (which is important since having to use two distinct intervals would double the evaluation cost), the proof can be found in appendix A.2 along with the method to compute the clipping in a stable way. In appendix B.3.1, we can find additional information for more numerically stable evaluation of the scalar field.

An example of scalar field obtained is shown in figure 3.8(c). We can note that the kernel support smoothly changes along the skeleton, making it easier to model shapes with variation of radius. It was also the case for Hornus integral surfaces, but the surfaces didn't reproduce the wanted radius at the end point (under a given weight the surface does not exist anymore so objects with sharp features are difficult to create).

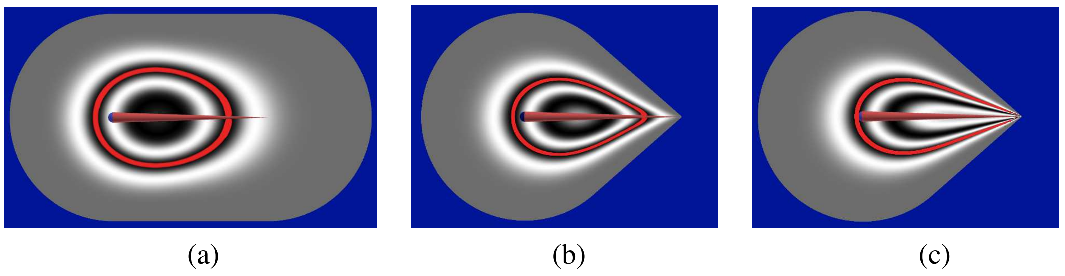


FIGURE 3.8 – *Scalar fields generated integrating a compact support kernel along a segment of linearly varying weight, with a 0 value at one end. The iso-surface of interest is highlighted in red. (a) Standard formulation (eq. 1.14), (b) Hornus formulation (eq. 1.15), (c) Scale-invariant formulation (SCALIS). As we can see, only SCALIS formulation enables the creation of the nice sharp point.*

Other curves with linear radius

Helix primitives computed by warping to segments, as in chapter 2, can be used without problem since we have the formula for segments. For arcs of circle, only the *Inverse* kernel of even degree lead to closed form formula for polynomial weight (deduced directly from the weighted convolution with high polynomial degree). Other kernel are only useable with constant weight (method by warping).

Triangle-primitive with linear radius

In the case of triangle-skeletons, a number of closed-form solutions have been developed for various kernels, but only for convolution with constant radius [She99d, JTZ09, Hub12, ZJLZ12]. We directly derive our close-form SCALIS solution in the constant radius case from these expressions, using the warping formulation.

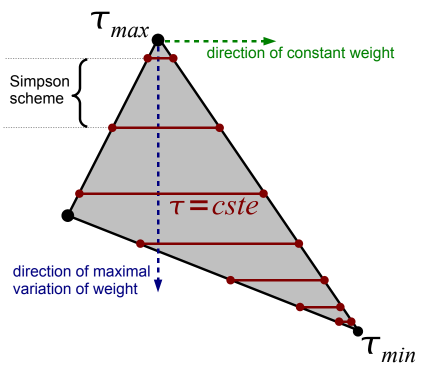

τ_{max}
direction of constant weight
Simpson scheme
$\tau = cste$
direction of maximal variation of weight
τ_{min}

FIGURE 3.9 – *Numerical integration scheme for triangle skeletons.*

We have not been able to derive closed-form formula for triangle primitives with linear radius. Nevertheless, such primitives are of great interest. For this reason, we introduce a numerical scheme that aims at being both accurate and fast to evaluate. In order to do so, we use the fact that if a linear weight is defined on a triangle, there is at least one direction where the radius is constant. Because the case of constant radius for SCALIS is less expensive to compute (notably for compact polynomial), we decided to favor the direction of constant radius in the integration. Thus, we perform a semi-numerical integration scheme. We first integrate in the direction of constant radius using closed-form formula of the integral (one should remember that the normalization of arc-length is $\frac{1}{\tau^2}$ instead of $\frac{1}{\tau}$). We perform the integration in the second direction (orthogonal to the previous one and corresponding to the maximal radius variation) using multiple Simpson scheme which size depends on the local weight τ, the smallest the weight the denser should be the subdivision (see Figure 3.9).

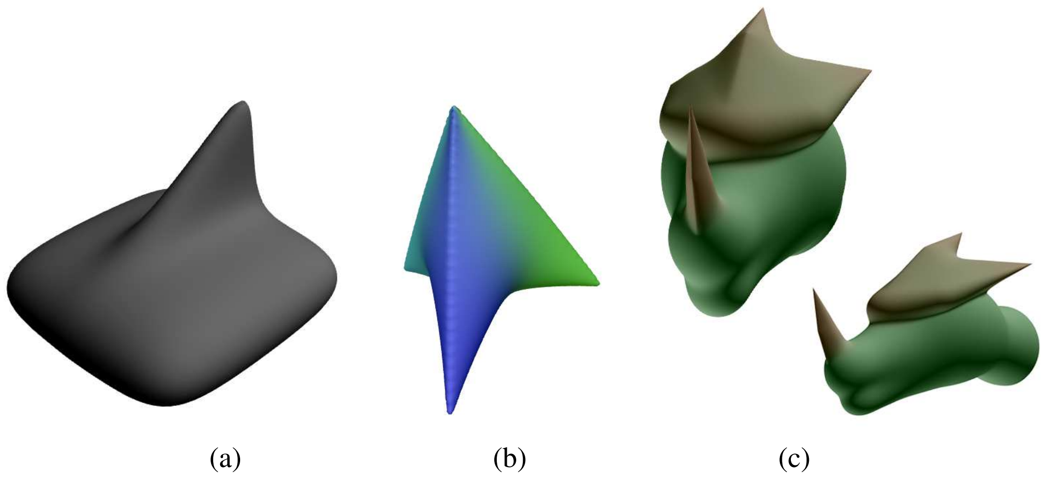


FIGURE 3.10 – *Objects created using triangles primitives : (a) triangles with a constant thickness using a closed-form solution ; (b) triangles with linearly varying thickness using numerical integration ; (c) sharp edges are created by making thickness tend toward zero along one of the edges of triangles.*

Note that the integration scheme could still be improved either by taking into account the query point position to better capture the function to integrate or by using better integration scheme.

Figure 3.10 depicts two objects created using triangle primitives, note that by prescribing radius close to zero we are able to obtain sharp edges.

3.4 ACHIEVING RADIUS CONTROL

This section presents our last contribution to the SCALIS model : we enhance scale-invariant integral surfaces (SCALIS) with direct control of the surface radius along a skeleton. We first present a simple modification of the model enabling to directly use τ to set the desired radius. This radius is reached wherever the skeleton is long enough relative to the local width of the kernel. In Section 3.4.2, we improve the model in order to provide radius control in other cases as well for segment-skeletons.

3.4.1 Relating τ to the radius

Let us come back to the general expression for SCALIS on line-segments (equation (3.3)). We first remark that a slight modification of this expression can bring an important benefit : the model can be set so that $\tau(s)$ directly sets the radius of the resulting shape. This is done in the following way : Let us consider a line (a segment-skeleton of infinite length) of constant weight τ_c, and define $h(\tau_c, d)$ as the field value at a distance d from the line (see Figure 3.11). Formula for the main kernels (eq. (1.11), (1.12) and (1.13)) are given in table 3.5 (for the computation of this value, as

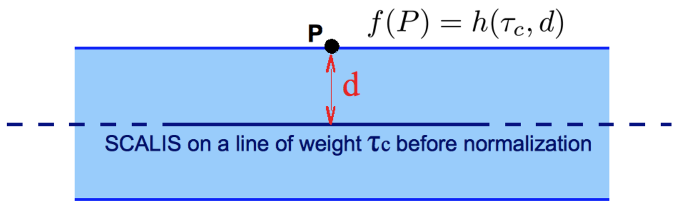


FIGURE 3.11 – *Line used to define $h(\tau_c, d)$ which is used for normalization of the scalar field.*

Kernel	Seed values		Recurrence relation
Cauchy	$h_2(\tau_c, d) = \pi\sigma \frac{1}{\sqrt{\left(\frac{d}{\tau_c\sigma}\right)^2 + 1}}$	$h_3(\tau_c, d) = 2\sigma \frac{1}{\left(\frac{d}{\tau_c\sigma}\right)^2 + 1}$	$h_i(\tau_c, d) = \frac{i-3}{i-2} \frac{1}{\left(\frac{d}{\tau_c\sigma}\right)^2 + 1} h_{i-2}(\tau_c, d)$
Inverse	$h_2(\tau_c, d) = \pi\sigma \left(\frac{\tau_c\sigma}{d}\right)$	$h_3(\tau_c, d) = 2\sigma \left(\frac{\tau_c\sigma}{d}\right)^2$	$h_i(\tau_c, d) = \frac{i-3}{i-2} \left(\frac{\tau_c\sigma}{d}\right)^2 h_{i-2}(\tau_c, d)$
Comp. Poly.	$h_0(\tau_c, d) = 2\sigma\sqrt{1 - \left(\frac{d}{\tau_c\sigma}\right)^2}$		$h_i(\tau_c, d) = \frac{i}{i+1} \left(1 - \left(\frac{d}{\tau_c\sigma}\right)^2\right) h_{i-2}(\tau_c, d)$

TABLE 3.5 – ***Recurrence relation for the computation of*** $h(\tau_c, d)$ ***defined as the scalar field generated at a distance*** d ***of a line of constant weight*** τ_c ***for scale-invariant integral surfaces before normalization (eq. 3.3).***

well as the one for 2D skeletons, we can refer to Appendix B.2). If we look at the $h(\tau_c, \tau_c)$, it happens that this value is independent of τ_c (this is due to the scale invariance of our model), it only depends on the convolution kernel used. Let us now define $N_k(c) = \frac{c}{h(1,1)}$, and then renormalize the scalar field by the inverse of this value. This sets the surface of interest, associated to the iso-value c around the skeleton, to the prescribed distance τ_c from the line. The normalization factor being independent from τ_c :

- we redefine SCALIS primitives generated by segment-skeleton as :

$$f(\mathbf{p}) = \frac{1}{N_k(c)} \int_{s\in\Omega} K\left(\frac{\|\overrightarrow{P\Gamma(s)}\|}{\tau(s)}\right) \frac{ds}{\tau(s)} \tag{3.7}$$

  with $N_k(c)$ a normalization factor depending only on the iso-value of interest c.
- Similarly, we redefine SCALIS triangle-primitives as :

$$f(\mathbf{p}) = \frac{1}{N_k(c)} \int_{(u,v)\in\Omega} K\left(\frac{\|\overrightarrow{P\Gamma(u,v)}\|}{\tau(u,v)}\right) \frac{dudv}{\tau(u,v)^2} \tag{3.8}$$

Just note that the normalization factor depend on the dimension of the skeleton (to compute the normalization factor for a triangle, one should compute the field value at a distance one of an infinite plane).

This improved model provides an explicit control of the surface radius, directly controlled by the choice of τ. Moreover, for *Compact Polynomial* kernels, the prescribed value is exactly reached wherever there is a sufficient portion of segment-skeleton (respectively, of surface-skeleton) of similar weight near the point of interest, which is due to their compact support. The required length L to get a radius τ is given by the Pythagore theorem (see figure 3.12) : $\tau^2 + (\frac{L}{2})^2 = (\sigma\tau)^2$, so the needed length is :

$$L = 2\tau\sqrt{\sigma^2 - 1}.$$

Note that it is proportional to τ, so small, thin primitives can be set to a prescribed radius as easily as large, wide ones.

Following is a small digression about *Cauchy* and *Inverse* kernels. As explained by Hubert in [Hub12], there is only little difference between this two kernels from an algebraic point of

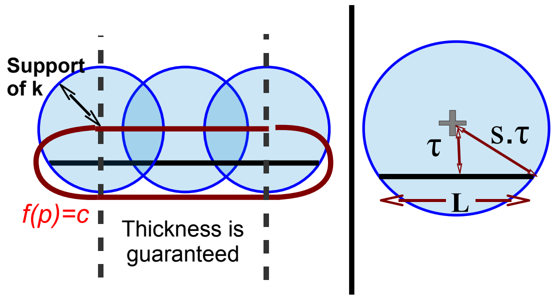


FIGURE 3.12 – *Exact radius control for compact polynomial primitives. The portion of skeleton that influences a point in space are the ones that are at a distance smaller than the kernel radius. Therefore, a point at distance τ of the skeleton has to be in the area of influence of a length L of skeleton for the iso-surface to pass through.*

view. In this new normalized formulation we have the *Cauchy* scalar field that tend toward the *Inverse* scalar field has σ tend toward zero. In fact, the *Cauchy* kernel is equivalent to the new kernel introduced in [Hub12]. It is also important to note that for the formulation 3.7 the *Inverse* scalar field does not depend on σ : the *Inverse* kernel has no real smoothing parameter in contrary to *Cauchy* and *Compact Polynomial* kernels.

The radius of the generated primitive can still be smaller than τ at extremities of primitives, for short skeletons and for extreme radii variations. A last extension of SCALIS that handles these cases is presented next.

3.4.2 Guaranteeing radius in extreme cases

As mentioned in the previous section, the radius of a SCALIS primitive will be smaller than the prescribed value τ if a given minimal length of skeleton of similar radius is not available around the query point ; this length is proportional to the prescribed radius. Thus, the shape typically gets thinner in a few typical situations :

- near the end-point of a skeleton,
- near parts of skeleton with a local maxima of radius.

We develop a practical solution when the skeleton is a graph of poly-lines, which eases skeleton-based modeling and improves the reconstruction in sketch-based modeling applications (see Section 6). It consists of automatically adding the lacking length of integration thanks to a local analysis of the skeleton graph. More specifically, we apply a correction at some of the nodes, depending on their "1-ring" neighborhood.

Extremity : At the extremities of the skeleton, we add an extra segment-skeleton of constant weight in continuation of the ending skeleton branch, set so that the desired radius τ is exactly reached at the end-point of the shape. The required length for this new skeleton is easy to compute thanks

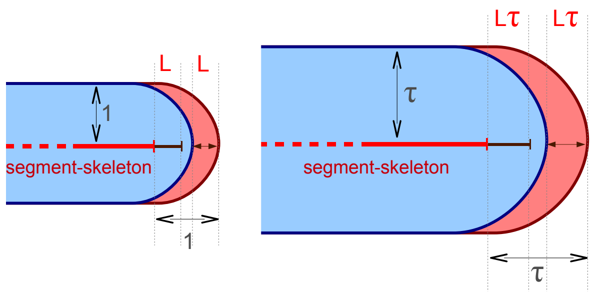


FIGURE 3.13 – *Correction at end point of the skeleton, the length of skeleton to be added is proportional to the wanted radius (thanks to the scale-invariance).*

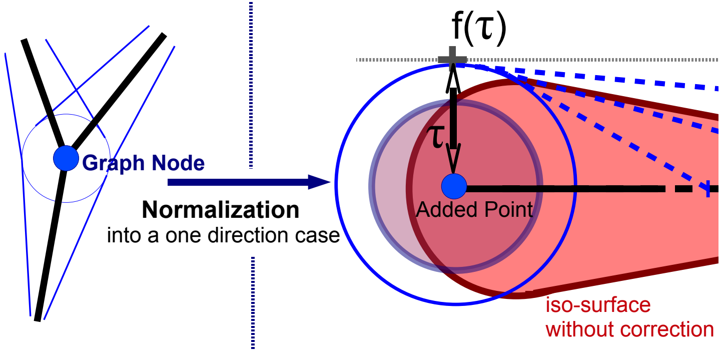


FIGURE 3.14 – *Correction of the scalar field around a node of maximal radius in the skeleton graph. A "normalized" version of the node is used to compute the lacking field value.*

to the scale-invariance property of SCALIS primitives : in the case of constant radius, this length is proportional to τ. So, we just need to pre-compute, for the kernel used, the length l_1 to be added for basic convolution (i.e. SCALIS when $\tau = 1$), and set the length of the extra skeleton to $\tau\, l_1$ in the other cases (see figure 3.13).

Maximal radius : The second case is more difficult to handle since there is no clear direction in which to add a segment-skeleton. For this reason we choose to concentrate all the needed length of skeleton at the point S_{max} of the skeleton with the maximal desired radius. This is equivalent to adding a Dirac in the field integral or to adding a standard point-skeleton primitive to the skeleton.

$$f(P) = w\, K\left(\frac{\|\overrightarrow{PS_{max}}\|}{\tau_{max}}\right) \tag{3.9}$$

with τ_{max} the desired radius in S_{max} and w a weighting value computed as to get this required radius. The weight w is computed by looking at a "normalized" version of the surface behavior around this node of the skeleton graph : we consider that all the segments connected to this graph node point to the same direction and that they continue until their prescribed radius falls to zero (see figure 3.14). This was chosen to be independent from skeleton subdivision, and it enables to have only a local processing. In this configuration, the lacking field value at a distance τ is given by $c - f(\tau)$, where $f(\tau)$ is the field value of the normalized case at a distance τ in a direction orthogonal to the segments primitives (see figure 3.14). From this, we get $w = \frac{c - f(\tau)}{k(1)}$. In addition to independence toward skeleton subdivision, this normalized case also aim at avoiding strange behavior which would happen if we choose to evaluate the original scalar field at a point around the graph node to be corrected. Furthermore, we would not know where to evaluate the original scalar field.

In practice : The case of correction at the end point is, in fact, a bit more complicated. Indeed, they can be both local maxima of wanted radius and end-points. If the variation of wanted radius is important it is much more interesting to use a correction created by a point instead of a segment (it will provide a better correction and is faster to evaluate). For this reason, we propose in practice to interpolate between our two solutions, which can easily be done since we only use summation blend. First, we compute them separately. Then if the variation of radius $\Delta\tau_u$ (per length unit) is greater than 0.5 then we only use the point corrector. Otherwise, we do a spline interpolation between the correction using the following formula for the weight of

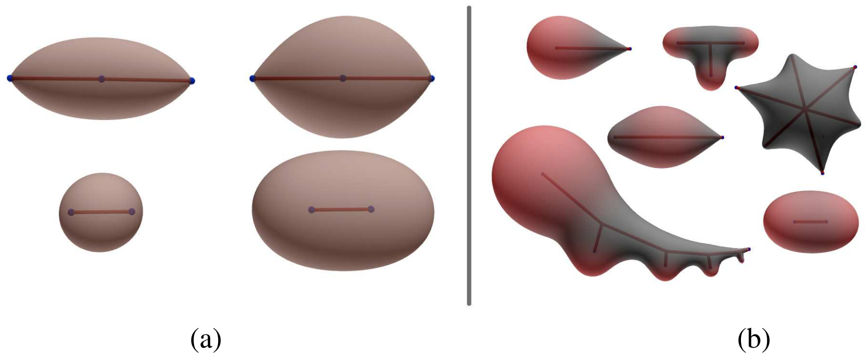


FIGURE 3.15 – *(a) Comparison between SCALIS models before (left) and after (right) radius correction, (b) some simple correction of radii (appearing in red), just note that the "star" shape does not have correction since the blending of all leaving branches is sufficient to obtain the wanted radius in the "normalized" configuration.*

the point :

$$w_{point} * = 1 - (1 - 2\Delta\tau_u)^2$$

and adding a constant weight on the segment :

$$w_{segment} = (1 - 2\Delta\tau_u)^2$$

Note that this solution for insuring prescribed radii does not require any optimization step, which makes it fast and stable, and is coherent with the pleasant summation blend. Due to the addition of a skeleton point, the level of regularity of the resulting field is now the one of the kernel. Some base results are depicted in figure 3.15(b) and the correction is also used in more complex models (Teaser figure and 3.17).

Barycentric blending of correction

Multiple corrections in the same area could lead to unwanted bulging in some special cases. Even if those cases are unusual, it is possible to add safeguards by using a barycentric blending to blend the correctors together :

$$\sum_{c_i:corrector} \frac{w_{c_i} f_{c_i}}{w_{c_i}}$$

where w_{c_i} are chosen in order to use the main contribution when there is only one and lower contribution when there is multiple correction with different influence, thus individual field contribution f_{c_i} are a good choice. This would prevent the added correction to be more important than the maximal one.

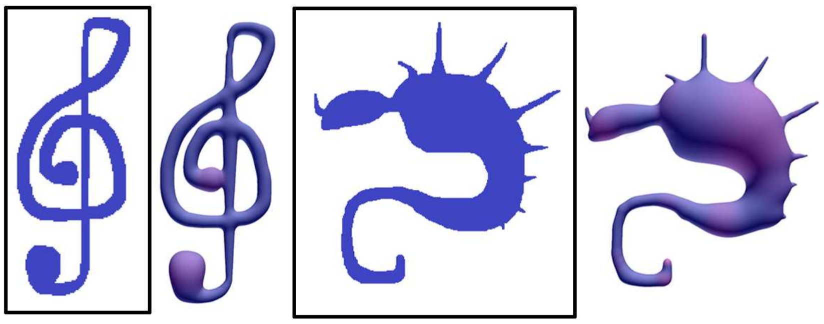


FIGURE 3.16 – *Sketch based modeling examples, area affected by the radius correction method appear in purple.*

3.5 RESULTS AND DISCUSSION

3.5.1 Implementation details

Kernel choice :

For all interactive applications, we used the *Compact Polynomial* kernel (equation (1.13)) of order $i = 6$. This choice is dictated by several consideration : thanks to its local support, it saves computational time while limiting blending at distance. The improvement of efficiency is due to the restriction of evaluation to tight bounding-boxes (with other kernel this would impact field regularity and create local artifact). As the degree i increases both the degree of continuity of the surface and computational cost increase. Thus we advice to use a degree $i = 6$ as a trade-off : it is the first one for which the Hessian of the field is continuous (ie - continuous curvature along the surface). The shape of figure 3.8(b) cannot be modeled with a support of constant radius (so, with standard convolution), since it has both a large smooth part and thin sharp part.

Rendering method :

In order to provide interactive feedback during interactive modeling sessions, we used a local marching cube method [Blo94], enabling to recompute only the edited parts of the surface. Computation times for the whole meshes are given in Table 3.6. Just note that the efficiency is impacted by the resolution required to get small details. This can become really problematic when the difference between minimal and maximal radius becomes large. This is the case if we want to mesh the whole ant model with a resolution that enable us to obtain good detail quality that allows a close-up of the mandibles. For this reason, an improvement would be to use a dual marching cube method on an octree [JLSW02] whose construction would be guided by the BlobTree. First tests show that for a shape such as the ant, we can benefit from a speed-

	Number of Triangle	Computation Times
Dancer Teaser (a)	93 984	0.32s
Elf Teaser (b)	214 364	2.64s
Ant (middle Res) Fig.3.17(a)	190 058	0.72s
Ant's Head (HiRes) Fig.3.17(b)	166 568	0.73s

TABLE 3.6 – *Computation time of a non-adaptative marching-cube on an Intel Core 2 Duo (2.4 GHz, 1 core used).*

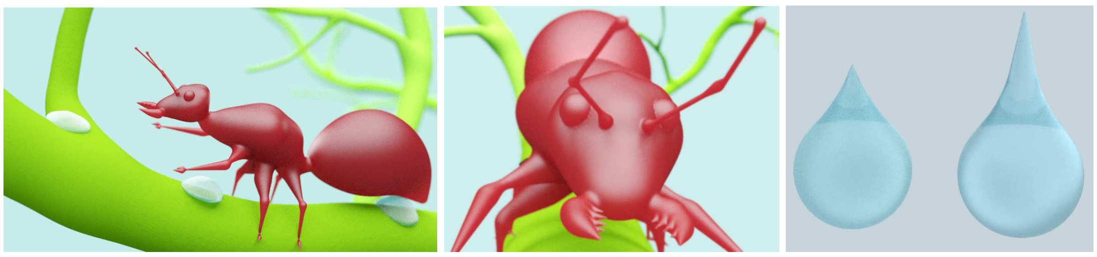


FIGURE 3.17 – *Implicit ant on implicit branch. The zoom on the head of the ant show that the details on the mandible are not blurred (they still look to have sharp points). The right picture shows 2 drops of water created from 3 segment scale-invariant primitives each and by applying the method of thickness correction.*

up factor of ten, despite the fact that the code used can still be further optimized. For some additional details on this method we can refer to appendix B.4.

3.5.2 Applications

We tested Scale-invariant Integral Surfaces (SCALIS) in three use cases : interactive constructive modeling with skeletons, shape reconstruction from medial axis data, and sketch-based modeling. In practice, we use our new implicit primitives in a BlobTree structure, enabling the use of classical composition operators.

Skeleton-based modeling :

To validate the fact that skeleton-based modeling is a good paradigm at conceptual stages of design, we set up a prototype of an interactive modeling system, where a user can create a shape from scratch by manipulating segment and triangle skeletons and prescribing the desired radii at the nodes of the resulting graph. Teaser picture depicts a dancer model created by a computer artist, novice to our system, in less than one hour. Note that this model includes

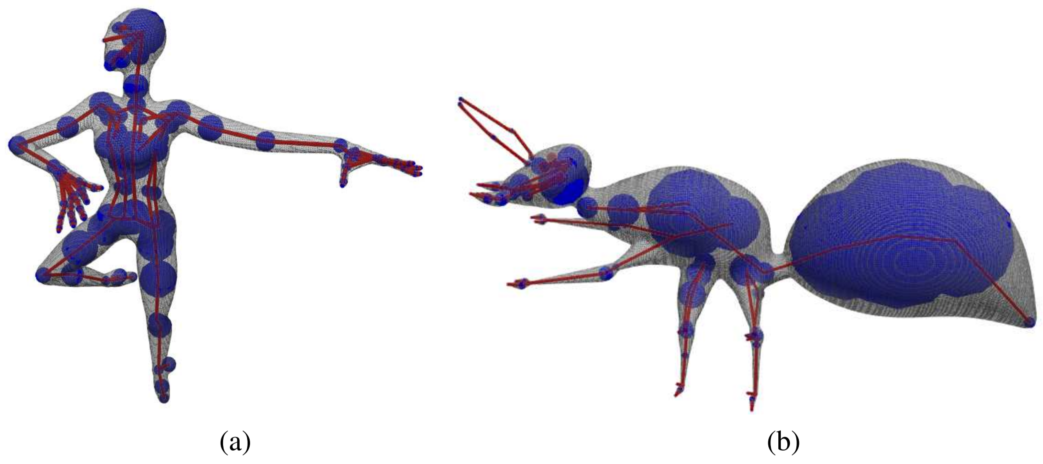


FIGURE 3.18 – *Skeletons of the dancer and ant with the model in wireframe. The size of VertexHandle in blue corresponds to the desired radius at extremities of segment skeletons.*

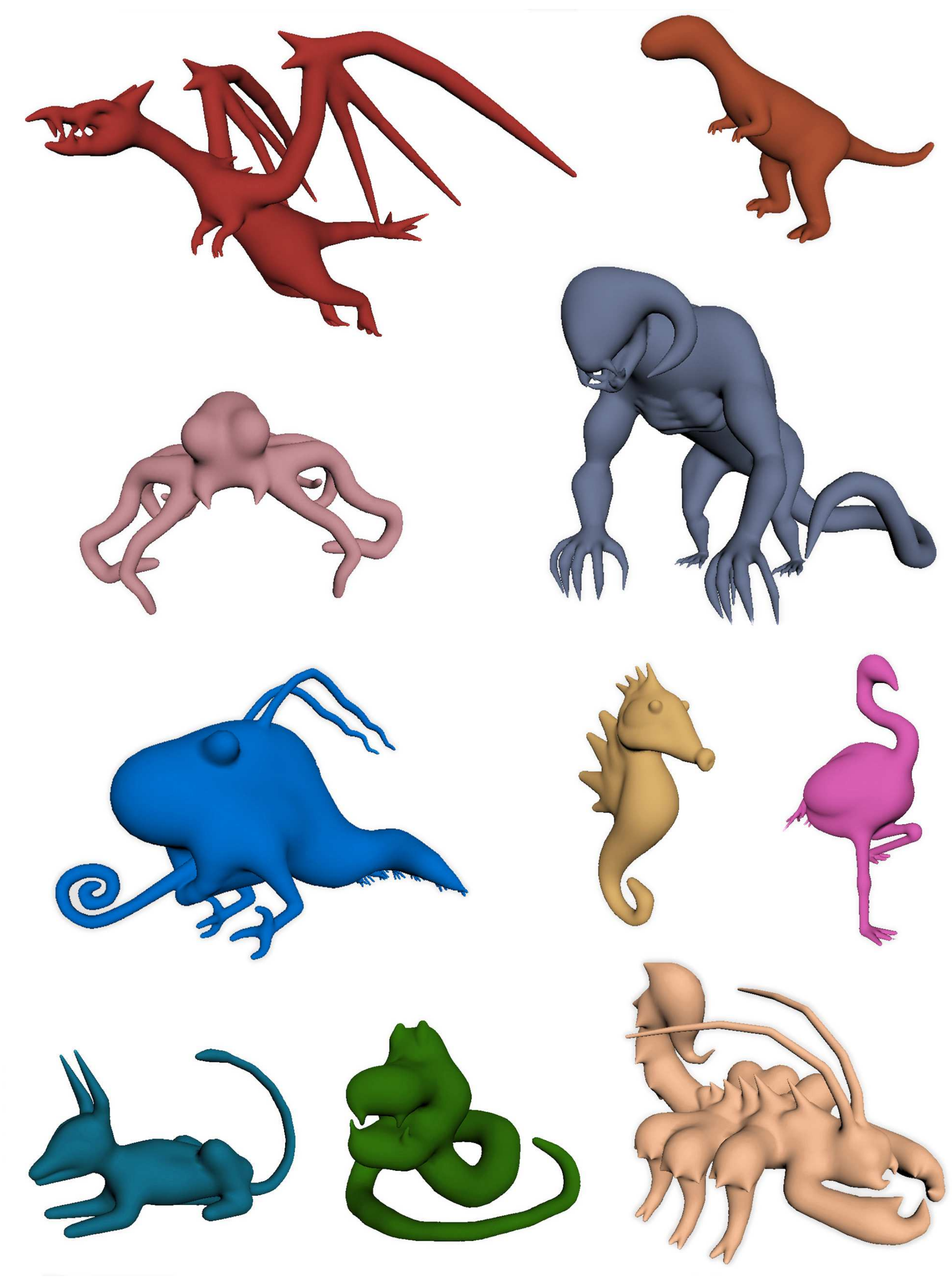


FIGURE 3.19 – *Skeleton-based modeling : Examples modeled in SkimLab (a start-up company with a modeling framework embedded in a web browser* `www.skimlab.com`*).*

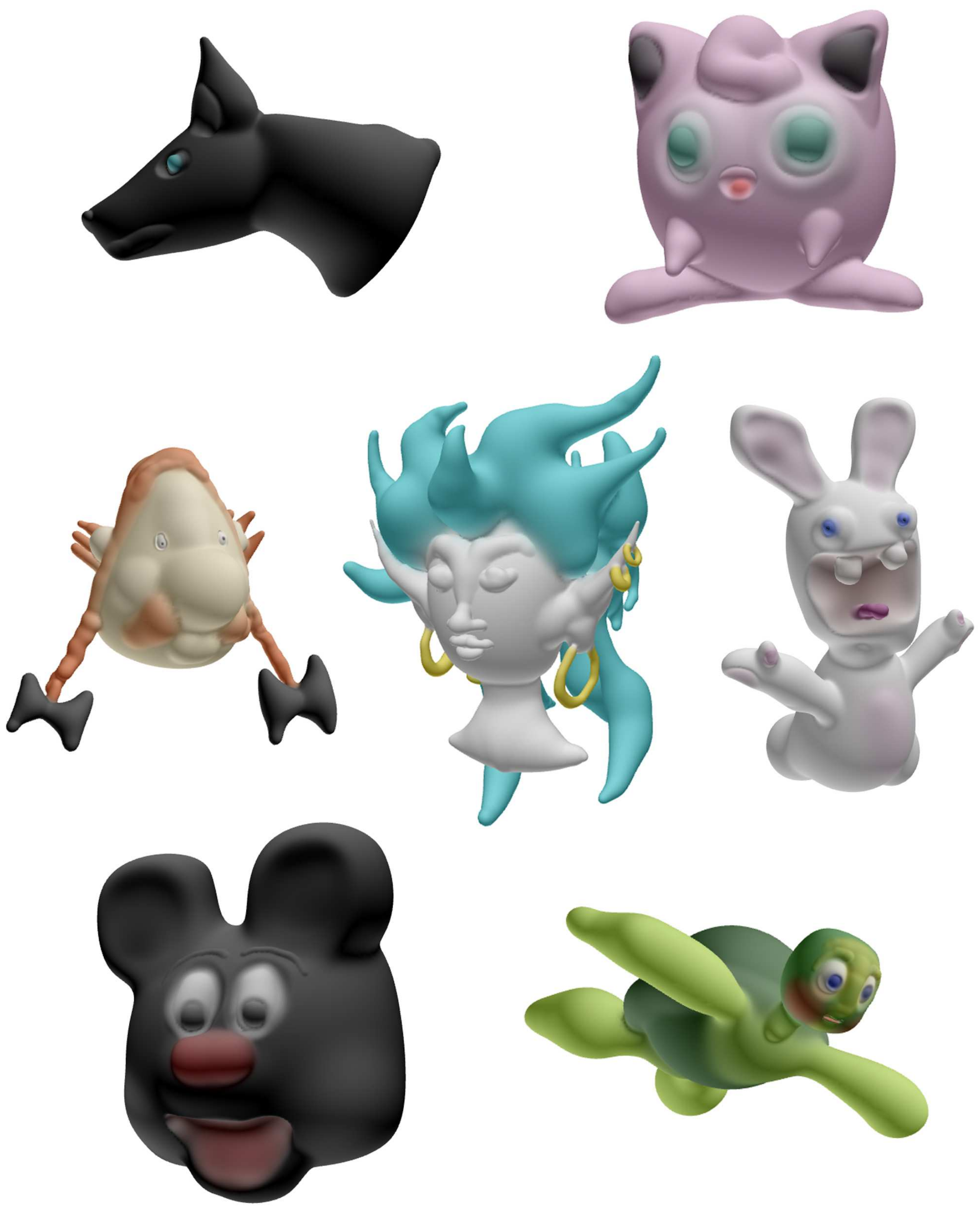


FIGURE 3.20 – *Sketch-based modeling : Examples modeled in MATISS [BPCB08] using our method.*

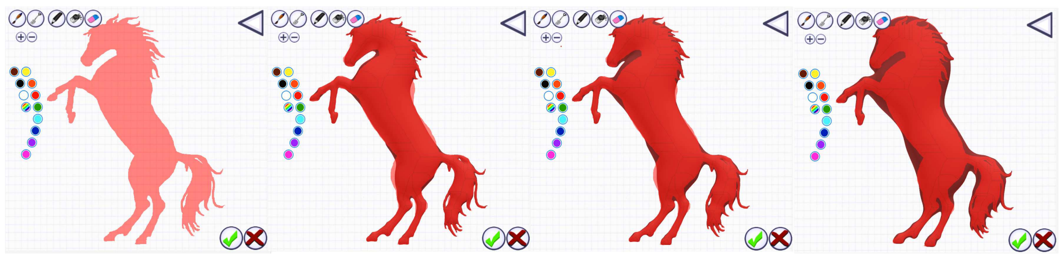


FIGURE 3.21 – *Reconstruction of a drawing using different value of the parameter σ for* Compact Polynomial *kernel (picture on the left is the initial drawing, for pictures on the right σ is respectively equal to* 1.2*,* 2.0 *and* 3.0 *from left to right).*

parts of quite different scales (the whole body and the hand), validating the benefits of our approach. In practice, the hand model was designed at a larger scale and then scaled down to be blended with the arm still enabling further editing.We also used our system to model an ant with very sharp details (figure 3.17) validating the fact that thin details are not smoothed out with SCALIS. Furthermore, we can note on figure 3.18 that the thickness of the iso-surface matches the radii given at skeleton vertices. Additional objects created by a skeleton-based method are depicted in Figure 3.19.

Sketch-based modeling :

In practice, moving skeletons in a 3D space can be intricate for novice users : A good way to make interactive shape design even more intuitive is to set up a progressive sketch-based modeling system. SCALIS is particularly well adapted to this end, since it is able to generate 3D shapes that fit 2D contours directly, with no optimization step, and to seamlessly blend them into the current model using a simple sum of field contributions. Precise fitting of the sketched 2D contour is achieved by converting it to a medial axis representation, and to use it as the skeleton of a SCALIS model, as was done in [lTZkF04] for convolution surfaces. Since for 2D figures, the medial axis is just a graph of polylines, with no surface parts, the methods we described in Section 3 are sufficient to get a good fit of the contour (see figure 3.16). It is now possible to choose the smoothness of the reconstruction (see figure 3.21).

In figure 3.20, the central model (the elf head) was created by using solely a summation blend. It is possible to use different σ parameter on the different part of an object in order to obtain different smoothness such as illustrated on figure 3.22. However, we also demonstrate in this modeling framework that our primitives can be used with other kind of blending. The other examples of figure 3.20 were created by using Ricci blending and digging by difference to combine the shape generated from each drawing (of course, the base shape are created with a summation blend). Just note that, for digging, we use a slightly modified blend by difference in order to obtain a positive scalar field with C^1 continuity.

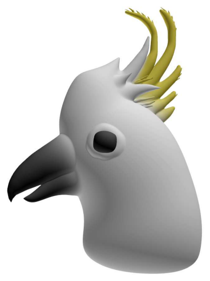


FIGURE 3.22 – *A Cacatoes created using different σ parameter to create more or less sharp part. For instance the beak and the yellow crest use a smaller value of σ to obtain sharper blending.*

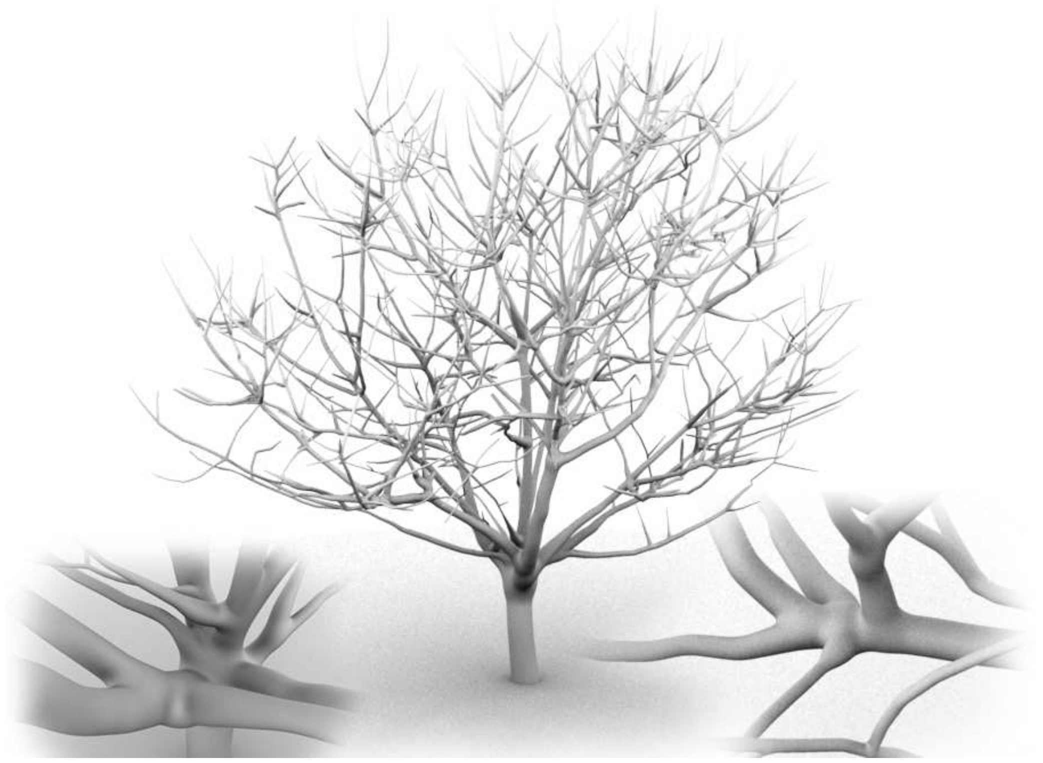


FIGURE 3.23 – *A cherry tree reconstructed from its medial axis.*

We just want to note that there is still space for improvement in the skeletonization process used : the main one is the use of a filtering method on the skeleton, such as the Scale-axis transform [GMPW09] (applied on the medial axis in 3D and not 2D since it doesn't have the same behavior), in order to remove unneeded skeleton elements (which can cause oscillation on the surface due to the blending) .

Reconstruction from medial-axis :

Our last application is the off-line reconstruction of a model from a scanned object. We used SCALIS to directly generate the shape from a medial axis already extracted from scanned data of a real tree. SCALIS is a very good choice for this application : firstly, the field function enables a pseudo-distance between data-points and the iso-surface of interest to be computed, providing an error measure on the quality of reconstruction ; in addition, the resulting model is very compact, and can be easily edited through direct skeleton manipulation ; lastly, the scale-invariant blending behavior of SCALIS produces smooth and self-similar branching at different scale, a very desirable behavior for a fractal-like, self-similar object such as a tree (see Figure 3.23).

3.5.3 Discussion

We already mentioned the advantages of our model over convolution, namely radius control, non-blurring and non-vanishing of details, and scale invariant blending. Based on these properties, SCALIS is able to capture smooth shapes with sharp features, as those of figure 3.17, 3.24 and 3.25 (see the ant's mandible), that could not easily be modeled with previous implicit mo-

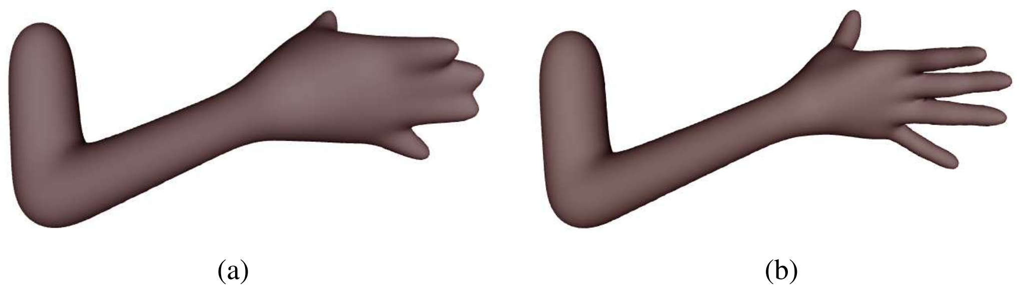


FIGURE 3.24 – *(a) Hand of the dancer modeled with standard convolution, note the unwanted blurring of the fingers, (b) hand modeled with SCALIS.*

deling techniques. If we take a look at figure 3.24 with the hand of the dancer (in this case modeled with *Cauchy* kernel), we can see that the shape we would have obtained using classic convolution is not acceptable : fingers are nearly lost. Furthermore, ant's abdomen would have been difficult to design with standard convolution model with *Compact Polynomial* kernel since there is a large difference between maximal and minimal radius.

One should also note that SCALIS works with any of the Kernel functions of equations (1.11), (1.12) and (1.13), and in particular for any choice of their parameter σ. Since our model maintains the prescribed radius around skeletons and that σ controls the based width of kernel support (hence the blending properties), our model can reconstruct shapes that tend toward an exact reconstruction of the medial axis data : this is achieved by making σ tends towards 1 for *Compact Polynomial* kernels. A last benefit of our approach is to provide closed form solutions for both the field values and their gradient. A closed-form gradient is without doubt more precise than a numerical one (no error of discretization) but it is also more efficient, which is useful to accelerate tessellation or ray-tracing of the models. Numerical scheme rely on either 4 or 6 evaluations of the base function in order to compute its gradient. In our case the closed-form evaluation of the gradient needs at most twice the number of operation as the the basic function (see table 3.2, for *Compact Polynomial* kernel, the evaluation of the gradient is even cheaper than the evaluation of the function itself). Furthermore, if both f and its gradient should be computed, it can be done more efficiently since they have some term in common.

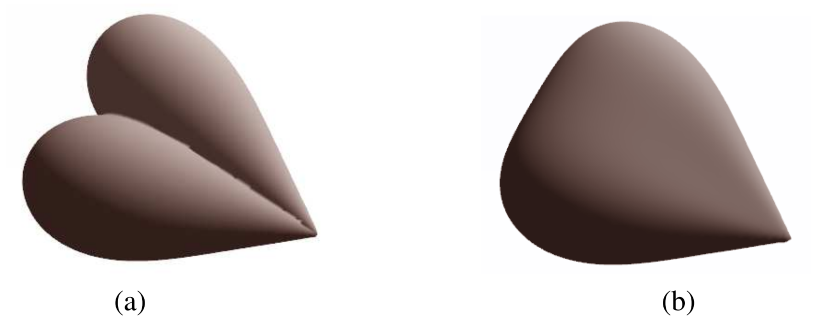


FIGURE 3.25 – *The positioning of two scale-invariant primitives in a "V" shape illustrates the good behavior of the SCALIS blending, which is the same at both extremities of the shape (we can note that the thin extremity is not blurred). (a) blending with a "max", (b) blending with a "+".*

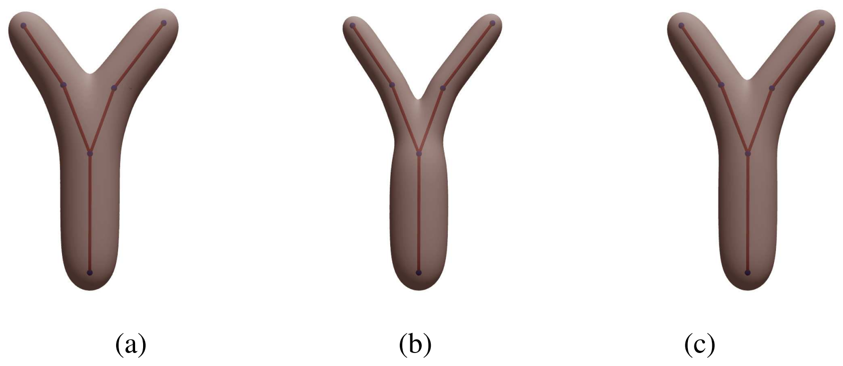


FIGURE 3.26 – *Comparison between two types of branching junction : (a) Standard convolution model with* $\tau_1 + \tau_2 = \tau$*, (b) SCALIS with* $\tau_1 + \tau_2 = \tau$ *(*τ *now controls the radius), (c) SCALIS with* $\tau_1^2 + \tau_2^2 = \tau^2$.

SCALIS also brings a few drawbacks, discussed below :

- SCALIS is more expensive to compute than former models (see Table 3.2 for the comparison between SCALIS and standard weighted convolution). However, SCALIS is particularly well adapted to compact support kernels, which enable local field queries. This makes the use of our model still possible in interactive applications.
- Scale invariant blending may not be always sufficient : with *Compact Polynomial* kernel, when a thin segment is blended into a thick one, the blend is sharper than one would expect. A practical solution for the user is to split the thin segment and manually add the desired transition of radius.
- Loss of "superposition" property : One of the advantages of standard convolution surfaces (equation (1.14)) is that whenever we stack two skeletons with weight τ_1 and τ_2, then the resulting surface is equivalent to the one generated from a single segment of weight $\tau_1 + \tau_2$. This property made it easy to create "Y" branching junctions where a main branch subdivides into two thinner ones without suffering from any bulge. The loss of this property is to be mitigated : the quality of the shape produced highly depends on the kernel used and on its σ parameter (see figure 3.26 (a) versus figure 3.27(a)). In case the shape produced by the superposition property is intuitive as in figure 3.26 (a), SCALIS enables to model it using another rule : $\tau^2 = \tau_1^2 + \tau_2^2$ (see figure 3.26(c)). τ being the radius of the model, this new rule expresses the fact that the area of the cross section of the shape before and after the branching point is preserved. In other cases (see figure 3.27), we may need a different skeleton graph to get the same shape. Then our skeleton is closer to the medial axis of the shape and thus more intuitive to manipulate.

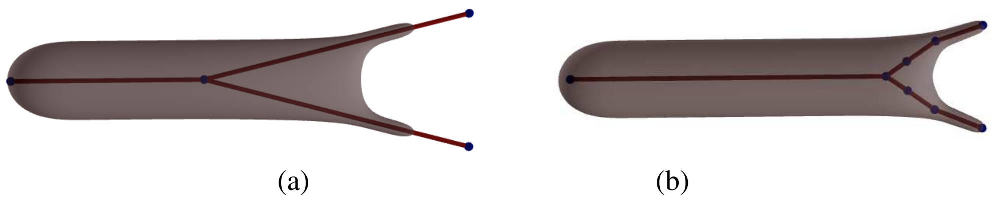


FIGURE 3.27 – *(a) Superposition property for standard convolution with Compact Polynomial kernel of degree* 6 *with* $\sigma = 2.0$ *and* $\tau = 1.16$*, (b) similar shape designed with scale-invariant integral surfaces. Note that the skeleton of (b) is much closer to the medial axis of the shape.*

3.6 Beyond SCALIS

On one hand, with scale-invariant integral surfaces, we have a model providing better results than weighted convolution and whose behavior is both more intuitive and easier to analyze (indeed it is the same at all scale). On the other hand, superposition property of weighted convolution is still interesting in some cases. However this two models are not necessarily antinomic, they can be combined in order to obtain the interesting properties of both models. This is done by adding back a new density ω on the skeleton :

$$f(\mathbf{p}) = \frac{1}{N_k(c)} \int_{s\in\Omega} \omega(s)\, K\left(\frac{\|\overrightarrow{P\Gamma(s)}\|}{\tau(s)}\right) \frac{ds}{\tau(s)} \tag{3.10}$$

This density would be constant and equal to 1 when the scale-invariant property is wanted. In order to obtain closed-form expression of equation 3.10 for segment skeleton with linear wanted radius τ and linear density ω, we re-use the recurrence relationship presented in table 3.3 and 3.4.

In the rest of this section, we present some examples in order to show what could be achieved using this combination of existing models. It is important to note that those examples are not achieved with automatic method but are the result of skeleton manipulation by hand.

Creating shape with non-circular cross section

SCALIS with density could ease the creation of non circular cross section easy to manipulate. In order to do so, we can subdivide a skeleton in sub-skeletons which summed density equal to one, this skeleton can be moved in order to change the cross section of the object. Since the "energy" generated by the skeletons is constant, the volume of the shape does not vary too much when sub-skeletons are moved apart and if all the sub-skeletons are superposed then the cross section is circular. Examples of objects created using this method are given in Figures 3.28 and 3.29.

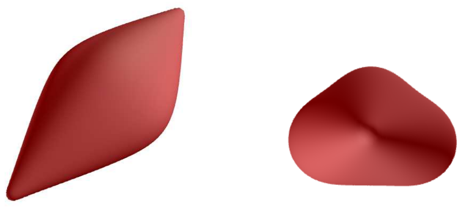


Figure 3.28 – *Non circular cross section.*

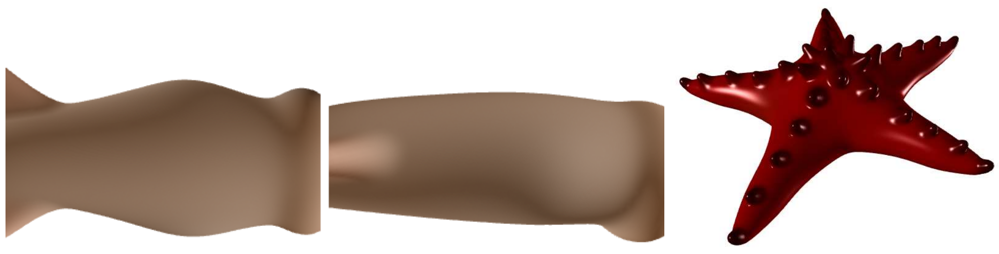

(a) (b)

Figure 3.29 – *(a) the body of the the squid of picture 2.17 have been created using weighted SCALIS. (b) main part of the starfish have been created with weighted SCALIS.*

Adding smooth material to an object

For sketch-based modeling, it would be interesting to be able to add material on a surface while choosing the smoothness of the added material. On figure 3.30, different smoothness of material have been created by using both τ and ω. The main difficulty to be overcome is to succeed to choose τ and ω automatically depending on the shape to be deformed (basically τ of the added material should tend toward the one of the base shape while ω should decrease to preserve the generated energy).

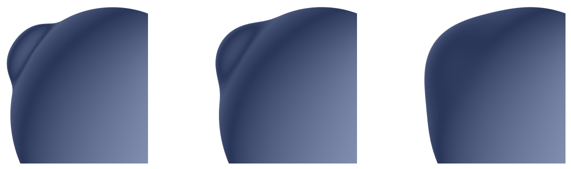


FIGURE 3.30 – *Different degree of smoothness of the added material.*

Reducing Bulges at skeleton junction

In the same spirit as segment shortening in [Blo95a], weighted SCALIS could help reduce the bulging effect near branching junction of skeleton. The idea is to reduce the density ω of the skeletons around the branching junction. The main constraint is having the sum of density of each segment equal to 1 on the junction point. In figure 3.31, we present what kind of result could be expected for different kinds of branching.

The idea would be to automatize the spread of density reduction in all branches by taking into account wanted radius of each branches and the angles between them. Furthermore, the potential of a line (equation 3.5) and the one of a half-infinite line (see Appendix B.2) should help perform the automatization.

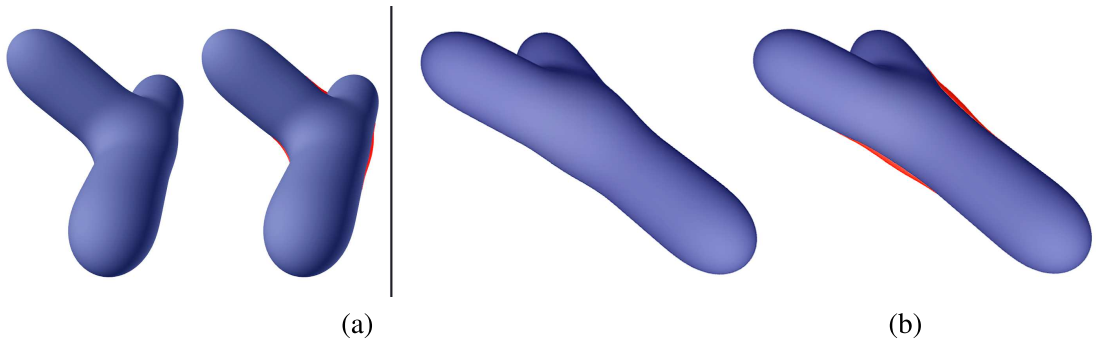


FIGURE 3.31 – *Two examples of bulge reduction using weighted SCALIS. The difference of silhouette appears in red.*

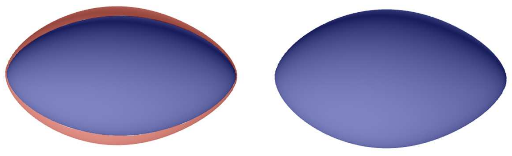


FIGURE 3.32 – *Adding density on segment could help improve radius correction when the local maxima of radius is a small segment with segments of fast decreasing radius on both side.*

Improving correction of radius

It should be possible to perform a better correction of radius than the one presented in section 3.4.2. In order to do so, the idea would be to perform a less local analysis but still using the idea of normalized cases which avoid the problem of global optimization (as for current correction method all the skeleton would be collinear in the normalized case). Typically, the case that is problematic in previous correction is the case of a short segment of constant radius which is a local maxima with segment of fast decreasing radius on both side, example of what could be obtained is shown in figure 3.32.

Note that a simpler alternative to improve the correction would be to add some constraints on the skeletons that could be created.

Intuitiveness and automatization

While this combination of models enables to obtain interesting results, there is still a lot of things to do before being really usable in practice. For bulge reduction and better correction a full automatization transparent to the user is needed. For non-circular cross section, skeleton manipulation should take into account some constraint such as maximal distance between sub-skeleton to prevent creases to appear.

3.7 Conclusion and Future work

Despite the compactness of representation they provide and their interest for conceptual shape design, skeleton-based implicit surfaces are not often used as a modeling primitive. In this work, we solved the problem of generating smooth, solid shapes from skeletal structures and radius information. This was done by introducing Scale invariant Integral Surfaces (SCALIS). Built on convolution surfaces, SCALIS inherited their good properties such as blending with a sum, and shape independence from skeleton refinement. But contrary to convolution surfaces, they bring radius control mechanisms and behave similarly at different scales. This allows reconstruction from medial data and yields intuitive editing.

Despite of the closed-form expressions we provided for f and for its gradient for segment skeletons, there is room for improvement concerning the efficiency of the model : avenues for future work include finding closed-form expressions for triangle with tri-linearly varying weight (or improve the numerical integration scheme), or improving and parallelizing the evaluation of the scalar field and the meshing process.

Lastly, implicit surfaces are well known for their undesired blending at distance behavior. All blending problems of implicit surfaces have been solved recently using complex binary blending [BBCW10, GBC*13]. However such blending do not preserve independence to skeleton subdivision. Furthermore, contrary to N-ary operators, their use complicates the Blobtree, making its manipulation more cumbersome for the user. For this reason, in next section, we aim at improving behaviors of our surfaces while keeping independence to skeleton subdivision, our main focus being reducing topological problems.

CHAPTER

4 TOWARDS TOPOLOGY-PRESERVING INTEGRAL SURFACES

Teaser Figure : *A dragon created with a gradient-based field warping to prevent unwanted topology changes : when the dragon's tail fold back onto itself, it does not merge.*

THE most interesting property of Scale-invariant Integral Surfaces is independence from skeleton subdivision through blending by summation. However, our new model, as all implicit surfaces, does not guarantee that the topology of the resulting surface is the one of the skeleton (not even the one of an infinite union of balls of desired thickness spread along the skeleton, see Figure 4.1). The main cause is the use of the sum of fields contributions which blend elements at distance leading to the well known "unwanted blending" problem. As discussed in the state-of-the-art, a lot of work has been done in order to overcome this problem. However, all the proposed solution rely on complex binary blending operators. By using them, we would lose a fine property of our integral surfaces, which is their independence from skeleton subdivision. Furthermore, a heavy use of binary operators make the handling of the BlobTree more complicated. Besides, the resulting surface then depends on the blending order.

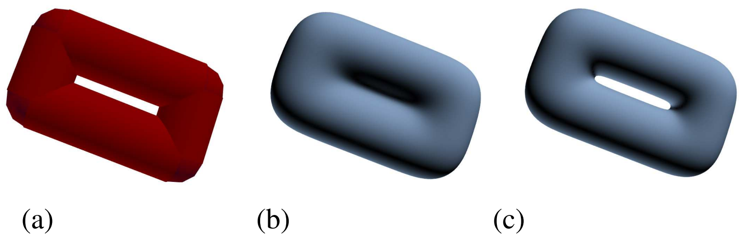


FIGURE 4.1 – *Example of unwanted blending behavior causing change in topology : (a) union of ball defined thanks to the wanted radius around skeletons, (b) blending by summation does not preserve the topology of the union of ball, (c) wanted behavior, the topology is preserved.*

Our aim in this section is to avoid unwanted blending while keeping the independence to skeleton subdivision. Improving the topological behavior of our surfaces is our main focus, however it is not our sole interest. Indeed, while SCALIS primitives greatly improve blending in comparison to other Integral Surfaces their behavior is still perfectible : beside topological problems, bulges can still appear and really large difference of radius between primitives lead to low quality blends (as mentioned in [WW00]). Our second goal is therefore an improved blending behavior in these cases.

In this chapter we present two different approaches to solve these problems : the first one is skeleton-based while the second one is a gradient-based solution.

4.1 FIRST DISCUSSION AND BASE CASES

The first thing to note is that when we refer to *"topology preservation"*, we talk about the topology of the skeleton given the specified radii around it, hence the topology of the union of spheres whose centers are the skeleton points and whose radii are skeleton weights. The new model should also be independent from skeleton subdivision as convolution surfaces were.

In order to obtain this property we believe that classic approach that combines field contributions of each element of the skeleton cannot succeed (with the obvious exception of the sum). Our point of view is that N-ary blending is far from being sufficient (for instance Ricci blending would break the independence to skeleton subdivision property of integral surfaces) and that we should find formulations that remain strongly linked to the summation blend : the first method we propose relies on the choice of the part of the skeleton to be used to compute a blend by sum while the other analyzes the result of a summation blend and corrects it to preserve topology.

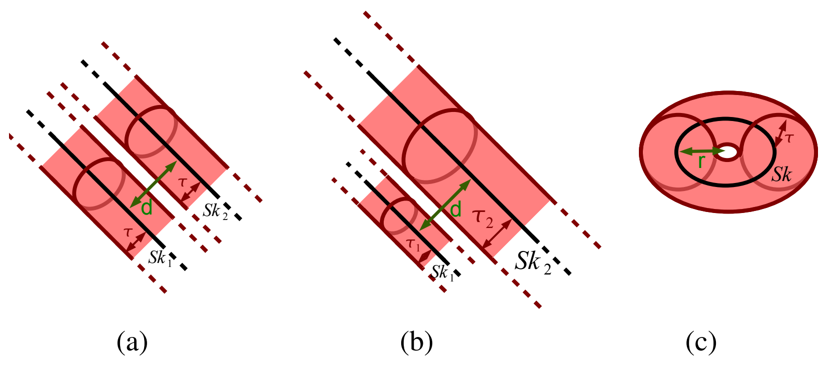


FIGURE 4.2 – *Base cases that are representative of most situations where unwanted change in topology can occur : (a) two parallel line skeletons with equal wanted thickness, (b) two parallel line skeletons with different wanted thickness, (c) a torus skeleton (change of topology happens when its radius is close to the desired thickness).*

Here are the base cases for the skeleton and the desired thickness that we will use to study the topological behavior of our new methods :

1. two parallel infinite lines of the same thickness,
2. two parallel lines with a large difference of thickness,
3. a circle of constant thickness.

The first one is the most basic test that we could think of with the goal of having the shapes blend if and only if their distance is smaller than twice the radius. The second test is useful to check the efficiency of the method in multi-scale cases. Lastly the circle is the "worse" case since all the length of skeleton is at the same distance from its center : the problem here is the persistence of the central hole until the radius is equal to the circle radius : see Figure 4.2.

Of course besides the base cases we also show more complex examples to illustrate blending behaviors.

4.2 WITNESSED BLENDING

The method presented in this part is inspired by several works : the graph-based blending methods by Angelidis [AC02] and Hornus [HAC03] (presented in the state-of-the-art 1.2.2) and by distance function to a probability measure [CCSM11] presented to us by Quentin Merigot. The aim of this last method is to provide good geometric reconstruction from noisy data (the method is robust to outliers). The idea behind this method, from a practical point of view, is to compute a distance-like function to a set of point : the distance to the k-Nearest Neighbors :

$$d_{\mu_\Omega}(X) = \left(\frac{1}{k} \sum_{P \in NN^k_\Omega(X)} \|\overrightarrow{PX}\|^2 \right)^{\frac{1}{2}}$$

with $NN^k_\Omega(X)$ being the k nearest neighbors to X in Ω the initial point cloud.

Our method is named after the approximation scheme that is used to compute efficiently this function : the Witnessed k-distance [GMM11]. Because the computation of k-distance would require an exponential number of distance evaluations, it is approximated by the distance to the barycenter of a set of k points (and a measure of the grouping of the set), one barycenter being computed from point of the initial set, the number of required distance computations is linear with the size of the initial set.

4.2.1 From Witnessed k-distance to Witnessed blending

The main problem of graph-based method for avoiding unwanted blending of convolution surfaces [AC02, HAC03] is that they are not fully independent from the skeleton subdivision. The method we introduce tries to overcome this drawback. We first present the method for convolution of unit weight (for historical reason since at the time we started working on this model we had not introduced SCALIS primitives yet) and we will next see that it naturally extends to SCALIS primitives. Note that we will only study our method for skeleton-curves.

The idea in common with [AC02, HAC03] is that we want to perform our convolution on a subset of the initial skeleton, while the idea taken from k-nearest neighbor is that we will use a fixed amount of data to perform our computations. For the k-nearest points, the initial set is countable which is not the case for segments seen as a set of points. For this reason, instead of using a fixed number of points, we will use a fixed amount of *length*. The basic idea consists in considering the part of the data that is the closest to the computation point (which can be seen as having more interest). Finding a given length L of skeleton that is the closest to a given point P in space is equivalent to finding the radius of a ball centered in P that contains the wanted length of skeleton (we will later see that this is the way the skeleton is clipped in practice). Thus, the new definition of the field function for a given skeleton is :

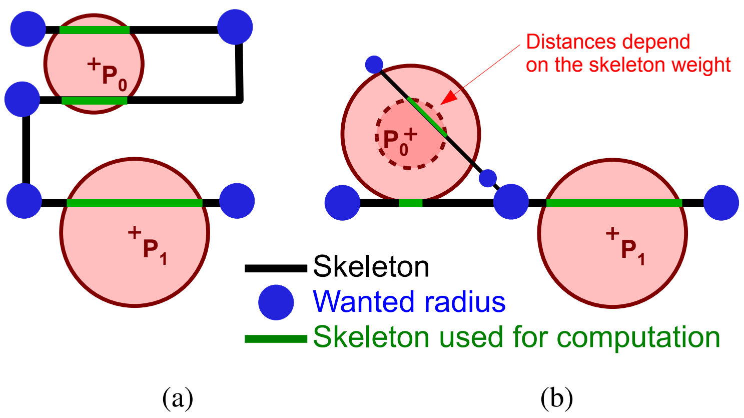
$$f_{\mathcal{S}k}(P) = \int_{Clipping(P,\mathcal{S}k,L)} g_{\mathcal{S}k}(Q)\, K\left(\|\overrightarrow{PQ}\|\right)\, \mathrm{d}Q,$$

with $Clipping(P, \mathcal{S}k, L)$ the ball centered on the computation point that contains the length L of skeleton. Efficient ways to implement this approach will be discussed in Section 4.2.2, in the more general case of SCALIS primitives.

As we can see, the main difference with [AC02, HAC03] is the set on which convolution is computed ; because the length of skeleton to be used is not dependent of the skeleton subdivision (it is just the closest one in space, a segment can therefore be only partly used), this yields the property of independence to skeleton subdivision we were seeking for. We can note that this is not the same thing as using a kernel with compact support since the radius of the ball in which we are selecting skeleton parts varies with the local density of skeleton (see Figure 4.3). This is exactly what will improve the blending behavior.

Varying radius : The question is what happens when one wants to deal with primitives of varying radius ? Fortunately, things don't change that much when using scale-invariant integral surfaces (it would not be so simple for weighted convolution and Hornus integral surfaces).

Indeed, looking for closest skeleton points is the same as looking for the points that have the most influence at the computation point (this is due to the fact that kernels are decreasing functions of the distance). But looking for points with the greatest influence is quite easy for SCALIS primitives due to their scale-invariant nature : the skeleton points of greatest field contribution are simply the closest skeleton points in warped space. (which we will now called homothetic space). In order to maintain coherence we should also consider the length of the skeleton in the homothetic space : it is simply multiplied by $\frac{1}{\tau}$; we will use the term mass instead of length when we speak about length in homothetic space in order to avoid confusion.



Distances depend on the skeleton weight
P0
P1
P0
P1
Skeleton
Wanted radius
Skeleton used for computation
(a)
(b)

FIGURE 4.3 – *With this new blending method we only used a fixed length of skeleton to perform computation. This length of skeleton corresponds to the closest one to the computation point. As we could see in (a), it is not the same thing as using the skeleton inside of a fixed region (as kernels with compact support do), indeed the size of region to be used adapts to the local density of skeleton. (b) smaller radius implies both higher distance and density of skeleton in homothetic space leading to smaller clipped distance in real space.*

Thus, the definition of our new blending becomes :

$$f_{\mathcal{S}k}(P) = N_K(c,m) \int_{HomotheticClipping(P,\mathcal{S}k,m)} g_{\mathcal{S}k}(Q)\, K\left(\frac{\|\overrightarrow{PQ}\|}{\tau_{\mathcal{S}k}(Q)}\right) \frac{\mathrm{d}Q}{\tau_{\mathcal{S}k}(Q)},$$

with $HomotheticClipping(P, \mathcal{S}k, L)$ being the ball of center P that contains exactly the mass m of skeleton in homothetic space, ie - the ball whose radius r verifies the equation :

$$\int_{\frac{\|\overrightarrow{PQ}\|}{\tau(Q)}<r} g_{\mathcal{S}k}(Q)\, \frac{\mathrm{d}Q}{\tau_{\mathcal{S}k}(Q)} = m.$$

Note the new normalization factor $N_K(c, m)$ that replaces the one used in SCALIS. Indeed the latter is not adapted anymore since it assumed an infinite length of skeleton which is obviously not the case anymore. This new factor is just equal to $\frac{c}{h_{m1,1}}$ which is the field generated at a distance 1 of a segment of length m and weight 1 in the plane that is orthogonal to the segment and goes through its middle : this will ensure that the wanted radius is obtained along linear parts whatever the clipped mass.

From scale-invariant integral surfaces to scale-invariant distance surfaces : When we make the clipped mass tend toward zero, we tend toward scale-invariant distance surfaces which are presented in Appendix A. Indeed, when the mass tends towards zero, all the points used will be at the same distance d_w (in warped space) and so have the same contribution $K(d_w)$. For this reason, the integral will tend to be equal to :

$$f = N_K(c,m)\, K(d_w) \int_{HomotheticClipping(P,\mathcal{S}k,m)} g_{\mathcal{S}k}(Q)\, \frac{\mathrm{d}Q}{\tau_{\mathcal{S}k}(Q)}\ ,$$

which, by definition of the clipping, is equal to :

$$f = N_K(c,m)\, K(d_w)\ m\ .$$

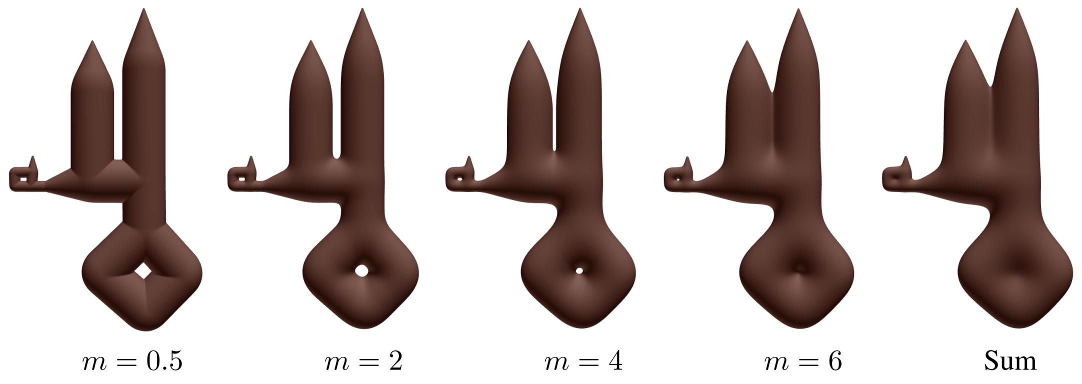


FIGURE 4.4 – *Effect of the mass m to be clipped (Cauchy 4 kernel with $\sigma = 0.3$ used) : when m tends toward zero, we tend toward the union of balls with no blending while when m tends toward the total mass of skeleton present in space we tend towards a sum.*

The normalization factor $N_K(c, m)$ will tend toward $\frac{c}{mK(1)}$, which implies :

$$f = \frac{c\,K(d_w)\,m}{m\,K(1)} = \frac{c\,K(d_w)}{K(1)}.$$

This will be equal to the iso-value c if and only if $d_w = 1$. This means that when the clipped mass changes, we vary between an exact reconstruction of the generalized cylinders associated to the skeleton (union of spheres) and scale-invariant integral surfaces (see Figure 4.4). In the first case, the topology is guaranteed but there is no blending while in the second case there is blending but no guarantee. For this reason, we can choose the amount of topology preservation. It is important to note that the method has the most impact on the region where there is a high density of skeleton, these regions are also the place where the blending is the most problematic. This is why it should be possible to choose a mass value that will improve topological behavior without impacting too much the smoothness of the surface.

4.2.2 Practical considerations

In the previous part, we have presented the theory behind our new blending method. However, some questions have to be answered in order to apply the method in practice.

Computation of the effective clipping

The main problem to solve in order to apply the method is to compute the clipping ball around a given point in space that contains the wanted mass. Since we are not able to do it analytically, we are obliged to do it through optimization. In order to do so, we have to be able to solve the following sub-problem : *"What is the mass inside of a given ball in homothetic space ?"*. Since we want to apply the method when the skeleton is a set of segments, we can obviously solve the sub-problem for each segment individually then add the mass that is inside of the ball in homothetic space for each segments. This can be subdivided into two steps : first, computing the clipping, then, computing the mass of the segment clipped. The first step has already been studied in previous chapter for the purpose of computing SCALIS with *Compact Polynomial* kernel (section 3.3.4 and Appendix A.2). Knowing the clipping in homothetic space, it is quite easy to compute the clipped mass in function of the end point parameters l_1

and l_2 of the clipped part of the segment :

$$\begin{aligned} m_{clipped} &= L \int_{l_1}^{l_2} \frac{1}{\Delta\tau t + \tau_0} \mathrm{d}t = L \left[\frac{1}{\Delta\tau} \ln(\Delta\tau t + \tau_0) \right]_{l_1}^{l_2} \\ &= \frac{1}{\Delta\tau_u} \ln \left(\frac{\Delta\tau l_2 + \tau_0}{\Delta\tau l_1 + \tau_0} \right) \end{aligned}$$

with $\Delta\tau_u$ the variation of radius per length unit. Note that when the latter becomes close to zero the formula becomes numerically unstable, in this case we use a finite expansion of order 2.

Optimization : Knowing how to solve the subproblem, we have to perform an optimization depending on a single parameter : the ball radius, until the ball contains the desired mass. In order to do so, we use a Newton method (note that we can compute the closed-form derivatives of the clipped mass). However, there are a few things that should be taken into account : each time the sphere crosses a new segment end-point there is a tangent discontinuity in the function to be optimized. This added to the fact that Newton method can perform badly in presence of inflection points forced us to add safeguard. We use bounds on minimal and maximal possible radius that are updated at each step using the fact that the function to optimize is increasing in function of the ball radius, if Newton step doesn't fall in-between this bounds, we use a dichotomy step instead.

In order to initialize the optimization, we use a simple heuristic for computing the first radius and initial bounds. The maximal radius is set either to the infinity or to the kernel radius size factor σ depending if the kernel has a compact support. The minimal radius r_{min} is set to the minimal distance between the computation point and the segment in homothetic space. The initial radius is chosen such as to directly clip the right amount of mass if the computation point is along a straight skeleton line with constant desired radius (see Figure 4.5), which give the following formula :

$$r_{init} = \sqrt{\left(\frac{m}{2}\right)^2 + r_{min}^2}$$

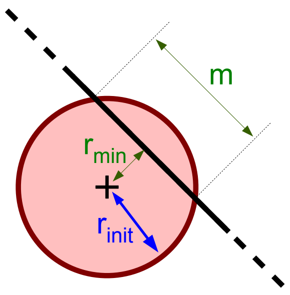


FIGURE 4.5 – *Choice of the initial radius for the optimization method from the mass m to be clipped and from the minimal distance to skeleton in homothetic space r_min.*

There is one problematic case for the clipping : there can be a countable number of points in space where there is an infinite mass of skeleton at the same distance. These query points can arise in two particular positions : at the center of an arc of circle skeleton, but also at a single point on the line defined by a segment skeleton with varying wanted radius (this point correspond to the point of cancellation of desired radius if the segment was prolonged with the same variation of radius, this is demonstrated in Appendix A.2). This fact has not yet been taken into account in our implementation, we just limit the maximum number of iteration to 30 in the optimization method in order to prevent infinite loops ; the main difficulty to solve this problem is to detect it during the optimization. Indeed all the points having the same influence $K(r)$ (r being the ball radius just before the problematic points are clipped) we just have to compute the lacking mass $m - m_{clipped}$ before they are clipped and add $(m - m_{clipped})K(r)$ to the field (note that this can create a gradient discontinuity at the problematic point in space).

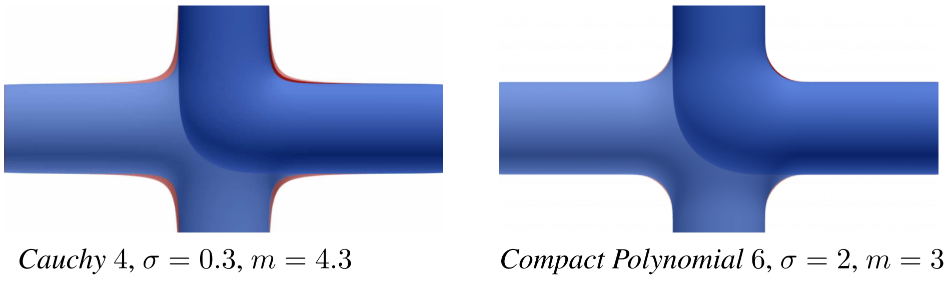

Cauchy 4, $\sigma = 0.3$, $m = 4.3$
Compact Polynomial 6, $\sigma = 2$, $m = 3.6$

FIGURE 4.6 – *In blue is the shape created with Witnessed blending, while in red is the shape created with the blending by summation (light color correspond to a cross skeleton while dark color correspond to an orthogonally bend skeleton). Note that the blend is only affected in a small vicinity of the skeleton intersection and that for* Cauchy *kernel there is less change in the blending shape between cross and bend skeleton for Witnessed blending than for the summation.*

Mass to be clipped

There is an important remaining question : how should we choose the mass to be clipped in order to suppress or reduce unwanted blending ? Our point of view is that the chosen mass should not change too much the blending in the case of an orthogonal bend in comparison to a blending by sum. Furthermore, the blending should be the same for an orthogonal bend and a cross. Because these are not exact constraints, we should choose the "minimal" mass meeting them, indeed, this is the one that will reduce the most the unwanted blending behavior.

Here are the values we found empirically for the kernels we use the most : for *Cauchy* 4 with $\sigma = 0.3$ we use a clipped mass of 4.3 (which slightly reduces the blending in the case of the orthogonal bend but keeps the same blend for the cross, which is not the case for the sum) while for *Compact Polynomial* 6 with $\sigma = 2$ we use $m = 3.6$ which produce nearly the same results has the sum. See Figure 4.6.

Computation of the gradient

Because computing numerical gradients is not efficient and the result tends to be noisy when multi-scale objects are created, we would prefer to have a closed form gradient for witnessed integral surfaces, or at least a closed-form approximation scheme if it is sufficiently precise. Our idea for the approximation scheme is to compute the closed-form gradient using the local clipping value. This means that we neglect the fact that the radius of clipping sphere is changing while evaluating the gradient. This is much faster to evaluate than a numerical gradient due to the fact that we only need one clipping and that closed-form SCALIS gradient is not much more costly than the evaluation of the field. We performed some tests using this approximation scheme and found that the mean relative error with the numerical gradient

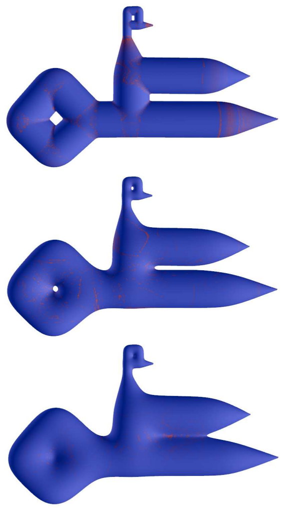


FIGURE 4.7 – *Relative error between numerical gradient and the analytical approximation with constant clipping values, for different values of the clipping length. The maximum relative error is in red and corresponds to* 10^{-4} *while the mean relative error is around* 10^{-6}.

is around 10^{-6} which is quite small. Furthermore the maxima of relative error tends to be scattered over the shape (see Figure 4.7) and are still not very important (around 10^{-4}) which makes us think that they are mainly due to the numerical gradient.

Since the gradient using a local constant clipping value seems to be a good approximation and that it is the exact gradient for both limit cases (clipped length approaching either zero or the total length of skeleton present in the object), we use it in practice and suspect that it is in fact the actual closed-form gradient of Witnessed blending (which we have not demonstrated).

4.2.3 Behavior of the model

As stated earlier in this section, Witnessed blending provides a way to smoothly transition from a union of generalized cylinders to scale-invariant integral surfaces (SCALIS) by switching the mass parameter from 0 to the mass of the whole skeleton. It is important to note that when the mass tends toward zero the quality of the scalar field is decreased (indeed for $m = 0$ there can be discontinuities of the gradient). We will present with more details how this new model behaves.

Postponing topology changes

Since the area used to compute the field change with respect to the local density of skeleton, the change of topology will tend to be postponed when two skeletons come close to one another. Indeed, in this case, the local density of skeleton will increase leading to a smaller clipping ball. The difference with the sum become more and more important as the local density of skeleton increase, so the effect will be more important at the center of a circle than between two segments. The difference of behavior for base cases (parallel segment of equal and different radius and circle of constant radius) is shown in Figure 4.8. Note that even if the difference between sum and Witnessed blending seems small in the graph for Compact Polynomial kernel in the case of parallel segments it is really visible in practice as show Figure 4.9.

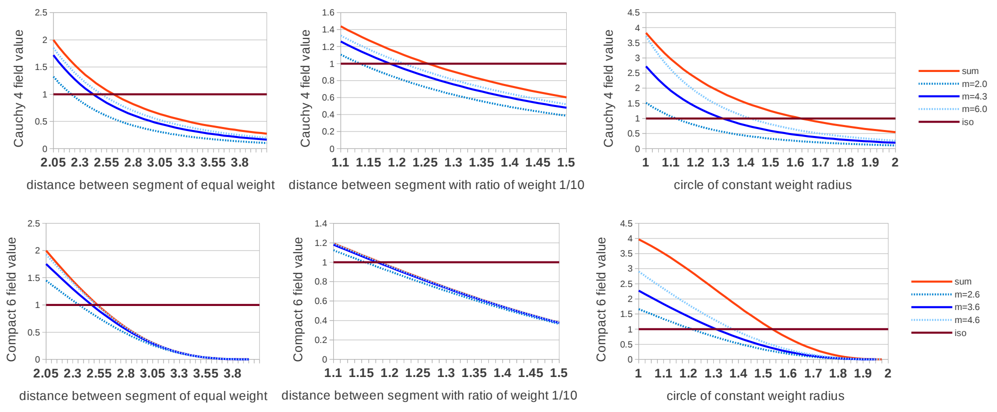


FIGURE 4.8 – *Graph of the minimal field between skeletons (where the gradient is null) value in function of distance/radius for base cases. The interest of minimal field value is that when it crosses the iso-value the topology changes. Note that the topology of the union of generalized cylinders is lost for the minimal abscissa value for each case. We can see that the witnessed blending has the more impact for the circle cases (which is due to a higher local density of skeleton).*

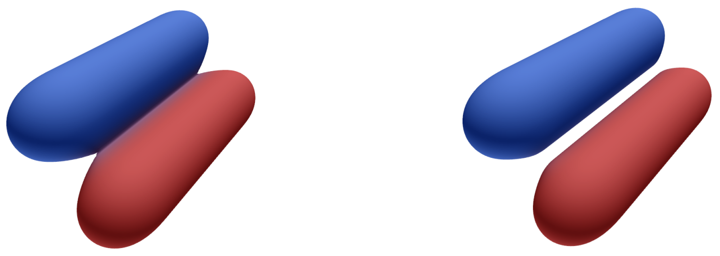


FIGURE 4.9 – *Comparison between SCALIS (left) and witnessed blending (right) for segment primitives for* Compact Polynomial 6 *kernel with a clipped mass of* 3.6 *: the witnessed solution avoids the unwanted blend in this case.*

It is important to note that even if we can postpone change in topology, we are not able to prevent them : indeed the only mass value which would give guarantee is 0 but it would not create a smooth surface.

Reducing unwanted bulges

There is a maximal amount of bulge that can be created whatever the density of skeleton used, it will be obtained when there is a mass m of skeleton concentrated on a unique point in space. In this case, the field generated around this skeleton (assuming it is of constant wanted thickness τ) is :

$$f(d) = N_K(c,m)\; m\; K(\frac{d}{\tau})$$

so the maximal amount of bulge is :

$$\frac{d}{\tau} = K^{-1}\left(\frac{1}{mN_K(c,m)}\right)$$

In Figure 4.10, we can see the effect of the bulge reduction.

Removing unwanted blending due to the global support of the kernel

In the case of kernel with infinite support, the total mass of skeleton can affect the obtained radius : as the global mass augments the shape can slightly inflate which can become problematic for preserving the topology of details. Witnessed blending adds a kind of local influence to each primitive leading to a behavior more similar to kernel with compact support. An example of this behavior is given in Figure 4.11.

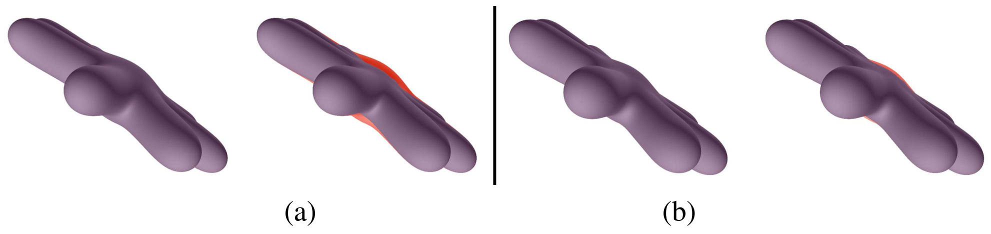


FIGURE 4.10 – *Witnessed blending compared to sum (superimposed in red on the right part of the figure) : (a)* Cauchy 4 *kernel with* $m = 4.3$, *(b)* Compact Polynomial 6 *kernel with* $m = 3.6$. *Note that the witnessed solution reduces the bulge, but is not able to avoid it.*

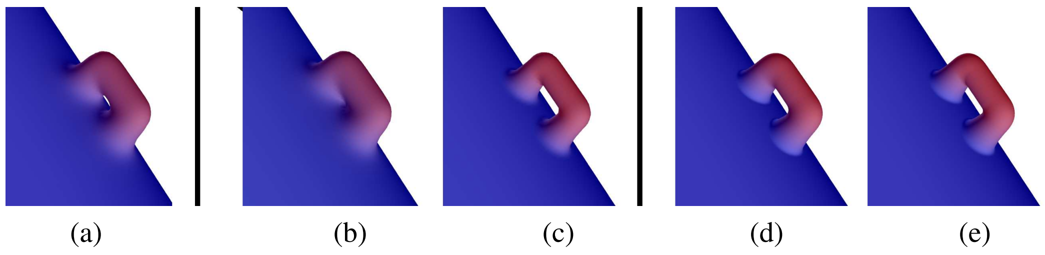


FIGURE 4.11 – *Localized influence with Witnessed blending : (a) a small handle is blending with a sum into a larger primitive, Cauchy kernel is used. In (a,b,c,d) a second large primitive is blended at the opposite of the handle. In (b), with* Cauchy *kernel and a blending by sum, the topology is lost partly due to a subtle enlargement of the blue surface, while in (c) Witnessed blending succeeds to preserve the topology. In (d) sum, and (e) witnessed blend, the* Compact Polynomial *kernel is less affected by the additional primitive.*

4.2.4 Discussion

Improvement to the standard blending problems : As we have seen in previous section, Witnessed blending provides much better results than summation blending by reducing the unwanted blending behavior. However, such a method can't provide guarantee on the topology, the only solution would be to use a mass to be clipped of 0 which would completely remove blending.

Computation time : While the computational time is not really impacted for global support kernel (due to a lower number of integral computations), it is less the case for compact support kernel (see Table 4.1).

If the method is to be applied to complex skeletons, it will become important to use optimization structures to reduce the number of segments to be taken into account when performing the optimization. The main problem is that we are not working in an Euclidian space (for which most optimization structures are intended) but in the warped space. In order to overcome this problem a solution would be to group skeletons by range of wanted radius. Using the maximal radius of a range to compute distance and the minimal radius to compute the clipped length would simplify the problem of choosing the initial set of segments while ensuring to avoid underestimation (not doing so would prevent the optimization to succeed).

Kernel	*Cauchy* 4				*Compact Polynomial* 6			
Mass m used	0.5	2	4	6	0.5	2	4	6
Computation time in percentage of sum	86%	102%	127%	132%	116%	140%	159%	259%
Average number of optimization step per field evaluation	0.77	2.18	3.05	3.57	0,79	2,20	2,60	15,2

TABLE 4.1 – *Computation time in percentage of the blending by sum and average required number of optimization step to perform a single evaluation of scalar field, the error threshold on the clipped mass being* 10^{-6}*. Test was done on the skeleton used in figure 4.4.*

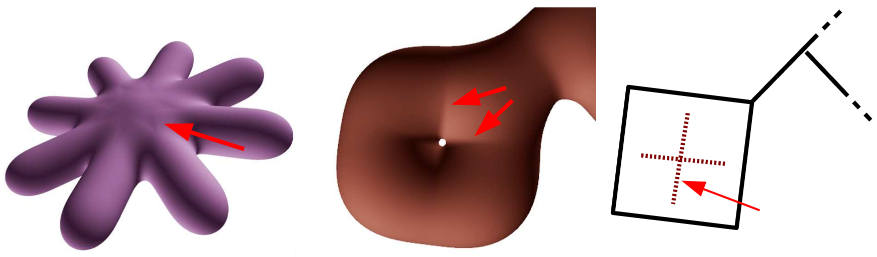


FIGURE 4.12 – *The lack of smoothness of the clipping can impact the smoothness of the surfaces creating visual artifacts.*

Toward smooth clipping The main problem of the method has it stands is the smoothness of the clipping : its lack of regularity (a point of the skeleton can't be partially clipped, there is no smooth transition between the clipped state and the ignored state) leads to a poor field regularity. This is clearly visible in some cases as in Figure 4.12. This mainly happens when the smallest contribution of a skeleton point is not small enough. It should be possible to remedy to this problem by introducing a smooth clipping : instead of just selecting a mass m of skeleton, we could select an additional mass m_t to smooth the clipping. This additional mass would be used with an additional density as in Equation 3.10 : this density should be equal to 1 on the boundary of the main clipping ball and equal to 0 on the boundary of the new ball corresponding to the transition area. In order to do so, the simplest way is to use a weighting function that depends on the ball radius :

$$w(r(t)) = \left(1 - \left(\frac{r(t)^2 - r_{main}^2}{r_{transition}^2 - r_{main}^2}\right)^2\right)^2$$

with $r(t)$ the distance between a skeleton point of parameter t and the computation point (the clipping ball center). The reason for the use of $r(t)^2$ is to obtain a polynomial weighting function : Indeed for segment skeletons $r(t)^2$ is a second degree polynomial. The use of this weighting function would ensure a C^2 clipping.

This idea has not yet been implemented. There is no theoretical reason against its application (we already know how to compute a clipping, and formula are the same as SCALIS with density of section 3.6). However the double clipping combined with an increased computation cost in the transition area due to the new weighting function would probably increase too much the computational time.

4.2.5 Conclusion

While the method we just presented gives interesting results in term of shape control for all kernels, the one that really benefits from it are global kernels since the method gives them a bounded region of influence. An interesting feature of this blending method is the fact that the mass parameter used to tune clipping describes whether we want an exact reconstruction of the required radius or a smoothed reconstruction. It should be interesting to have the mass used vary in space in order to give more control on the blending.

The main drawback of the method is that it would require a smooth clipping in order to be really usable in practice, however, such a clipping would probably increase too much computation time. Moreover, when it comes to control of topology we think that the gradient-based method presented in next section is more promising.

4.3 Gradient-based SCALIS

An important fact when it comes to the topology of implicit surfaces is that points where topological changes can take place are characterized by the fact that the scalar field gradient at this point is null [Har97].

This, combined to the recently introduced gradient-based implicit blend [GBC*13], suggest that the use of gradient is mandatory to produce high quality blending. In [GBC*13], the angle between gradients is used to parametrized the blend. The base parametrization ("camel" controller), basically uses these three facts :

– collinear gradients require no blending to prevent unwanted bulges from appearing ;
– opposite gradients require no blending either to insure that the topology of the skeleton (union of balls) is maintained ;
– orthogonal gradients, on the contrary, require blending to occur.

We cannot apply this method as it stands since it is intrinsically binary due to the use of angle between two gradients . Moreover, applying this method to integral surfaces would ruin the independence from skeleton subdivision. In this section, we try to adapt the idea of defining blending through their gradient to integral surfaces, while preserving their good properties. In order to do so, we introduce a *gradient-based field warping* which can be seen as a unary node operator to add above a sum in the Blobtree.

Context : While we point out the fact that global support is not acceptable, we focus in this part on the *Inverse* kernel. While this can seems irrational, this is motivated by some properties of this kernel that will make our ideas easier to test. We will present in the discussion section our idea to overcome the problems that arise with other kernels. Furthermore, our method is first presented for skeleton-curves and later extended to skeleton of other dimension.

4.3.1 Gradient-based field warping

Basic idea : analyzing the gradient field of a "failed" blend

Since we would like to keep the property of independence from skeleton subdivision, let us try to analyze the behavior of the gradient during a classical summation blend in order to deduce when unwanted bending occurs. Our aim would then be to compute a corrected field :

$$f_{corrected} = warp(f_S, \nabla f_S)$$

that will reduce the field f_S obtained through a summation blending when the latter was too important.

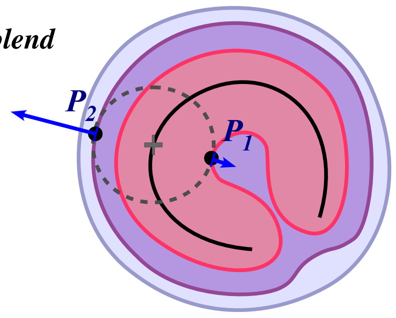


FIGURE 4.13 – *The point P_1 in the inner part of the shape has a field value more important than P_2 but a smaller gradient, this is due to multiple contribution that are scattered around the query point in the first case. In the second case, the field is closer to the one of a line skeleton.*

Let us characterize when we want the field to remain unchanged versus when we want to reduce it the most :

- Our reference for thickness being an infinite line skeleton (see Section 3.4.1), we would like to keep the field unchanged in this case.

- On the contrary, points that lie in-between skeletons (such as at the center of a torus, or between two line skeletons) tend to have a smaller gradient than expected : while each contribution of infinitesimal length of skeleton sums up together to obtain the resulting field, it is not the case for the gradient for which small contributions cancel each other since they tend to have opposite direction.

If we take a look at Figure 4.13, we can be more precise on the way to measure the unwanted blending : the point P_1 which is at the outside of the shape have a field value and gradient that are closer to the one of a line skeleton than the point P_2 which has a small gradient (due to opposite contribution) in comparison to its field value. Thus, we should analyze gradient behavior in comparison to field value.

Our main insight is to use this fact in order to compute a measure of the angle between gradients - or the amount of unwanted blending - in a N-ary way. Using the result of a "failed" summation blend to parametrize our blend, instead of explicitly computing angle between gradients, is the main difference with classic gradient-based blends.

However, we cannot just use the field gradient as is in order to parametrize our correction, since it depends on some other parameters. We will explain how it is used in practice.

Scale-invariant gradient

Scale-invariance is important to analyze things independently from the skeleton size (i.e. desired radius). While the scalar field of Scale-invariant Integral Surfaces (SCALIS) is invariant through scaling of the skeleton, it is not the case of their gradient. This is a problem for obtaining a measure of the topological problem. For this reason, we will analyze a scale-invariant gradient instead of the classical one. From this point we will note $\nabla f_{\mathcal{H}}$ the scale-invariant gradient and $f_{\mathcal{S}}$ the scale-invariant scalar field (SCALIS). From Equation 3.6, we can easily note that we just have to multiply the classic gradient by τ in order to obtain scale-invariance. This lead to the following definition :

$$\nabla f_{\mathcal{H}}(P) = \nabla \left(\int_{\mathbb{R}^3} \frac{g_{\mathcal{S}k}(Q)}{\tau_{\mathcal{S}k}(Q)^{d-1}} \, K \left(\frac{\|\overrightarrow{PQ}\|}{\tau_{\mathcal{S}k}(Q)} \right) \mathrm{d}Q \right) \tag{4.1}$$

Note that in the case of skeletons of dimension 1, the gradient in (4.1) is equal to the gradient of the Hornus integral surfaces [HAC03]. Furthermore, a closed-form solution can easily be computed for classic kernels in the same way we have computed the gradient of SCALIS in previous section. Lastly computing both $\nabla f_{\mathcal{H}}$ and $f_{\mathcal{S}}$ can be done efficiently since their is some redundant computation.

Reference case : Skeletons of infinite length

There is a remaining problem to obtain information on the amount of unwanted blending in the classical summation blend : the norm of the scale-invariant gradient $\nabla f_{\mathcal{H}}$ does not only tend to zero when there is topological problems, but also when the scalar field $f_{\mathcal{S}}$ tends to zero.

Because we have a reference case that should stay unmodified by our correction method - the case of infinite line skeleton - we will try to characterize the relationship between $\|\nabla f_{\mathcal{H}}\|$ and $f_{\mathcal{S}}$ in this canonical case in order to be able to correct the blending based on the relationship between this two values.

This is where *Inverse* kernel is interesting : indeed it is the only classical kernel that brings a *bijective mapping* between $\|\nabla f_{\mathcal{H}}\|$ and $f_{\mathcal{S}}$ in the case of a line skeleton (partly due to absence of inflection point in the kernel definition). The formula for the scalar field $f_{\mathcal{S}}$ is obtained from Table 3.5. By taking into account the normalization factor, we obtain :

$$f_{\mathcal{S}}(\tau_c, d) = \left(\frac{\tau_c}{d}\right)^{i-1} \tag{4.2}$$

From this formula, we first compute the norm of the SCALIS gradient (which is equal to the partial derivative in function of the distance d to the line) and multiply it by τ to obtain the formula for the scale-invariant gradient :

$$\|\nabla f_{\mathcal{H}}\|(\tau_c, d) = (i-1)\left(\frac{\tau_c}{d}\right)^{i} \tag{4.3}$$

We can easily find the bijective mappings between these two values :

$$\|\nabla f_{\mathcal{H}}\| = (i-1) {f_{\mathcal{S}}}^{\frac{i}{i-1}} \tag{4.4}$$

and

$$f_{\mathcal{S}} = \left(\frac{\|\nabla f_{\mathcal{H}}\|}{i-1}\right)^{\frac{i-1}{i}} \tag{4.5}$$

where i is the degree of the *Inverse* kernel. The mapping are thus respectively defined by the two following functions :

$$g(x) = (i-1)x^{\frac{i}{i-1}} \text{ and } h(y) = \left(\frac{y}{i-1}\right)^{\frac{i-1}{i}} \tag{4.6}$$

which verify $h \circ g(x) = x$.

Replacing $f_{\mathcal{S}}$ by a corrected field

We are trying to define a warping operator $warp : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ that will reduce the value $f_{\mathcal{S}}$ when unwanted blending is detected.

We now have the relationship between the scale-invariant field and the norm of the scale-invariant gradient in the case of an infinite line, defined through the bijective mapping g. As stated earlier, the case where the skeletons form a partition of a line is our reference case where blending should act like a sum. Thus, if $\|\nabla f_{\mathcal{H}}\| = g(f_{\mathcal{S}})$ is verified then our correction should leave $f_{\mathcal{S}}$ unchanged :

$$warp(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|) = f_{\mathcal{S}} \text{ if } \|\nabla f_{\mathcal{H}}\| = g(f_{\mathcal{S}})$$

Secondly, if we look at $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ as a 2D point P, and name $\mathcal{G}$ the graph of g, what will push away P from $\mathcal{G}$ is the blending (either desired or not). Let us have a look at two basic cases to get more insight on the gradient behavior during blending :

1. two infinite line skeletons forming a cross (in this case the blending is desired), we will focus on field behavior for points at equal distance from the two skeletons,
2. two parallel line skeletons (in this case some unwanted blending occurs if the two union of balls come too close to each other), we are interested on field behavior in-between the two skeletons,

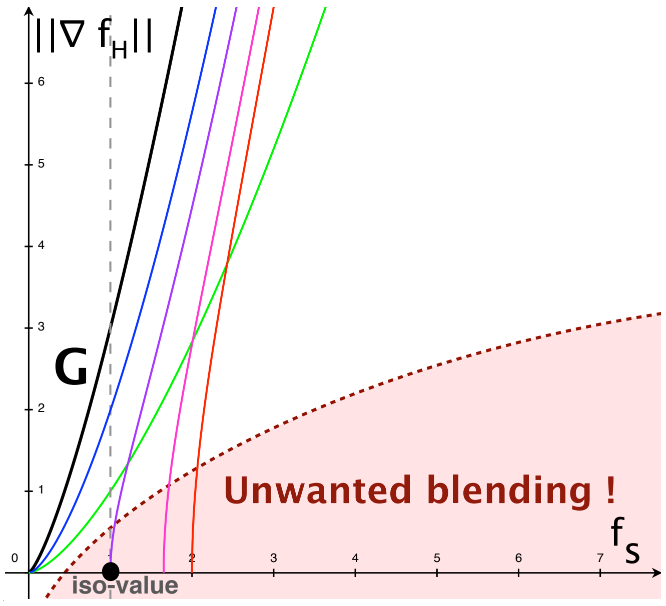


FIGURE 4.14 – *Comparison between the case of an infinite line skeleton (the graph $\mathcal{G}$ in black) with several kind of blending behavior (curves drawn are graph of $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ in some particular cases). The kernel used for the example is the* Inverse *of degree* 3*. First case are the point that lies at equal distance from two line skeleton forming a cross, this is the region where the maximal blending should occur. The second case are the point that lie between two parallel lines (ratio between distance and wanted radius displayed for* 2*,* 2.2*,* $\sqrt{8}$ *and* 20 *: color varying from red to blue), depending of the distance between this skeletons the blending can become problematic. If the two lines are far from one another (blue curve) then blending is not a problem, on the contrary if the union of balls is in tangency (red curve) then there is too much blending (the field is equal to* 2 *instead of* 1*), for the sum, the topology is lost for a ratio of* $\sqrt{8}$ *instead of* 2*.*

in both case, we are only interested in what happen in the plane defined by the two line skeletons. For both cases, curve corresponding to $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$, are drawn in Figure 4.14. We can see that unwanted blending behavior occur in a restricted portion of the 2D plane (basically under the green curve).

For this reasons, our idea is to push back P closer to $\mathcal{G}$ when it is too far from it, this would give us a new point Q. We are only trying to compute a new field value, not the associated gradient, which mean that the new field value will simply be the abscissa of the new point Q :

$$warp(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|) = Q_x.$$

Since we want point $P = (f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ to remain unchanged when it is on $\mathcal{G}$, projection seems a good way to do define the new point Q. Furthermore, it would have more influence when P is far from $\mathcal{G}$ which is the desired behavior.

Various possible correction

There are different basic ways to project P onto the graph $\mathcal{G}$: vertical and horizontal projections, projection in arbitrary fixed direction and projection to the closest point (see Figure 4.15).

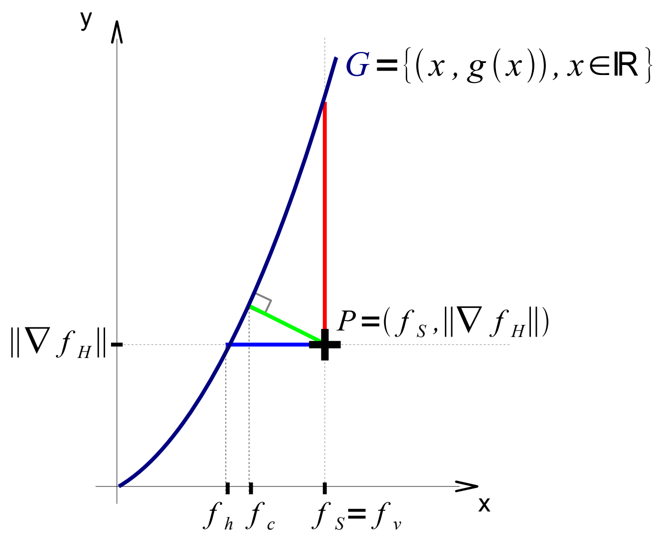


FIGURE 4.15 – *Different strategies for projecting $P = (f_\mathcal{S}, \|\nabla f_\mathcal{H}\|)$ onto the graph of $\mathcal{G}$: vertical projection in red, horizontal in blue and projection onto the closest point of $\mathcal{G}$ in green.*

Vertical projection : One of the simplest ways to pull back P onto the canonical case is to perform a projection on $\mathcal{G}$ in the vertical direction, so the projected point $Q = (q_x, q_y)$ is such that $Q \in \mathcal{G}$ and $q_x = f_\mathcal{S}$ which imply $q_y = g(f_\mathcal{S})$. Thus, we have :

$$Q = \left(f_\mathcal{S},\ (i-1)f_\mathcal{S}^{\frac{i}{i-1}}\right).$$

However, doing so the abscissa of the projected point will be left unchanged which mean that we will keep the same scalar field : this is classical blending by summation of scale-invariant integral surfaces $f_v = f_\mathcal{S}$. The behavior of this blending has already been studied in the previous chapter.

Horizontal projection : The horizontal projection is more interesting : we are looking for $Q = (q_x, q_y)$ such that $Q \in \mathcal{G}$ and $q_y = \|\nabla f_\mathcal{H}\|$, so we should have $q_x = g^{-1}(\|\nabla f_\mathcal{H}\|)$ which in turn give $q_x = h(\|\nabla f_\mathcal{H}\|)$. So, in this case, the projected point is

$$Q = \left(\left(\frac{\|\nabla f_\mathcal{H}\|}{i-1}\right)^{\frac{i-1}{i}}, \|\nabla f_\mathcal{H}\|\right)$$

and the new scalar field is :

$$f_h = \left(\frac{\|\nabla f_\mathcal{H}\|}{i-1}\right)^{\frac{i-1}{i}}$$

which means that the new blending in this case only depends on the scale-invariant gradient. There is no direct dependence to the SCALIS field. However, it is important to note that if the skeleton is a partition of an infinite line the obtained result is the same as SCALIS : the field is only affected wherever there is significant blending. In unwanted blending area, f_h will be smaller than $f_\mathcal{S}$ (P is under the graph $\mathcal{G}$), which will increase the chance to preserve topology. However, as we will see later, this solution would reduce blending too much.

Projection on closest point : Another possible correction is the projection onto the closest point of $\mathcal{G}$, this is equivalent to solve :

$$Q = \underset{x \in \mathbb{R}^+}{\operatorname{argmin}} \left\| \begin{pmatrix} f_{\mathcal{S}} \\ \|\nabla f_{\mathcal{H}}\| \end{pmatrix} - \begin{pmatrix} x \\ (i-1)x^{\frac{i}{i-1}} \end{pmatrix} \right\|^2$$

which in turn is equivalent to :

$$i(i-1)X^{i+2} + X^{i-1} - i\|\nabla f_{\mathcal{H}}\|X^2 - f_{\mathcal{S}} = 0$$

where $X = \sqrt[i-1]{x}$ (as shown in Appendix C.1). This can be solved analytically for kernel degree i up to 3.

As for the horizontal projection, this correction tend to reduce too much the blending.

Projection in some arbitrary fixed direction : This can be seen as an interpolation between the two first projections. We will defined the direction of projection as an angle ψ, 0 being the horizontal projection and $\frac{\pi}{2}$ being the vertical one.

The projected point is at the intersection between the graph $\mathcal{G}$ and the line $\mathcal{D}$ passing through P with the prescribed direction :

$$\mathcal{D} = \{(x, -\tan(\psi)x + \|\nabla f_{\mathcal{H}}\| + \tan(\psi)f_{\mathcal{S}}) \ / \ x \in \mathbb{R}\}.$$

The intersection can be computed analytically for low kernel degree (see Appendix C.2.1).

Practical solution : Analytical solution of the projection cannot be computed for all kernel degree, instead we use an approximation scheme (which is faster and simpler to implement) based on the two first projections.

We compute the projection onto the line passing through the two first solutions instead of the projection onto the graph $\mathcal{G}$ (see Figure 4.16). The precision of this approximation increases as the point P comes closer to the reference case (since in this case the two previous projections are the identity), it also increase with the kernel degree i.

Let l_H being the distance between P and the horizontal projection :

$$l_H = f_{\mathcal{S}} - h(\|\nabla f_{\mathcal{H}}\|)\,,$$

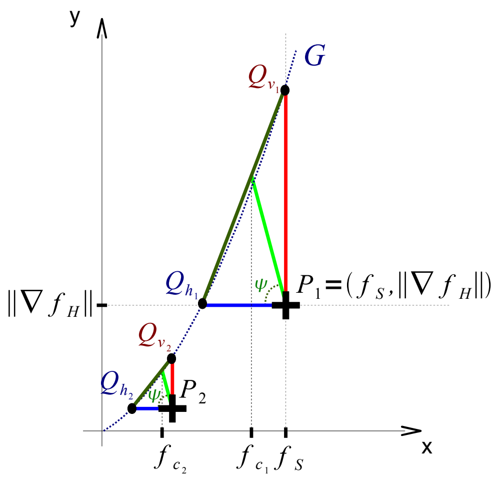


FIGURE 4.16 – *We approximate closest point location by closest point to the line passing through vertical and horizontal projection.*

and l_V being the distance between P and the vertical projection :

$$l_V = g(f_{\mathcal{S}}) - \|\nabla f_{\mathcal{H}}\|\,,$$

then the new corrected scalar field is defined as :

$$f_\psi = f_{\mathcal{S}} - \frac{l_H l_V}{l_V + l_H \tan(\psi)}\,.$$

Details to obtain this expression are given in Appendix C.2.2.

4.3.2 Blending behavior in function of the projection :

In comparison to classical blending by summation (vertical correction), the horizontal projection, and more generally projections in intermediate directions, have two main advantages :

– they help preserving the topology (see Figures 4.17 , 4.18 and 4.19),

– they help preserving shape of thin features (see Figures 4.19, 4.20 and 4.21).

In practice, the correction introduced by the horizontal projection is too important, delaying too much the change of topology (see Figure 4.17) and limiting too much smooth blending between shapes with large difference of radius (see Figure 4.20 and 4.21).

The projection to closest point is equivalent to a projection with a parametrized angle, this angle is lower or equal to $\frac{\pi}{4}$ for field value higher to 0.5 (note that we use an iso-value equal to 1.0) for all *Inverse* kernel of degree higher or equal to 3, which mean that this projection will do the same overcorrection as described above.

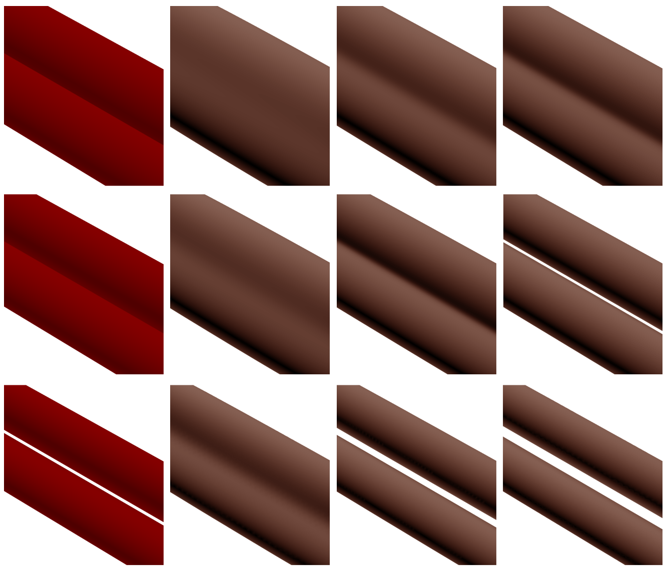


FIGURE 4.17 – *Blending between parallel segments of equal radius. From left to right : union of balls, vertical projection (SCALIS), projection with angle $\psi = \frac{\pi}{4}$ and horizontal projection. The two latter help preserving the topology. However, for horizontal projection, the correction can be too important (which increases the space between the two surfaces or can even postpone too much the change of topology).*

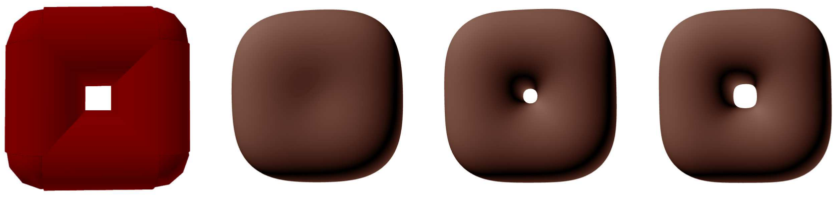


FIGURE 4.18 – *Shape with a hole. From left to right : union of balls, vertical projection (SCALIS), projection with angle $\psi = \frac{\pi}{4}$ and horizontal projection. The two latter help preserving the topology of the skeleton.*

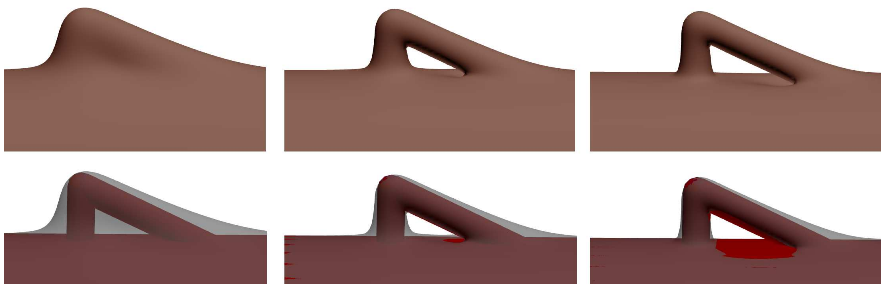


FIGURE 4.19 – *Small handle blended into a larger segment. From left to right : vertical projection (SCALIS), projection with angle $\psi = \frac{\pi}{4}$ and horizontal projection. Top row presents the obtained iso-surface, bottom row presents the union of balls corresponding to the skeleton in red. While the two latter help preserving the topology, they can create creases where the angle between the small segment and larger one is small (in this case the gradient is small).*

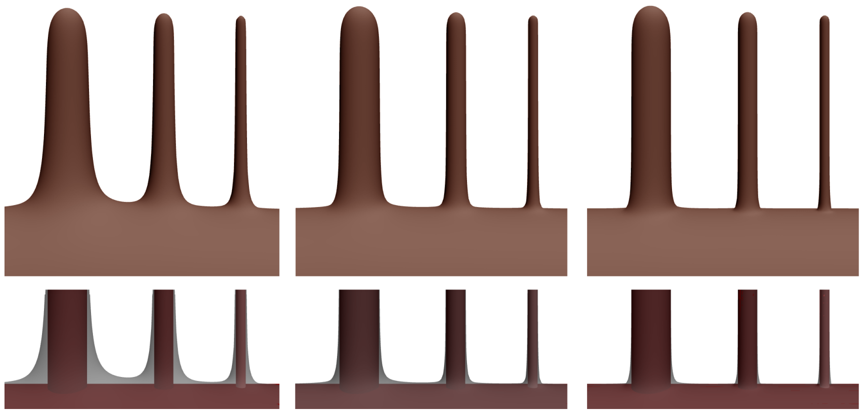


FIGURE 4.20 – *Blending between primitives with a large difference of radius, the vertical primitives have respectively a radius of $\frac{1}{5}$, $\frac{1}{10}$ and $\frac{1}{20}$ of the horizontal one. From left to right : vertical projection (SCALIS), projection with angle $\psi = \frac{\pi}{4}$ and horizontal projection. Top row presents the obtained iso-surface, bottom row presents the union of balls corresponding to the skeleton in red. The two latter projections better preserved the shape of thin features.*

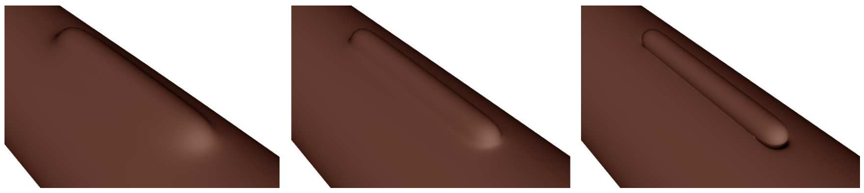


FIGURE 4.21 – *Blending between primitives with a large difference of radius (note that the small segment is placed in tangency with the wanted radius of the larger skeleton, thus the union of balls associated to each skeleton overlap). From left to right : vertical projection (SCALIS), projection with angle $\psi = \frac{\pi}{4}$ and horizontal projection. While the intermediate correction helps better preserving the shape of the thin primitive, the horizontal correction is too important creating a crease in the larger primitive.*

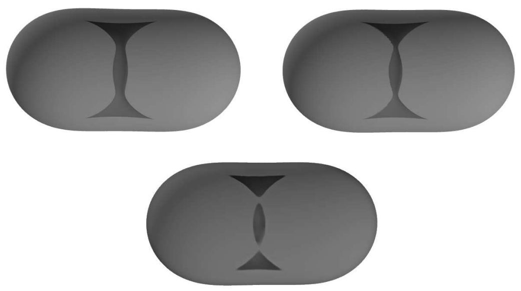


FIGURE 4.22 – *While the horizontal projection and the projection of angle $\psi = \frac{\pi}{4}$ help preserving the topology, they have a major drawback, when the correction is too important : when a shape with a hole is progressively closing, or when two parallel segments come too close to one another, a small crease and then an internal cavity appear.*

Despite some interesting properties, projections with an inclined correction present major drawbacks (it is also the case of horizontal projection) : they can create creases and even cavities. Creases can be seen in Figure 4.19 while cavity is visible in Figure 4.22 (note that it also happens in the case of parallel segments in Figure 4.17). Fortunately, this problem always arise in similar cases : high field values with gradient norm close to zero, which gives hope to correct this drawback by using a direction of projection parametrized by $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$.

Note that the projection angle that provides the best results depends on the degree of the kernel used, for instance for degree 3, best angles are around 0.25π while for degree 4 they are around 0.35π. In Figure 4.23, we compare SCALIS to the gradient-based field warping for projection angle of 0.35π which is close to the required value to preserve topology for *Inverse* kernel of degree 4 if there wasn't the handle problem. We can note that the topology of the resulting shape is much closer to the one of the skeleton whatever the desired radius. Furthermore, shapes still smoothly blend when they come closer to one another and overlap.

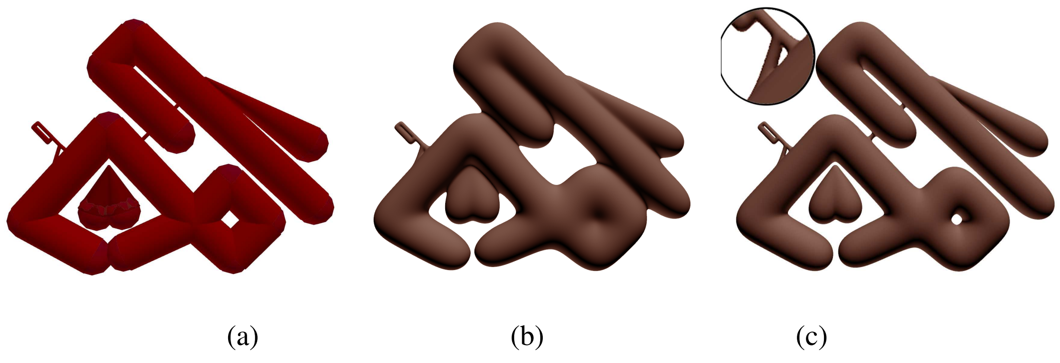


FIGURE 4.23 – *Comparison between blending by summation and projection with angle $\psi = 0.35\pi$ for* Inverse 4 *kernel. We can note that the obtained shape is much closer to the skeleton in (c) than in (b). It almost preserves the topology of the union of balls, except at extremities (note that we have not applied the correction method presented in previous chapter).*

4.3.3 Extension to skeletons of other dimensions

While we have applied our correction methods to one dimensional skeletons (i.e. segments), it is possible to do the same thing for skeletons of other dimension (point and triangle). Furthermore, provided that the kernel is chosen according to the dimension, it is possible to simply sum the fields of all primitives independently from their dimension, and apply the correction afterward.

The condition is that the canonical case for the pair kernel/dimension are the same function of the distance. For instance if a kernel of degree i is used for segments then a kernel of degree $i-1$ should be used for points, and degree $i+1$ for triangles. Result for points skeletons and segment skeletons used together are illustrated in Figure 4.24.

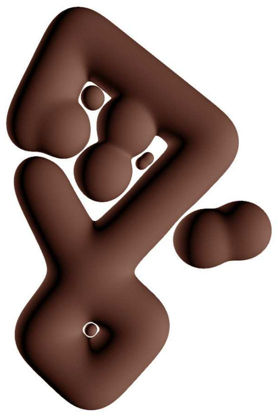


FIGURE 4.24 – *Point with* Inverse *kernel of degree* 3 *and segment with* Inverse *kernel of degree* 4 *are used at the same time.*

4.3.4 Discussion and Future improvements

Since our method is only based on a summation blend, it does not depend on the skeleton subdivision, so it preserves the main property of integral surfaces. In order to perform a test in a more complex case, we have modeled a dragon with large difference of radius between the primitives used (see Figure 4.25). Our method better preserves the topology than SCALIS with the same kernel but also than *Compact Polynomial* kernel.

It is important to note that since our operator is based on a summation blend, it should help to use efficient caching system for modeling, indeed if we store the field before correction, we just have to subtract the contribution of the skeleton under modification to the cache and add the new field value instead of recomputing all the field value in the modified area.

While the first results seem promising, there are still a number of improvements which would be needed for our method to be usable in practice.

Analysis of the required behavior and parametrization of the correction

The handle problem previously described is probably the main problem to be solved. Indeed, creating creases and/or central cavities is problematic in the case of two shapes close to being tangent to each other (see Figure 4.26). It seems to be problematic only in the region where the gradient is both small, and small in comparison to the field value.

Because the problem seems to appear in particular cases, we think it is possible to correct it by introducing a projection method parametrized by the couple of values $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$. In order to do so, one should study the case of two parallel lines.

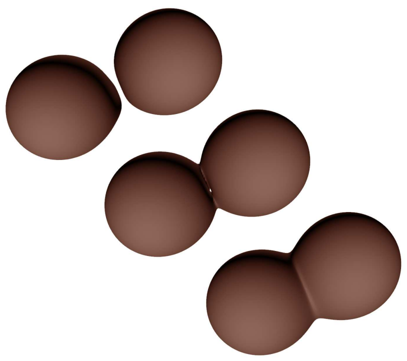


FIGURE 4.26 – *Two parallel segments with distance (normalized by primitive radius) respectively equal to* 2.1, 1.9 *and* 1.7. *We can see that the handle problem creates an awkward blend in the middle image.*

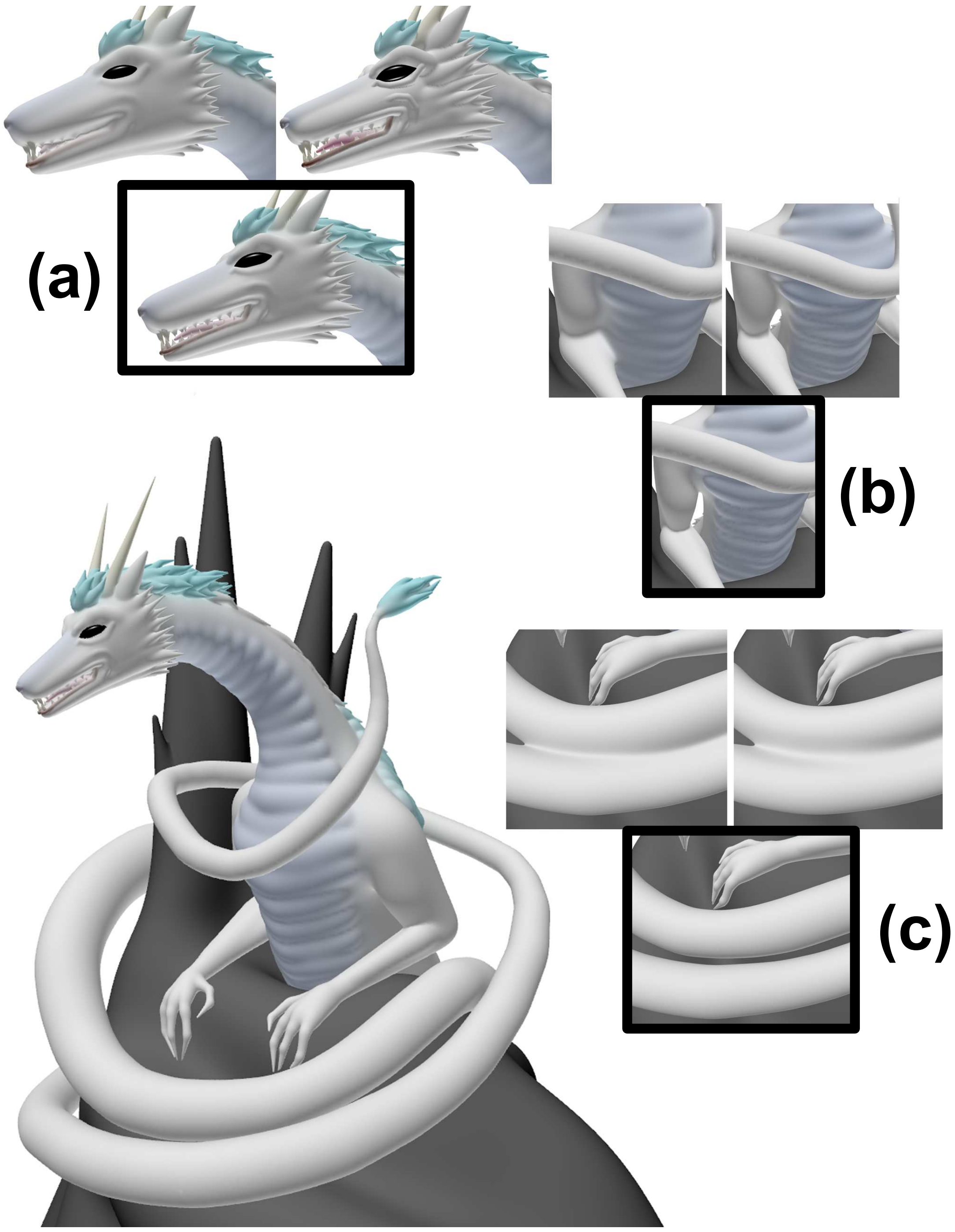


FIGURE 4.25 – *Dragon created with only two Blobtree nodes : a max operator is used to combine the eyes with the rest of the dragon which is created from a single gradient-based field warping (used with* Inverse 4 *kernel). We present close-up with comparison to standard blending by sum for the same kernel (upper left vignette) but also for* Compact Polynomial *kernel of degree* 6 *(upper right vignette), the gradient-based field warping is presented in the framed bottom central vignette. Note that gradient-based field warping is the only method that keeps the topology in (c).*

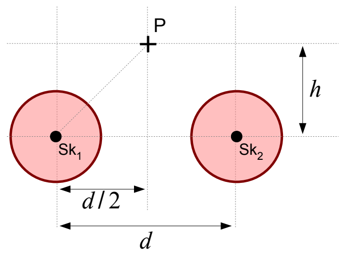


FIGURE 4.27 – *Base case to study in order to adjust blending : two parallel lines of unit radius (shown here in cross-section). We are mainly interested by the field behavior at equal distances from the two skeletons.*

There are two important properties that should be verified by the parametrization of the projection :

1. the field defined in the plane at equal distance from the two lines should be a decreasing function of h whatever the distance d between the two lines (see Figure 4.27).
2. the field value at equal distance of the two segments for $h = 0$ should be a decreasing function of the distance between the two segments, and it should be equal to 1 in for $d = 2$ (which is the case of tangency).

Of course, in order to limit the computation cost, the function we are looking for should return $\tan(\psi)$ and not the direction ψ itself.

The first thing to note is that for a given value of $\|\nabla f_{\mathcal{H}}\|$, the projection in a fixed direction already implies that the blending is a decreasing function of the field value $f_{\mathcal{S}}$ (at least for the analytical projection).

Furthermore, Figure 4.28 highlights two important facts. First, it explains the handle problem : it is normal to obtain it in the case of a projection in a fixed direction. Indeed, in the base case of Figure 4.27, for a given distance d between two parallel line-skeletons, there are two points with different values of h that have the same projection on $\mathcal{G}$. Secondly, the different graphs of $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ for different distances d between the line primitives do not cross each other (this is proved in Appendix C.3). This means that it is possible to respect the first constraint we previously described (preventing the appearance of the handle problem).

The parametrization of the correction that would meet our constraints is probably not unique which means that we can have some freedom on the blend we will obtained (more or less sharpness).

An important thing to note is that our field warping correction will not be able to exactly keep the topology of the union of balls in both the case of parallel lines and in the case of a circle skeleton. In both cases, the point where the topology should change (the one where the union of spheres is made of tangential parts) has a null field gradient. In the case of the circle skeleton the field value will be

$$f_{circle} = \frac{1}{i-1}(2\pi\tau)\left(\frac{\tau}{\tau}\right)^i \frac{1}{\tau} = \frac{2\pi}{i-1},$$

whereas for two parallel line skeletons the field value is equal to 2. None coincidence of this two values is the reason why topology cannot be guaranteed at the same time in both cases. However, note that for the *Inverse* kernel of degree 4, the field values would not be that far from one another (the ratio between them is around 1.04) which means that the topology in the circle case would be preserved much longer than when using a sum.

Compact support

Using kernels of local influence, which can be either provided by compact support kernels or by Witnessed blending, is a necessity for both efficiency and local shape control.

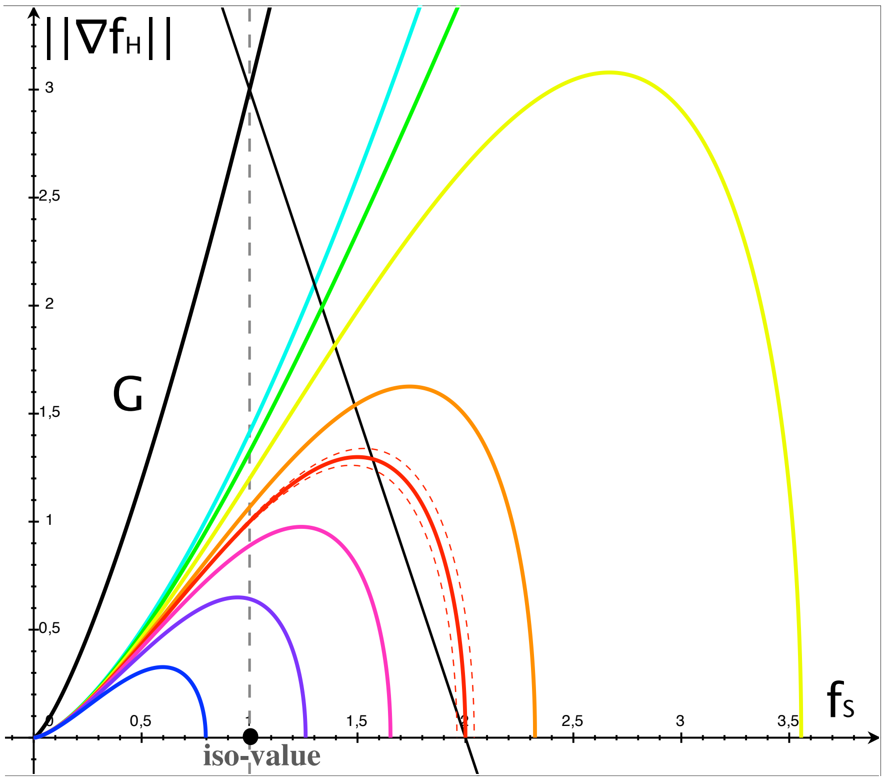


FIGURE 4.28 – *Each colored graph correspond to the graph of* $(f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ *for the blending of two line skeleton in the plane situated at equal distance of each skeleton from a distance of* 0 *in cyan to a distance of* 3.16 *in blue (the equation of the curves are given in Appendix C.3). In red is the distance of* 2 *which corresponds to the tangential case between the two shapes to be bended : the inclined line leaving from* $(2, 0)$*, is the required direction of projection in order to obtain exactly the wanted iso-value in the point of null gradient in the tangential case. This figure emphasizes the facts that make us think that there exists a correction corresponding to our constraint to obtain a good blending : the colored graphs do not cross one another which means that we do not have contradictory constraint.*

The first thing to note is that the behavior in a local neighborhood of the skeleton is not that important if one wants to later convert the field into a field with an inner bound [CGB13]. Indeed, in order to do so, a transfer function should be used that will map all values above a given threshold to a constant value (so field with infinite value on the skeleton are not so much problematic provided that the generated field is converted before applying operations such as digging).

The main difficulty in the application of the gradient-based field warping method to classic local support kernels, is that they have a null gradient on the skeleton and they present an inflection point, thus the inversion strategy we used cannot be applied anymore. We see three main kinds of solutions to overcome this problem :

– The first one is to choose a classic *Compact Polynomial* kernel with parameter such that inflection point of the scalar field is located far enough inside the iso-surface, so that the relationship between normalized gradient and scalar field is invertible in the region where the unwanted blending is problematic. Furthermore the method should be applied with a polynomial of high degree in order to have a smooth gradient. A degree of 8 with

a σ value of 1.8 should meet the previous constraints. The main difficulty will be to find the relationship between $f_{\mathcal{S}}$ and $\|\nabla f_{\mathcal{H}}\|$ in the interval where it is invertible (or use pre-computation).
– An alternative solution would be to introduce a new kernel which would be a product of an *Inverse* kernel and a *Compact Polynomial* one. This would ensure both a compact support and the absence of inflection point.
– The last alternative would be to continue using an *Inverse* kernel in combination with either a Witnessed blending to compute the field to be analyzed (provided the future work on smooth clipping gives good results, see section 4.2.4) or to use a fixed (and smooth) clipping around the computation point (which would probably be the easiest solution to implement).

Closed-form gradient

We do not have yet closed-form expressions for the gradient of the warped field. In order to compute it, we should compute the gradient of $\|\nabla f_{\mathcal{H}}\|$: a Hessian which could be computed efficiently since it as a lot of term in common with the gradient. However, the main difficulty is probably to compute the part of the gradient that depends on the correction, we think that the easiest way would be to rely on precomputed partial derivatives as it was done for gradient blending.

4.3.5 Conclusion

We have presented a blending method based on summation blend followed by field warping. It relies on the analysis of a summation blend in order to provide a better topological behavior. Since the method only relies on a sum, it is intrinsically independent from the skeleton subdivision. While the first results are promising, some work still has to be performed in order to make the method really usable in practice.

In addition to the improvements already discussed, it would be interesting to introduce a family of projections described by a single parameter (which would describe the sharpness of the required blending). We could then set this parameter as a skeleton attribute and interpolate it on the fly during the evaluation in order to obtain a blending with a sharpness directly controlled by the skeleton without making the Blobtree more complicated.

4.4 Conclusion

In this chapter, we have presented two approaches to improve the blending behavior of integral surfaces while preserving their independence to skeleton subdivision. They rely on two very different strategies : the first one is a skeleton based method which selects a restricted part of the skeleton for each field query, while the second one is a gradient-based method combining the sum with some adequate warping of the resulting scalar field. Both methods show very promising results such as reducing the blur of smooth details and improving topology preservation. However, both still require some work to be really useable in practice.

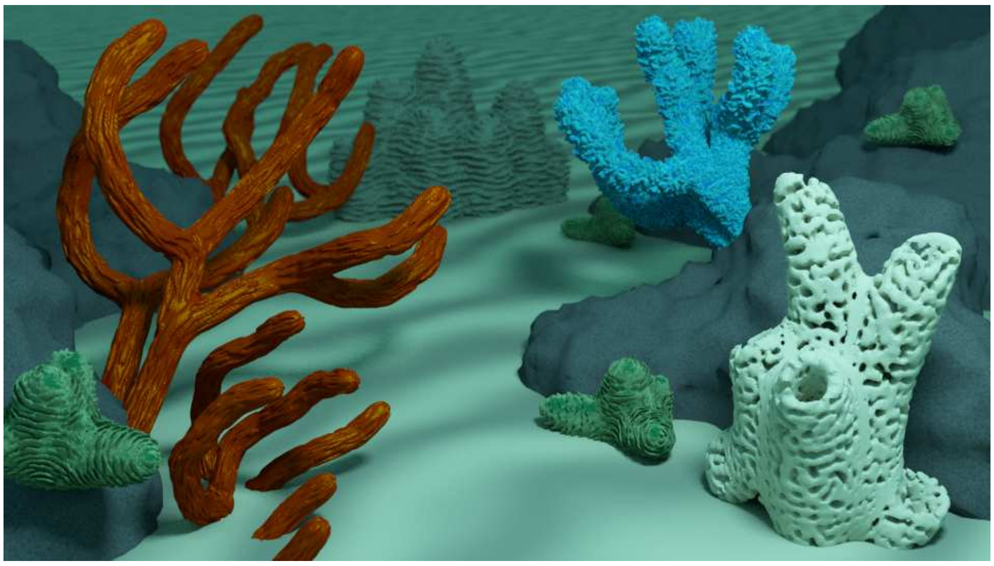
It is important to note that these methods do not aim at replacing complex binary operators such as gradient blending, since they already provide much more control. But being N-ary, they enable to limit the use of such binary operators to the few cases when they are really required : this would greatly reduce the height of the Blobtree, making it much easier to modify.

Our insight is that combining the two methods we presented with some skeleton analysis, such as proposed in section 3.6 (in order to reduce the bulge) would give the best results in term of quality. However, from a computational point of view, it would probably be better to focus only on gradient-based integral surfaces, while using skeleton analysis to reduce bulges.

Lastly, it would be interesting to find a way to add a sharpness parameter on the skeleton in order to control the smoothness of the blending.

CHAPTER

5 GEOMETRIC DETAILS ON SKELETON-BASED IMPLICIT SURFACES



Teaser Figure : *Implicit modeling of a coral reef. All small scale details have been added with our method.*

Publication :

ZANNI C., BARES P., LAGAE A., QUIBLIER M., CANI M.-P. : Geometric Details on Skeleton-based Implicit Surfaces. In *Eurographics 2012 : Short Paper* (Cagliari, Italie, May 2012).

As we have seen in part 1.4, implicit surfaces have long been restricted to the modeling of smooth shapes, since most methods for adding geometric details require some local parameterization. As stressed by Sherstyuk [She99c], the two main ways to enhance implicit surfaces with geometric details are either to add small implicit primitives near the surface, or to deform the scalar field with volumetric noise such as hypertextures [PH89]. In the first case, in addition to the burden of adding many primitives by hand, small details may blur when blended, requiring the use of costly blending mechanisms instead of the sum [BBCW10]. In the second case, controlling the distribution and orientation of details on the surface would be difficult due to the volumetric nature of hypertextures, and small disconnected components could appear near the main surface.

In this chapter we introduce the first extension of constructive implicit modeling to surfaces with procedural geometric details. Our method enables the addition of well-distributed anisotropic details over implicit primitives, provides an intuitive control of their orientation from the skeletons that define the primitives, and contrary to [BIT04], extends the blending properties of implicit surfaces without causing the details to blur. Although presented in the context of skeleton-based implicit surfaces, our method is applicable to other implicit models as well, but the user then needs to define some skeletal abstraction for the main shape feature.

5.1 Background : Anisotropic Gabor noise on surfaces :

Our method for generating details on implicit surfaces is based on Gabor noise [LLDD09]. This is a sparse convolution noise that presents several interesting properties : accurate spectral control, anisotropic noise, non-periodicity and lastly, it provide high quality filtering (which is important for texturing). In addition to these qualities, it also has some practical advantages : small memory footprint and fast evaluation.

The most basic form of Gabor noise, 2D anisotropic Gabor noise, is defined as

$$n\left(\mathbf{x}; K, a, F_0, \omega_0\right) = \sum_i w_i g\left(\mathbf{x} - \mathbf{x_i}; K, a, F_0, \omega_0\right), \tag{5.1}$$

where the noise parameters K, a, F_0 and ω_0 control the amplitude, bandwidth, frequency and direction of anisotropy of the noise. The random weights $\{w_i\}$ are distributed according to a standard uniform distribution and the random positions $\{\mathbf{x_i}\}$ are distributed according to a Poisson process with mean λ. The Gabor kernel g is defined by :

$$g(x, y; a, F_0, \omega_0) = Ke^{-\pi a^2(x^2+y^2)} cos[2\pi F_0(x\cos\omega_0 + y\sin\omega_0)].$$

By using random direction of anisotropy ω_0, an isotropic noise is obtained ; this can also be done more efficiently by using a modified noise kernel [LLD11]. In Figure 5.1, we can see

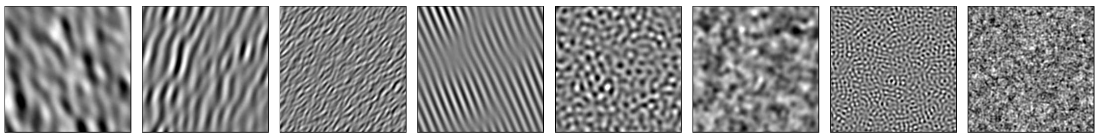


FIGURE 5.1 – *Various noise patterns generated with different noise parameters, both anisotropic noise and isotropic one are presented. Figure from [LLDD09].*

various examples of noise (both isotropic and anisotropic) obtained using different set of parameters. In practice Gabor kernels are truncated in order to be generated on the fly inside a grid which cell size r is chosen according to noise properties.

Note that this 2D Gabor noise naturally extend to solid (3D) noise which can be used to texture surfaces as if they were carved into a material. However, the extension that has the main interest in our case is Surface Gabor noise.

Surface Gabor noise : This generalization of 2D Gabor noise to surfaces provides a way to map a noise onto a surface while preserving its 2D pattern which is not the case of solid noise.

In order to obtain surface noise, noise kernels are locally projected onto the surface. This is done by projecting the 3D distribution $\{\mathbf{x_i}\}$ onto the plane defined by the query point and its normal ; note that only points $\mathbf{x_i}$ in a local neighborhood (a cylinder of radius r and height $2r$) are projected. Instead of generating the random weights $\{w_i\}$, weights are computed thanks to the distance d from $\mathbf{x_i}$ to its projection using the formula $w_i = \frac{1-d}{r}$.

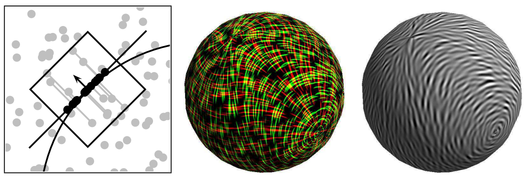


FIGURE 5.2 – *(a) Creation of a local kernel distribution by projection onto the local tangent plane, (b) frame field used to orientate noise anisotropy, and (c) resulting anisotropic surface noise. Figure from [LLDD09].*

Surface Gabor noise is independent of the geometric representation of the surface. Evaluating it only requires reference frames to be defined over the surface to express the direction of anisotropy. In order to keep good properties of the noise, texture details should be small in comparison to geometry detail.

While we will use this version of Gabor noise, it is important to note that they exist several extensions, such as noise with controllable band-limits, which can be combined to weighted sum of noises in order to obtain more complex noise patterns.

5.2 PROCEDURAL NOISE ON SKELETON-BASED IMPLICIT SURFACES

Let $\mathcal{S}$ be the implicit surface to be enhanced with details and $f_{\mathcal{S}}$ be the scalar field that defines it. $f_{\mathcal{S}}$ is generated thanks to a set of skeletons $Sk = \{Sk_i\}$ generating individual fields f_i. This section explains how we define a smooth, anisotropic Gabor noise on $\mathcal{S}$ that seamlessly follows this geometry.

Smooth vector field :

Generating a surface Gabor noise requires the definition of continuously varying local frames over the surface. To reduce user input, we provide a method for automatically generating them from a meaningful spatial vector field, defined as a weighed sum of the skeleton directions. Since the skeletons control the shape of $\mathcal{S}$, this ensures that the frames will coherently follow the surface.

Let us develop the method when the skeleton $\{Sk\}$ defining $\mathcal{S}$ is a set of segments. Similarly to the way smooth field values are computed from skeletons in convolution surfaces, we generate a smoothly varying vector field by integrating the directions $\mathbf{Sk}(s)$ of the skeleton, weighted by the field contribution of the associated infinitesimal arc length :

$$\mathbf{Wd}_{\mathbf{Sk}}(P) = \int_{s \in Sk} K(d_{Sk}(s,P)).\mathbf{Sk}(s)\,\mathrm{d}s. \tag{5.2}$$

The additivity of the integral leads to an additive property for the resulting vector field :

$$\mathbf{Wd}_{\mathbf{Sk}}(P) = \sum_i \mathbf{Wd}_{\mathbf{Sk}_i}(P) \tag{5.3}$$

Moreover, the direction $\mathbf{Sk}_i$ being constant on a single segment, equation (5.2) combined with the formula of convolution (equation 1.10)), yields :

$$\mathbf{Wd}_{\mathbf{Sk}_i}(P) = f_i(P).\mathbf{Sk}_i \tag{5.4}$$

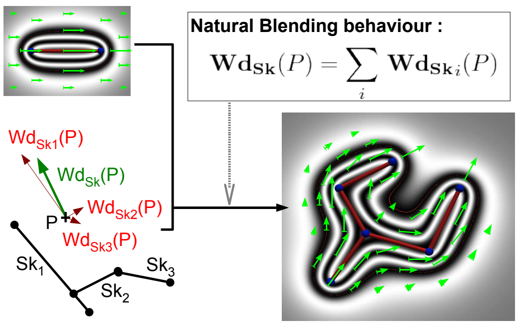

Natural Blending behaviour :
$\mathbf{Wd}_{\mathbf{Sk}}(P) = \sum_i \mathbf{Wd}_{\mathbf{Sk}_i}(P)$
$Wd_{Sk1}(P)$
$Wd_{Sk}(P)$
$Wd_{Sk2}(P)$
P
$Wd_{Sk3}(P)$
Sk_1
Sk_2
Sk_3

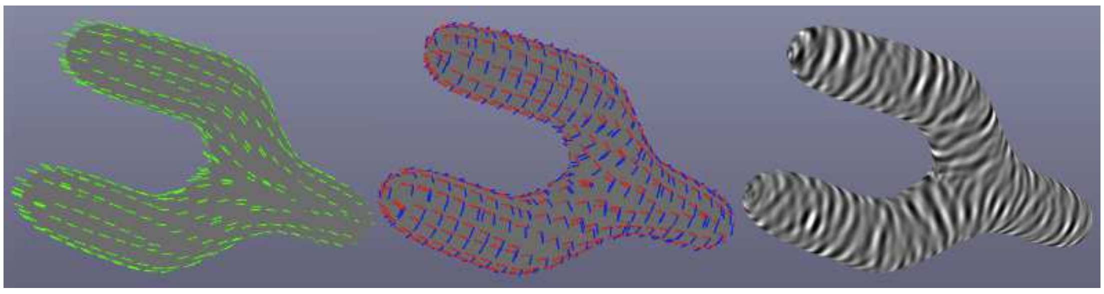
FIGURE 5.3 – *Weighted direction naturally lend themselves to blending by summation.*

This results into a continuous space vector field $\mathbf{Wd}_{\mathcal{S}}(P)$ that encodes the shape of $\mathcal{S}$, as depicted on Figure 5.4(a). Note that we can easily adapt the definition of the weighted direction to all kind of integral surfaces (such as Hornus integral surfaces and SCALIS).

We extend the definition of $\mathbf{Wd}$ to all kinds of implicit primitives by using equation (5.3) and (5.4), replacing $\mathbf{Sk}$ by $\mathbf{Sk}(P)$, a meaningful direction which may vary in space, defined from the primitive's skeleton. For instance, for triangles we use any vector included in its defining plane ; note that the orientation as well as the direction defined for each skeleton affects the resulting vector field.

The implicit primitive is now associated with two smooth vector fields : the field gradient (defining the normal $\mathbf{N}$) and the weighted direction $\mathbf{Wd}_{\mathcal{S}}$ we just defined. We use them for computing smoothly varying local frames $(\mathbf{n}, \mathbf{t}, \mathbf{b})$ (normal, tangent, bitangent) over the surface by setting :

$$\{\mathbf{n} = \mathbf{N}(P),\ \mathbf{b} = \mathit{normalized}\,(\mathbf{n} \wedge \mathbf{Wd}_{\mathcal{S}}(P))\,,\ \mathbf{t} = \mathbf{n} \wedge \mathbf{b}\}$$



(a) (b) (c)

FIGURE 5.4 – *(a) weighted skeleton direction, (b) procedural frames, (c) resulting anisotropic Gabor surface noise.*

The resulting local frames are depicted on Figure 5.4(b) ; in the case of tubular shapes, $\mathbf{t}$ tends to be in the same direction as the weighted direction. In practice, we generate the frames efficiently using closed-form formula for convolution field gradients, derived from closed form-formula for field function [HC12]. Note that our frame field is not only coherently defined on the surface, but also in the surrounding space ; we will use this for another purpose in section 5.3.

Anisotropic surface noise :

Once the frames are defined, they are used to generate surface Gabor noise as explained in Section 5.1 (equation 5.1). The parameters defining the noise are the noise magnitude K, the bandwidth a, and the orientation with respect to the local frame ω. For instance, we can define noise that follows the main orientation of the surface, that is orthogonal to it or that progressively swirls around it. Such anisotropic Gabor surface noise is depicted on Figure 5.4(c). It can directly be used for texturing an implicit surface.

5.3 Generating geometric details from surface noise

We now explain how we generate geometric details over an implicit surface from the noise we just defined. We assume that the details to be added are small compared to the geometric features of the initial shape, i.e. that there is enough room to add them in concave regions.

Basic method :

The goal is to use the noise for locally deforming the scalar field that defines the input implicit surface. To save computational time, we only apply this deformation in a local neighborhood of the iso-surface of interest. Let the interval $[iso_d^-; iso_d^+]$ define the region, chosen from the size of details, where the field is to be deformed (figure 5.5).

As we would have done with displacement mapping, the noise value on the surface will be used to quantify the deformation. Thus, to deform the field around the surface, we must associate to each point in space a point on the surface, in order to get the amount of deformation.

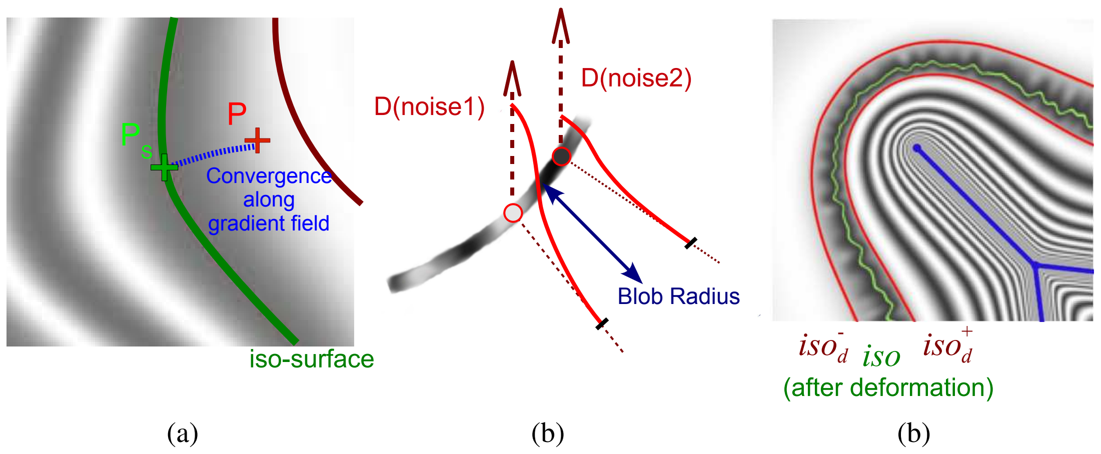


FIGURE 5.5 – *Principle of the method : (a) projection on surface to get noise value, (b) the noise value is use to modulate the influence of a blob, (c) in red : iso-surfaces $f_S = iso_d^-$ and $f_S = iso_d^+$ defining the deformation region.*

To compute this association in a coherent way, we project the evaluation point P on the surface along stream-lines of the field, defined by following the field gradient ; this was also used in [ZGVdF97], but here the aim is to obtain a space/surface mapping. This gives us both a point $P_{\mathcal{S}}$ on the surface (with the associated noise value) and a distance to it. The scalar field $f_{\mathcal{S}}(P)$ of the input surface is then modulated as if there was a blob centered at $P_{\mathcal{S}}$. The blob intensity is set from the noise value. Note that we also apply a cancellation factor to the deformation factor in order to guarantee to have no influence outside of the range $[iso_d^-; iso_d^+]$.

Note that projecting space points along streamlines insures that the resulting details remain coherent in concave areas (neighboring details do not intersect nor blend). Moreover, the method can be used both for adding material and for carving the input surface, depending on the positive vs. negative value of the noise.

Generating free-form and tilted details :

Our method provides easy ways to control the shape and orientation of details, since each step of the algorithm below can be adequately parameterized : First, one can play on the noise parameters to obtain a variety of details, the main parameters for the user are the frequency, the bandwidth and the direction of anisotropy for the version of Gabor noise we are currently using (band-limited Gabor noise would give some additional parameters and the possibility to interpolate between anisotropic and isotropic noise). Furthermore, both the blob radius and the deformation function D (that gives the blob height in function of the noise value) can be parametrized in order to obtain a greater variety of details (for the influence of the function D we can refer to Figure 5.6). Just note that the blob radius should be chosen in a coherent way with the range of field value that are affected by the noise.

The algorithm for computing the new field value $f(P)$ is thus :

$$\textit{If } f_{\mathcal{S}}(P) \notin [iso_d^-; iso_d^+] \textit{ then } f(P) = f_{\mathcal{S}}(P).$$
$$\textit{Else :} \left[\begin{array}{l} let\ P_{\mathcal{S}} = proj(P, \mathbf{v}_{\alpha,\theta}), \\ let\ noise = n\left(P_{\mathcal{S}}; k, a, \omega\right), \\ f(P) = f_{\mathcal{S}}(P) + D(noise).K\left(\frac{\|PP_{\mathcal{S}}\|}{R}\right). \end{array} \right.$$

where (k, a, ω) are the noise parameter on the surface, $\mathbf{v}_{\alpha,\theta}$ is the direction of projection, D is a map that gives the amount of deformation from the noise value and R is the blob radius. We use $K(r) = (1 - r)^6$ to define the blob's contribution.

Using an arbitrary direction $\mathbf{v}_{\alpha,\theta}$ of projection from P to $P_{\mathcal{S}}$ instead of projecting P along streamlines enables us to tilt geometric details with respect to the normal of the main surface,

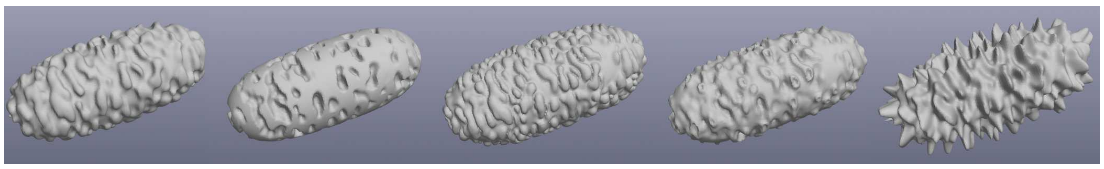


FIGURE 5.6 – *Different kind of details obtained from the same noise by playing only on the deformation map D. We currently use a cubic spline by part in order to define this function.*

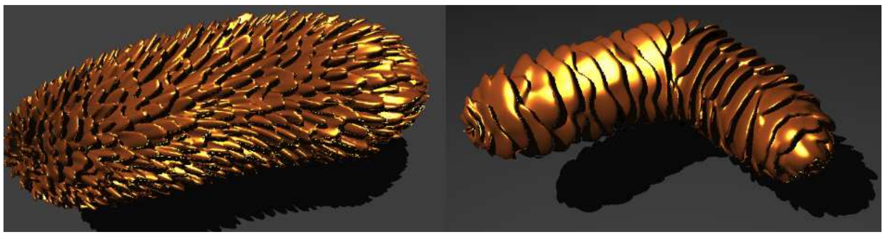


FIGURE 5.7 – *Effect of using a tilted direction of projection. The direction of projection is parametrized in the local frame defined from the weighted direction : here it is the median direction between tangent vector* $\mathbf{t}$ *and normal vector* $\mathbf{n}$.

leading to the generation of details that are not mere high fields (see figure 5.7). $\mathbf{v}_{\alpha,\theta}$ is defined within the frame computed in previous section.

Blending implicit surfaces carrying details :

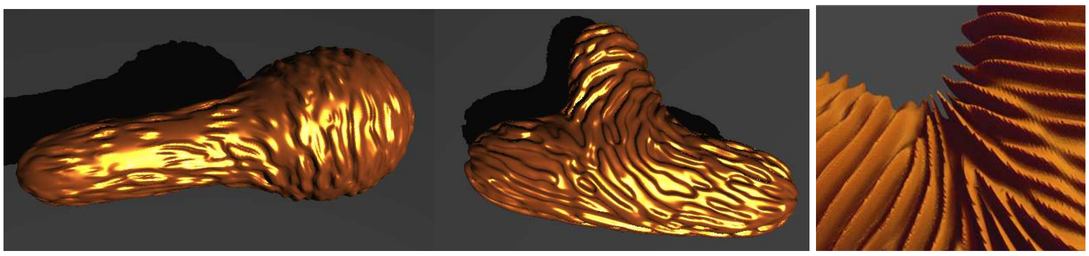
The parameters defining geometric details can be set to vary from an implicit primitive to the next. During blending, these parameters are smoothly interpolated according to the respective field contribution of each primitive at the evaluation point, or at the projected point for noise parameters :

$$param(P) = \frac{\sum_i f_i(P).param_i(P)}{\sum_i f_i(P))}.$$

This leads to a smooth change of detail orientation and shape in blending regions. Note that only scalar parameters can be interpolated this way ; yet, all the parameters of base Gabor noise are scalars. This would not be the case anymore if one wants to use a combination of Gabor noises, then a more complex interpolation scheme should be found.

5.4 RESULTS

As our results show, the method generates details that behave nicely in concave regions (Figure 5.8(c)), instead of self-colliding as when displacement mapping is used : this is due to the continuity of the gradient of the scalar field used which will guarantee that all the "bumps" will be preserved in the concave region. Moreover, when implicit surfaces blend, the noise values are smoothly interpolated, resulting in a natural behavior of details ; when details have



(a) (b) (c)

FIGURE 5.8 – *(a,b,c) Smooth changes in details direction.*

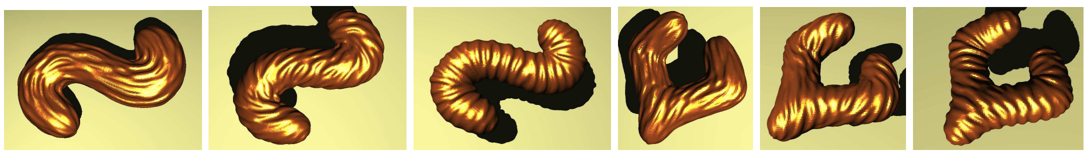


FIGURE 5.9 – *Examples emphasizing the variety of choices that can be made for details orientation, all coherent with input shape.*

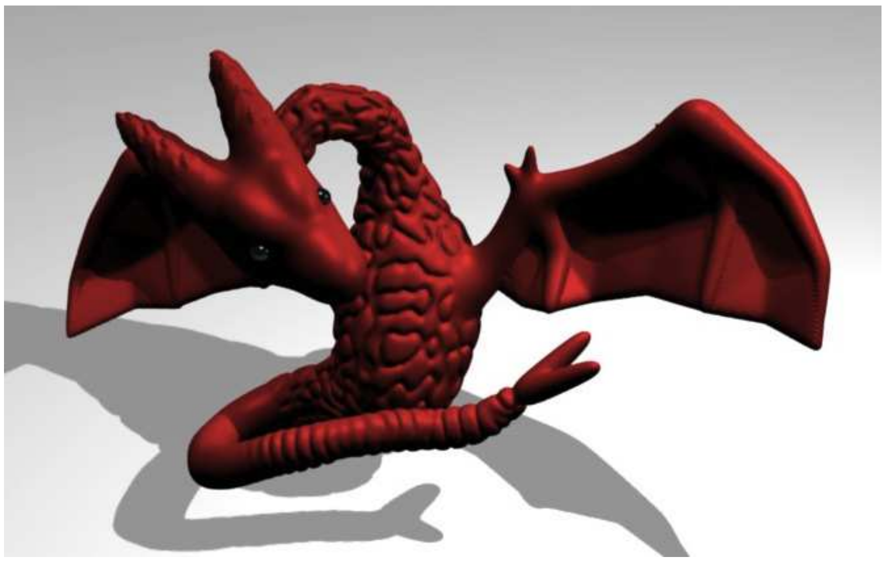


FIGURE 5.10 – *Dragon model showing the variety of details that can be generated.*

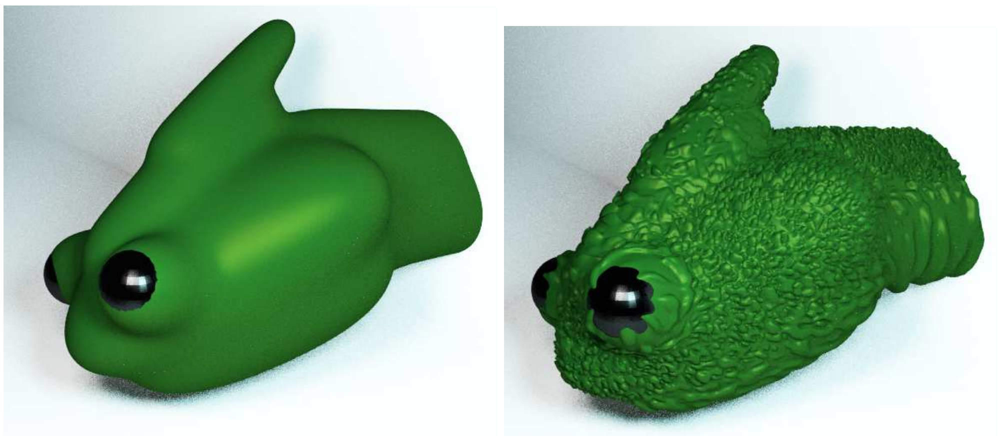


FIGURE 5.11 – *Object before and after applying details*

different orientations on the initial surfaces, their orientation smoothly varies in the blended regions of the resulting shape (Figure 5.8(a,b)). Note that details variation depends on the range of influence of the fields : for instance, the T-junction in Figure 5.8(b) affects the orientation of details on the other side of the main cylinder, which would not be the case using sharper field functions. More complex objects enhanced with details are depicted in Figure 5.10 and 5.11.

Limitations :

The complexity of the details we can generate is limited by the range of patterns that can be obtained through Gabor noise. Moreover, generating frame field without any singularity is not possible on some surfaces. In our case, singularities arise when $\mathbf{Wd}_{\mathcal{S}}(P)$ and $\nabla f_{\mathcal{S}}(P)$ are collinear. Fortunately, computing the scalar product between these two directions gives us a distance to singular point, enabling us to locally cancel the noise. A solution would be to use isotropic surface noise in this region, which can handle singularities.

Computational efficiency :

Timings are given in Table 5.1 ; due to the fact that migration along stream-lines can be computationally expensive, we replaced it in practice by a projection in the initial gradient direction. From our experiments, this solution is much more efficient and still leads to good results due to the small size of details with respect to the variation of the main field function. There is room for a lot of improvements in the performances of our method, from an efficient caching system to the derivation of closed-form equations for the gradient of the part of the field that represents details.

	Triangles in Meshes	Computational Time
Coral reef - (average per mesh)	326 920	102.3s
Dragon (figure 5.10)	226 828	88.4s
Figure 5.7 - (average per mesh)	285 622	90.8s
Figure 5.8(a,b) - (average per mesh)	125 174	9.7s

TABLE 5.1 – *Computation time on a 2.4 GHz Intel Core 2 Duo (only one core used).*

5.5 CONCLUSION

We have presented one of the first methods for extending constructive implicit modeling to surfaces with geometric details, based on Gabor noise. Note that our method can be adapted to non-skeleton-based primitives, provided that one or several reference axis are set to define their global orientation. In addition to solving the problems listed in the discussion of results, our future work will focus on improving the way details are defined : First, sketching the profile of details should be a fast and intuitive way to set their parameters. Secondly, detail parameters could be stored along the skeleton, which would enable them to vary over individual implicit primitives.

CHAPTER

6

CONCLUSION

THE WORK presented in this thesis can be divided into two main types of contributions to implicit modeling : firstly, widening the kind of shapes that can be modeled ; secondly, improving the blending behavior.

We have introduced a new implicit primitive with convolution-like properties in order to ease the creation of more complex shapes : an helical primitive from warping. It is partly based on the classical twisting operator. However, we modified it in order to reduce the deformation of the cross section of the base primitive. This new primitive should make the creation of surfaces with free-form skeletons easier.

Lack of details is one of the main problems of implicit surfaces. In order to remedy this drawback, we have introduced geometric procedural surface details. This method is based on the use of Gabor noise, providing a large choice of detail patterns. A coherent orientation of the details with respect to the underlying shape is obtained by using the skeleton of the implicit surface to parametrize the details. In addition to defining geometric details, this can also be used to texture implicit surfaces.

Our second aim was to improve the blending behavior of skeleton-based implicit surfaces while preserving independence to skeleton subdivision, which is the main benefit of convolution surfaces. The first step was the introduction of a new model to generate surfaces with varying thickness along their skeleton : SCALe-invariant Integral surfaces (SCALIS). As their name suggests, one of the main properties of this surface is to obtain scale-invariance of the generated shape leading to more intuitive behavior of the model. SCALIS enable radius to be easily prescribed around skeleton and it also improves blending behavior of small details. Furthermore, it enables an intuitive use of kernel with compact support that are required when working with complex skeletons.

The second step we presented to improve blending behavior builds upon the properties of SCALIS that make the behavior of the model easier to analyze. We present two different approaches in order to improve blending with a main focus on topological behavior, toward having an implicit shape that always reflects the topology of its skeleton. Our aim was, while doing so, to keep the property of independence from skeleton subdivision, which was never achieved so far. The first method is a skeleton-based approach that selects a subset of skeleton depending on the position of the query point : the Witnessed blending. While it does improve behavior of the blending, it does not provide guarantee on the topology of the resulting shape. On the opposite, the second approach, based on the use of the gradient of a blending by summation does provide guarantee in base cases such as parallel line skeletons. This method consists in analyzing the relationship between the field value of SCALIS and a scale-invariant gradient, in order to correct the field generated by the sum. Therefore, this can be seen as a field warping. The second solution is more promising than the skeleton-based approach in term of topology preservation. However it still needs to be adapted to compact support kernels in order to be usable in practice with large skeletons. Note that both solutions could be combined in order to obtain better guarantee of topology.

In the future, we would like to further improve blending of integral surfaces still without breaking their property of independence from skeleton subdivision : our new goal would be to provide more control over the sharpness of the blending by setting sharpness parameters in addition to radius on the skeleton. While this could be done using a two step convolution, this is not an acceptable solution since it would increase by a factor of two the computational cost. The alternatives we have in mind are either a modification of the Witnessed blending we presented, since it provides a way to interpolate between the union of balls and SCALIS, or a parametrization of the correction of our gradient-based field warping.

What is really still lacking in skeleton-based modeling is the possibility of creating shapes with sharp edges. While we can create them in some cases thanks to triangles primitives with varying radius, this is not an intuitive way nor an easy one to define them. A path to investigate would be a modification of the Warp Curve method of Sugihara [SWS10], the main difficulty being the replacement of the differential warp which ensures a C^2 continuity.

It would be important to improve the efficiency of the visualization which is the main remaining lock to the use of implicit surfaces. First, existing algorithms could be adapted to better use the graphics hardware : for instance a rough mesh with topological guarantee could be computed on the CPU before being tessellated on the GPU (which are more adapted to brute force algorithms) by taking into account the point of view to reduce the computational time. In addition to octree-based method, B-Meshes are an interesting alternative to generate a mesh since they provide quad-meshes of high quality : however the method should be adapted to take into account skeletons without any constraint.

Skeleton manipulation by user should be improved, for instance it should take into account hierarchy of skeletons (such as muscles on an arm). Furthermore, using at the same time sketch-based modeling to create base shapes and then skeleton manipulation to deform them if desired would be interesting.

There is probably some work to be done on animation of integral surfaces. For instance, it would be interesting to introduce constraints such as constant volume deformation during animation or to provide skeleton with special behavior to model muscle. Hierarchal deformation would also be required : for instance, if a skeleton has been created at the surface of an existing shape, this property should be preserved during animation.

Lastly, we are working on a new application of integral surfaces to animation : the generation of animated volume hair such as the one present in cartoon animation : implicit surfaces would provide the way to generate wisps that freely and smoothly merge and divide during animation. In order to obtain more natural blending behavior, wisps that are orthogonal should not merge, this could be obtained by parametrizing the blending thanks to the weighted direction we introduced for detail generation.

As a conclusion, we hope that this work will help making skeleton-based implicit more popular for both modeling and animation.

APPENDIX

A

SCALE-INVARIANT DISTANCE

The aim of this appendix is the study of scale-invariant distance to a segment, which we will also call distance in homothetic space. This "distance" is defined as the Euclidian distance to a skeleton point scaled by the inverse of the skeleton local weight :

$$d_H(P, Q) = \frac{\|\overrightarrow{PQ}\|}{\tau(Q)}$$

There are two main reasons behind this study. The first one is that, given a point in space, it is useful for both SCALIS with *Compact Polynomial* and Witnessed blending to find the skeleton points whose individual influence are above a given threshold. Because kernel functions are decreasing this is equivalent to finding the skeleton point whose homothetic distance are lower to a given threshold. The second reason is that scale-invariant distance primitives is the limit case of the Witnessed blending when the length used tend toward zero.

Notation : Assume we have a line segment $[AB]$, with the following parametrization defined on $[0; 1]$

$$\Gamma(t) = A + t\,\overrightarrow{AB},$$

and a weighting function defined on the same interval

$$\tau(t) = \tau_0 + \Delta\tau\, t.$$

Then, we have (with $P \in \mathbb{R}^3$ a query point) :

$$\|\overrightarrow{P\Gamma(t)}\|^2 = \|\overrightarrow{AB}\|^2\, t^2 - 2\overrightarrow{AB}.\overrightarrow{AP}\, t + \|\overrightarrow{AP}\|^2$$

We will name $\overrightarrow{u}$ the unit vector that define the segment direction.

Hypothesis : We assume that we only work with segments of positive weight :

$$\forall t \in [0;1],\ \tau(t) > 0$$

Furthermore, without loss of generality, we consider that $\Delta\tau$ is positive, thus that the maximal radius on the segment is in B. This is done to reduce the number of cases to be treated, results for negative value of $\Delta\tau$ can easily be obtained by inverting the parametrization of the segment. This means that the only parameter space that could have an interest is $]-\frac{\tau_0}{\Delta\tau};+\infty[$ (indeed we have then $\tau(t) > 0$).

A.1 Minimal distance to a segment

We are looking for the minimal distance in scale-invariant space between a point $P \in \mathbb{R}^3$ and a weighted segment $[AB]$:

$$\min_{t\in[0;1]} \frac{\|\overrightarrow{\Gamma(t)P}\|}{\tau(t)}$$

$\|\overrightarrow{\Gamma(t)P}\|$ and $\tau(t)$ are positive value, thus this is equivalent to studying :

$$\min_{t\in[0;1]} \frac{\|\overrightarrow{\Gamma(t)P}\|^2}{\tau(t)^2} = \min_{t\in[0;1]} g(t)$$

with :

$$g(t) = \frac{\|\overrightarrow{AB}\|^2\, t^2 - 2\overrightarrow{AB}.\overrightarrow{AP}\, t + \|\overrightarrow{AP}\|^2}{(\tau_0 + \Delta\tau\, t)^2}.$$

Studying g ***on*** $]-\frac{\tau_0}{\Delta\tau};+\infty[$

In order to obtain the minimal value of g the simplest way to proceed is to study the sign of its derivative and the function behavior at the limits of the interval of definition.

Limits :

$$\lim_{t\to -\frac{\tau_0}{\Delta\tau}} g(t) = +\infty$$

$$\lim_{t\to+\infty} g(t) = \frac{\|\overrightarrow{AB}\|^2}{\Delta\tau^2} > 0$$

Derivative of g :

$$g'(t) = \frac{2}{(\tau_0 + \Delta\tau.t)^3}\left(t(\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}) - (\tau_0\overrightarrow{AB}.\overrightarrow{AP} + \Delta\tau\|\overrightarrow{AP}\|^2)\right)$$

Derivative sign on $]-\frac{\tau}{\Delta\tau};+\infty[$ ***:***

First, as stated above, the denominator is positive on this interval, thus we only have to study the sign of the numerator.

$$g'(t) \geq 0 \Leftrightarrow t(\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}) \geq (\tau_0\overrightarrow{AB}.\overrightarrow{AP} + \Delta\tau\|\overrightarrow{AP}\|^2)$$

Two cases arise in function of the sign of $\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}$, which is also the sign of :

$$\tau_0 + \Delta\tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right) = \tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right)$$

which is the weight of the projection of the computation point on the segment. Two cases should be studied depending of its sign.

Case 1 : $\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} \leq -\frac{\tau_0}{\Delta\tau} \Leftrightarrow \tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right) \leq 0$

In this case, we have :

$$\begin{aligned} g'(t) < 0 \quad &\Leftrightarrow \quad t > \frac{\tau_0 \overrightarrow{AB}.\overrightarrow{AP} + \Delta\tau\|\overrightarrow{AP}\|^2}{\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}} \\ &\Leftrightarrow \quad t > \frac{\tau_0\,\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} + \Delta\tau\left\|\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right\|^2}{\tau_0 + \Delta\tau\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}} \end{aligned}$$

From the Pythagore's Theorem, we have $h \in \mathbb{R}$ such that :

$$\left\|\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right\|^2 = \left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right)^2 + h^2$$

We obtain :

$$\begin{aligned} g'(t) < 0 \quad &\Leftrightarrow \quad t > \overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\ \frac{\tau_0 + \Delta\tau\ \overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} + \Delta\tau\frac{h^2}{\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}}}{\tau_0 + \Delta\tau\ \overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}} \\ &\Leftrightarrow \quad t > \overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} + \frac{\Delta\tau\ h^2}{\tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right)} \end{aligned}$$

We have $\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} \leq -\frac{\tau_0}{\Delta\tau}$ and $\tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right) \leq 0$ and $\Delta\tau\ h^2 > 0$ so :

$$\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} + \frac{\Delta\tau h^2}{\tau\left(\overrightarrow{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right)} \leq -\frac{\tau_0}{\Delta\tau}$$

Because, we have $t \in]-\frac{\tau_0}{\Delta\tau}; +\infty[$, the condition for $g'(t) < 0$ is always satisfied. Thus g is decreasing which imply

$$\operatorname*{argmin}_{t\in[0;1]} g(t) = 1$$

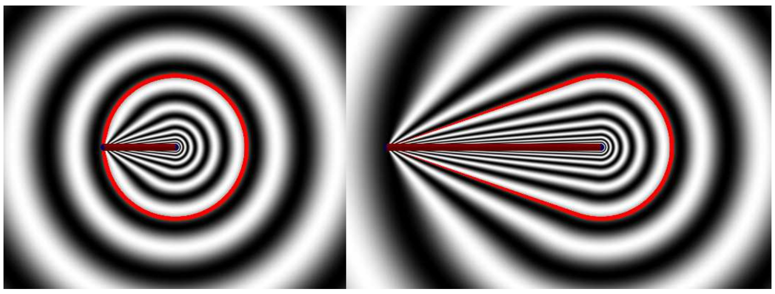


FIGURE A.1 – *Homothetic distance to a skeleton of varying radius (from $\tau = 0$ to $\tau = 1$) with two different weight speed variation, the distance 1 is in red.*

Case 2 : $\vec{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} > -\frac{\tau_0}{\Delta\tau} \Leftrightarrow \tau\left(\vec{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}\right) > 0$

By re-using the reasoning of case 1, we can check that :

$$g'(t) < 0 \Leftrightarrow t < \vec{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|} + \frac{h^2}{\tau_0 + \Delta\tau\ \vec{u}.\frac{\overrightarrow{AP}}{\|\overrightarrow{AB}\|}}$$

We can easily deduce that the derivative of g is first positive then negative on $]-\frac{\tau_0}{\Delta\tau};+\infty[$ which show that the minimum of g is obtained for the cancellation of its derivative. Thus we obtain :

$$g'(t) = 0 \Leftrightarrow t(\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}) - (\tau_0\overrightarrow{AB}.\overrightarrow{AP} + \Delta\tau\|\overrightarrow{AP}\|^2) = 0,$$

from which we can deduce :

$$\underset{t\in]-\frac{\tau_0}{\Delta\tau};+\infty[}{\operatorname{argmin}}\ g(t) = \frac{\tau_0\overrightarrow{AB}.\overrightarrow{AP} + \Delta\tau\|\overrightarrow{AP}\|^2}{\tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP}}$$

Conclusion

To summarize we have :

$$\underset{t\ /\ \tau(t)>0}{\operatorname{argmin}} \frac{\|\overrightarrow{\Gamma(t)P}\|}{\tau(t)} = \begin{cases} \frac{\tau_0\overrightarrow{AB}.\overrightarrow{AP}+\Delta\tau\|\overrightarrow{AP}\|^2}{\tau_0\|\overrightarrow{AB}\|^2+\Delta\tau\overrightarrow{AB}.\overrightarrow{AP}} & \text{if } \tau_0\|\overrightarrow{AB}\|^2 + \Delta\tau\overrightarrow{AB}.\overrightarrow{AP} > 0 \\ +\infty & \text{otherwise} \end{cases}$$

Having the parameter for which minimal homothetic distance is obtained, the latter can be easily computed (just note that the parameter's value should be clipped in order to belong to the segment skeleton).

A.1.1 Geometric interpretation

It is interesting to have a geometric interpretation of previous result. Assume that $P \in \mathbb{R}^3$, is such that $\overrightarrow{AB}$ and $\overrightarrow{AP}$ are not collinear, we will work in the plane defined by (A, B, P) and we will use as abscissa axis the direction $\vec{u} = \frac{\overrightarrow{AB}}{\|\overrightarrow{AB}\|}$. In this new frame, we have :

$$A = (0,0),\ B = (L,0),\ P = (P_x, P_y).$$

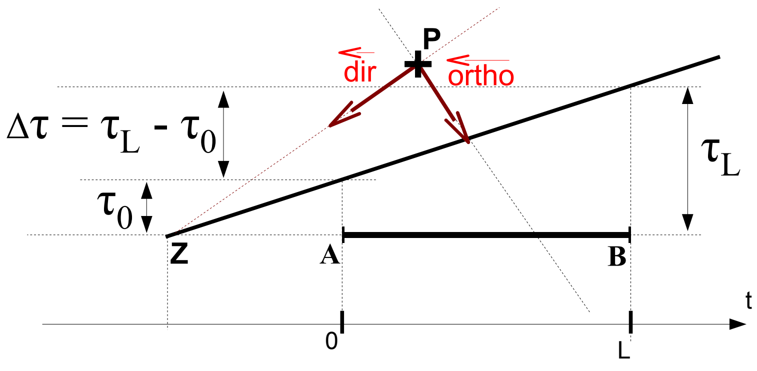


FIGURE A.2 – *Geometric interpretation of the closest point on a scale-invariant segment.*

We use a parametrization of the weight τ by curvilinear abscissa : $\tau(t) = \tau_0 + \frac{t}{L}\Delta\tau$. We can easily obtain the intersection of the line representing radius (see Figure A.2) with the abscissa axis :

$$Z = (-\frac{\tau_0 L}{\Delta\tau}, 0),$$

and we can define a local frame around P :

$$\overrightarrow{dir} = \overrightarrow{PZ} = (-\frac{\tau_0 L}{\Delta\tau} - P_x, -P_y) \text{ and } \overrightarrow{ortho} = (-P_y, \frac{\tau_0 L}{\Delta\tau} + P_x)$$

We can define a line Ω of direction $\overrightarrow{ortho}$ and going through P :

$$\Omega = \{P + \omega\, \overrightarrow{ortho}\ /\ \omega \in \mathbb{R}\}$$

We look for the intersection point between Ω and the abscissa axis :

$$\left(P + \omega\, \overrightarrow{ortho}\right)_y = 0 \Leftrightarrow \omega = \frac{-P_y}{\frac{\tau_0 L}{\Delta\tau} + P_x}$$

By injecting this expression in the definition of the line Ω we obtain the abscissa of the intersection :

$$t = P_x + \frac{\Delta\tau {P_y}^2}{\tau_0 L + \Delta\tau P_x}$$

which can be re-writed :

$$t = \frac{\tau_0(P_x L) + \Delta\tau({P_x}^2 + {P_y}^2)}{\tau_0 L + \Delta\tau P_x}$$

If we renormalize the axis abscissa in order to fall back on the initial parametrization of the segment, we obtain :

$$\tilde{t} = \frac{\tau_0(P_x L) + \Delta\tau({P_x}^2 + {P_y}^2)}{\tau_0 L^2 + \Delta\tau(P_x L)},$$

which is the expression we previously demonstrate.

A.2 Sphere clipping of segment in homothetic space

Clipping a segment with a sphere in the homothetic space is equivalent to finding the part of a segment which are included in a sphere of radius R scaled by the local weight defined on the segment, this is equivalent to finding the following set :

$$\left\{ t \in [0;1]\ /\ \frac{\|\overrightarrow{\Gamma(t)P}\|}{\tau(t)} \le R \right\},$$

which in turn is equivalent to studying the following second degree inequation :

$$(\|\overrightarrow{AB}\|^2 - R^2\Delta\tau^2)t^2 - 2(\overrightarrow{AB}.\overrightarrow{AP} + R^2\Delta\tau\tau_0)t + \|\overrightarrow{AP}\|^2 - R^2\tau_0^2 \le 0$$

This can be easily solved by studying the discriminant Δ of the equation, however some geometric considerations can be taken into account in order to simplify the implementation, furthermore solving second degree equation can be numerically ill-conditioned depending on the polynomial coefficients.

Geometric consideration

For each sign of the discriminant Δ there is interesting geometrical consideration to take into account in order to simplify the clipping. In order to study the different cases, we will introduce new notations. The new parametrization is chosen in order to have $\tau(0) = 0$:

$$\tau(t) = \Delta\tau t.$$

We then have

$$\|\overrightarrow{\Gamma(t)P}\|^2 = h^2 + (t-d)^2$$

with d the distance between the query point P and the point of the segment of minimal weight. The obtained inequation is :

$$t^2(1 - R^2\Delta\tau^2) - 2td + h^2 \leq 0,$$

its discriminant is :

$$\Delta = d^2 - (1 - R^2\Delta\tau^2)h^2$$

Δ is negative : In this case, there can be no clipping. The only case that would lead to solutions is if the coefficient of the second degree of the polynomial $a = (1 - R^2\Delta\tau^2)$ is negative which is impossible. Indeed, since we have $\Delta < 0 \Leftrightarrow d^2 < ah^2$ that would require h^2.

Δ is null : In this case there is only one point in space that could lead to an effective clipping. Indeed, if a is negative then the set of solution is of measure zero (which is useless when it comes to integration). On the other hand, we have $d^2 = ah^2$ which combine to $a \leq 0$ means that the only case it could happen is $d = 0$ and $h = 0$.

Δ is positive : In theory, a second degree inequation can have two unconnected interval, however, we can show that the solution of interest are a connected set which is important from a computational view-point. In order to have two disconnected intervals of solution, the coefficient of degree 2 should be negative, thus $(1 - R^2\Delta\tau^2) < 0$. The roots are :

$$\frac{-d \pm \sqrt{d^2 + (R^2\Delta\tau^2 - 1)h^2}}{R^2\Delta\tau^2 - 1}$$

We can see that $\sqrt{d^2 + (R^2\Delta\tau^2 - 1)h^2} \geq d$ that lead to $-d + \sqrt{\Delta} \geq 0$ and $-d - \sqrt{\Delta} \leq -2d$. Because d is positive we can come to the conclusion that the two roots are of opposed sign. The only interval of interest it thus :

$$\left[\frac{-d + \sqrt{d^2 + (R^2\Delta\tau^2 - 1)h^2}}{R^2\Delta\tau^2 - 1}; +\infty\right[$$

indeed, on

$$\left]-\infty; \frac{-d + \sqrt{d^2 + (R^2\Delta\tau^2 - 1)h^2}}{R^2\Delta\tau^2 - 1}\right] \subset \mathbb{R}^-$$

we have $\tau(t) < 0$ (our initial hypothesis is that we are only working with positif radius).

To summary for $\Delta \leq 0$ there is only one point in space that can lead to an effective clipping and for $\Delta > 0$ there can be only a connected set as solution.

Numerical consideration : Stable clipping

From an implementation view-point, one should be careful with second degree (in)equation since numerical error can easily appear when the coefficient of the second degree tend toward zero (and in some other cases). The document "Scilab Is Not Naive" is interesting since it explain the way solutions should be computed in order to avoid instability which we combine with knowledge of our problem to minimize the number of required test.

The two roots of the inequation $at^2 - 2bt + c \leq 0$ when $\Delta > 0$ are $x_1 = \frac{b-\sqrt{\Delta}}{a}$ and $x_2 = \frac{b-\sqrt{\Delta}}{a}$ which are linked by the two following relationships :

$$x_1 x_2 = \frac{c}{a} \text{ and } x_1 + x_2 = \frac{2b}{a}$$

which can be used to rewrite x_1 in a numerically more stable way :

$$x_1 = \frac{c}{b + \sqrt{\Delta}}.$$

Let us just remind that we are studying the case where $\Delta\tau \geq 0$, as it is the one we actually use in practice (see Appendix B.3.1) we present a way to minimize the number of required cases to be treated (indeed, conditional jump can slow done the computation speed). For both sign of a we should compare the value x_1 to 1 as if it is lower there will be no clipping (see Figure A.3). In the case where a is strictly positive, on should also check the sign of the second root since if it is negative, there is nothing to clipped. The second root being equal to $\frac{c}{ax_1}$ and a being positive, it is possible to check x_2 sign by looking at the sign of cx_1. Note that this last value is always positive for $a \leq 0$, this mean that we could just check the two previous statements to know if a clipping occurs. Here is the effective clipping code that we are using :

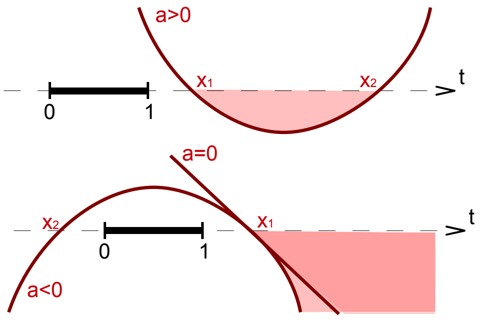


FIGURE A.3 – *When $\Delta\tau \geq 0$, comparison of x_1 to 1 enable to discard cases for all sign of a.*

Algorithm 1 Clipping computation

```
Require: $\Delta\tau \geq 0$
  $\Delta = b^2 - ac$
  if $\Delta \geq 0$ then
    $\delta = b + \sqrt{\Delta}$
    if $(\delta \geq 0)$ & $(\delta > c)$ then
      $x_1 = c/\delta$
      $l_1 = (x_1 < 0)$ ? $0 : x_1$
      $l_2 = (2b < a(x_1 + 1))$ ? $c/(ax_1) : 1$
      return $true$
    end if
  end if
  return $false$
```

When we know that a clipping a occur, we still have to find the solution that are inside $[0;1]$, which is done by comparing x_1 to 0 for all sign of a and by comparing x_2 to 1 for $a > 0$. Note that this latter condition could be reworked to be independent from the sign of a.

Particular case

Provided that $\Delta\tau \neq 0$, there is one point in space at equal distance (in homothetic space) to all the point of a segment skeleton. In order to demonstrate it, we will study the case of an half-infinite line with the first point in $(0,0)$ with a weight$\tau(0) = 0$ and direction $(0,1)$. So, we are looking for $P(x,y)$, such that :

$$\forall t \in [0,+\infty[, K(\frac{\|\overrightarrow{\Gamma(t)P}\|}{\tau(t)}) = cste,$$

the kernel K being a decreasing positive function this is equivalent to :

$$\forall t \in [0,+\infty[, \frac{\|\overrightarrow{\Gamma(t)P}\|^2}{\tau(t)^2} = cste \text{ with } \|\overrightarrow{\Gamma(t)P}\|^2 = (x-t)^2 + y^2 \text{ and } \tau(t)^2 = \Delta\tau^2 t^2.$$

So if $\Delta\tau \neq 0$, we have

$$(x-t)^2 + y^2 = cste\ \Delta\tau^2 t^2$$

Yet, two polynomial are equal if and only if all there coefficient are equals, so there is equality if and only if

$$cste = \frac{1}{\Delta\tau^2} \text{ and } x = 0 \text{ and } y = 0,$$

which mean that there is only one point in space at equal distance to all the skeleton points, this point being the point of the segment where τ fall to zero.

APPENDIX

B SCALE-INVARIANT INTEGRAL PRIMITIVES : ADDITIONAL DETAILS

In this Appendix, we will give some additional details on Scale-Invariant Integral surfaces : proof of one of its main property (namely independence to skeleton subdivision), additional information on efficiency of closed-form gradient evaluation, the way to easily compute characteristic values of the field function and lastly some practical details for *Compact Polynomial* kernel.

B.1 Proof of the additivity of SCALIS in relation to the skeleton

In order to demonstrate the additivity of the SCALIS field function $f_{\mathcal{S}k}$ in relation to the skeleton, we will use the formulation of equation (3.6) that is independent from any parametrization (with the skeleton $\mathcal{S}k$ defined in eq. (1.1)). We are mainly interested in demonstrating the additivity of two skeletons of the same dimension (the proof could also be done for skeletons of different dimension but with less convenient notations).

We just remind the definition of the distribution $g_{\mathcal{S}k}$ that help define the integral surface for a skeleton of constant dimension $i_{\mathcal{S}k}$:

$$g_{\mathcal{S}k}(P) = \delta_{i_{\mathcal{S}k},\mathbf{S}_{\mathcal{S}k}}(P)$$

where $\delta_{i_{\mathcal{S}k},\mathbf{S}_{\mathcal{S}k}}$ is the distribution that enable to fall back to an integral corresponding to the dimension of the skeleton. For skeletons of dimension 1, we have $\int_{\mathbb{R}^3} \delta_{i,\mathbf{S}}(P).f(P)\ \mathrm{d}P = \int_I f(\Gamma(s))\mathrm{d}s$, where Γ is the curvilinear parametrical representation of the skeleton $\mathcal{S}k$.

Given two skeletons $\mathcal{Sk}_1$ and $\mathcal{Sk}_2$ with the same dimension $i = i_{\mathcal{Sk}_1} = i_{\mathcal{Sk}_2}$, we will show that if we have :

$$\int_{\mathbb{R}^3} g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cap \mathbf{S}_{\mathcal{Sk}_2}}(P)\, \mathrm{d}P = 0 \quad (H_1)$$

then $f_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2} = f_{\mathcal{Sk}_1} + f_{\mathcal{Sk}_2}$. If $i = 3$, we would like to point out the hypothesis (H_1) is equivalent to $\mathbf{S}_1 \cap \mathbf{S}_2$ being a measure-zero set. Because $\tau_{\mathcal{Sk}}(Q)$ is the wanted radius around a skeleton point Q, the logical definition of $\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}$ and $\tau_{\mathcal{Sk}_1 \cap \mathcal{Sk}_2}$ is :

$$\begin{aligned}
\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q) &= \max(\tau_{\mathcal{Sk}_1}(Q), \tau_{\mathcal{Sk}_2}(Q)) \\
\tau_{\mathcal{Sk}_1 \cap \mathcal{Sk}_2}(Q) &= \min(\tau_{\mathcal{Sk}_1}(Q), \tau_{\mathcal{Sk}_2}(Q))
\end{aligned}$$

We will show that $f_{\mathcal{Sk}_1}(P) + f_{\mathcal{Sk}_2}(P) - f_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(P)$ is null under the hypothesis (H_1). First, we can note that $\mathbf{S}_{\mathcal{Sk}_1} \setminus \mathbf{S}_{\mathcal{Sk}_2}$, $\mathbf{S}_{\mathcal{Sk}_2} \setminus \mathbf{S}_{\mathcal{Sk}_1}$ and $\mathbf{S}_{\mathcal{Sk}_1} \cap \mathbf{S}_{\mathcal{Sk}_2}$ form a partition of $\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}$ (and that the distributions $g_{i,\mathbf{S}_{\mathcal{Sk}_1}}$, $g_{i,\mathbf{S}_{\mathcal{Sk}_2}}$ and $g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}$ are null on $\mathbb{R}^3 \setminus (\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2})$), which imply :

$$\begin{aligned}
f_{\mathcal{Sk}_1}(P) + f_{\mathcal{Sk}_2}(P) &- f_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(P) = \\
&\int_{\mathbf{S}_{\mathcal{Sk}_1} \setminus \mathbf{S}_{\mathcal{Sk}_2}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q)}{\tau_{\mathcal{Sk}_1}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1}(Q)}\right) - \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \quad \mathrm{d}Q \\
&+ \int_{\mathbf{S}_{\mathcal{Sk}_2} \setminus \mathbf{S}_{\mathcal{Sk}_1}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_2}(Q)}\right) - \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \quad \mathrm{d}Q \\
&+ \int_{\mathbf{S}_{\mathcal{Sk}_1} \cap \mathbf{S}_{\mathcal{Sk}_2}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q)}{\tau_{\mathcal{Sk}_1}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1}(Q)}\right) + \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_2}(Q)}\right) \\
&\qquad - \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \quad \mathrm{d}Q.
\end{aligned}$$

Now, for all point $Q \in \mathbf{S}_{\mathcal{Sk}_1} \setminus \mathbf{S}_{\mathcal{Sk}_2}$ (and reciprocally), we have $g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q) = g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)$ and $\tau_{\mathcal{Sk}_1}(Q) = \tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)$ which imply :

$$\int_{\mathbf{S}_{\mathcal{Sk}_1} \setminus \mathbf{S}_{\mathcal{Sk}_2}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q)}{\tau_{\mathcal{Sk}_1}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1}(Q)}\right) - \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \mathrm{d}Q = 0.$$

Thus, we have :

$$\begin{aligned}
f_{\mathcal{Sk}_1}(P) + f_{\mathcal{Sk}_2}(P) &- f_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(P) = \\
&\int_{\mathbf{S}_{\mathcal{Sk}_1} \cap \mathbf{S}_{\mathcal{Sk}_2}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q)}{\tau_{\mathcal{Sk}_1}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1}(Q)}\right) + \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_2}(Q)}\right) \\
&\qquad - \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \quad \mathrm{d}Q.
\end{aligned}$$

Because all the manipulated quantities are either positives or null, this is equivalent to

$$\begin{aligned}
|f_{\mathcal{Sk}_1}(P) + f_{\mathcal{Sk}_2}(P) &- f_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(P)| \leq \\
&\int_{\mathbf{S}_{\mathcal{Sk}_1} \cap \mathbf{S}_{\mathcal{Sk}_2}} \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1}}(Q)}{\tau_{\mathcal{Sk}_1}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1}(Q)}\right) + \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_2}(Q)}\right) \\
&\qquad + \frac{g_{i,\mathbf{S}_{\mathcal{Sk}_1} \cup \mathbf{S}_{\mathcal{Sk}_2}}(Q)}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{Sk}_1 \cup \mathcal{Sk}_2}(Q)}\right) \quad \mathrm{d}Q.
\end{aligned}$$

On $\mathbf{S}_{\mathcal{S}k_1} \cap \mathbf{S}_{\mathcal{S}k_2}$, we have $g_{i,\mathbf{S}_{\mathcal{S}k_1}}(Q) = g_{i,\mathbf{S}_{\mathcal{S}k_2}}(Q) = g_{i,\mathbf{S}_{\mathcal{S}k_1}\cup\mathbf{S}_{\mathcal{S}k_2}}(Q) = g_{i,\mathbf{S}_{\mathcal{S}k_1}\cap\mathbf{S}_{\mathcal{S}k_2}}(Q)$, $\tau_{\mathcal{S}k_1\cap\mathcal{S}k_2}(Q) \leq \tau_{\mathcal{S}k_1}(Q) \leq \tau_{\mathcal{S}k_1\cup\mathcal{S}k_2}(Q)$ and $\tau_{\mathcal{S}k_1\cap\mathcal{S}k_2}(Q) \leq \tau_{\mathcal{S}k_2}(Q) \leq \tau_{\mathcal{S}k_1\cup\mathcal{S}k_2}(Q)$, which combined to the fact that K is a decreasing function on $\mathbb{R}^+$ gives :

$$|f_{\mathcal{S}k_1}(P) + f_{\mathcal{S}k_2}(P) - f_{\mathcal{S}k_1\cup\mathcal{S}k_2}(P)| \leq 3\int_{\mathbf{S}_{\mathcal{S}k_1}\cap\mathbf{S}_{\mathcal{S}k_2}} \frac{g_{i,\mathbf{S}_{\mathcal{S}k_1}\cap\mathbf{S}_{\mathcal{S}k_2}}(Q)}{\tau_{\mathcal{S}k_1\cap\mathcal{S}k_2}(Q)^i} K\left(\frac{\|P-Q\|}{\tau_{\mathcal{S}k_1\cup\mathcal{S}k_2}(Q)}\right) \mathrm{d}Q$$

and thanks to hypothesis (H_1), we have :

$$|f_{\mathcal{S}k_1}(P) + f_{\mathcal{S}k_2}(P) - f_{\mathcal{S}k_1\cup\mathcal{S}k_2}(P)| \leq 0$$

which lead to :

$$f_{\mathcal{S}k_1}(P) + f_{\mathcal{S}k_2}(P) = f_{\mathcal{S}k_1\cup\mathcal{S}k_2}(P) \text{ under the condition } (H_1).$$

B.2 Special value computation

In [HC12], an extended analysis of convolution is performed containing proof of the location of particular direction in a scalar field generated by a segment skeleton. This direction correspond to maximal and minimal distance between the skeleton and a given iso-value. This direction are respectively the line (in 2D) going through the middle point of the segment and orthogonal to it and the line defined by the segment.

1D skeletons :

Here, we are only interested in the special case of segment with infinite length since it is required for the normalization of SCALIS. First, we will compute the convolution of constant weight equal to one with a line as a function h of the distance d to the line. Knowing h, we can deduce by applying integration by substitution the result for both Weighted convolution, Hornus integral surfaces and Scale-invariant integral surfaces (all with a constant weight τ). Indeed, we have the following relationship :

– *Weighted convolution surfaces* :

$$h_{\mathcal{C}}(d) = \tau h(d) \tag{B.1}$$

– *Hornus integral surfaces* :

$$h_{\mathcal{H}}(d) = \tau h\left(\frac{d}{\tau}\right) \tag{B.2}$$

– *Scale-invariant integral surfaces (before normalization)* :

$$h_{\mathcal{S}}(d) = h\left(\frac{d}{\tau}\right) \tag{B.3}$$

In order to compute h more easily, we can rely on a half line (see Figure B.1). The distance between the computation point and the point of the line of parameter t is defined by :

$$d(t)^2 = d^2 + t^2$$

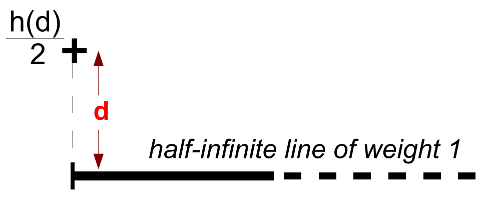


FIGURE B.1 – *Simplified computation of special field values h.*

For both *Inverse* and *Cauchy* kernel we have an infinite domain of integration. For instance for the *Cauchy* kernel, we have (with i the degree of the kernel) :

$$\frac{h_i(d)}{2} = \int_0^{+\infty} \frac{1}{\left(\frac{t^2}{\sigma^2} + \frac{d^2}{\sigma^2} + 1\right)^{\frac{i}{2}}} \,\mathrm{d}t$$

which is equivalent to :

$$h_i(d) = 2\left(\frac{\sigma^2}{d^2+\sigma^2}\right)^{\frac{i}{2}} \int_0^{+\infty} \frac{1}{\left(\frac{t^2}{d^2+\sigma^2} + 1\right)^{\frac{i}{2}}} \,\mathrm{d}t = 2\left(\frac{\sigma^2}{d^2+\sigma^2}\right)^{\frac{i}{2}} \left[I_{0,i}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty}$$

We just have to pass the recurrence formula of Table 3.3 to the limit, to obtain the result. First, we can note that we have :

$$\left[\frac{at-b}{(at^2-2bt+c)^{\frac{i-2}{2}}}\right]_0^{+\infty} = \begin{cases} a^{\frac{1}{2}} \text{ if } i = 3 \\ 0 \text{ if } i > 3 \end{cases}$$

So, if $i > 3$, we have

$$(i-2)\frac{1}{d^2+\sigma^2}\left[I_{0,i}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty} + \frac{1}{d^2+\sigma^2}(3-i)\left[I_{0,i-2}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty} = 0$$

$$\left[I_{0,i}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty} = \frac{i-3}{i-2}\left[I_{0,i-2}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty}$$

By injecting this equality in the expression of h_i, we obtain :

$$h_i(d) = 2\left(\frac{\sigma^2}{d^2+\sigma^2}\right)^{\frac{i}{2}} \frac{i-3}{i-2}\left[I_{0,i-2}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty}$$

$$\Leftrightarrow h_i(d) = \left(\frac{\sigma^2}{d^2+\sigma^2}\right) h_{i-2}(d) = \left(\frac{1}{\left(\frac{d}{\sigma}\right)^2 + 1}\right) h_{i-2}(d)$$

We just have to compute the initial case of the recurrence, hence h_2 and h_3. For $i = 2$, we have :

$$\left[I_{0,2}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty} = \sqrt{d^2+\sigma^2}\,(\arctan(+\infty) - \arctan(0)) = \sqrt{d^2+\sigma^2}\frac{\pi}{2}$$

and for $i = 3$:

$$\left[I_{0,3}(\frac{1}{d^2+\sigma^2}, 0, 1)\right]_0^{+\infty} = \left(d^2+\sigma^2\right)^{\frac{1}{2}}$$

By injecting this two formula in the expression of h_i, we obtain :

$$h_2(d) = \pi\frac{\sigma^2}{(d^2+\sigma^2)^{\frac{1}{2}}} = \pi\sigma\frac{1}{\sqrt{\left(\frac{d}{\sigma}\right)^2 + 1}}$$

and

$$h_3(d) = 2\frac{\sigma^3}{(d^2+\sigma^2)} = 2\sigma\frac{1}{\left(\frac{d}{\sigma}\right)^2 + 1}$$

We just have to apply equation (B.3) in order to obtain the expression presented in Table 3.5. The formula for the *Inverse* kernel can be computed the same way.

The main difference for the *Compact Polynomial* kernel, is that the interval of integration is limited and depend on the kernel support. By applying the Pythagorean theorem, we directly obtain the domain of integration :

$$[0; \sqrt{\sigma^2 - d^2}]$$

Thus, we have :

$$h_i(d) = 2\int_0^{\sqrt{\sigma^2-d^2}} \left(1 - \frac{d^2}{\sigma^2} - \frac{t^2}{\sigma^2}\right)^{\frac{i}{2}} \, \mathrm{d}t$$

which is equivalent to

$$h_i(d) = 2\left(1 - \frac{d^2}{\sigma^2}\right)^{\frac{i}{2}} \int_0^{\sqrt{\sigma^2-d^2}} \left(1 - (\sigma^2 - d^2)t^2\right)^{\frac{i}{2}} \, \mathrm{d}t$$

$$h_i(d) = 2\left(1 - \frac{d^2}{\sigma^2}\right)^{\frac{i}{2}} \left[I_{0,-i}(-(\sigma^2 - d^2), 0, 1)\right]_0^{\sqrt{\sigma^2-d^2}}$$

From this point, the same method as for *Cauchy* kernel should be applied.

2D skeletons :

The normalization factor to be used with SCALIS depend on the dimension of the skeleton. For 2D skeleton, one should be able to compute the convolution with weight 1 of an infinite plane in order to obtain the required normalization factor :

$$h_{i,2D}(d) = \int_{\mathbb{R}^2} K(\sqrt{d^2 + x^2 + y^2}) \, \mathrm{d}x\mathrm{d}y$$

The simplest way to compute this function is to express the integration in polar coordinates :

$$h_{i,2D}(d) = \int_{[0;+\infty]\times[0;2\pi]} K(\sqrt{d^2 + r^2}) \, r \, \mathrm{d}r\mathrm{d}\theta$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \int_{[0;+\infty]} K(\sqrt{d^2 + r^2}) \, r \, \mathrm{d}r$$

For *Inverse*, we obtain :

$$h_{i,2D}(d) = 2\pi \int_0^{+\infty} \frac{r}{\left(\frac{d^2+r^2}{\sigma^2}\right)^{\frac{i}{2}}} \, \mathrm{d}r$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \left[-\frac{\sigma^2}{i-2} \frac{1}{\left(\frac{d^2+r^2}{\sigma^2}\right)^{\frac{i-2}{2}}} \right]_0^{+\infty}$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \frac{\sigma^2}{i-2} \frac{1}{\left(\frac{d}{\sigma}\right)^{i-2}}$$

For *Cauchy* kernel, we have :

$$h_{i,2D}(d) = 2\pi \int_0^{+\infty} \frac{r}{\left(1+\frac{d^2+r^2}{\sigma^2}\right)^{\frac{i}{2}}} \,\mathrm{d}r$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \left[-\frac{\sigma^2}{k-2} \frac{1}{(1+\frac{d^2+r^2}{\sigma^2})^{\frac{i-2}{2}}} \right]_0^{+\infty}$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \frac{\sigma^2}{i-2} \frac{1}{(1+\left(\frac{d}{\sigma}\right)^2)^{\frac{i-2}{2}}}$$

And finally, for *Compact Polynomial* kernel, we get :

$$h_{i,2D}(d) = 2\pi \int_0^{\sqrt{\sigma^2-d^2}} r \left(1 - \frac{d^2+r^2}{\sigma^2}\right)^{\frac{i}{2}} \mathrm{d}r$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \left[\frac{-\sigma^2}{i+2} \left(1 - \frac{d^2+r^2}{\sigma^2}\right)^{\frac{i+2}{2}} \right]_0^{\sqrt{\sigma^2-d^2}}$$

$$\Leftrightarrow h_{i,2D}(d) = 2\pi \frac{\sigma^2}{i+2} \left(1 - \frac{d^2}{\sigma^2}\right)^{\frac{i+2}{2}}$$

Remark :

It is interesting to note that the field of infinite skeleton (line and plane) of constant weight can be expressed, for all classic kernel, as a kernel of the same family which degree depend on the dimension of the skeleton.

B.3 SCALIS SEGMENT PRIMITIVES WITH LINEAR RADIUS : ADDITIONAL DETAILS

B.3.1 *Compact Polynomial* kernel : Better formulation for computation

With *Compact Polynomial* kernel, directly applying recurrence relationship to formula presented in 3.1 lead to a numerically ill-conditioned expression which can lead to numerical instability when $\Delta\tau$ tend toward zero.

A first solution to the problem would be to introduce an approximation scheme : the factor $\rho(t) = \frac{1}{(\Delta\tau\ t+1)^{j+1}}$ which causes all the trouble could be approximated by a polynomial function when $\Delta\tau$ tend toward zero. It is possible to either use a finite expansion in 0 or an Hermite interpolation with a spline of degree 3 of $(\rho(0), \rho'(0), \rho(l), \rho'(l))$. By doing polynomial approximation, we fall back on known primitives (see Table 3.3), however either the accuracy is impaired or the computation cost increased (for polynomial of high degree).

We present in the following how to re-write the expression in a way that is both numerically stable, has great accuracy and does not cost more to evaluate. The increase in stability is particularly useful if one want to perform computation on a GPU which use single precision floating point.

Reformulating the integral :

Wether the segment is parametrized from maximal radius to minimal radius or the inverse is important for stability : we discover empirically that $\Delta\tau$ should be positive to provide better stability, we choose our parametrization accordingly.

The factor $\frac{1}{(\Delta\tau\, l_{1,2}+\tau_0)^{j+1}}$ that appear in the closed-form formula jeopardize stability if $\Delta\tau\, l_{1,2} + \tau_0$ is to small. In order to remove this problem we factorize τ_0 from denominator and bring it in the inside of the power $\frac{i}{2}$ of the numerator leading to the formula :

$$\mathcal{R}_{g,i}(P) = \frac{\|\overrightarrow{AB}\|}{\tau_0} \int_{l1}^{l2} \frac{\left((\left(\frac{\Delta\tau}{\tau_0}\right)^2 - \frac{l^2}{\tau_0^2})t^2 - 2(-\frac{\Delta\tau}{\tau_0} - \frac{uv(P)}{\tau_0^2})t + 1 - \frac{d(P)^2}{\tau_0^2} \right)^{\frac{i}{2}}}{(\frac{\Delta\tau}{\tau_0}\, t + 1)^{i+1}} \, \mathrm{d}t.$$

This combined with the fact that $\Delta\tau$ is positive ensure that $\frac{1}{(\Delta\tau\, l_{1,2}+\tau_0)^{j+1}}$ is never smaller than 1. Because the inverse of the coefficient $\frac{\Delta\tau}{\tau_0}$ of the denominator appear in the closed-form formula, we apply a change of variable : $t \mapsto \frac{\tau_0}{\|\overrightarrow{AB}\|}\tilde{t}$ which give the following formula :

$$\mathcal{R}_{g,i}(P) = \int_{\tilde{l1}}^{\tilde{l2}} \frac{\left((\Delta\tau_u^2 - \frac{1}{\sigma^2})\tilde{t}^2 - 2(-\Delta\tau_u - \frac{1}{\sigma^2}\frac{\overrightarrow{u}.\overrightarrow{P_0P}}{\tau_0^2})\tilde{t} + 1 - \frac{1}{\sigma^2}\frac{\|\overrightarrow{P_0P}\|^2}{\tau_0^2} \right)^{\frac{i}{2}}}{(\Delta\tau_u\, \tilde{t} + 1)^{i+1}} \, \mathrm{d}\tilde{t}.$$

with $\Delta\tau_u$ the variation of radius per unit of length. The last step in the reformulation is the use of a "local segment" which will simplify the expression by having zero as the first bound of the integral : after computing the clipping in the classical way, we compute the new formula through Pythagorean theorem in order to minimize the number of required operations. Here is what the new code to evaluate both f and its gradient looks like :

Algorithm 2 SCALIS with *Compact Polynomial* kernel

```
Require: $\Delta\tau_u \geq 0$
  $uv = (\overrightarrow{u}|\overrightarrow{P_0P})$
  $d2 = \|\overrightarrow{P_0P}\|^2$
  $coeff[3] = \{\tau_0^2 - \frac{1}{\sigma^2}\, d2,\ -\Delta\tau_u\, \tau_0 - \frac{1}{\sigma^2}\, uv,\ \Delta\tau_u^2 - \frac{1}{\sigma^2}\}$
  if HomotheticClipping($coeff$, $l_1$, $l_2$)) then
    $\omega_m = 1.0/(\tau_0 + l_1\, \Delta\tau_u)$
    $coeff[0] = 1.0 - \frac{1}{\sigma^2}(l_1(l_1 - 2uv) + d2)\omega_m^2$
    $coeff[1] = -\Delta\tau_u - \frac{1}{\sigma^2}(uv - l_1)\omega_m$
    G_seg_FGradF_i( $(l_2 - l_1)\omega_m$, $\Delta\tau_u$, $coeff$, $F0F1F2$)
    $f = N_{1D}\ F0F1F2[0]$
    $F0F1F2[1]* = \omega_m$
    $\nabla f = (\frac{i}{\sigma^2} N_{1D}\omega_m)((F0F1F2[2] + l_1\ F0F1F2[1])\overrightarrow{u} - F0F1F2[1]\ \overrightarrow{P_0P})$
  else
    $f = 0$
    $\nabla f = \overrightarrow{0}$
  end if
```

Simplifying the obtained expression :

By applying the recurrence formula of Table 3.4, we arrive on an expression with division by $\Delta\tau_u$ which can become problematic when too close to zero. We will see how to remove all these divisions (note that we apply our simplification on the expression obtained through maple optimization).

The first kind of problematic expression is :

$$a_j \left(... \left(a_1 \left(a_0 \left(1 - \frac{1}{(\Delta\tau_u\, l + 1)^k} \right) \frac{1}{\Delta\tau_u} - \frac{l}{(\Delta\tau_u\, l + 1)^{k+1}} \right) \frac{1}{\Delta\tau_u} - .... \right) \frac{1}{\Delta\tau_u} - ... \right) \tag{B.4}$$

with k belonging to $[1; i]$ and $k + j$ in $[k; i]$. First, we have :

$$1 - \frac{1}{(\Delta\tau_u\, l + 1)^k} = \frac{(\Delta\tau_u\, l + 1)^k - 1}{(\Delta\tau_u\, l + 1)^k}$$

Using the following equality :

$$(\Delta\tau_u\, l + 1)^k - 1 = (\Delta\tau_u\, l) \sum_{n=0}^{k-1} (\Delta\tau_u\, l + 1)^n$$

we obtain :

$$1 - \frac{1}{(\Delta\tau_u\, l + 1)^k} = (\Delta\tau_u\, l) \sum_{n=1}^{k} \frac{1}{(\Delta\tau_u\, l + 1)^n}$$

This provide a way to simplify the division by $\Delta\tau_u$. By applying recursively this method in the expression (B.4), we arrive on the following kind of expression :

$$\left(\frac{l}{\Delta\tau_u\, l + 1} \right)^k \sum_{n=0}^{j} \frac{b_n}{(\Delta\tau_u\, l + 1)^n}$$

However, there is one case where this simplification can't be applied. This is due to the presence of a logarithm :

$$a_i \left(... \left(a_1 \left(a_0 \left(\ln(\Delta\tau_u\, l + 1) \frac{1}{\Delta\tau_u} - \frac{l}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} - \frac{l^2}{(\Delta\tau_u\, l + 1)^2} \right) \frac{1}{\Delta\tau_u} - ... \right) \frac{1}{\Delta\tau_u} - ... \right) \tag{B.5}$$

In order to simplify the division by $\Delta\tau_u$, we should express the logarithm as a series. While we could use the classic finite expansion of the logarithm $\ln(1 + x)$ around 1, this series does not converge outside $[-1; 1]$. Instead, we use the following series for the logarithm :

$$\ln(x + 1) = 2 \sum_{k=0}^{+\infty} \frac{1}{2k + 1} \left(\frac{x}{x + 2} \right)^{2k+1}$$

this series converge for all value of x in $[-1; +\infty]$ and has a better convergence rate on $[-1; 1]$ that the classic finite expansion.

Here is the first step of the simplification :

$$
\begin{aligned}
I &= a_0 \left(2 \frac{1}{\Delta\tau_u} \sum_{k=0}^{+\infty} \frac{1}{2k+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k+1} - \frac{l}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \left(2 \frac{l}{\Delta\tau_u\, l + 2} \sum_{k=0}^{+\infty} \frac{1}{2k+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k} - \frac{l}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \frac{l}{\Delta\tau_u\, l + 2} \left(2 \sum_{k=0}^{+\infty} \frac{1}{2k+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k} - \frac{\Delta\tau_u\, l + 2}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \frac{l}{\Delta\tau_u\, l + 2} \left(2 \sum_{k=1}^{+\infty} \frac{1}{2k+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k} + 2 - \frac{\Delta\tau_u\, l + 2}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \frac{l}{\Delta\tau_u\, l + 2} \left(2 \sum_{k=1}^{+\infty} \frac{1}{2k+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k} + \frac{2(\Delta\tau_u\, l + 1) - (\Delta\tau_u\, l + 2)}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \frac{l}{\Delta\tau_u\, l + 2} \left(2 \sum_{k=0}^{+\infty} \frac{1}{2k+3} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k+2} + \frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \frac{l}{\Delta\tau_u\, l + 2} \left(2 \frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \sum_{k=0}^{+\infty} \frac{1}{2k+3} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k+1} + \frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 1} \right) \frac{1}{\Delta\tau_u} \\
&= a_0 \left(\frac{l}{\Delta\tau_u\, l + 2} \right)^2 \left(2 \sum_{k=0}^{+\infty} \frac{1}{2k+3} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k+1} + \frac{\Delta\tau_u\, l + 2}{\Delta\tau_u\, l + 1} \right)
\end{aligned}
$$

In addition to removing the division by $\Delta\tau_u$, this result in the cancellation of the first term of the series. By applying recursively this kind of simplification to the expression B.5, we arrive to the following formula (for degree i even) :

$$
\left(\frac{l}{\Delta\tau_u\, l + 2} \right)^{i+1} \left(2 \sum_{k=0}^{+\infty} \frac{1}{2k+i+1} \left(\frac{\Delta\tau_u\, l}{\Delta\tau_u\, l + 2} \right)^{2k} + \frac{1}{i} \frac{(\Delta\tau_u\, l + 2)^{\frac{i-2}{2}}}{(\Delta\tau_u\, l + 1)^i} poly_{i-2}(\Delta\tau_u\, l) \right)
$$

where $poly_{i-2}(\Delta\tau_u\, l)$ is a polynomial of degree $i-2$ in $\Delta\tau_u\, l$. Note that this simplification is made possible by the particular definition of the coefficients $\{a_j\}$. There is no more divisions by $\Delta\tau_u$ and the first terms of the series that represent the logarithm has been canceled.

In practice, for kernel of degree 6, we truncate the new series at the rank 7 and evaluate it through Horner's method. This is equivalent to having use a truncation at rank 10 of the logarithm series minus possible massive cancellation. The resulting evaluation code uses 70 multiplications, 43 additions and subtractions, 3 divisions and no special functions. For this reason, in addition to being more stable than the naive computation code, it is also slightly faster.

B.4 SKELETON-BASED DUAL-MARCHING CUBES

In order to mesh objects with large difference of prescribed radii, we use a slightly modified version of the dual-contouring algorithm [JLSW02].

We proceed in two main stages : firstly, the octree is subdivided in order to retrieve all the unconnected part of the surface. In order to do so, we subdivide the octree in order to have the length of cell edges lower or equal to the primitive radius in their bounding-boxes (we use several bounding-boxes per primitive to handle variation of radius as well as long non axis aligned primitives). The second step is a subdivision according to a smaller value (proportional to the radius) which aims at obtaining a good looking surface : we subdivide only the cubes from which at least one edge is crossed by the iso-surface until the edge length is smaller than the given threshold.

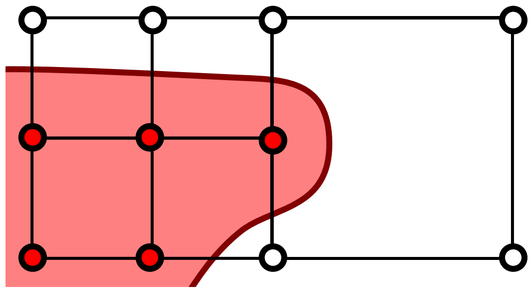


FIGURE B.2 – *If all the edges of a cell do not cross the surface (no change of sign between the corner), then this cell cannot create a vertex, which can be problematic. Furthermore, in the second step of our subdivision method, only the cells where the surface is detected are subdivided which would lead us to ignore a part of the surface in the presented case.*

However, this can introduce problematic cases (see Figure B.2). In order to prevent them, we introduce an additional correction step : we propagate cell-subdivision to neighbor when red-white-red or white-red-white edges are detected.

One of the advantages of such a meshing method is that it easily provide incremental surfaces improvement (the precision is not the same at rest and during user manipulation).

There is still a lot of possible improvements from better data-structure to the use of additional subdivision criterion (in order to provide better reconstruction of blending or better topological guarantee). Furthermore, in the context of dual meshing, several methods have been presented for the reconstruction of sharp edges.

APPENDIX

C

GRADIENT-BASED SCALIS : ADDITIONAL DETAILS

In this Appendix, we present additional information on gradient-based field warping of Chapter 4. We remind that the graph $\mathcal{G}$ is defined as :

$$\{(x, g(x)), x \in \mathbb{R}\}$$

with $g(x) = (i-1)x^{\frac{i}{i-1}}$, i being the degree of the *Inverse* kernel.

C.1 Projection on closest point

The projected point of $P = (f_{\mathcal{S}}, \|\nabla f_{\mathcal{H}}\|)$ onto closest point of $\mathcal{G}$ is defined by :

$$Q = \underset{x \in \mathbb{R}^+}{\operatorname{argmin}} \left\| \begin{pmatrix} f_{\mathcal{S}} \\ \|\nabla f_{\mathcal{H}}\| \end{pmatrix} - \begin{pmatrix} x \\ (i-1)x^{\frac{i}{i-1}} \end{pmatrix} \right\|^2$$

This is equivalent to find the cancellation of the derivative :

$$\Big((f_{\mathcal{S}} - x)^2 + (\|\nabla f_{\mathcal{H}}\| - (i-1)x^{\frac{i}{i-1}})^2\Big)' = 0$$

Since x is positive, we will look for $X = x^{\frac{1}{i-1}}$, which lead to :

$$\begin{aligned} &\big((f_{\mathcal{S}} - X^{i-1})^2 + (\|\nabla f_{\mathcal{H}}\| - (i-1)X^i)^2\big)' = 0 \\ \Leftrightarrow\ & 2(-(i-1)X^{i-2})(f_{\mathcal{S}} - X^{i-1}) + 2(-(i-1)iX^{i-1})(\|\nabla f_{\mathcal{H}}\| - (i-1)X^i) = 0 \\ \Leftrightarrow\ & 2(i-1)X^{i-2}\left(X^{i-1} - f_{\mathcal{S}} + iX^2((i-1)X^i - \|\nabla f_{\mathcal{H}}\|)\right) = 0 \end{aligned}$$

So, we obtain a polynomial equation in X :

$$i(i-1)X^{i+2} + X^{i-1} - i\|\nabla f_{\mathcal{H}}\|X^2 - f_{\mathcal{S}} = 0$$

which can be solved analytically for i up to 3.

C.2 Projection an arbitrary direction

C.2.1 Closed-form solution

The intersection between $\mathcal{G}$ and the line :

$$\mathcal{D} = \{(x, -\tan(\psi)x + \|\nabla f_{\mathcal{H}}\| + \tan(\psi)f_{\mathcal{S}}) \ / \ x \in \mathbb{R}\}.$$

verify :

$$(i-1)x^{\frac{i}{i-1}} = -\tan(\psi)x + (\|\nabla f_{\mathcal{H}}\| + \tan(\psi)f_{\mathcal{S}})$$

which can be rewritten, by using $X = x^{\frac{1}{i-1}}$, as :

$$(i-1)X^i + \tan(\psi)X^{i-1} - (\|\nabla f_{\mathcal{H}}\| + \tan(\psi)f_{\mathcal{S}}) = 0$$

This is a polynomial equation, therefore it can be solved analytically for degree i lower or equal to 5. It seems that this equation has only one positive root (we do not have proved it), which we will note r_+. Then, the abscissa of the projected point is :

$$f = \sqrt[i-1]{r_+}.$$

C.2.2 Approximation scheme

We want to find the intersection between the line $\mathcal{D}$ and the line defined by horizontal projected point Q_H and vertical projected point Q_V.

In order to find it, we just have to do some trigonometry, let l_H being the distance between P and the horizontal projection, and l_V being the distance between P and the vertical projection (see Figure C.1, all other following notations refer to this figure). In this case we will defined the abscissa of the intersection as :

$$f_\psi = f_{\mathcal{S}} - c$$

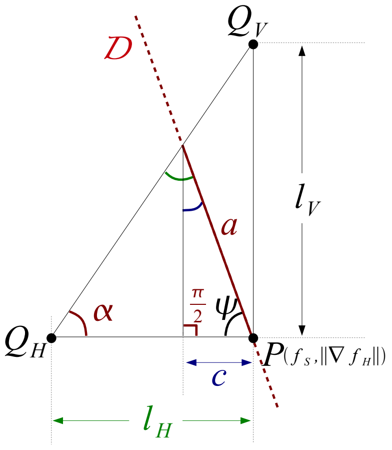


Figure C.1 – *Triangle formed by the point P and vertical and horizontal projection, as well as the line $\mathcal{D}$.*

By applying the law of sines in two different triangles, we obtain

$$\frac{l_H}{\sin(\pi - (\alpha + \psi))} = \frac{a}{\sin(\alpha)}$$

and

$$a = \frac{a}{\sin(\frac{\pi}{2})} = \frac{c}{\sin(\pi - (\frac{\pi}{2} + \psi))}$$

which implies :

$$c = l_H \sin(\alpha) \frac{\sin(\frac{\pi}{2} - \psi)}{\sin(\pi - (\alpha + \psi))} = l_H \sin(\alpha) \frac{\cos(\psi)}{\sin(\alpha + \psi)}.$$

Using the fact that $\sin(\alpha + \psi) = \sin(\alpha)\cos(\psi) + \cos(\alpha)\sin(\psi)$, we obtain :

$$c = l_H \frac{\tan(\alpha)}{\tan(\alpha) + \tan(\psi)},$$

and by taking into account that $\tan(\alpha) = \frac{l_V}{l_H}$, we get :

$$c = \frac{l_H l_V}{l_V + l_H \tan(\psi)}.$$

C.3 Parallel skeleton analysis

In order to obtain the graph of figure 4.28, one have to find the relationship between $f_{\mathcal{S}}$ and $\|\nabla f_{\mathcal{H}}\|$ in the case of point lying at equal distance from two parallel infinite line skeletons (presented in figure 4.27). In this case the field value depend from two values : the distance d between the line and the distance h to the plane containing the line. Since the contribution of each line simply sum up, we directly obtained the field value from Equation 4.2 :

$$f_{\mathcal{S}} = \frac{2}{((\frac{d}{2})^2 + h^2)^{\frac{i-1}{2}}}.$$

Since gradient of the line skeletons are not collinear, we have to use their unit direction $\overrightarrow{u}$ in addition to Equation 4.3 :

$$\overrightarrow{u} = \frac{1}{\sqrt{\frac{d}{2}^2 + h^2}} \begin{pmatrix} \pm\frac{d}{2} \\ h \end{pmatrix}.$$

By taking into account that the first coordinate of gradients cancel each other, we obtain :

$$\|\nabla f_{\mathcal{H}}\| = (i-1)\frac{2h}{((\frac{d}{2}^2 + h^2))^{\frac{i+1}{2}}}.$$

In order to obtain the desired graph, we should express $\|\nabla f_{\mathcal{H}}\|$ in function of $f_{\mathcal{S}}$. First, we have to express h in function of $f_{\mathcal{S}}$:

$$h = \sqrt{\left(\frac{2}{f_{\mathcal{S}}}\right)^{\frac{2}{i-1}} - \left(\frac{d}{2}\right)^2}.$$

By injecting this expression in the gradient formula, we obtain :

$$\|\nabla f_{\mathcal{H}}\| = 2(i-1)\sqrt{\left(\frac{2}{f_{\mathcal{S}}}\right)^{\frac{2}{i-1}} - \left(\frac{d}{2}\right)^2}\left(\frac{f_{\mathcal{S}}}{2}\right)^{\frac{i+1}{i-1}}$$

So the graphs we are looking for are :

$$\{\mathcal{M}_d = (x, m_d(x)), x \in \mathbb{R}\}$$

with $m_d(x) = 2(i-1)\sqrt{\left(\frac{2}{x}\right)^{\frac{2}{i-1}} - \left(\frac{d}{2}\right)^2}\left(\frac{x}{2}\right)^{\frac{i+1}{i-1}}$.

We can easily check that these graphs have no point in common for distinct value of d. If it was the case, we would have a x value such that $m_{d_1}(x) = m_{d_2}(x)$, by computing $2\sqrt{\left(\frac{2}{x}\right)^{\frac{2}{i-1}} - \left(\frac{1}{2}\left(\frac{2}{x}\right)^{\frac{i+1}{i-1}} m_{d_i}(x)\right)^2}$ we directly arrive to $d_1 = d_2$ which imply that two distinct graphs have no point in common.

REFERENCES

[AC02] ANGELIDIS A., CANI M.-P. : Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *7th ACM Symposium on Solid Modeling and Applications, June, 2002* (Saarbrucken, Allemagne, June 2002), pp. 45–52.

[ADBA09] ANDERSEN V., DESBRUN M., BÆRENTZEN J. A., AANÆS H. : Height and tilt geometric texture. In *Proceedings of the 5th International Symposium on Advances in Visual Computing : Part I* (2009), ISVC '09, Springer-Verlag, pp. 656–667.

[AG01] AKKOUCHE S., GALIN E. : Adaptive implicit surface polygonization using marching triangles. *COMPUTER GRAPHICS FORUM 20* (2001), 67–80.

[AHLD07] AUJAY G., HÉTROY F., LAZARUS F., DEPRAZ C. : Harmonic Skeleton for Realistic Character Animation. In *ACM-SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), ACM.

[AJC02] ANGELIDIS A., JEPP P., CANI M.-P. : Implicit Modeling with Skeleton Curves : Controlled Blending in Contact Situations. In *International Conference on Shape Modeling and Applications (SMI'02)* (Banff, Canada, 2002), ACM, IEEE Computer Society Press, pp. 137–144.

[BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L. : Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3 (July 2006), 1180–1187. Special issue : SIGGRAPH'06.

[Bar84] BARR A. H. : Global and local deformations of solid primitives. *Special issue : SIGGRAPH84 - Comput. Graph. 18* (January 1984), 21–30.

[BBCW10] BERNHARDT A., BARTHE L., CANI M.-P., WYVILL B. : Implicit blending revisited. *Comput. Graph. Forum 29*, 2 (May 2010), 367–375.

[BDS*03] BARTHE L., DODGSON N. A., SABIN M. A., WYVILL B., GAILDRAT V. : Two-dimensional potential fields for advanced implicit modeling operators. *Comput. Graph. Forum* (2003), 23–34.

[BGA05] BARBIER ACCARY A., GALIN E., AKKOUCHE S. : A framework for modeling, animating and morphing textured implicit models. *Graphical Models 67*, 3 (2005), 166–188.

[BGSF08] BIASOTTI S., GIORGI D., SPAGNUOLO M., FALCIDIENO B. : Reeb graphs for shape analysis and applications. *Theor. Comput. Sci. 392*, 1-3 (Feb. 2008), 5–22.

[BIT04] BHAT P., INGRAM S., TURK G. : Geometric texture synthesis by example. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), SGP '04, ACM, pp. 41–44.

[BJD*12] BOROSÁN P., JIN M., DECARLO D., GINGOLD Y., NEALEN A. : Rigmesh : automatic rigging for part-based shape modeling and deformation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 198 :1–198 :9.

[BL99] BLOOMENTHAL J., LIM C. : Skeletal methods of shape manipulation. In *Proceedings of the International Conference on Shape Modeling and Applications* (1999), SMI '99, IEEE Computer Society.

[Bli82] BLINN J. F. : A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (July 1982), 235–256.

[Blo88] BLOOMENTHAL J. : Polygonization of implicit surfaces. *Comput. Aided Geom. Des. 5*, 4 (Nov. 1988), 341–355.

[Blo94] BLOOMENTHAL J. : Graphics gems iv. 1994, ch. An implicit surface polygonizer, pp. 324–349.

[Blo95a] BLOOMENTHAL J. : Bulge elimination in implicit surface blends, 1995.

[Blo95b] BLOOMENTHAL J. : *Skeletal Design of Natural Forms*. Ph.d. dissertation, University of Calgary, 1995.

[Blo97] BLOOMENTHAL J. (Ed.) : *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997.

[Blo02a] BLOOMENTHAL J. : Medial-based vertex deformation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 147–151.

[Blo02b] BLOOMENTHAL J. : Medial-based vertex deformation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 147–151.

[BN07] BOUTHORS A., NESME M. : Twinned Meshes for Dynamic Triangulation of Implicit Surfaces. In *Graphics Interface (GI '07)* (Montréal, Canada, 2007), Healey C. G., Lank E., (Eds.), ACM Press, pp. 3–9.

[BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L. : Matisse : Painting 2D regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM 2008, June, 2008* (Annecy, France, June 2008), Alvarado C., Cani M.-P., (Eds.), pp. 57–64.

[Bro05] BRONSTEIN M. : Symbolic integration. *I, 2nd Edition. Vol. 1 of Algorithms and Computation in Mathematics* (2005).

[BS91] BLOOMENTHAL J., SHOEMAKE K. : Convolution surfaces. In *Proceedings SIGGRAPH '91* (1991), ACM, pp. 251–256.

[BWdG04] BARTHE L., WYVILL B., DE GROOT E. : Controllable binary csg operators for "soft objects". *International Journal of Shape Modeling 10*, 2 (2004), 135–154.

[Can93] CANI M.-P. : An implicit formulation for precise contact modeling between flexible solids. In *20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93))* (Anaheim, United States, 1993), ACM, ACM SIGGRAPH, pp. 313–320.

[CCM13] CHERIN N., CORDIER F., MELKEMI M. : Modeling piecewise helix curves from sketches. *Computer-Aided Design, special issue of SPM 2013* (2013).

[CCSM11] CHAZAL F., COHEN-STEINER D., MÉRIGOT Q. : Geometric inference for probability measures. *Foundations of Computational Mathematics 11*, 6 (2011), 733–751.

[CGB13] CANEZIN F., GUENNEBAUD G., BARTHE L. : Adequate Inner Bound for Geometric Modeling with Compact Field Function. *Computer & Graphics (proceedings of SMI 2013) 35*, 5 (July 2013), –.

[CGD97] CANI-GASCUEL M.-P., DESBRUN M. : Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (Jan. 1997), 39–50.

[Coo84] COOK R. L. : Shade trees. *SIGGRAPH Comput. Graph. 18* (January 1984), 223–231.

[CS07] CERMAK M., SKALA V. : Polygonisation of disjoint implicit surfaces by the adaptive edge spinning algorithm of implicit objects. *Int. J. Comput. Sci. Eng. 3*, 1 (July 2007), 45–52.

[dAJ05] DE ARAUJO B. R., JORGE J. A. P. : Adaptive polygonization of implicit surfaces. *Computers & Graphics 29*, 5 (2005), 686 – 696.

[DG95] DESBRUN M., GASCUEL M.-P. : Animating soft substances with implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 287–290.

[dG08] DE GROOT E. P. : *Blobtree Modelling*. Ph.d. dissertation, University of Calgary, 2008.

[dGWvdW09] DE GROOT E., WYVILL B., VAN DE WETERING H. : Locally restricted blending of blobtrees. *Computers & Graphics 33*, 6 (2009), 690 – 697.

[DJBDT13] DEROUET-JOURDAN A., BERTAILS-DESCOUBES F., THOLLOT J. : Floating tangents for approximating spatial curves with g1 piecewise helices. *Comput. Aided Geom. Des. 30*, 5 (June 2013), 490–520.

[DTpG95] DESBRUN M., TSINGOS N., PAULE GASCUEL M. : Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Computer Graphics Forum* (1995), pp. 171–185.

[FGW01] FOX M., GALBRAITH C., WYVILL B. : Efficient use of the blobtree for rendering purposes. In *Proceedings of the International Conference on Shape Modeling & Applications* (Washington, DC, USA, 2001), SMI '01, IEEE Computer Society, pp. 306–.

[FLCB95] Fleischer K. W., Laidlaw D. H., Currin B. L., Barr A. H. : Cellular texture generation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), SIGGRAPH '95, ACM, pp. 239–248.

[GBC*13] Gourmel O., Barthe L., Cani M.-P., Wyvill B., Bernhardt A., Paulin M., Grasberger H. : A gradient-based implicit blend. *ACM Trans. Graph. 32*, 2 (Apr. 2013), 12 :1–12 :12.

[GMM11] Guibas L. J., Mérigot Q., Morozov D. : Witnessed k-distance. In *Proceedings of the twenty-seventh annual symposium on Computational geometry* (New York, NY, USA, 2011), SoCG '11, ACM, pp. 57–64.

[GMPW09] Giesen J., Miklos B., Pauly M., Wormser C. : The scale axis transform. In *Proceedings of the twenty-fifth annual symposium on Computational geometry* (New York, NY, USA, 2009), SCG '09, ACM, pp. 106–115.

[GPP*10] Gourmel O., Pajot A., Paulin M., Barthe L., Poulin P. : Fitted bvh for fast raytracing of metaballs. *Computer Graphics Forum 29*, 2 (May 2010), xxx–xxx.

[GW95] Guy A., Wyvil B. : Controlled blending for implicit surfaces using a graph. In *Implicit Surfaces '95* (Apr. 1995), pp. 107–112.

[HAC03] Hornus S., Angelidis A., Cani M.-P. : Implicit modelling using subdivision curves. *Visual Comput. 19*, 2-3 (May 2003), 94–104.

[Har94] Hart J. C. : Sphere tracing : A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12* (1994), 527–545.

[Har97] Hart J. C. : Morse theory for implicit surface modeling. In *Mathematical Visualization* (1997), Springer-Verlag, pp. 257–268.

[HC12] Hubert E., Cani M.-P. : Convolution surfaces based on polygonal curve skeletons. *Journal of Symbolic Computation 47*, 6 (2012), 680 – 699.

[HH85] Hoffmann C. M., Hopcroft J. E. : *Automatic Surface Generation in Computer Aided Design.* Tech. rep., Ithaca, NY, USA, 1985.

[HIIT97] Hilton A., Illingworth J., Implicit D., Triangulation S. : Marching triangles : Delaunay implicit surface triangulation, 1997.

[HT11] Harary G., Tal A. : The natural 3D spiral. *Computer Graphics Forum (Proceedings of Eurographics) 30*, 2 (2011).

[Hub12] Hubert E. : Convolution surfaces based on polygons for infinite and compact support kernels. *Graphical Models 74*, 1 (2012), 1 – 13.

[JLSW02] Ju T., Losasso F., Schaefer S., Warren J. : Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 339–346.

[JLW10] Ji Z., Liu L., Wang Y. : B-mesh : A modeling system for base meshes of 3d articulated shapes. *Computer Graphics Forum (Proceedings of Pacific Graphics) 29*, 7 (2010), 2169–2178.

[JT02a] Jin X., Tai C.-L. : Analytical methods for polynomial weighted convolution surfaces with various kernels. *Computer Graphics 26*, 3 (2002), 437–447.

[JT02b] JIN X., TAI C.-L. : Convolution surfaces for arcs and quadratic curves with a varying kernel. *The Visual Computer 18*, 8 (2002), 530–546.

[JTFP01] JIN X., TAI C.-L., FENG J., PENG Q. : Convolution surfaces for line skeletons with polynomial weight distributions. *J. Graph. Tools 6*, 3 (2001), 17–28.

[JTZ09] JIN X., TAI C.-L., ZHANG H. : Implicit modeling from polygon soup using convolution. *Vis. Comput. 25*, 3 (Feb. 2009), 279–288.

[KB89] KALRA D., BARR A. H. : Guaranteed ray intersections with implicit surfaces. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 297–306.

[LC87] LORENSEN W. E., CLINE H. E. : Marching cubes : A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 163–169.

[LLD11] LAGAE A., LEFEBVRE S., DUTRÉ P. : Improving Gabor noise. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (August 2011), 1096–1107.

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P. : Procedural noise using sparse Gabor convolution. *ACM Trans. Graphics 28*, 3 (2009), 54 :1–54 :10.

[Lon98] LONCARIC S. : A survey of shape analysis techniques. *Pattern Recognition 31* (1998), 983–1001.

[lTZkF04] LAN TAI C., ZHANG H., KIN FONG J. C. : Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum 23* (2004), 71–83.

[Mit90] MITCHELL D. P. : Robust ray intersection with interval arithmetic. In *Proceedings on Graphics interface '90* (Toronto, Ont., Canada, Canada, 1990), Canadian Information Processing Society, pp. 68–74.

[MS98] MCCORMACK J., SHERSTYUK A. : Creating and rendering convolution surfaces. *Computer Graphics Forum 17*, 2 (1998), 113–121.

[NHK*85] NISHIMURA H., HIRAI M., KAWAI T., KAWATA T., SHIRAKAWA I., OMURA K. : Object modeling by distribution function and a method of image generation. *Transactions IECE Japan 4* (1985), 718–725.

[Nie04] NIELSON G. M. : Dual marching cubes. In *Proceedings of the conference on Visualization '04* (Washington, DC, USA, 2004), VIS '04, IEEE Computer Society, pp. 489–496.

[NN94] NISHITA T., NAKAMAE E. : A method for displaying metaballs by using bézier clipping. *Computer Graphics Forum 13* (1994), 271–280.

[NY06] NEWMAN T. S., YI H. : A survey of the marching cubes algorithm. *Computers & Graphics 30* (2006), 854 – 879.

[OC97] OPALACH A., CANI M.-P. : Local deformation for animation of implicit surfaces. In *Spring Conference on Computer Graphics (SCCG)* (Bratislava, Slovakia, Jun 1997).

[PASS95] PASKO A., ADZHIEV V., SOURIN A., SAVCHENKO V. : Function representation in geometric modeling : concepts, implementation and applications. *The Visual Computer 11*, 8 (1995), 429–446.

[PBFJ] PORUMBESCU S. D., BUDGE B., FENG L., JOY K. I. : Shell maps. In *ACM SIGGRAPH 2005 Papers*, ACM, pp. 626–633.

[PCP10] PIHUIT A., CANI M.-P., PALOMBI O. : Sketch-based modeling of vascular systems : a first step towards interactive teaching of anatomy. In *Sketch-Based Interfaces and Modeling, SBIM 2010, June, 2010* (Annecy, France, June 2010), Alexa M., Do E. Y.-L., (Eds.), Eurographics Association, pp. 151–158.

[Pea85] PEACHEY D. R. : Solid texturing of complex surfaces. *SIGGRAPH Comput. Graph. 19* (July 1985), 279–286.

[Ped95] PEDERSEN H. K. : Decorating implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 291–300.

[Per85] PERLIN K. : An image synthesizer. *SIGGRAPH Comput. Graph. 19* (July 1985), 287–296.

[PGMG09] PEYTAVIE A., GALIN E., MERILLOU S., GROSJEAN J. : Arches : a Framework for Modeling Complex Terrains. *Computer Graphics Forum (Proceedings of Eurographics) 28*, 2 (2009), 457–467.

[PH89] PERLIN K., HOFFERT E. M. : Hypertexture. *SIGGRAPH Comput. Graph. 23* (July 1989), 253–262.

[PPIK02] PASKO G., PASKO A., IKEDA M., KUNII T. : Bounded blending operations. In *Proceedings of the Shape Modeling International 2002 (SMI'02)* (Washington, DC, USA, 2002), SMI '02, IEEE Computer Society, pp. 95–.

[PPK05] PASKO G. I., PASKO A. A., KUNII T. L. : Bounded blending for function-based shape modeling. *IEEE Comput. Graph. Appl. 25*, 2 (Mar. 2005), 36–45.

[Ric73] RICCI A. : Constructive geometry for computer graphics. *Computer Journal 16*, 2 (1973), 157–60.

[RO85] ROCKWOOD A., OWEN J. C. : Blending surfaces in solid modeling. *Proceedings of SIAM Conference on Geometric Modelling and Robotics* (1985).

[Roc89] ROCKWOOD A. P. : The displacement method for implicit blending surfaces in solid models. *ACM Trans. Graph. 8*, 4 (Oct. 1989), 279–297.

[RPC*10] ROHMER D., POPA T., CANI M.-P., HAHMANN S., SHEFFER A. : Animation wrinkling : augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 157 :1–157 :8.

[SAC*99] STORA D., AGLIATI P.-O., CANI M.-P., NEYRET F., GASCUEL J.-D. : Animating lava flows. In *Proceedings of the 1999 conference on Graphics interface '99* (San Francisco, CA, USA, 1999), Morgan Kaufmann Publishers Inc., pp. 203–210.

[SdGWS08] SUGIHARA M., DE GROOT E., WYVILL B., SCHMIDT R. : A sketch-based method to control deformation in a skeletal implicit surface modeler. In *Proceedings of the 5th Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2008), pp. 65–72.

[SGW06] SCHMIDT R., GRIMM C., WYVILL B. : Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph. 25*, 3 (July 2006), 605–613.

[SH97] STANDER B. T., HART J. C. : Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of the 24th* *annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 279–286.

[She99a] SHERSTYUK A. : *Convolution Surfaces in Computer Graphics*. Ph.d. dissertation, School of Computer Science and Software Engineering, Monash University, 1999.

[She99b] SHERSTYUK A. : Fast ray tracing of implicit surfaces. *Computer Graphics Forum 18*, 2 (1999), 139–147.

[She99c] SHERSTYUK A. : Interactive shape design with convolution surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications* (1999), IEEE Computer Society, pp. 56–.

[She99d] SHERSTYUK A. : Kernel functions in convolution surfaces : A comparative analysis. *The Visual Computer 15*, 4 (1999), 171–182.

[Sny92] SNYDER J. M. : Interval analysis for computer graphics. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 121–130.

[SPB96] SHERBROOKE E., PATRIKALAKIS N., BRISSON E. : An algorithm for the medial axis transform of 3d polyhedral solids. *Visualization and Computer Graphics, IEEE Transactions on 2*, 1 (1996).

[SS11] STAM J., SCHMIDT R. : On the velocity of an implicit surface. *ACM Trans. Graph. 30*, 3 (May 2011), 21 :1–21 :7.

[SW04] SCHAEFER S., WARREN J. : Dual marching cubes : Primal contouring of dual grids. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (Washington, DC, USA, 2004), PG '04, IEEE Computer Society, pp. 70–76.

[SWG05] SCHMIDT R., WYVILL B., GALIN E. : Interactive implicit modeling with hierarchical spatial caching. In *Proceedings of the International Conference on Shape Modeling and Applications 2005* (Washington, DC, USA, 2005), SMI '05, IEEE Computer Society, pp. 104–113.

[SWS10] SUGIHARA M., WYVILL B., SCHMIDT R. : Warpcurves : A tool for explicit manipulation of implicit surfaces. *Computers & Graphics 34*, 3 (2010), 282 – 291. <ce :title>Shape Modelling International (SMI) Conference 2010</ce :title>.

[SWSJ07] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A. : Shapeshop : sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), SIGGRAPH '07, ACM.

[TCW99] TIGGES M., CARPENDALE S., WYVILL B. : Alternate distance metrics for implicit surface modeling. *Graphics Interface 1999 Poster Session,Kingston Canada.* (June 1999).

[TW98] TIGGES M., WYVILL B. : Texture mapping the blobtree. In *In Proceedings of the Third International Workshop on Implicit Surfaces* (1998), pp. 123–130.

[TW99] TIGGES M., WYVILL B. : A field interpolated texture mapping algorithm for skeletal implicit surfaces. *Proc. CG International 99* (1999), 25–33.

[VBG*13] VAILLANT R., BARTHE L., GUENNEBAUD G., CANI M.-P., ROHMER D., WYVILL B., GOURMEL O., PAULIN M. : Implicit skinning : real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July 2013), 125 :1–125 :12.

[WGG97] WYVILL B., GALIN E., GUY A. : The Blob Tree, Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *University of Calgary technical report* (July 1997).

[WGG99] WYVILL B., GUY A., GALIN E. : Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2 (1999), 149–158.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B. : Data structure for soft objects. *The Visual Computer 2*, 4 (Aug. 1986), 227–234.

[WO97] WYVILL B., OVERVELD K. V. : Warping as a modelling tool for csg/implicit models. In *Proceedings of the 1997 International Conference on Shape Modeling and Applications (SMI '97)* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 205–214.

[WT90] WYVILL G., TROTMAN A. : Ray-tracing soft objects. In *CG International.* Springer Japan, 1990, pp. 469–476.

[WW00] WYVILL B., WYVILL G. : Better blending of implicit objects at different scales. *ACM Siggraph 2000 presentation* (2000).

[WWM87] WYVILL G., WYVILL B., MCPHEETERS C. : Solid texturing of soft objects. In *CG International '87 on Computer graphics 1987* (New York, NY, USA, 1987), Springer-Verlag New York, Inc., pp. 129–141.

[YT13] YU J., TURK G. : Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph. 32*, 1 (Feb. 2013), 5 :1–5 :12.

[ZGV*98] ZONENSCHEIN R., GOMES J., VELHO L., DE FIGUEIREDO L., TIGGES M., WYVILL B. : Texturing composite deformable implicit objects. In *Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision* (1998), SIBGRAPHI '98, IEEE Computer Society, pp. 346–.

[ZGVdF97] ZONENSCHEIN R., GOMES J., VELHO L., DE FIGUEIREDO L. H. : Texturing implicit surfaces with particle systems. In *Visual Proceedings : The art and interdisciplinary programs of SIGGRAPH '97* (1997), ACM, p. 172.

[ZGVdF98] ZONENSCHEIN R., GOMES J., VELHO L., DE FIGUEIREDO : Controlling texture mapping onto implicit surfaces with particle systems. In *Proceedings of the Third International Workshop on Implicit Surfaces* (July 1998), pp. 131–138.

[ZJLZ12] ZHU X., JIN X., LIU S., ZHAO H. : Analytical solutions for sketch-based convolution surface modeling on the gpu. *The Visual Computer 28*, 11 (2012), 1115–1125.