N-ary Implicit Blends with Topology Control

author preprint

The definitive version is available at `www.sciencedirect.com`

C. Zanni[a], M. Gleicher[b], M.-P. Cani[a]

[a] *Laboratoire Jean Kuntzmann (Grenoble University, CNRS) and Inria*
[b] *Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA*

Abstract

Constructive implicit surfaces are attractive for modeling and animation because they seamlessly handle shapes with complex and dynamic topology. However, the way they merge shapes is difficult to control. This paper introduces a solution: an improved blend operator that provides control over how topology changes are handled. It is based on a correction applied to the standard blending operator: the sum. Building on summation preserves the n-ary nature of the blend, providing the simplicity of arbitrary (e.g. flat) construction trees and segmentation invariance. The correction is based on projection to a reference case in the variation-space defined by the field and the norm of its gradient. It provides a single parameter, allowing for tuning behavior to achieve effects ranging from avoiding topological combination, through merging only during overlap, to merging at a distance. Dynamic adjustment of the parameter allows for context-dependent effects. Applications range from skeleton-based modeling, where shapes keep the topology of their skeleton, to objects that change topology during animation, with controllable merging. We illustrate the latter with Manga-style hair, where merging depends on the angle between hair wisps.

Figure 1: Topology control for implicit models can be used to prevent blending at distance and guarantee the topology of skeleton-based surfaces (left) or to control the dynamic topology of Manga-style hair (right), where blending should not only depend on distance but also on the angle between neighboring hair-wisps.

1. Introduction

Implicit surface blending is an attractive tool for modeling and animation. Its ability to adapt to the underlying topology means that it can be used to create seamless, smooth surfaces from a collection of parts. This frees the user from being concerned with how to connect different pieces, or from how the topology may change as parts move and interact with each other. However, this topological freedom has also limited the application of implicit modeling: while users do not need to consider how pieces will combine, they also cannot control

this combination. So while shape components tend to merge when they come closer and to separate when they move apart, there is no way to control when topological changes will occur. For instance implicit drops of water will deform and merge before they collide or the arm of an implicit character may merge with its body if they come too close. With no control over the topology, implicit blending is limited to "soft" objects without crisp boundaries, and is unable to model selective combination effects.

While some recent methods addressed the topology control issue, they only did so through binary combination operators. This brings several drawbacks: Firstly, being restricted to binary blends complicates modeling and increases the cost of field queries, due to the larger depth of binary trees compared to n-ary ones. Secondly, it causes the loss of *segmentation-invariance*, the ability to create skeletal surfaces that do not depend on the way their skeleton is tessellated, enabling seamless refinement and tuning during interactive design. So far, these drawbacks have limited the use of topology-controlled blending in skeletal implicit modeling and animation. In contrast, N-ary blending enables to combine an arbitrary number of primitives using a single blending node. For example, summation is an N-ary blending operator that provides a simple way to smoothly blend implicit surfaces in an order-independent manner, however, it cannot provide control over topology. See Figure 2.

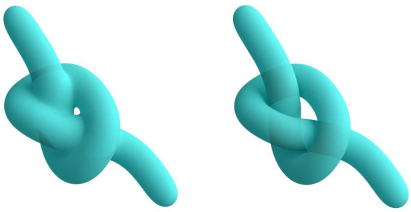

Figure 2: A large number of segment-skeletons are used to generate an implicit surface, built with the SCALIS model. Using blending by summation (a) brings segmentation invariance, but does not produce the expected topology. Using our method (b) enables us to keep the topology of the union of the input primitives.

The goal of this work is to develop an N-ary blending operator that provides topology control while maintaining segmentation invariance, and is applicable to any family of skeleton-based implicit surface. The topological control we are looking for should allow a range of different merging behaviors both for static and dynamic examples. Figure 3 depicts four examples of behaviors

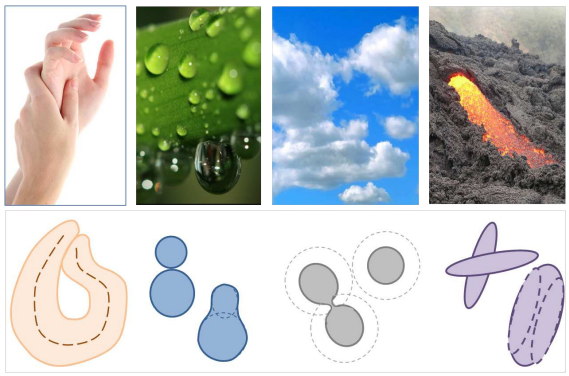

Figure 3: Taxonomy of blending behaviors, with real example on top and schematic illustration at the bottom. From left to right: Skeletal-blend, contact-blend, distance-blend and context-dependent-blend. Note that context is the temperature for the lava flow, but is the orientation of the primitives in the bottom sketch, where we illustrate the angle-dependent blending we need for hair wisps.

that we can observe either in the real world, or for existing imaginary objects. We seek to provide this range:

Skeletal-blend is needed to model organic shapes such as characters, animals or trees. While implicit modeling is useful in these cases to generate a smooth shape around a ramified skeletal structure, shape-parts should remain distinct rather than merge when they come to contact. Therefore, the topology of this type of shapes should always remain the one of their internal skeleton, such as the dragon in Figure 1 left.

Contact-blend is designed for animating shapes that should deform and merge if and only if they come into contact. Moreover, when they start merging, they should do so in a progressive way, as drops of water. The topology of this type of shape always remains the same as the one of the union the blended primitives.

Distance-blend captures shapes that deform and merge when they come close to each other, before colliding. Clouds are an example of such behavior among natural phenomena. This behavior was also used for implicit deformers modeling dynamic garment folds [1]. The topological genus of this type of shape is smaller or equal to the one of the union the blended primitives.

Context-dependent blend is used in cases when the blending behavior, among the three above, should change in space and/or time depending on context. In nature, this is the case for instance for lava flows, where blocks of lava melt and merge at high temperature, but separate to form a highly granular crust when the flow cools down. Their behavior then spans from distance-blend to contact-blend and then to skeletal-blend, de-

pending on the temperature. This range of behaviors can also be observed in the quite different case of imaginary, volumetric hair, where the context is given by the local distribution of wisp directions. Neighboring wisps should merge at small distance when their orientation is similar, but squash and repulse each other in other cases (Figure 15, right).

In this work, we introduce an N-ary blending operator enabling to span between objects that deform rather than merge, merge on contact, or merge at distance. The new operator can be used to combine any number of arbitrary skeleton-based implicit primitives, provided they are all generated using the same kernel function. This includes sets of primitives defined using skeletons of different dimensions such as points, curves or surfaces, which makes our solution applicable to a large range of shapes. Topology control is performed by tuning a single parameter. This provides a simple way to model context-dependent blending effects by dynamically adjusting the parameter based on context, enabling effects such as materials with clumping phenomena like hair.

The basic idea of our approach is to use the standard sum operator, while correcting its results to avoid unwanted topological changes. This allows us to retain the flexibility of n-ary additive blends, but also to achieve topological control. The key challenge is to define the adjustment anywhere in space, and for a wide range of implicit surface types. To provide this generality, we express the correction as a function of the result of the sum operator expressed in *variation space* (defined as the cartesian product of possible field values and norms of associated gradients). The enabling insight is to define the correction through projection, in this space, onto a reference case that none of our adjustments should change. The projection angle provides the topology-control parameter we were looking for: tuning it enables to switch between the different merging behaviors we already listed.

To present our approach, we first review previous solutions for controlling implicit blends in Section 2. We then introduce our general method for defining n-ary, controllable blends in Section 3. We detail the implementation of our projection operator in Section 4. Our solutions for using the projection angle for topology control are presented in Section 5. We finally discuss use cases in Section 6.

2. Related Work

Skeleton-based implicit surfaces, built by blending primitive shapes, where introduced as an easy way to model complex and dynamic topology [2, 3]. They have been used since then for rendering water and viscous material [4, 5, 6], for combining shape components in sketch-based modeling applications [7], for animating garment folds that merge rather than collide [1] and recently, for improving character skinning [8].

Blobby surfaces [2, 3] are defined by decreasing functions of the distance to skeleton-points. Distance surfaces extend this to decreasing functions of the distance to any geometric skeleton (a curve, a surface, or even a volume). However, using the sum of field contributions as blending operator generates non-desired bulges at junctions for this model, when graphs of 1D or 2D skeletal components are used. Convolution surfaces solve this problem by defining the field as the integral of point-contributions along the skeleton [9, 10]. Therefore, they generate smooth shapes that are independent from the way the skeleton is split into primitives. This model was recently extended to scale-invariant integral surfaces (SCALIS) providing radius control and limiting the blur of details [11].

Unfortunately, using the standard summation blend makes the amount of blending between primitives, and in particular whether topology changes are going to occur, difficult to predict. These changes depends on the slope of the kernel function used. Moreover, the fact that shapes blend at distance is often un-desired: for instance implicit water droplets will start to merge before they collide, and the hand of a character may blend with its body if they come close. Therefore, the topology of a skeleton-based implicit shape does not always reflect the one of its skeleton, in contrast to the skeletons built in shape analysis methods, which are there to encode the shape topology [15]. This *unwanted blending problem*[1] was identified years ago, leading to a number of solutions.

One early approach monitored topology changes during animations, in order to improve meshing [14]. Other early methods used static or dynamic *blending graphs* to define which primitives were allowed to blend [16, 5], but this generated shape discontinuities. [17] improved the blending graph approach by introducing decay functions to avoid discontinuities, but this complex solution was restricted to connected skeleton graphs, and did not handle arbitrary branchings.

Recent solutions use binary blending operators for preventing blending at distance, and more precisely for insuring that the model always keeps the topology of

[1] Note that this problem does not occur in level set implicit models [12, 13], but we focus here on skeleton representation, which enable direct shape modeling and animation.

the union of the input surfaces [18, 19, 20]. The first ones restrict blending to the inside of a geometric primitive defined around the intersection. [20] rather defines a blending operator depending not only on field values, but also on the angle between the gradients of the two input primitives.

In this paper, we present the first solution which provides topology control for n-ary blends. In contrast with previous work, our approach not only captures blends that preserve the topology of the union of the input shapes, but also provides control to achieve other effects when desired. As [20], we use the field gradient to identify regions where blending should be allowed, but this is done using the norm of the resulting gradient after an additive blend, in contrast with the original method based on the angle between two input gradients. Therefore, unlike the prior work, our approach is not limited to binary blends.

3. N-ary blending operator based on summation

This section first gives some background on implicit modeling to clarify our notations. It then presents the three key-ideas to our method: defining our new blending operator as a correction to the additive blend; using the norm of the resulting gradient - together with the field value - to characterize blending regions; and lastly, using projection to a reference case in the resulting variation space, to define the correction.

3.1. Background: Skeleton-based implicit modeling

Implicit surfaces are defined as the set of points $\mathbf{p}$ in space where $f(\mathbf{p}) = iso$, where f is a given scalar field and iso an iso-value [21]. The interior of an object are the points of space where $f(\mathbf{p}) > iso$. The iso-value is a part of the shape definition. In this work, we therefore keep it to a fixed value. Different surfaces can be blended by combining their fields' contributions. The most simple blending operator is the *sum*, used to smoothly blend an arbitrary number of the input primitives:

$$f(\mathbf{p}) = \sum_i f_i(\mathbf{p}) \tag{1}$$

where f_i is the field function defining the ith primitive. The combination of a number of input primitives using various successive blends is stored into a construction tree.

We now give the equations for all the implicit models we will use; all these primitives are based on distance to skeleton computations. Given a point in space $\mathbf{p}$ and a point on the skeleton $\mathbf{q}$, we will note the distance computation :

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| \tag{2}$$

Let k be the decreasing function of the distance (or *kernel*) used to define our implicit primitives. Let τ_{S_i} be the radius control parameter, which can either be constant or vary over the skeleton S_i. We define scale-invariant distance to a skeleton point $\mathbf{q}$ with radius $\tau_{S_i}(\mathbf{q})$ as :

$$e_{S_i}(\mathbf{p}, \mathbf{q}) = \frac{d(\mathbf{p}, \mathbf{q})}{\tau_{S_i}(\mathbf{q})} \tag{3}$$

Then the distance primitive generated by S_i and τ_{S_i} is defined in a scale-invariant way using:

$$f_i(\mathbf{p}) = \min_{\mathbf{q} \in S_i} k \circ e_{S_i}(\mathbf{p}, \mathbf{q}) \tag{4}$$

Similarly, the scale-invariant integral primitive [11] for skeleton S_i and radius τ_{S_i} is defined through:

$$f_i(\mathbf{p}) = \frac{1}{N_k} \int_{S_i} \frac{k \circ e_{S_i}(\mathbf{p}, \mathbf{q})}{\tau_{S_i}(\mathbf{q})^\delta} \, \mathrm{d}\mathbf{q} \tag{5}$$

where δ is the dimension of S_i (0 for points, 1 for line-skeletons, 2 for surface-skeletons) and N_k a normalization factor depending only on the kernel used.

Note that if S_i is a point-skeleton, both formula give the same result, which is the usual way of defining the field function of a blob:

$$f_i(P) = k \circ e_{S_i}(\mathbf{p}, \mathbf{q}) \tag{6}$$

The method developed in this paper holds for all of these implicit models, which can be seamlessly combined in the same scene using our blending method.

Union of balls: In the remainder of the paper, the union of balls refers to the infinite union of all the balls centered on the skeleton of an implicit shape and whose radius is the local radius value τ at the center point. When the skeleton is composed of several distinct pieces, the union of balls corresponds to a sharp union of the unions of balls of the different skeletal parts, with no blending nor smoothing. In the remainder of this paper (e.g. Figures 6, 7, 12), unions of balls will be used for comparison with the new blends we are defining.

3.2. Correction on top of a sum

Sum is a simple, well-known blending operator. It has a number of important properties that any new operator must preserve. Appart from Ricci's blending [22], it is the only smooth n-ary blending operator known so

far. It fits well with convolution and SCALIS surfaces defined from an integral along a skeleton, since the linearity of the integral (the integral of a sum of functions is the sum of the integrals of these functions) brings the segmentation-invariant property.

To preserve these nice properties of the sum operator, our new operator is built on top of it: the method first computes a sum of all field contributions, and then applies a correction C that maps field values to other ones:

$$\hat{f} = C(f) = C\left(\sum_i f_i\right)$$

This is a key point of our solution: it enables us to keep the n-ary nature of the blend and therefore to handle an arbitrary number of objects and complex skeleton graphs through a single blend of all their individual primitives.

3.3. Using the norm of a scale-invariant gradient

Using not only fields, but also gradients, proved useful for controlling binary blends [20]. However, while the previous method used a binary blend parameterized by the angle between the input gradient vectors, we adopt a different strategy, enabling us to preserve the n-ary nature of the blend: we use the norm of the resulting gradient vector, after a standard sum. Indeed, space points where topology changes are singular points of the gradient field [14], i.e. points where the norm of the gradient vanishes. Therefore, the norm of the gradient tells us something about how close the query point is to regions where topology is going to change.

To give more insight, let us consider the simple scenario depicted in Figure 4. At the right of the figure, the field computed using sum tends to be too high in regions where the norm of the gradient is small, such as at point $\mathbf{p}_1$, where the shape folds onto itself. We would therefore like to decrease the field value there. In contrast, the ratio between the field value resulting from the sum and the norm of its gradient remain the same on $\mathbf{p}_2$ when the shape folds. We have no correction to apply there, since there is no risk for unwanted blending.

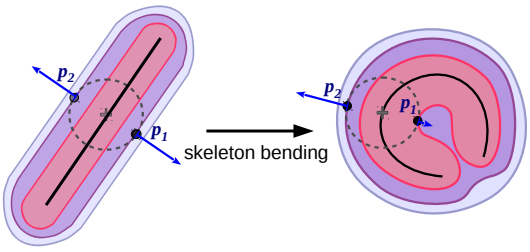

Figure 4: Characterizing regions where blending is too high when a shape bends (right picture): Too much blending at distance occurs in inner regions, such as at $\mathbf{p}_1$, while no unwanted blending occurs at $\mathbf{p}_2$. These two cases can be identified by comparing the proportionality between the norm of the gradient and the field value, compared to the reference case on the left.

The norm of the gradient should therefore be one of the input of the correction function C. However, to be able to get a measure which is applicable to both small scale and large scale objects, we need a *scale invariant gradient*.

While the scalar field of implicit primitives defined from equations (4), (5) and (6) are invariant through scaling of the shape, this is not true for their gradient fields. Indeed, small shapes have sharper fields and therefore larger gradient norms. Mathematically, when we derive the term $k \circ e_{S_i}(\mathbf{p}, \mathbf{q})$ that appears in (4), (5) and (6) with respect to the distance $d(\mathbf{p}, \mathbf{q})$, we get $\frac{1}{\tau_{S_i}(\mathbf{q})}\, k' \circ e_{S_i}(\mathbf{p}, \mathbf{q})$. The factor $\frac{1}{\tau_{S_i}(\mathbf{q})}$ explains the change of scale of the gradient, which is inversely proportional to the primitive size. We can therefore easily define a scale-invariant gradient as $\nabla_S f$ by using $\tau_{S_i}(\mathbf{q})\ k$ instead of k when computing the space derivatives of the field.

We add the norm of the resulting scale invariant gradient as an input of the correction operator C:

$$\hat{f} = C(f, \|\nabla_S f\|) \tag{7}$$

We are now looking for a function C defined over the 2D cartesian product space of all $(f, \|\nabla_S f\|)$ values taken by the sum of the input primitives. We call this space the *variation-space* of f.

3.4. Projection to a reference case

We define the correction to be applied to the sum of the input fields through projection, in the variation space we just defined, onto a reference case that none of our adjustments should change. This is done as follows:

As shown in Figure 4, extra blending in the concave parts of a shape may cause unwanted topological changes. But when the skeleton is a line, there is no risk for extra blending, so no correction is necessary. Making a correction in this case would even break the segmentation invariance property: the shape generated by aligned segments forming a line would then be subject to bulges or creases where segments joint. Therefore, we choose an infinite line skeleton as our reference case, where the field value given by the sum should be kept unchanged.

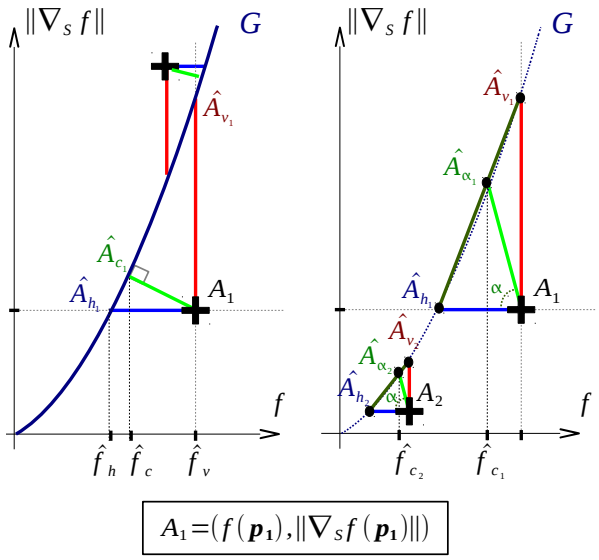

Figure 5: Left: Curve $\mathcal{G}$ representing the reference case in variation space. Vertical, horizontal, and closest point projections are depicted for point A_1. Right: Projection in a fixed direction, approximated using the intersection between the line in green and the line passing through vertical and horizontal projections.

Let us look at the specific correspondence between field and norm of gradient values created everywhere in the 3D space by the reference case we just defined (an infinite line skeleton). When plotting these values in variation space (see Figure 5), we note that all points fall onto a single curve $\mathcal{G}$, that entirely spans both axes. Mathematically speaking, our reference case defines a bijective mapping between the field and the norm of the scale-invariant gradient for all kernels k of interest (i.e. when both k and its first derivative k' are strictly decreasing functions). For such kernels, there exist a bijective mapping g such that

$$\forall \mathbf{p} \in \mathbb{R}^3 \ , \ \|\nabla_S f(\mathbf{p})\| = g(f(\mathbf{p})),$$

and $\mathcal{G} = \{f, g(f)\}$ is the graph of this function. The exact formula for this curve depends on the kernel k used and will be derived in Section 4.1. Note that all isolated implicit primitive defined using the same kernel k (for instance generated by a point-skeleton, a segment-skeleton or a triangle-skeleton) exactly match the reference case in variation space. Apart from them, forming a curve in variation space is specific to the reference case: Most implicit shapes, such as the one at the right of Figure 4 span non-bounded 2D regions in variation space.

To keep the field unchanged in the reference case, the operator C we are looking for should output the original scalar field value f for points located on $\mathcal{G}$ in variation space; C should thus coincide with the abscissa operator when applied on the curve $\mathcal{G}$.

Our key insight for defining C everywhere else in variation space is to use a *projection* to the curve $\mathcal{G}$ before taking the abscissa:

$$C(f, \|\nabla_S f\|) = \hat{A}_x.$$

where $\hat{A}$ is a projection of $A = (f, \|\nabla_S f\|)$ onto the curve. Using a projection insures that neither the reference case nor any isolated implicit primitive (sharing the same graph in variation space) will not be affected by our corrections. Moreover, the further A is from $\mathcal{G}$, the larger the correction is (i.e. the difference between input and output abscissas), which is the desired behavior: this will enable large field values to be reduced sufficiently when the associated norm of gradient is too small, such as at point A_1 in Figure 5.

Note that the operator C corrects field values only. The gradient is recomputed numerically from the local corrected field values around a query point.

Let us now discuss the choice of the projection operator: There are different simple ways to project A onto the curve $\mathcal{G}$, Each leading to a different correction to the blending behavior: we could think for instance of vertical or horizontal projections, and projection to the closest point, as depicted in Figure 5 (left)). If we use a vertical projection onto the curve, the field will not change. Our new operator will then be identical to blending by summation. In contrast, horizontal projection corresponds to the largest possible correction: the field will heavily decrease for all points at the right of the curve (smaller gradient than the reference case for a given field value), which reduces blending but may also create undesired empty gaps between primitives. Using projection to the closest point on $\mathcal{G}$ could seem the most natural solution: however, for a given input field value, the correction would become closer to the maximal one (horizontal projection) when the norm of gradient increases, while we would like the opposite. Moreover, using this single, predefined projection would not provide the control we need for capturing the different behaviors depicted in Figure 3.

In this work, we rather use projection in an arbitrary, fixed direction: The projection angle then provides the continuous parameter we were looking for to allow topology control. It enables to span between horizontal and vertical projection, the two extreme behaviors we just described. Moreover, if the kernel is well chosen, projection can be computed efficiently and key

values of the projection angle parameter can be explicitly computed. The methods are detailed in the next section.

4. Implementing the operator

4.1. Practical choices

In practice, implementing the correction C is easier with a kernel that provides a closed-form solution for the bijective mapping between the field and the norm of scale-invariant gradient in the reference case. This is the case for inverse kernels of arbitrary degree n, defined by $k(d) = \left(\frac{1}{d}\right)^n$. When integrated along an infinite skeleton line with formula 5, it yields:

$$f(\tau, d) = \left(\frac{\tau}{d}\right)^{n-1} \tag{8}$$

$$\|\nabla_{\mathcal{S}} f\|(\tau, d) = (n-1)\left(\frac{\tau}{d}\right)^{n} \tag{9}$$

where d is the distance to the line and τ the required radius along it (derivation of the formula is provided in section 1.1 of the supplemental material). From equations (8) and (9) we get the bijective mapping :

$$\|\nabla_{\mathcal{S}} f\| = (n-1) f^{\frac{n}{n-1}} \text{ and } f = \left(\frac{\|\nabla_{\mathcal{S}} f\|}{n-1}\right)^{\frac{n-1}{n}} \tag{10}$$

The reference curve $\mathcal{G}$ in variation space is therefore defined as

$$g(f) = (n-1) f^{\frac{n}{n-1}}, \text{ and } \mathcal{G} = \{f, g(f)\} \tag{11}$$

In the remainder of this paper, we develop our solution for this specific kernel, which we use for blobs, for distance surfaces and for integral surfaces.

Skeletons of arbitrary dimensions. Although being one dimensional, the reference case we rely on can be used for any set of input implicit primitives. Because the primitives we use are all scale-invariant (see Section 2), their images in variation space are the same as for point-skeletons: if we add the extra constraint of choosing inverse kernels of degree $n+1$ for point-skeletons, n for curve-skeletons and $n-1$ for surface-skeletons, then the image in variation space (i.e. Eq. 11) of all integral primitives generated from skeleton of "infinite" size (line and plane) is the same as for point-skeletons. This holds true for other kernels such as the Cauchy kernel and the Compact Polynomial kernel. The derivation of the formula for plane is similar to the above derivation for a line, and can be found in section 1.2 of the supplemental material.

In practice, we impose this extra constraint on the degrees of the inverse kernels used, while n can be used to tune the smoothness of the resulting shapes. Since the fields of all input surfaces refer to the same reference case, we can seamlessly blend them whatever their dimension by directly summing their fields and then applying a single correction, through projection in variation space.

Projection in a fixed direction. The projection we use in equation 7 is computed in a fixed direction defined by a projection angle α, as shown in Figure 5, right. Note that while the parameter usually varies between 0 and $\frac{\pi}{2}$, negative projection angles can be used as well. The projection of point $A = (f, \|\nabla_{\mathcal{S}} f\|)$ in variation space is given by the intersection between the curve $\mathcal{G}$ and the line $\mathcal{D}$ passing through A with the prescribed direction :

$$\mathcal{D} = \{(x, -\tan(\alpha)x + \|\nabla_{\mathcal{S}} f\| + \tan(\alpha) f) \;/\; x \in \mathbb{R}\}.$$

Although this intersection could be computed analytically for low kernel degrees, we simplify the computation by locally approximating $\mathcal{G}$ with a line. We choose two points on $\mathcal{G}$, corresponding to the horizontal and vertical projections, to define the approximation (Figure 5 (right)). This insures the exact values are computed for these two key points. The distances to these two points are computed in closed form. Let l_H being the signed distance between A and the horizontal projection:

$$l_H = f - \left(\frac{\|\nabla_{\mathcal{S}} f\|}{n-1}\right)^{\frac{n-1}{n}},$$

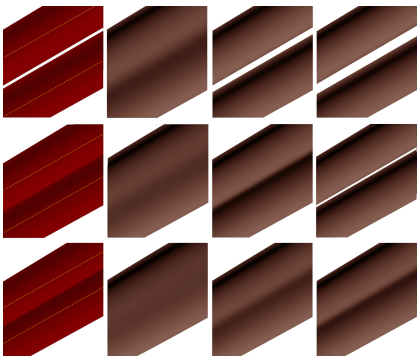

Figure 6: Top to bottom: blending when segment-primitives of equal radius come close to each other. From left to right: union of balls inside the shapes, vertical projection (sum), projection with angle $\alpha = \frac{\pi}{4}$, and horizontal projection.

and l_V being the signed distance between A and the vertical projection:

$$l_V = (n-1) f^{\frac{n}{n-1}} - \|\nabla_{\mathcal{S}} f\|,$$

then the corrected scalar field can be computed by interpolation:

$$\hat{f}_\alpha = f - \frac{l_H l_V}{l_V + l_H \tan(\alpha)}. \tag{12}$$

Note that in the case of horizontal projection ($\alpha = 0$), the new field value will only be defined by the scale-invariant gradient :

$$\hat{f}_{\alpha=0} = \left(\frac{\|\nabla_{\mathcal{S}} f\|}{n-1}\right)^{\frac{n-1}{n}} \tag{13}$$

The quality of this approximation increases as the computation point position in variation space moves closer to the reference case (in which case there is no approximation error). Note that an alternative to this approximation scheme would be the use of numerical root finding.

The projection angle α provides a single control parameter that provides a variety of behaviors, see Figures 6 and 7 for examples. In order to avoid numerical instabilities when α tend toward $\frac{\pi}{2}$, it is possible to slightly change the way equation 12 is computed. The alternative formulation is provided in the supplemental material.

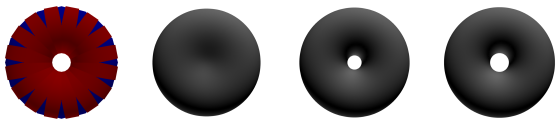

Figure 7: Torus modeled from a circular skeleton, using different values for the projection angle.

4.2. Avoiding unwanted cavities

One problem with the above method is that the correction may introduce small unwanted voids inside the blended interfaces when merging is reduced, but not eliminated. This *cavity problem* can be seen in Figure 8. The problem comes from non-monotonic projection values in variation space, when we travel in the tangent plane to a shape. We analyze this problem to give some insight on our solution.

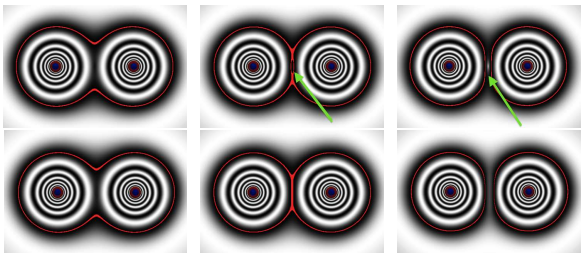

Figure 8: Scalar field of point-skeletons for different correction angle (in cross-section). The top row illustrates the cavity problem, where a small cavity inside the shapes can be observed in the center and right pictures. The bottom row shows our correction, which enforces shape to consistently switch between blending and non-blending behaviors depending on the angle.

Analysis. Without loss of generality, let us consider two point-skeletons, and look at the field correction we apply for points that lie on a half line of interest located at equal distance from two primitives (top right of Figure 9). For each values of the distance d between the two point-skeletons, the relationship between f and $\|\nabla_{\mathcal{S}} f\|$ is given by $m_d(f) = \|\nabla_{\mathcal{S}} f\|$ with

$$m_d(x) = 2(n-1)(x/2)^{\frac{n+1}{n-1}} \sqrt{(2/x)^{\frac{2}{n-1}} - (d/2)^2}.$$

This formula is derived in section 3.1 of the supplemental material. We call $\mathcal{M}_d$ the graph of this function in

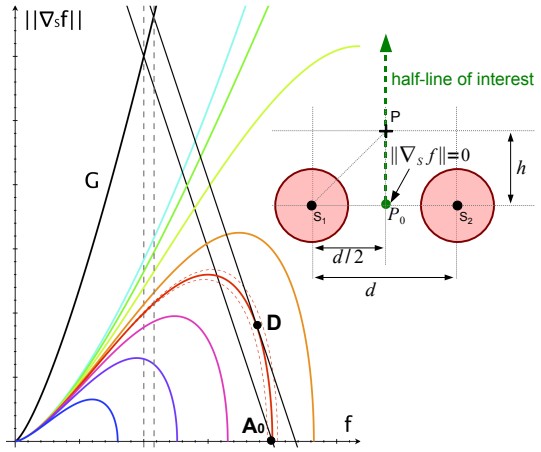

Figure 9: Reference case for solving the cavity problem: Each colored curve $\{\mathcal{M}_d = (x, m_d(x)), x \in \mathbb{R}\}$ corresponds to the graph of $(f, \|\nabla_{\mathcal{S}} f\|)$ for the blending of two segment-skeletons in the plane located at equal distance to each skeleton, from a distance d of 1.59 in green to a distance of 3.16 in blue. The inclined line leaving from A_0 is a given direction of projection.

variation-space, see Figure 9. Note that the function is not bijective: If we look at the projection of A_0 (associated to $\mathbf{p}_0$, the point of null gradient shown in the top right), we can see that it crosses $\mathcal{M}_d$ in two points. This means that there is another point on the half-line with the same corrected field, and that all the points of the curve $\mathcal{M}_d$ above the line have a higher corrected field. These non-monotonic variations of our correction explain the cavity problem, and are unwanted: indeed, the point $\mathbf{p}_0$ is the closest point to both segment and therefore should have the highest field value to maintain a consistent blending behavior.

A good field correction should ensure that the resulting field does not increase as h increases. Therefore, we should change the method so that the resulting field (abscissa of the projected point) does not increase along any curve $\mathcal{M}_d$, when f decreases.

Correction of the cavity. Our correction is based on the fact that, for a given curve $\mathcal{M}_d$, the derivative of the corrected field along this curve is null at the point D where the direction of projection is colinear to the tangent of the curve. The simplest way to improve the field is then to output a constant corrected field along the part of $\mathcal{M}_d$ which is at the right of D, i.e. with higher f values. Here is our algorithm for doing so. To compute $\hat{f}_\alpha(\mathbf{p})$:

1. Find the curve $\mathcal{M}_d$ on which $A = (f(\mathbf{p}), \|\nabla_S f(\mathbf{p})\|)$ lies (solve $m_d(f(P)) = \|\nabla_S f(\mathbf{p})\|$ in d),
2. Compute the point D of $\mathcal{M}_d$ whose tangent is colinear to the projection direction (solve $m_d(D_x)' = \tan(\alpha)$, deduce D_y),
3. if $f > D_x$, then $\hat{f}_\alpha = \hat{D}_x$, else $\hat{f}_\alpha$ is given by equation (12).

Steps 1 and 2 can be solved analytically. Details are provided in the supplemental material. Results are shown in Figure 8 (second row). Note that in practice, the cavity correction only adds 20 lines of code to the implementation.

5. Topology control

In this section, we describe how the projection angle α is chosen to provide the various types of behavior we wish to create. The value of α is either chosen by the user (for example, using a slider) and assigned to a subset of the primitives (or all of them), or is computed for each point in a context-dependent manner. We first discuss how specific values of α can be chosen to achieve the various behaviors, and then describe how they can be computed based on context.

5.1. Key Parameter Values

The first three behaviors in Figure 3 are generated using appropriate values of α. Low values of alpha lead to blend avoidance (objects deforming when they come in contact, and only blending when they overlap sufficiently). High values of alpha lead to blending at distance. The degree of these effects depends on the value of α, for example lower values require more overlap before blending occurs.

The case of contact blend, where primitives begin to blend exactly when they come in contact, occurs for a specific key value of α. For values of α above this threshold, blending at a distance will occur; for values below, blending will be avoided until a sufficient overlap. Unfortunately, this key value where contact blend occurs varies depending on the shapes being blended. Here we derive the key value for two extreme cases: when two convex primitives come together, and when an infinite number of convex primitives come together (the center of a torus). The first case is important in practice, while the second can be used as a lower bound for the interval where key-values for arbitrary shapes are to be looked for.

Contact-blend between two convex primitives. The case of two convex primitives coming into contact is common, for example when a spherical drop comes into contact with a planar surface for a stylized depiction of rain. This same situation occurs for contacts in the convex regions of more complex shapes, such as when the arms of the dragon come to contact, or near point $\mathbf{p}_2$ for the bent shape of Figure 4.

Without loss of generality, we consider the specific case of two parallel line primitives of radius 1 coming to contact parallel to each other. When no blending occurs, the generated shapes are two cylinders. Given our correction to the cavity problem, topology changes occurs, as expected, at the mid points of segment orthogonally joining the two lines.

We are looking for the distance d where the change of topology occurs as a function of α. Closed-form expression can be derived in two cases. From Equation (13), we obtain that for $\alpha = 0$ (horizontal projection), the topology change occurs at a distance $d = \sqrt{n}\left(\frac{2}{\sqrt{n+1}}\right)^{\frac{n+1}{n}}$, for the inverse kernel of degree 4, $d \approx 1.7396 < 2$. From Equation (8), we obtain that for $\alpha = \frac{\pi}{2}$ (summation blending) the change of topology occurs for $d = 2^{\frac{n}{n-1}}$. We compute intermediate values numerically using a dichotomy on d for each value of α. The graph of d in function of α is the blue curve in Figure 10.

This graph give us the key-value we are looking for; it is the α value for which primitives of radius 1 blend when their union of balls are tangent, i.e. when $d = 2$. We compute it numerically using a dichotomy on α. For the inverse kernel of degree $n = 3$ we get $\alpha \approx 0.93$, for degree 4 we get $\alpha \approx 1.16$ and for degree 5 we have $\alpha \approx 1.28$. Using these specific values of α to parametrize our operator enables us to get the *contact-blend* behavior for any pair of implicit primitives regardless of their skeleton respective dimensions (points, curves, surfaces), as well as when convex parts of any pair of arbitrary shapes approach each other.

Case of toroidal shapes. When more than two objects (or parts of objects) approach each other, the derivation for 2 primitives does not apply. This situation occurs frequently in practice in concave regions of shapes, such as at $\mathbf{p}_1$ in Figure 4 (left). In such situations, more blending occurs, so a smaller value of α will be required to preserve the topology of the union of balls. To provide some insight for choices of α in such cases, we derived the value of α required to preserve the topology of the union of balls in the extreme case of a point being fully surrounded by implicit primitives, i.e. in the case of a circular skeleton whose radius is equal to the radius τ of the shape around it (using integral surface from Equation (5)) . A similar shape is pictured in Figure 7.

At the center of such primitive for kernel of even degree n, the field is equal to :

$$f(\mathbf{p}) = 2 \prod_{j=2}^{n/2} \frac{2j-2}{2j-3}$$

This value is greater or equal to 4 for all $n \geq 4$, whereas in the tangential case of two parallel segments the field value was equal to 2. However, in both cases the gradient is null. This explains the need for a different value of α to preserve the topology of a torus versus the one of two segment primitives coming in contact. We can compute numerically the key value required to preserve the topology of the torus: for a kernel of degree $n = 4$, α should be equal to -0.56. This new value of α, which corresponds to a negative projection angle, is a lower bound on the value of α to be used in order to obtain a contact-blend.

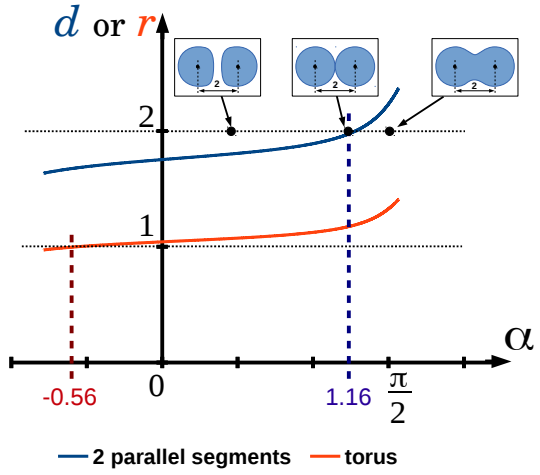

Figure 10: Relationship between the angle value α and the distance d at which topology changes for two parallel segments of radius 1 (depicted in cross section), and for a torus with major radius r and minor radius 1.

5.2. Dynamic Parameter Values

In order to obtain context-dependent effects, the control parameter α can also be set to vary in space or time. The simplest context-dependent effects would associate different values of α to different regions of space. The parameter could also be computed based on animation or simulation parameters. For instance, α could be computed from the local temperatures value in the case of a lava flow (see Figure 3): the higher the temperatures, the higher the blending. Here we describe two effects that we have found useful.

Per-skeleton blending. A value for the α parameter can be included within the description of each skeleton S_k. In this case, the fields of the set of primitives can be used to interpolate the parameter at each query point:

$$\alpha(\mathbf{p}) = \frac{\sum_i f_i(\mathbf{p})\alpha_i}{\sum_i f_i(\mathbf{p})}. \qquad (14)$$

Directional blending. More complex behavior can be obtained using additional information from the skeletons. For instance we can define blending depending on the local distribution of skeletal orientations in order to enable merging when neighboring skeletons are parallel, while preventing it when they are orthogonal. This effect can be obtained through the following formula :

$$\alpha(\mathbf{p}) = \frac{\sum_{i\neq j} f_i(\mathbf{p}) f_j(\mathbf{p}) \gamma(\mathbf{u}_i.\mathbf{u}_j)}{\sum_{i\neq j} f_i(\mathbf{p}) f_j(\mathbf{p})}, \qquad (15)$$

where $\mathbf{u}_i$ is the direction of skeleton S_i and γ is a polynomial function used to interpolate between two extremal amounts of blending α_{min} and α_{max} in function of the cosines of the angle between two skeletons.

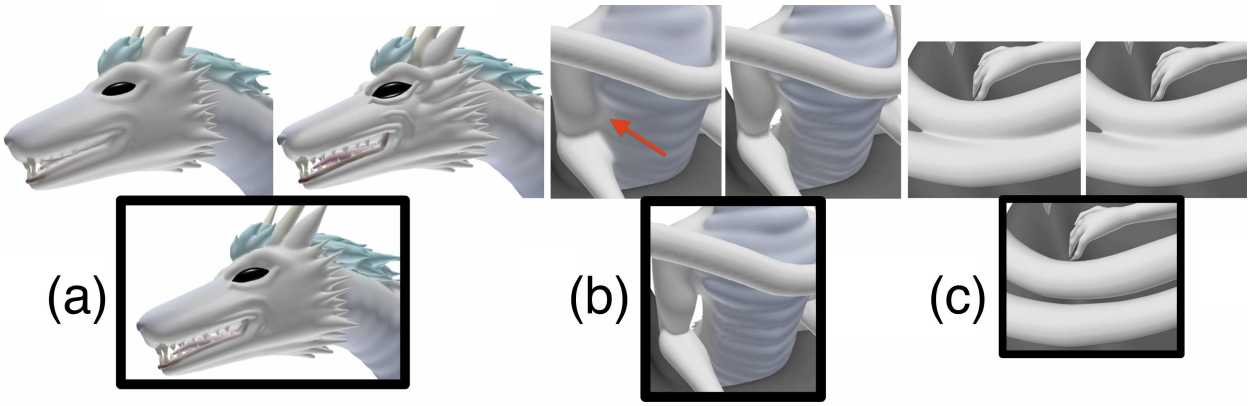

Figure 12: Close-up on the middle dragon from Figure 1. It was created using two Blobtree nodes : a max operator combines the eyes with the rest of the dragon, which is created from a single n-ary blend with topology control. The close-ups show comparison to sum of the same degree 4 inverse kernel primitives (upper left vignette) and to sum of compact Polynomial kernel primitives of degree 6 (upper right vignette). Our solution is the only one that preserves the skeleton topology in (c).

In our implementation, we use:

$$\gamma(x) = (\alpha_{max} - \alpha_{min})x^8 + \alpha_{min}. \quad (16)$$

This arbitrary function provides a fast change of behavior for angles in the interval $[0; \frac{\pi}{4}]$ and a nearly constant behavior in the interval $[\frac{\pi}{4}; \frac{\pi}{2}]$. The result of this context-based blending is depicted in a simple case in Figure 11, while an example of application is given in Section 6, with the hair animation example.

6. Results and Discussion

6.1. Applications

Modeling organic shapes. The dragon model depicted in Figures 1 and 12 illustrates the *skeletal-topology behavior* in the taxonomy of Figure 3. It was created by blending 867 SCALIS segment-primitives through a single nary blending node. We use the inverse kernel of degree 4 to generate the primitives. The blend parameter was set lower than the key value for contact-blend, in order to prevent merging even when different parts overlap slightly (see Figure 13). Figure 12 compares our result with two previous models, using respectively the sum, and a sum of sharper primitives modeled using a compact polynomial kernel. The latter (the upper right in the close-up pictures) gives almost the same results as our method in (b) but reduces blending too much for small scale primitives (a), while failing to preserve the topology constraint (c). Note that since our method preserves the segmentation-invariant property of integral surfaces, this dragon model can be further refined by refining its skeleton, for instance to get a smoother tail.

Figure 14, gives a second example of organic shape, where both segment and triangle skeletons were used. Note that using a combination of 1D and 2D skeleton primitives enables to approximate the skeleton of any 3D shape.

Animating Manga-style hair. Manga-style, volumetric hair, such as those depicted in Figures 1 and 15 is an example of context-dependent blending behavior: wisps of hair should smoothly blend when they have similar orientations, while they should squash on each other

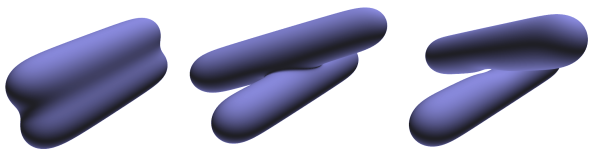

Figure 11: Effect of directional blending between two segments with an angle ranging from 0 to $\frac{\pi}{2}$. Although overlapping, the orthogonal segments at the right do not merge.

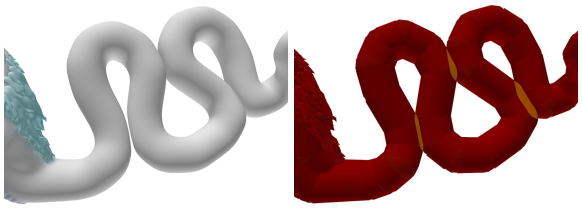

Figure 13: Close-up on the left dragon in Figure 1, where our choice of α prevents unwanted blending (left) even though the union of balls along the skeleton (right) slightly overlap.

and bounce when they come to contact perpendicularly to each other [23]. Furthermore, wisps tend to separate in a sharp way. No previous geometric model, to our knowledge, was able to automatically achieve this behavior. Our solution uses Super-Helices [24] for animating either straight or curly guide-hairs which correspond to centerline of volume wisps. These guide-hairs are tessellated into segments and serve as skeletons for SCALIS implicit primitives. Blending is controlled using a combination of skeleton-based properties and of context-based, directional blending: Different values $\alpha_{i,min}$ and $\alpha_{i,max}$ of the projection angle α are set along each guide-hair, in order to get smooth blending near the scalp and sharp blending at the tip of the hair wisps, while enabling the amount of blending to vary depending on the angle with neighbors. When the field is computed at a query point $\mathbf{p}$, the $\alpha_{i,min}$ and $\alpha_{i,max}$ values within each guide-hair H_i are first combined at $\mathbf{p}$ using equation (14). A similar equation is used to associate a direction vector $\mathbf{u}_i$ to H_i at $\mathbf{p}$:

$$\mathbf{u}_i(P) = \frac{\sum_{j \in H_i} f_{i,j}(\mathbf{p}) \mathbf{u}_{i,j}}{\sum_{j \in H_i} f_{i,j}(\mathbf{p})}$$

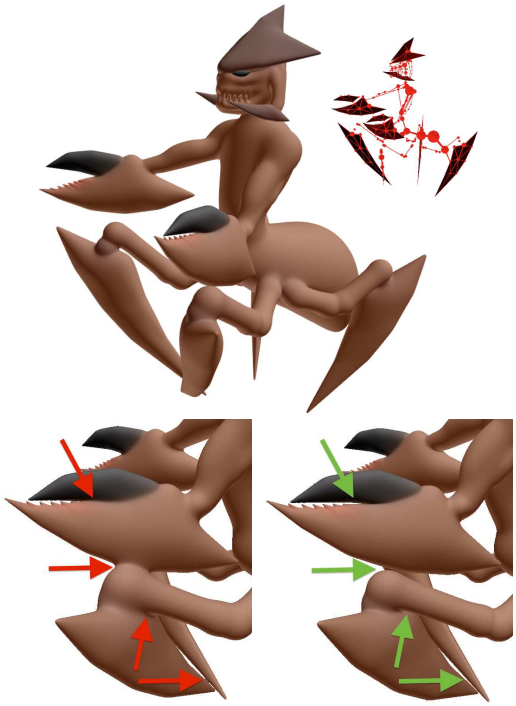

Figure 14: Our method applied to an object created from point, segment and triangle skeletons. The skeleton is displayed on the top right. Bottom left : summation blending ($\alpha = \frac{\pi}{2}$). Bottom right : our method with contact blend ($\alpha = 1.16$).

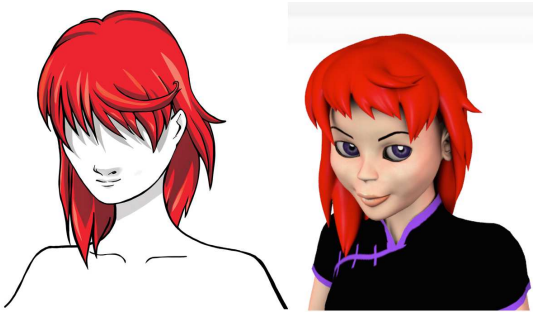

Figure 15: Left : a 2D drawing of a character's hair in the style of manga hair. We can note the sharp creases where wisps separate as well as the absence of blending between non parallel wisps. Right : Result of our method in a similar case.

Figure 16: Result for curly hair, with a close view on the left.

where $u_{i,j}$ is the direction and $f_{i,j}$ the field generated by the j-th segment skeleton of the i-th guide-hair. Then, $\alpha(\mathbf{p})$ is computed using equation (15), where f_i is the field contribution of guide-hair H_i, and where the bounds of the function γ are set from the minimal and maximal α values for H_i at $\mathbf{p}$.

Figure 17 provides some comparison between this new contextual blending and the summation blending, as well as results extracted from animations. This method can be applied to both straight and curly hair (see Figure 16).

6.2. Performances

Implicit surface extraction for the examples were performed using the Marching Cubes algorithm. The use of a flat construction tree enabled us to easily implement a spatial optimization structure to accelerate field queries, namely a fitted bounding volume hierarchy [25]. Computational times are provided in Table 1. Their high values are due to the use of small grid cells which are required to retrieve fine features such as teeth of the dragon model and sharp end-points of the hair. Using a coarser grid (increasing grid's cell size by a factor

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

Figure 17: Left: Comparison between summation blend (a) and context dependent blend (b) : the wisp with a different orientation is not blended with the other ones and the separation between wisps is sharper than with a summation blend. Right : Two examples extracted from animations. See also the attached video.

5) enables interactive rate; a dragon generated a this lower resolution and interactively edited is shown in Figure 18. Besides improvement in the meshing method used, computational times could also be improved by deriving a closed-form gradient for our blending operator.

Meshing an implicit shape defined with our new operator takes 7% more time than with a summation blend in the dragon example (867 primitives). This additional cost decreases when the number of primitives increases. The extra cost of our operator increases to 65% more than the summation blend in the case of hair (from 205 to 700 primitives) . This is due to the evaluation of the context-based control parameter.

6.3. Advantages and limitations

These examples and the video illustrate the ability of our method to capture different types of topological effects while performing n-ary blends. We preserve the ability of the additive blend to create smooth surfaces from multiple components. However, by changing the blend parameter, we can achieve a variety of merging behaviors. Moreover our method can safely be applied before the use of any other composition operator (such as union, difference, etc) since it defines a smooth scalar field everywhere in space.

At present, we have only derived the correction for infinite radius kernels. Kernels with local support should be compatible with our method. Extending our derivation to them would reduce the computation time while providing local shape control.

To further improve performance, one could also exploit the fact that our approach uses a single blend operator, rather than a sequence of binary operators. This enables the use of caching schemes for interactive modeling, where only the field of the edited primitive and the correction would be recomputed.

Low values of the blending parameter are used to prevent blending. They cause slightly-overlapping primitives to compress, rather than merge. This can only cor-

	Nb Primitives	Nb Triangles	Times
Dragon - Figure 1(middle)	867	3 790k	51.3s
Monster - Figure 14	259	1 340k	59.3s
Volume Hair - Figure 15	205	972k	17.8s
Dragon (mid-res) - Figure 18	867	172k	1.56s

Table 1: Computation time of marching-cube on an Intel Xeon (3.2 GHz, only one core used and no AVX).

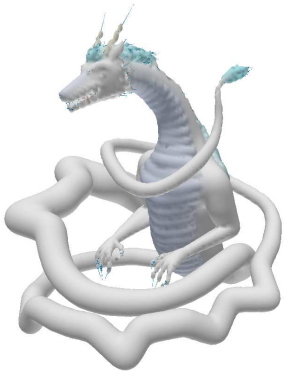

Figure 18: Lower quality mesh enabling to manipulate our dragon model in real time.

rect for small inter-penetrations: we still need to use collision detection and response for more general animation. Also our method does not necessarily create contact as there may be a gap between compressed parts. Leaving this small gap could be desirable for some situations, such as animating a dressed character. Exactly achieving contact would require determining another key parameter value for the input shapes.

In this work, we only provided exact values for contact blend in extreme cases. While some important cases, such as the point primitives for water drops, fit these, in other cases we can only approximate the key value, usually making a conservative estimate to avoid unwanted blending. This means that shapes that should touch may compress slightly to avoid contact. Prior approaches, such as [20], have the same problem. Dynamically computing the key-value value would allow exact contact blend. For instance, the angle α could be computed as a function of some local "density" of skeletons.

Lastly, in future work, we hope to better explore the range of possible behaviors that context-dependent blends can generate and find ways to make their control more intuitive for animators.

7. Conclusion

In this paper, we have introduced the first n-ary blending method that combines the benefits of summation operators with an easy to tune parameter that provides topology control over the resulting shape. The method is efficient and easy to implement. It seamlessly handles the most common primitives used in skeleton-based implicit modeling (blobs, distance surfaces and integral surfaces). Simple behaviors can be achieved by tuning a single control parameter, namely the angle α used for projection to a reference case in variation-space. We provide exact values of α for some specific behaviors, such as for pairs of primitives that start to merge upon contact. More complex blending behaviors can be easily parameterized, using skeleton-dependent or context-dependent α values. This provides an easy way to model organic shapes as it allows for smooth combination of parts without unwanted self-intersection. In particular, our new method handles the challenging case of animating volumetric Manga-style hair, where dynamic topology changes occur according to the angle between neighboring hair wisps. This demonstrates that n-ary blending with topology control makes skeletal implicit modeling applicable to a wider range of modeling and animation applications.

References

[1] D. Rohmer, T. Popa, M.-P. Cani, S. Hahmann, A. Sheffer, Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles, ACM Trans. Graph. 29 (6) (2010) 157:1–157:8. doi:10.1145/1882261.1866183.

[2] J. F. Blinn, A generalization of algebraic surface drawing, ACM Trans. Graph. 1 (3) (1982) 235–256.

[3] G. Wyvill, C. McPheeters, B. Wyvill, Data structure for soft objects, The Visual Computer 2 (4) (1986) 227–234.

[4] D. Terzopoulos, J. Platt, K. Fleischer, From goop to glop: Heating and melting deformable models, in: Proceedings of Graphics Interface, Vol. 2, 1989, pp. 219–226.

[5] M. Desbrun, M.-P. Cani, Animating soft substances with implicit surfaces, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, ACM, New York, NY, USA, 1995, pp. 287–290. doi:10.1145/218380.218456.

[6] M. Müller, D. Charypar, M. Gross, Particle-based fluid simulation for interactive applications, in: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 154–159.

[7] C. lan Tai, H. Zhang, J. C. kin Fong, Prototype modeling from sketched silhouettes based on convolution surfaces, Computer Graphics Forum 23 (2004) 71–83.

[8] R. Vaillant, L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, M. Paulin, Implicit skinning: real-time skin deformation with contact modeling, ACM Trans. Graph. 32 (4) (2013) 125:1–125:12. doi:10.1145/2461912.2461960.

[9] J. Bloomenthal, K. Shoemake, Convolution surfaces, in: Proceedings SIGGRAPH '91, ACM, 1991, pp. 251–256.

[10] A. Sherstyuk, Interactive shape design with convolution surfaces, in: Proceedings of the International Conference on Shape Modeling and Applications, IEEE Computer Society, 1999, pp. 56–.

[11] C. Zanni, A. Bernhardt, M. Quiblier, M.-P. Cani, SCALe-invariant Integral Surfaces, Computer Graphics Forum 32.

[12] R. W. K. Museth, D.E. Breen, A. Barr, Level set surface editing operators, ACM Transactions on Graphics (Proc. SIGGRAPH) 21 (3) (2002) 330–338.

[13] M. Eyiyurekli, D. Breen, Interactive free-form level-set surface-editing operators, Computers & Graphics 34 (5) (2010) 621–638.

[14] B. T. Stander, J. C. Hart, Guaranteeing the topology of an implicit surface polygonization for interactive modeling, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997, pp. 279–286. doi:10.1145/258734.258868.

[15] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, On-the-fly curve-skeleton computation for 3d shapes., Comput. Graph. Forum 26 (3) (2007) 323–328.

[16] A. Guy, B. Wyvil, Controlled blending for implicit surfaces using a graph, in: Implicit Surfaces '95, 1995, pp. 107–112.

[17] A. Angelidis, M.-P. Cani, Adaptive implicit modeling using subdivision curves and surfaces as skeletons, in: 7th ACM Symposium on Solid Modeling and Applications, June, 2002, Saarbrucken, Allemagne, 2002, pp. 45–52.

[18] G. I. Pasko, A. A. Pasko, T. L. Kunii, Bounded blending for function-based shape modeling, IEEE Comput. Graph. Appl. 25 (2) (2005) 36–45. doi:10.1109/MCG.2005.37.

[19] A. Bernhardt, L. Barthe, M.-P. Cani, B. Wyvill, Implicit blending revisited, Comput. Graph. Forum 29 (2) (2010) 367–375.

[20] O. Gourmel, L. Barthe, M.-P. Cani, B. Wyvill, A. Bern-

hardt, M. Paulin, H. Grasberger, A gradient-based implicit blend, ACM Trans. Graph. 32 (2) (2013) 12:1–12:12. doi:http://dx.doi.org/10.1145/2451236.2451238.

[21] J. Bloomenthal (Ed.), Introduction to Implicit Surfaces, Morgan Kaufmann Publishers Inc., 1997.

[22] A. Ricci, Constructive geometry for computer graphics, Computer Journal 16 (2) (1973) 157–60.

[23] E. Plante, M.-P. Cani, P. Poulin, Capturing the Complexity of Hair Motion, Graphical Models 64 (1) (2002) 40–58. doi:10.1006/gmod.2002.0568.

[24] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, J.-L. Lévêque, Super-helices for predicting the dynamics of natural hair, ACM Trans. Graph. 25 (3) (2006) 1180–1187, special issue: SIGGRAPH'06.

[25] O. Gourmel, A. Pajot, M. Paulin, L. Barthe, P. Poulin, Fitted bvh for fast raytracing of metaballs, Computer Graphics Forum 29 (2) (2010) xxx–xxx.