

Implicit Blending Revisited

Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, Bryan Wyvill

▶ **To cite this version:**

Adrien Bernhardt, Loic Barthe, Marie-Paule Cani, Bryan Wyvill. Implicit Blending Revisited. Computer Graphics Forum, 2010, 29 (2), pp.367-375. 10.1111/j.1467-8659.2009.01606.x . inria-00450722v2

HAL Id: inria-00450722

https://inria.hal.science/inria-00450722v2

Submitted on 27 Jan 2010

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

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

EUROGRAPHICS 2010 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Implicit Blending Revisited

Adrien Bernhardt[1], Loic Barthe[2], Marie-Paule Cani[1], Brian Wyvill[3]

[1]Université de Grenoble, CNRS (Laboratoire Jean Kuntzmann), INRIA Grenoble Rhône-Alpes, France
[2]IRIT-VORTEX, Université Paul Sabatier, Toulouse, France
[3]Department of Computer Science, University of Victoria, Canada

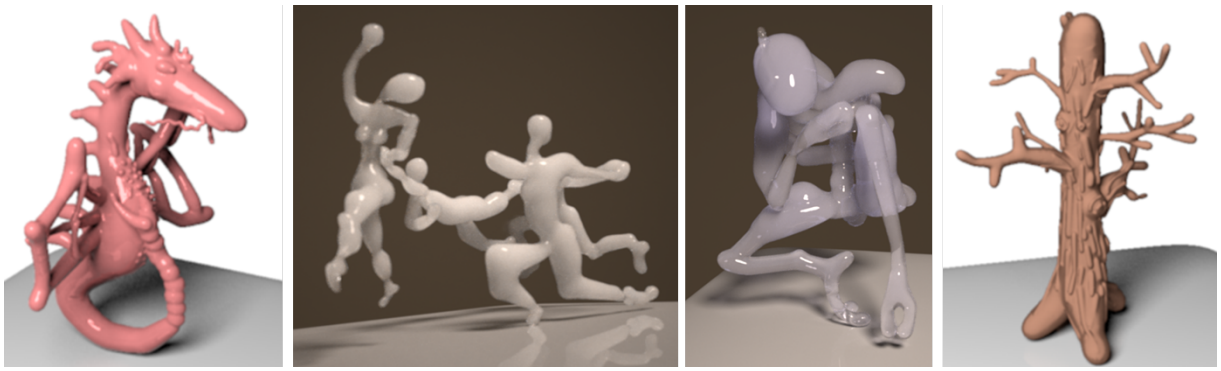


Figure 1: *Models generated interactively using a new approach to implicit blending. Note the capability of the shapes to come close to each other without blending. From left to right : The dragon, a dancer loop, the alien, a tree.*

Abstract

Blending is both the strength and the weakness of functionally based implicit surfaces (such as F-reps or soft-objects). While it gives them the unique ability to smoothly merge into a single, arbitrary shape, it makes implicit modelling hard to control since implicit surfaces blend at a distance, in a way that heavily depends on the slope of the field functions that define them. This paper presents a novel, generic solution to blending of functionally-based implicit surfaces: the insight is that to be intuitive and easy to control, blends should be located where two objects overlap, while enabling other parts of the objects to come as close to each other as desired without being deformed. Our solution relies on automatically defined blending regions around the intersection curves between two objects. Outside of these volumes, a clean union of the objects is computed thanks to a new operator that guarantees the smoothness of the resulting field function; meanwhile, a smooth blend is generated inside the blending regions. Parameters can automatically be tuned in order to prevent small objects from blurring out when blended into larger ones, and to generate a progressive blend when two animated objects come in contact.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object modelling—Solid, surface and object representation; Constructive solid geometry

1. Introduction

One of the major strength of implicit surfaces has been the capacity for objects to blend at a distance. By blending we mean that a new, seamless surface can be generated by smoothly merging two input volumes. The new surface is calculated by combining the field functions of the input objects and computing a new iso-surface. This feature has been popular in both Computer Animation to animate topological changes (e.g. liquids, melting objects [TPF89, DG95, BGOS06]) and in modelling where a constructive approach is used to assemble object components [PASS95, WGG99, SPS04, BPCB08, dGWvdW09].

The main problem with functionally-based implicit blending is its rather unpredictable nature: implicit objects may

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

blend at distance, or in regions where it is not required. For instance, in simple skeleton-based implicit modelling systems (such as blobby models, meta-balls, soft objects or convolution surfaces), the blend shape is computed by taking the sum of the field functions that define the input implicit primitives. Therefore, the amount and range of blending heavily depends on the slopes of these functions, without any intuitive user control. This results in the blurring of small, sharp features when blended into large, smooth surfaces [WW00]. Moreover, this is a major problem in Computer Animation, since implicit objects that come close together will start to deform and blend at a distance; for example a falling drop of water deforms the water surface before reaching it. Lastly, this method leads to blending in unexpected regions; for instance the hand of a character will blend with its body when it comes close enough, due to the blend allowed in the shoulder region.

Although improving control and localizing blends has been attempted for years through the definition of various ways to combine field functions (see section 1.2), no simple, generally applicable solution has emerged. This may be the main reason for the relatively small spread of implicit techniques among the set of practical tools used by modelling and animation professionals.

We describe an automatic way of defining a blend between arbitrary functionally-based implicit surfaces. Contrary to previous methods, it does not allow objects to blend at a distance, but rather automatically localizes the blend near the regions where their surfaces intersect, while the extent of the blend is controlled independently of the support size of the defining field functions. This method is generic in the sense that our solution supports both local and global support implicit primitives.

1.1. Technical background

An implicit surface is defined as a set of points verifying an equation of the type $f(P) = C$ where f is a field function which can either be of global or local support. Other terminologies have been used for these functions such as *potential fields* in the case of skeleton-based implicit surfaces or *implicit functions*, which is not mathematically correct since the function is explicitly defined. We prefer *field function*, which expresses the fact that that the implicit surface is an iso-surface of the scalar field they define. The field function can be defined either with a level-set, i.e. as a front propagation in a grid [Set99, OF02], or by a functionally-based representation such as f-reps [PASS95], soft objects [WMW86], convolution surfaces [She99], RBF [SPOK95], and MLS [Lev03].

For these representations, several conventions can be used: Most **global support** field functions used in solid modelling behave as a distance to the surface of interest: they are negative inside the object, and positive outside, C being equal to zero. In contrast many implicit modelling techniques rely on a **skeleton-based scalar field** (from Blinn's objects to the most recent convolution surfaces): in which case the field function is a decreasing function of the distance to a geometric primitive called the skeleton. The field values are larger inside than outside and C is non-zero. **Local support implicit surfaces** such as meta-balls and soft objects belong to that group, with fields that vanish at a given distance to the skeleton. All these representations have specific properties that lead to different composition operators for union, difference, blending, etc. For a full explanation please see [BWd04]. This paper does not focus on discretely sampled distance fields, like level sets, but rather on the blending of functionally based implicit surfaces.

1.2. Previous work

Over the years, improving the control of implicit blends and localizing them (i.e. solving the *unwanted blending problem*) has been a major area of research in implicit modelling and animation. While efficient solutions have been proposed for level set implicit surfaces [MBWB02,BMPB08,EGB09], no fully satisfactory technique has been developped for functionally-based implicit surfaces. Up to now, two families of solutions arose: The first ones used a blending graph to define pairs of primitives allowed to blend, i.e. to sum their field functions. Since the early solutions [GW95, DG95] did not insure that the resulting shape was continuous everywhere or could lead to sudden shape changes during animation, more elaborate methods were developed [CH01, AC02]. In particular, Angelidis [AC02] added decay functions to skeletal elements to insure the smoothness of the resulting shape, making the method quite intricate to implement and restricting the method to the specific case of convolution surfaces generated by skeletons made of line-segments and triangles.

A second group of methods developed more powerful ways than simple sums to combine scalar fields. In his seminal paper on set-theoretic R-functions [PASS95], A. Pasko introduced, in the framework of the global support surfaces used in solid modelling, Rvashev's R-function union operator that generates at the same time, the exact (sharp) union surface and a C^1 continuous field function everywhere else. In the remainder of this paper, we will call such an operator a *clean union* since it ensures that no unexpected crease appears during subsequent blends [BWd04]. More recently G. Pasko [PPK05] proposed a method for defining local smooth blends between some parts of two objects, while using a clean union elsewhere. This was done through the specification of a simple user-defined implicit primitive (sphere, ellipse, torus) of local support, indicating the blending region. The field function of the primitive was used to control the amount of blending. This method was designed for global support implicit surfaces and required manual user intervention to tune each blend.

In parallel, Barthe [BDS*03, BWd04] developed controllable ways to blend field functions. The method, first introduced for global support implicit surfaces, offered intuitive shape parameters and enabled the accurate location of

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

the blend between two given iso-values of the input implicit functions. However this does not allow some parts of the surfaces to come close without blending.

Barthe later extended his operator for soft objects (of local support) and defined the first clean union. However, his solution based on arcs of ellipses is computationally very costly, and thus not practical in an interactive modelling system. Lastly, the smooth blend operator was used in a sketch-based modelling system [BPCB08], where blends were artificially localized by re-computing only a part of the iso-surface. This method could lead to artefacts during subsequent blends due to the lack of implicit representation for the whole surface.

1.3. Overview and contributions

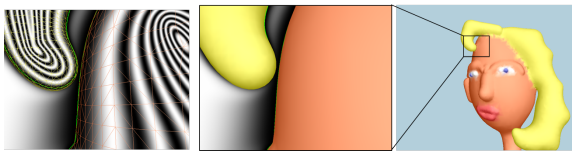


Figure 2: *Implicit primitives do not interfere with each other when they come close without intersection. The field function (left) remains C^1 continuous providing a clean union.*

The key idea of our method is that to be predictable and intuitive, blends should only occur where two objects overlap. Implicit primitives should not interfere with each other elsewhere, even if they come close (as illustrated in Figure 2). As suggested by Museth et al. [MBWB02] in the case of level sets, we localize blends automatically near intersections, by computing the intersection curve(s) between the two input surfaces, and use them as a skeleton to generate the region(s) where the objects are allowed to blend (see Figure 3). The thickness of the blending region is set to produce a progressive growth of the blend during animation.

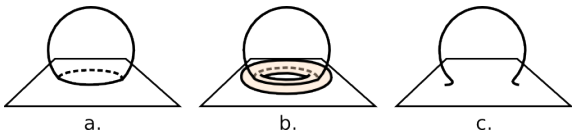


Figure 3: *Overview of revisited blending: a. intersection curve extraction; b. blending region generated by the curve. c. local blending inside the blending region while a clean union is used elsewhere.*

To achieve this, our contributions are to solve the 3 following issues:

- Firstly, we need an efficient yet precise method to compute the intersection curves between implicit objects and to define appropriate blending volumes around them.
- Secondly, we need an efficient clean union operator to be used outside of the blending volumes, whatever the nature of the implicit surface. While set theoretic R-functions [PASS95] provide a solution in the case of global support field functions, a more efficient operator than Barthe's arc-of-an-elipse union [BWd04] is required for local support field functions.
- Lastly, the smooth blend that takes place within the blending volume should automatically prevent the blur of small shape features when blended to larger shapes and ensure that the resulting field function is continuous across the border of the blending volumes.

Sections 2 to 4 respectively present our solutions to these problems. Section 5 illustrates our results on simple animation experiments and within an interactive modelling system where the user progressively creates simple shapes and blends them together. We then discuss the strengths and weaknesses of our approach.

2. Local blending volumes around intersections

Instead of asking the user to specify the region where each pair of surfaces should blend, we would like to blend them automatically where they intersect and just combine them with a union elsewhere. To achieve this, *blending volumes* must be automatically defined around each intersection. Since the field value of the blending volume will be used as an interpolation parameter between smooth blend and union, we need an implicit primitive of local support (see section 1.1) i.e. whose field function falls to zero at the border of the volume.

The intersection between two closed surfaces is a set of (possibly several) closed curves. See Figure 4. The idea is to use them as skeletons, in order to build a ring-shaped blending volume around each intersection. In the next subsection we discuss our method for an efficient, yet precise extraction of intersection curves. We then explain how the blending volumes around them are parameterized, with the goal of providing intuitive, progressive blends when animated objects come in contact.

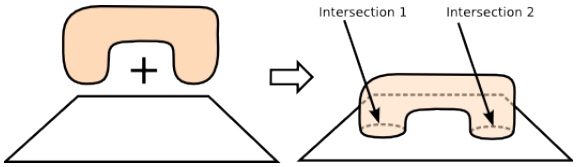


Figure 4: *Dealing with multiple intersections.*

2.1. Extracting intersection curves

The intersection curves between a pair of implicit surfaces may be described in two ways: as (possibly several) polylines, i.e. explicit lists of points at some approximation level, or defined analytically as a single implicit curve, i.e. as the set of points P simultaneously verifying:

$$f_1(P) = C \quad \text{and} \quad f_2(P) = C \tag{1}$$

where f_1 and f_2 are the field functions of the two input objects and C is the iso-value at which the surface is computed.

The goal is to use intersection curves as skeletons, which

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

generate implicit volumes, i.e. to enable quick yet precise distance queries from arbitrary points in space to the curves. In [MBWB02], the intersection curves computation relies on the discrete grid representation of the potential field. For functionally-based field functions, such a discrete representation is not directly available. In the first case, a distance query can be made efficient by embedding the poly-line into a tree of bounding boxes, but the resolution being fixed, precision may be lacking. In the second case, precision is good since the curve is analytically defined, but distance queries may not be efficient, since a numeric technique must be used to find the closest curve point.

To get both precision and efficiency, we use a hybrid approach: We first extract low resolution versions of the curves, and we locally refine each curve by splitting segments in which the mid-point does not meet the constraint presented in Equation 1. The extra points are made to converge on the curve using the algorithm described in [WvO96], i.e. by alternately marching in the two field gradient directions.

The extraction of the low resolution curve can be done in one of two ways. Firstly, by computing the intersection of the meshes used to display the input implicit surfaces; this is done in an efficient way using hierarchies of bounding boxes. The second approach uses the efficient marching-faces algorithm, introduced in [LY03]: A marching-cubes is performed on the set of voxels embedding surface intersections in order to locally polygonized each surface. The marching-faces algorithm then computes the intersection lines of polygones in each voxel. This second method is more general, since it does not require any precomputed meshing of the input surface. The selection of the voxels embedding the intersection curve required by this last technique can be greatly accelerated using an adaptive octree in the intersection of the combined objects bounding boxes. The octree splitting is guided by the well designed inclusion functions of f_1 and f_2 [FSSV06, Duf92, Sny92], which provide intervals that are close to optimal.

The method produces a fast computation of explicit yet precise curves. In the case of multiple intersection volumes, our method has the advantage of separating the intersection into disjoint curve primitives, whereas the analytical description would not. This allows us to generate blending volumes of appropriate size around each individual curve, as explained next.

2.2. Setting bounding volume parameters

At this point, the intersection is described as one or several closed curve(s). We build an implicit blending volume around each of them by using a finite support skeleton-generated field function, such as Wyvill's soft object function [WMW86] used in our current implementation. However, a parameter needs to be set: the thickness of the ring-shaped blending volumes, defined as the size of the support of the associated local field function.

As already mentioned, we are looking for a method which makes blending intuitive in both modelling and animation applications. In particular, two animated implicit shapes that come into contact need to blend progressively rather than jump immediately to a fully deformed shape (see Figure 5).

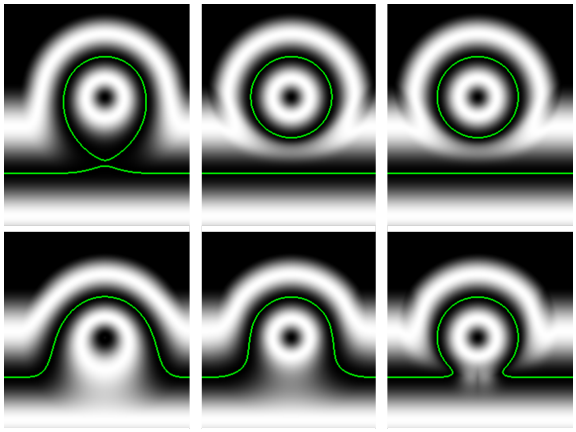


Figure 5: *2D view, with scalar field represented in grey levels, of a drop of water reaching the water surface. Left: standard blending, e.g. a sum or Barthe's operator [BWd04]. Center: our method with a constant radius blending volume: an instant large blend is generated when the drop touches the water. Right: our method with radius function of the size of intersection solves this problem by generating a continuously growing blend.*

This feature can be expressed as a constraint on the thickness of the blending volume. Our experience has shown that a suitable heuristic is that the thickness should be proportional to the diameter of the associated intersection curve; defined as the largest distance between curve points. In addition to enabling animation of progressive blends, this automatic parameter setting is quite intuitive in interactive modelling sessions, where the part of the object that penetrates the most will also be those that will have the largest region of smooth blend. If the result is not satisfactory the user can manually edit the thickness parameter of a blending volume, although in our initial experiments this has not proved necessary.

3. Efficient, clean union of implicit surfaces

This section describes the way we combine field functions outside of the blending volumes where a union operator is required to prevent unwanted blending effects. However, in order to prevent artefacts during subsequent smooth blends in the area, this operator needs to be a clean-union i.e. to generate a C^1 continuous field (except on the union surface), which is not the case for simple operators such as max. As already stated, Rvashev's R-function [PASS95] provided a simple solution in the global support case, but a more efficient solution than Barthe's [BWd04] is required in the case of a local support field functions.

In this section we extend Rvachev's operator to achieve

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

the clean-union of local support implicit surfaces, in an efficient manner, and more generally of any skeleton-based implicit surface, including global support blobs and convolution surfaces. Figure 6 illustrates the different steps of our method.

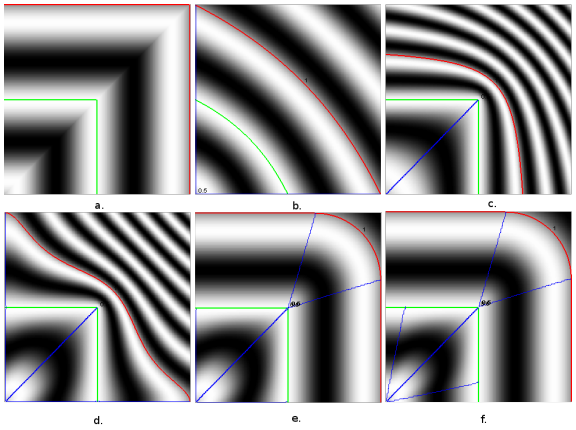


Figure 6: *2D comparison of different union operators: field values in sinusoidal grey scales are depicted in the (f_1, f_2) plane, the green and red lines being respectively the C (surface) and 2C (inside) iso-lines. a. The max operator is C^0 only; b. Rvachev's U_R is smooth but deforms the surface (green line) when directly applied to skeleton-based fields; c. U_{C_2} preserves the green line but contracts the field inside the shape (see the red line). d. U_{C_3} removes the contraction outside objects (see the region with small field values). e. Our final solution U_C for local support fields removes the contraction inside objects as well. f. Interpolation limits are added to properly deal with global support convolution primitives.*

3.1. Local-support clean-union operator

Rvashev's clean-union operator U_R (whose positive part is depicted in Figure 6b) was created to combine functions similar to a distance: zero iso-value, negative-values (respectively positive) inside (respectively outside) and un-bounded field values. This simple and efficient operator is defined by:

$$U_R(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \tag{2}$$

The goal is to make the operator applicable to skeleton-based scalar-fields, when surfaces are a C-iso surface with $C > 0$ and field values fall-off from inside to outside the shape. If they fall off to zero (local support), then the object resulting from the clean-union combination should also have a fall off to zero (i.e. the operator should be internal to the set of local support implicit surfaces). To achieve this, we translate and scale the U_R operator by defining:

$$U_{C_1}(f_1, f_2) = C + U_R(f_1 - C, f_2 - C) \tag{3}$$

As $U_{C_1}(0,0) = C(\sqrt{2} - 1) \neq 0$, we need to scale U_{C_1} so as to meet the zero value boundaries property. That gives us:

$$U_{C_2}(f_1, f_2) = C(1 - (U_R(f_1 - C, f_2 - C)/U_R(-C, -C)))$$

which can be rewriten as :

$$U_{C_2}(f_1, f_2) = \left(1 + \frac{\sqrt{2}}{2}\right) U_{C_1} - C\frac{\sqrt{2}}{2}$$

The new operator U_{C_2} (Figure 6c) respects fundamental local-support functions properties but contracts the field function. In particular, outside the boundary of f_2 (where $f_2 = 0$) we have:

$$U_{C_2}(f_1, 0) = C + \frac{f_1 - C + \sqrt{(f_1 - C)^2 + C^2}}{1 + \sqrt{2}}$$

which is different from the value of f_1, thus altering the original field function. As a consequence, the blending quality would be altered after multiple blending.

In order to better preserve the original field properties outside the objects, we interpolate between the standard $max(f_1, f_2)$ (which reproduces the fields of f_1 or f_2) and our function U_{C_2} as follows :

$$U_{C_3}(f_1, f_2) = (1 - \delta(f_1, f_2))\, max(f_1, f_2) + \delta(f_1, f_2) U_{C_2}(f_1, f_2)$$

with :

$$\delta(f_1, f2) = \begin{cases} 0 \text{ when } min(f_1, f_2) \leq 0 \\ \left(1 - \left(\frac{|f_1 - f_2|}{f_1 + f_2}\right)^2\right)^4 \text{ otherwise.} \end{cases} \tag{4}$$

Thus, as illustrated in Figure 6d the field is smooth everywhere except on the union surface $f_1 = f_2 = C$, and it tends close to f_1 (respectively f_2) when the other field function vanishes.

Unfortunately, the field inside the object (over the green line in Figure 6d) remains quite contracted, which could be a problem when objects are carved by combining them with negative-field primitives. We correct this by using Barthe's U_B operator [BDS*03] inside the objects, rather than the U_R operator in Equation 3. As shown in Figure 6e, our final clean union operator U_C, produces a high quality field function.

3.2. Adaptation to convolution surfaces

U_C can be applied to local support convolution surfaces, but not to those with infinite support. Such surfaces were made popular as in this case it is possible to compute an analytical expression of the convolution integral [She99, AC02]). As with local-support discussed above, these convolution surfaces are C-isosurfaces with $C > 0$, but rather than falling off to zero at a finite distance, the field value tends to zero at infinity. The adaptation of our compression-less operator to this case is quite straightforward:

If U_C was used directly, the interpolation applied outside the shape would never be used, since the field values never reach zero. The adaptation is just a modification of the limit

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

after which the operator reproduces f_1 and f_2 fields. To do so, we modify δ (Equation 4) so that $\delta_c(f_1,f_2)=0$ when $f_1=\alpha_1 f_2$ or $f_2=\alpha_2 f_1$:

$$\delta_c(f_1,f_2)=\begin{cases} 0 \text{ when } f_1\leq\alpha_1 f_2 \text{ or } f_2\leq\alpha_2 f_1 \\ \left(\frac{4(f_1-\alpha_1 f_2)(f_2-\alpha_2 f_1)}{((1-\alpha_2)f_1+(1-\alpha_1)f_2)}\right)^2 \text{ otherwise.}\end{cases}$$

Once this is done, we still use Barthe's U_B operator inside the objects in order to take into account the limit parameters α_1 and α_2 while providing a high quality field function (see Figure 6f).

4. Local, smooth blends

In this section we discusse our choice of operator inside the blending volume. As stressed in the introduction, the goal is to automatically calibrate a smooth blending operator in order to prevent the blurring of small shapes when blended into large, smooth ones. Our operator should also insure a seamless transition with the outer region, where the clean-union is used. The two following sections address these two problems.

4.1. Detail-preserving smooth blend

As discussed in the previous work section, a family of blending operators with some good range and shape control was already introduced by Barthe et al. [BDS*03, BWd04] in the case of both global support and local support implicit surfaces. More precisely, these operators restrict the range of blend between two, possibly different, iso-values of input implicit surfaces. The shape of the blend is then defined by a curve in the (f_1, f_2) domain, as depicted in Figure 7 (left).

To prevent small shapes from blurring, the distance at which blending is computed on the smallest shape (defined by an iso-value of the other surface) should be relatively small, while the other distance, on the larger shape, can be much larger. See Figure 7 (right). The limit iso-values chosen for the two surfaces can be set as the field values v_1 (respectively v_2) for points at the desired distance on S_1 (respectively S_2).

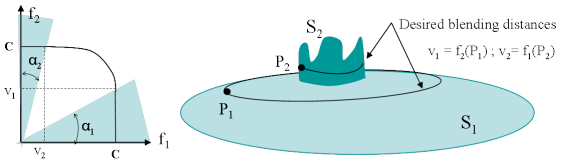


Figure 7: *Parameters of the blending operator U_B used to set the blending distance on each surface S_1 and S_2.*

We thus use Barthe's U_B blending operator (designed for local support implicit surfaces) and the problem becomes: how can we automatically define these desired distances along the surfaces, to automatically blend shapes as the user expects while saving them from the burden of tuning parameters v_1 and v_2 ?

The blending distance chosen for each surface should be proportional to the size of its smallest local feature, to prevent the latter from blurring. In some cases, depending on the types of implicit surfaces, the model comes with good indicators of smallest size, such as the local radius of the shape in the case of skeleton-based implicit surfaces, or a set of local bounding boxes. However, these indicators might as well be missing. In these cases, it can be pertinent to estimate the local size of shape feature from the local curvature of each surface along the intersection curve. Curvature information can be efficiently computed as presented in [Gol05]. A last indicator for skeleton based local support primitives and convolution surfaces is given by the gradient value at surface points. Indeed, the larger the gradient, the smallest the local detail of the shape.

Whatever the technique used, we compute an approximation of the curvature radius r_1 (resp. r_2) of the small details we want to preserve on surface S_1 defined by f_1 (resp. S_2 defined by f_2). The blending distances d_1 on S_1 and d_2 on S_2 are then computed as $d_i=r_i/2$ ($i=1,2$). Limit field values v_1 (resp. v_2) being at distance d_1 (resp. d_2) are approximated by evaluating f_2 (resp. f_1) at a point P_1 (resp. P_2) located at distance d_1 (resp. d_2) from a point P of the intersection curve:

$$v_1=f_2\left(p+d_1.\frac{\nabla f_2(p)}{\|\nabla f_2(p)\|}\right),\ v_2=f_1\left(p+d_2.\frac{\nabla f_1(p)}{\|\nabla f_1(p)\|}\right)$$

Thus the parameters of operator U_B are automatically set to these values. In case the progressive blend presented in Section 2.2 is not desired, the size of the support R of the ring-shaped blending volume can be set as well from the desired blending distances. Using $R=max(d_1,d_2)$ can be too large for preventing unwanted blending; since blending at the computed distance on the largest surface is not necessary, we rather used $R=2min(d_1,d_2)$ in our experiments.

4.2. Seamless transition between blend and union

Our goal is now to generate an operator U, which will provide a smooth transition between the smooth blend operator parameterized as discussed above and the clean-union operator used outside of the blending volume, in order to generate a C^1 continuous field function at the border of the blending volume. Our operator U is built from the interpolation of Barthe's U_B blending operator and our U_C operator as follows:

$$U(f_1,f_2)=(1-\beta)U_C(f_1,f_2)+\beta U_B(f_1,f_2)$$

where β is a power of the field value v of the ring-shaped blending volume.

5. Results

To demonstrate the usability of our new, local blend operator, we illustrate it in a simple animation experiment and within an interactive modelling system.

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

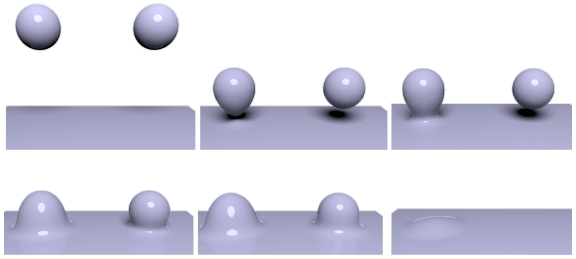


Figure 8: *A drop of water reaching the water surface. In each image : Left, with a sum operator, and right, with our new blending method.*

Figure 8 depicts a simple animation experiment, where a drop of water is dropped over a water surface. A local primitive (a soft object) is used for the drop and a global representation (a distance field) is used for the planar surface. Our result (the ball on the right in each of the image) validates our choice to start blending objects only when and where objects intersect. It also validates our parameterization for the size of the blending volume around the intersection curve.

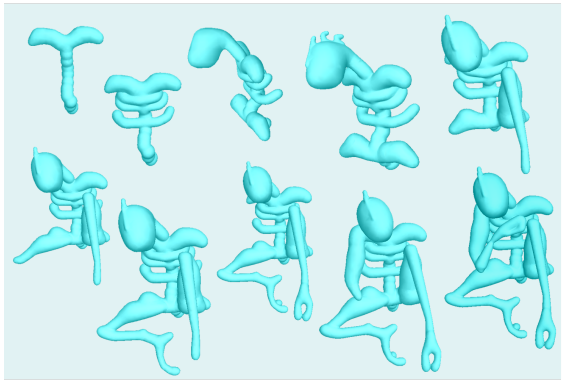


Figure 9: *Some of the creation steps of the alien character from the teaser image. The hand and the knee are very close, but do not blend.*

We also used our operator in an interactive modelling system where implicit primitives are progressively assembled, as illustrated on the alien in Figure 9. Primitives are convolution surfaces created interactively using skeleton curves. Four examples of different complexity were created: An alien, a group of dancers, a tree and a dragon. The two first examples illustrate the capability of the method to blend shapes only locally (such as the dancer's hand not blending with her body in the teaser image) and the two other illustrate the fact that small shape feature are well preserved, even when blended into larger shapes (Figures 10 and 11).

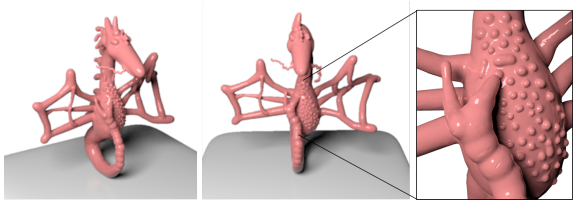


Figure 10: *More views of the dragon from the teaser image, illustrating the interest of blending to create models of non-trivial topology and the fact that small primitives do not blur when blended into large ones.*



Figure 11: *Some views of our tree model: blending is particularly challenging in this case, due to the complex shapes of the intersection curves between bark bumps and the trunk. Our method handles this case easily due to the automatic setting of the blending volumes.*

5.1. Computational time

The designer spent 30min to an hour to create each of the four examples. This time includes the progressive creation and blending of all the primitives. To give a better idea of the computational time necessary for each blend, we stored the construction-history for each example, and we measured the portion of time spent in blending when these examples were re-loaded. This time can then be divided by the number of primitives, to get an average time for a blend. Table 1 presents these results.

We also compare in Table 2 the evaluation cost of different blending operators. As expected, the more advanced the operator, the larger its computation time. However, the great advantage of operators such as U_B and U_C is their locality. Thus in a complex model, they require less evaluations even though these evaluations are more expensive.

U_R	U_R	U_B	U_C
0.08	0.115	0.78	0.95

Table 2: *Times in seconds for the one million evaluations of different composition operators required to compute images as those presented in Figure 6.*

5.2. Discussion

Our new method was found to be easy to use by our designer, since she just had to position primitives so that they slightly overlapped and to press a button for computing the blend. No

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

	Dragon	Dancer Loop	Alien	Tree
Nb of primitives	68	23	24	54
Reconstruction time from a script	277	34.3	41.9	475
Average time per primitive	2.65	1.24	1.31	6.3
Average time per blend	1.43	0.25	0.43	2.48

Table 1: *Computation times in seconds required by the creation of models presented in the teaser. The second row gives the number of primitives built by the user and then blended by the system. The third row gives, for each model, the total computational time spent when they are directly created from a script. The fourth and the fifth rows give the average of the times spent in respectively creating individual primitives and evaluating our blending operator per transition.*

extra primitives had to be created nor was it necessary to tune any parameters. The three long-standing problems identified for standard implicit blending, namely unpredictable blending at distance, lack of locality, and blurring of small details were solved by the system. The smoothness of the field function after each local blend, and the fact that our union operator does not create any unwanted contraction of the field, enabled the designer to create complex models without any problem or unexpected behaviour during successive blends.

A side benefit of our approach is the new *clean union operator* that can be used on its own for combining objects when a sharp union is desired.

We also identified some weaknesses of our methods, on which we are currently working:

Firstly, our current implementation defines blending volumes using a simple local support model (a soft object). Since the latter is based on the closest distance to the in- tersection curve, this model generates a field that may not be C^1 in the concavities of the curve. No artefact was ob- served in practice: the radius of the blending volume is often smaller than the curve radius and moreover, most curve concavities are inside objects. We can however see the problem on the bottom right of Figure 5, where the field looks strange inside the implicit shape, where the plane and the drop blend. Defining blending volumes from local-support convolution primitives would solve the problem since it would insure C^1 continuity of the scalar field everywhere and in all cases.

Secondly, the way we identify the size of small features for setting up the blending parameters may not be as general as we would like: in some cases, the two shapes we would like to blend may already have some tiny details (such as bumps on the surface), while the blend should take place at a much larger scale. In general, analysing the different scales of details in the frequency domain and enabling the user to choose the blending range among those scales (or doing it automatically from the size of the intersection curve) could be a better solution. If noisy reconstructed models from scanned real objects were used, such analyses would be necessary, however, this did not make our system less usable in practice, since objects are built from scratch and the designer naturally defined each object from coarse to fine.

Lastly, although our method is usable in interactive modelling sessions, the user sometimes has to wait for a few seconds before the surface updates. This is mainly due to the lack of optimization of our polygonizer, and making blending and subsequent local re-meshing real-time would be a good practical improvement. Future work includes more accurate incremental polygonization, a GPU implementation of our method using, for instance the Intel Larabee architecture.

6. Conclusion and future work

This work has taken a new look at functionally-based implicit surface modelling. Instead of letting surfaces blend rather imprecisely at a distance as in current systems, we propose limiting the blend volume to where surfaces overlap, while enabling models to come close to each other everywhere else without deforming. This new approach combines the strength of implicit surfaces, i.e. their ability to blend seamlessly, with the ease of use of more standard models that do not blend. Our solution is generic and makes implicit surfaces much easier to use for both implicit modelling and animation.

As most previous methods [BDS*03, BWd04, PPK05], our approach defines a binary blending operator, acting on a pair of input surfaces, whereas 'sum' could be n-ary.

Consequently, if, for example, a group of implicit primitives with three overlapping blending volumes, are to be combined, the final shape will depend on the order in which the blends are performed. This is not a problem in an interactive modelling system where the user defines and adds shape components one by one, but this could be a weakness for procedural modelling, where complex, symmetric shapes with multiple branching parts are to be processed. Defining multiple-local blend operators would thus be a good topic for future work.

Also, if the user designs a very complex object composed of both large primitives and a lot of very small details, all composed using our blending operator, the correct computation of the intersection curves will become an issue and a scalable technique will have to be developed.

Lastly, our method offers a good solution to the unwanted blending problem when different primitives are combined, but does not address the self-blending of a single, complex implicit surface. For example modelling a folded snake using a convolution surface defined by a spiral curve as skeleton

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.

is still impossible (except using intricate convolution specific methods [AJC02]), since the snake would self blend. In this case, artificially cutting the object into pieces and re-blending them would solve the problem, but this could be rather time consuming. A more general solution, that only enables blend where a complex shape self-overlaps, would be a good extension to our method.

In conclusion, we hope that our new vision of functionally-based implicit blending will be found inspiring, generate more research in the area and contribute to make implicit modelling more popular.

References

[AC02] ANGELIDIS A., CANI M.: Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Proceedings of the seventh ACM symposium on Solid modeling and applications* (2002), ACM New York, NY, USA, pp. 45–52.

[AJC02] ANGELIDIS A., JEPP P., CANI M.-P.: Implicit Modeling with Skeleton Curves: Controlled Blending in Contact Situations. In *International Conference on Shape Modeling* (2002), pp. 137–144.

[BDS*03] BARTHE L., DODGSON N. A., SABIN M. A., WYVILL B., GAILDRAT V.: Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum 22*, 1 (2003), 23–33.

[BGOS06] BARGTEIL A. W., GOKTEKIN T., O'BRIEN J. F., STRAIN J. A.: A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25*, 1 (2006), 19–38.

[BMPB08] BRODERSEN A., MUSETH K., PORUMBESCU S., BUDGE B.: Geometric texturing using level sets. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 277–288.

[BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L.: Matisse: Painting 2D regions for modeling free-form shapes. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM 2008, June, 2008* (Annecy, France, June 2008), Alvarado C., Cani M.-P., (Eds.), pp. 57–64.

[BWd04] BARTHE L., WYVILL B., DE GROOT E.: Controllable binary csg operators for "soft objects". *International Journal of Shape Modeling 10*, 2 (2004), 135–154.

[CH01] CANI M.-P., HORNUS S.: Subdivision-curve primitives: A new solution for interactive implicit modeling. *Shape Modeling and Applications, International Conference on* (2001), 0082.

[DG95] DESBRUN M., GASCUEL M.: Animating soft substances with implicit surfaces. In *SIGGRAPH'95 Conference Proceedings* (Los Angeles, CA, Aug. 1995), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 287–290.

[dGWvdW09] DE GROOT E., WYVILL B., VAN DE WETERING H.: Locally restricted blending of blobtrees. *Computers & Graphics* (2009).

[Duf92] DUFF T.: Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (1992), pp. 131–138.

[EGB09] EYIYUREKLI M., GRIMM C., BREEN D.: Editing Level-Set Models with Sketched Curves. In *Proceedings of Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling* (2009), pp. 45–52.

[FSSV06] FLÓREZ J., SBERT M., SAINZ M. A., VEHÍ J.: Improving the Interval Ray Tracing of Implicit Surfaces. In *Computer Graphics International 2006* (2006), pp. 655–664.

[Gol05] GOLDMAN R.: Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design 22*, 7 (2005), 632–658.

[GW95] GUY A., WYVILL B.: Controlled blending for implicit surfaces using a graph. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces* (Grenoble, France, Apr. 1995), pp. 107–112.

[Lev03] LEVIN D.: Mesh-Independent Surface Interpolation. In *Geometric Modeling for ScientiŰňĄc Visualization* (2003), pp. 181–187.

[LY03] LJUNG P., YNNERMAN A.: Extraction of intersection curves from iso-surfaces on co-located 3d grids. In *SIGRAD2003 Proceedings, ISSN 1650-3686* (2003), Linkoping Electronic Conference Proceedings, pp. 1650–3740.

[MBWB02] MUSETH K., BREEN D. E., WHITAKER R. T., BARR A. H.: Level Set Surfaces Editing Operators. In *Proceedings of SIGGRAPH'02* (2002), ACM, pp. 330–338.

[OF02] OSHER S. J., FEDKIW R. P.: *Level Set Methods and Dynamic Implicit Surfaces*, 1 ed. Springer, October 2002.

[PASS95] PASKO A., ADZHIEV V., SOURIN A., SAVCHENKO V.: Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer 11*, 8 (1995), 429–446.

[PPK05] PASKO G. I., PASKO A. A., KUNII T. L.: Bounded blending for Function-Based shape modeling. *IEEE Comput. Graph. Appl. 25*, 2 (2005), 36–45.

[Set99] SETHIAN J. A.: *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, 2 ed. Cambridge University Press, June 1999.

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

[Sny92] SNYDER J. M.: Interval analysis for computer graphics. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (1992), pp. 121–130.

[SPOK95] SAVCHENKO V. V., PASKO A., OKUNEV O. G., KUNII T. L.: Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum 14*, 4 (1995), 181–188.

[SPS04] SCHMITT B., PASKO A. A., SCHLICK C.: Constructive sculpting of heterogeneous volumetric objects using trivariate b-splines. *The Visual Computer 20*, 2-3 (2004), 130–148.

[TPF89] TERZOPOULOS D., PLATT J., FLEISHER K.: Heating and melting deformable models (from goop to glop). In *Graphics Interface'89* (London, Ontario, June 1989), pp. 219–226.

[WGG99] WYVILL B., GUY A., GALIN E.: Extending the csg tree - warping, blending and boolean operations in an implicit surface modeling system. *Comput. Graph. Forum 18*, 2 (1999), 149–158.

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data Structure for Soft Objects. *The Visual Computer 2*, 4 (February 1986), 227–234.

[WvO96] WYVILL B., VAN OVERVELD K.: Polygonization of Implicit Surfaces with Constructive Solid Geometry. *Journal of Shape Modelling 2*, 4 (1996), 257–274.

[WW00] WYVILL B., WYVILL G.: Better blending of implicit objects at different scales. In *ACM Siggraph 2000 Sketch Proceedings* (2000), ACM.

© 2010 The Author(s)
Journal compilation © 2010 The Eurographics Association and Blackwell Publishing Ltd.