DOI: 10.1111/cgf.12599

COMPUTER GRAPHICS *forum*
Volume 00 (2015), number 0 pp. 1–12

Convolution Filtering of Continuous Signed Distance Fields for Polygonal Meshes

Mathieu Sanchez[1], Oleg Fryazinov[1], Pierre-Alain Fayolle[2] and Alexander Pasko[1,3]

[1]The National Centre for Computer Animation, Bournemouth University, Poole, UK
{msanchez, ofryazinov, apasko}@bournemouth.ac.uk
[2]Division of Information and Systems, University of Aizu, Aizu-Wakamatsu city, Japan
fayolle@u-aizu.ac.jp
[3]Fusion Institute Global, Tokyo, Japan

Abstract

Signed distance fields obtained from polygonal meshes are commonly used in various applications. However, they can have C^1 discontinuities causing creases to appear when applying operations such as blending or metamorphosis. The focus of this work is to efficiently evaluate the signed distance function and to apply a smoothing filter to it while preserving the shape of the initial mesh. The resulting function is smooth almost everywhere, while preserving the exact shape of the polygonal mesh. Due to its low complexity, the proposed filtering technique remains fast compared to its main alternatives providing C^1-continuous distance field approximation. Several applications are presented such as blending, metamorphosis and heterogeneous modelling with polygonal meshes.

Keywords: geometric modelling, implicit surfaces, polygonal modelling, signed distance fields, distance function

ACM CCS: I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling—Geometric algorithms, languages and systems

1. Introduction

Polygonal meshes are the most common representation of object geometry in computer graphics. The mesh allows to represent geometry in a format easy to understand and convenient to modify. On the other hand, this format restricts the number of operations on geometric objects or does not allow to perform some of these operations easily. For example, the metamorphosis between two meshes can be hard to implement if these meshes have different genus. Therefore, an alternative representation of the mesh object is sometimes needed to provide greater flexibility and a larger spectrum of available operations such as, for example, implicit skinning and gradient-based blending [VBG*13, GBC*13].

The representation of polygonal meshes by discrete or continuous scalar fields has recently attracted a lot of attention in research as well as in application areas because of its properties. Scalar fields can be used efficiently for such operations as controllable blending, metamorphosis between meshes with different topology, surface offsetting, robust mesh repair and others. Applications include function-based heterogeneous object modelling and rendering, rapid prototyping and digital fabrication and simulation of different physical properties in medicine and geology (see [COSL98, FPRJ00, Ju04, LW11, PK08, JBS06] and references therein).

Methods for representing meshes using scalar fields can be distinguished as exact and approximate ones. For exact methods, the iso-value is guaranteed to be zero only on the initial mesh surface, meaning that all the features of the initial models are preserved by the scalar field representation. For approximate methods, such as Radial-Basis Functions (RBFs) [YT02, MYC*01], Multi-level Partition of Unity (MPU) [OBA*03] or Moving Least Squares (MLS) [SOS04] some given approximation error is allowed. Exact and approximate methods are both suitable for visualization, animation, reconstruction and other purposes. However, in some applications, such as medical simulations, approximation errors are not allowed and therefore approximate methods cannot be used.

While the iso-value of the scalar field is required to be zero on the polygonal mesh surface, it is not required to carry any geometric

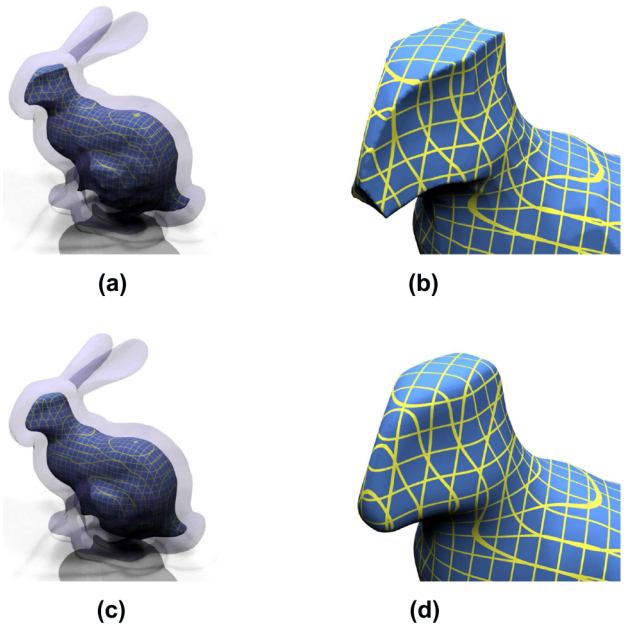

Figure 1: *Offset of the Stanford bunny, (a) and (b) using signed distances, (c) and (d) using our method.*

information in the general case. However, we can distinguish cases where the (absolute) field value corresponds to the Euclidean distance to the polygonal mesh. In this case, the scalar field is called a distance field. We distinguish between signed distance fields and unsigned distance fields. For unsigned distance fields, the iso-value is the distance (or sometimes squared distance) to the mesh and is positive everywhere but on the initial mesh surface. On the other hand, with signed distance fields, the sign is defined by the position of the evaluated point relatively to the interior or the exterior of the mesh object.

One problem of the distance function is that it can lack C^1 continuity, which results in creases when further operations, such as blending, are applied to the distance field. Gradient-based blending [GBC*13] also relies on C^1 continuity. Most approximate methods provide C^1 continuity, but for some applications, such as source-based material interpolation, the exact surface needs to be preserved and simultaneously C^1 continuity away from the surface is desired. A smooth approximation of the distance function can be obtained by taking the convolution with some smooth function (see Figure 1). The convolution is then numerically evaluated. In order to keep the method efficient, an efficient evaluation of the signed distance field is used.

2. Related Work

Methods of representing polygonal meshes with signed distance fields have been increasingly popular because of their numerous applications. These methods can be classified from different points of view into discrete and continuous distance fields, exact and approximate ones, signed or unsigned. In [JBS06], various methods are surveyed according to how the distance is evaluated and propagated across the discrete volume. Below we discuss several methods for computing the signed distance to meshes as well as applications of these methods.

Table 1: *Comparison between different methods for computing a distance field to a polygonal mesh. For the time complexity, n corresponds to the number of the triangles in the input mesh.*

Method	C^1 continuity	Exactness	Complexity
LP-distance [BFP13]	Yes	Yes	$O(n)$
BSP [FPA11]	Yes	Yes	$O(n)$ in the best case
RBF [CBC*01]	Yes	No	$O(n)$
MPU [OBA*03]	Yes	No	$O(n)$
CSRBF [MYC*01]	Yes	No	$O(\log n)$
Signed distances	No	Yes	$O(\log n)$

Approximate signed distance fields were presented in [WK03] where a piecewise linear approximation of the signed distance function was used. In [COSL98], the distance function was interpolated for a mesh with planar cross-sections. The signed distance field can be approximated from its values given at the nodes of a voxel grid, as first introduced in [RP66]. Some discrete approximation methods sample the signed distance or an approximation at the nodes of an octree grid [FPRJ00, Ju04]. An approximation of the distance field to a point cloud was presented in [CT11]. An approximation to the signed distance to noisy point cloud data is discussed in [MdGD*10]. A physics-based level set method can be used as well as in [ZO02]. Approximate methods can be very fast to evaluate the signed distance value, however they are inaccurate and cannot provide a continuous real function without a proper interpolation procedure and therefore cannot be used in a number of applications.

A continuous approximation of the signed distance to a polygonal mesh can be obtained by representing the object with set-theoretic operations on the half-spaces bounded by the planes passing through the polygonal faces [FPA11]. This method provides a continuous function, but the distance query can be slow and numerically unstable especially for large input meshes.

The signed distance to a polygonal mesh can be computed from the heat flow [CWW13] or by anisotropic diffusion [CDR00]. These methods use numerical solutions of partial differential equations and therefore require discretization with either a regular grid or a volumetric mesh resulting in inaccuracies.

Table 1 compares different methods for computing a distance field to a polygonal mesh. Most of the methods have linear (time) complexity with respect to the number of triangles in the input mesh. In addition, most of these methods are performing expensive operations per triangle or per vertex, making them slow in practice for large meshes.

In this paper, we deal with exact distance fields and therefore consider only methods for representing the input polygonal mesh exactly. These methods can be separated into two categories:

- Methods utilizing a full enumeration of all the geometric entities, i.e. vertices and faces or half-spaces built on the faces of the initial polygonal mesh;

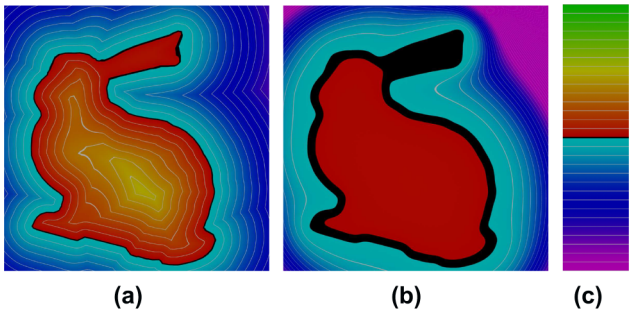

Figure 2: *(a) The signed distance field to the Stanford bunny; (b) An approximation of the distance field with the L_p distance field for $p = 1$; (c) The colour-map used in (a) and (b). The black line corresponds to values close to the zero-set (narrow range of zero).*

- Methods calculating the iso-value of the scalar field using pre-built data structures over the initial geometric data.

Methods from the first category usually have at least linear time complexity and are expensive for large input polygonal meshes. For example, L_p−distance fields [BFP13] provides a smooth approximation of the signed distance, however the computational cost is high, and the method is slow for large polygonal meshes. The particular case $p = 1$, corresponding to the inverse of the sum of Mean-Value Coordinates (MVC) weights [JSW05], is illustrated in Figure 2(b). Another example is the BSP-fields method [FPA11], where the resulting function is constructed by applying set-theoretic operations to the half-spaces bounded by the initial polygonal mesh faces. This method is more suitable for meshes with a small number of polygons, because each face supporting plane is involved at least once in the computation of the scalar field value at a given point. In the case of n faces, at least n half-spaces and $n-1$ set-theoretic operations are used, however if the Binary Space Partitioning (BSP)-tree is splitting faces during its construction, which happens in practice for most polygonal meshes, the plane equations and set-theoretic operations are used several times per split face. In addition, the implementation used for the set-theoretic operations affects the resulting field significantly. For example, min/max operations result in C^1 discontinuities. On the other hand, SARDF operations [FPS08] preserve C^1 and the approximate distance but are computationally more expensive.

The second category of exact methods includes optimized evaluations of the signed distance field (as used for example in Figure 2a). These are discussed in the next section.

In this work, we stay within the class of methods dealing with a continuous real function representing the signed Euclidean distance to the polygonal mesh, which was first introduced in [PT92]. These methods have been increasingly popular because of their numerous applications, such as collision detection [GBF03], rendering [Har96], intersection free mesh offset or shell [LW11, PK08] and heterogeneous object modelling [BST04]. Surveys of applications can be found in [JBS06]. In the following section, we discuss different techniques for the efficient evaluation of signed distance fields.

3. Signed Distance Fields

Formally, the signed distance function can be defined as:

$$f(\mathbf{p}) = sign(\mathbf{p};\Omega)\ dist(\mathbf{p};\Sigma),$$

where Ω is a point set bounded by the polygonal mesh Σ,

$$dist(\mathbf{p};\Sigma) = \inf_{\mathbf{x}\in\Sigma} \|\mathbf{x}-\mathbf{p}\|$$

is the unsigned distance from $\mathbf{p}$ to the polygon mesh Σ, and

$$sign(\mathbf{p};\Omega) = \begin{cases} +1 & \mathbf{p}\in\Omega \\ 0 & \mathbf{p}\in\Sigma=\partial\Omega \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

is the sign function.

Naive computation of the distance from a point to the mesh is an expensive procedure. However, it can be accelerated by using spatial structures, pre-processing and sorting the mesh polygons, application of different traversal strategies for the selected spatial structure and using hardware acceleration. In [SFP13], different spatial structures, building and traversal strategies are discussed in details and compared to present the optimal way to calculate the distance function to a polygonal mesh. This work also discusses how to improve the performance of a single query of the distance by using packet sampling. Often the function has to be sampled several times within a small volume. In ray-tracing, a similar problem occurs and is referred to as coherent rays processing. The related process is called packet sampling. Each sample is likely to go through the same branches of the tree, and it is inefficient to iteratively do each query separately. Instead, all the sample points are grouped and the tree is traversed once for all, which improves cache efficiency.

Evaluation of the sign can be done by different approaches. One of the fastest yet reliable methods is the angle-weighted pseudo normals method introduced in [BA05]. This method requires the input polygonal mesh to be watertight, i.e. free from holes and self-intersecting triangles. Ray-casting is another common solution [Req96], however it is at best $O(\log n)$ and requires robust ray–surface intersection procedures to avoid numerical errors. Slower, but more robust solutions exist for non-watertight meshes, for example, by computing the winding number [JKSH13]. In our experiments, we only considered watertight meshes and therefore used only the angle-weighted pseudo normals approach. Note that the sign computation generally does not depend on the distance computation and the distance can be evaluated for both watertight and non-watertight meshes, while the methods used for sign computation directly depends on the quality of the input mesh.

4. Convolution

As mentioned above, the (signed) distance function is generally not C^1 (non-differentiable at some points). However, a smooth function approximating the distance function can be obtained by convolution with some suitable functions. If f is the signed distance function, it can be replaced by the function g defined by

$$g(\mathbf{p}) = \int_{\mathbb{R}^3} f(\mathbf{p}-\mathbf{s}\,h(\mathbf{p}))\,w(\mathbf{s})d\mathbf{s}, \quad (2)$$

where w is a normalized smooth kernel function, h is a function that controls the kernel size, such that $h(\mathbf{p}) = 0$ if $f(\mathbf{p}) = 0$, and $\mathbf{s}$ is a displacement vector over the whole space where the signed distance field f is defined. Some examples of functions w and h are discussed below.

The scalar field defined by the function g has the same zero level set as the function f. It comes from the requirements to the function h: if $f(\mathbf{p}) = 0$ then $h(\mathbf{p}) = 0$, $\mathbf{p} - \mathbf{s}\,h(\mathbf{p}) = \mathbf{p}$. Therefore, $f(\mathbf{p} - \mathbf{s}\,h(\mathbf{p})) = 0$ if $f(\mathbf{p}) = 0$.

If we perform the substitution $\mathbf{u} = \mathbf{s}h(\mathbf{p})$, we have $d\mathbf{u} = h(\mathbf{p})d\mathbf{s}$ and, providing that $h(\mathbf{p}) \neq 0$, we can introduce the function $w_h(\mathbf{u}) = \frac{1}{h(\mathbf{p})} w(\frac{\mathbf{u}}{h(\mathbf{p})})$ and rewrite (2) as

$$g(\mathbf{p}) = \int_{\mathbb{R}^3} f(\mathbf{p} - \mathbf{u})\, w_h(\mathbf{u}) d\mathbf{u}. \tag{3}$$

From (3), g looks like a convolution: $g = f \star w_h$. The smoothness of g as defined in (2) will depend on the definition of w and h.

4.1. Kernel size function *h*

The function h is introduced to smoothly interpolate the values from 0 to 1 as we move away from the zero level set. It should be monotonically increasing on $\mathbb{R}^+$, smooth and such that $h(0) = 0$ and $h(t) = 1$, for $t \geq f_c$, given a capping value f_c. One of ways to define such a function h is to use the smoothstep function:

$$h(\mathbf{p}) = 3r(\mathbf{p})^2 - 2r(\mathbf{p})^3, \tag{4}$$

where $r(\mathbf{p}) = min(\frac{|f(\mathbf{p})|}{f_c}, 1)$. Note that this function does not depend on the sign of f. Reducing the kernel size to zero on the mesh surface preserves the exact surface representation by g.

4.2. Kernel function *w*

The kernel function w should take its maximum at $\mathbf{0}$ and smoothly converge to 0 as its argument goes to infinity with any of its coordinates. There are various ways to define such functions; examples are the Gaussian function and the bump function.

The Gaussian function is defined as $w_a(\mathbf{u}) = \left(\frac{a}{\pi}\right)^{\frac{3}{2}} e^{-a||\mathbf{u}||^2}$, where the parameter $a \in \mathbb{R}^+$ controls the width of the Gaussian (the standard deviation $\sigma = 1/\sqrt{2a}$ is also sometimes used). The larger a is, the closer g will approximate f. Gaussian curves for different values of a (1, 10, 20 and 30) are illustrated in the one-dimensional case in Figure 3.

For the bump function, we use the following definition:

$$w(\mathbf{u}) = \begin{cases} Ce^{\frac{1}{||\mathbf{u}||^2 - 1}} & \text{if } ||\mathbf{u}|| < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

$$w_b(\mathbf{u}) = b^{-3} w\left(\frac{\mathbf{u}}{b}\right), \tag{6}$$

where C is such that $\int w = 1$ and the parameter b controls the width of the bump function. The smaller b is, the closer g will approximate

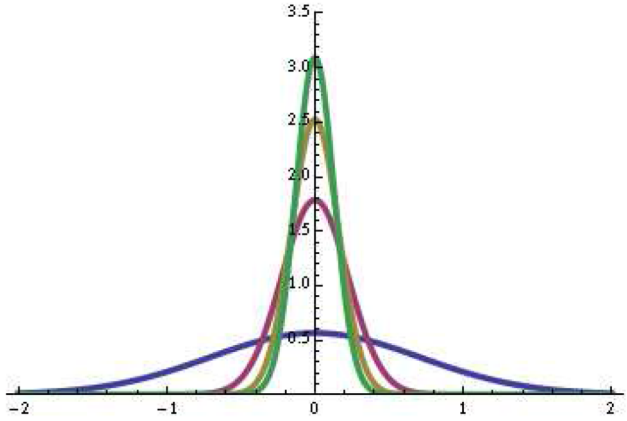

Figure 3: *Gaussian weight function for different values of the parameter a:* 1 *(blue),* 10 *(purple),* 20 *(yellow) and* 30 *(green).*

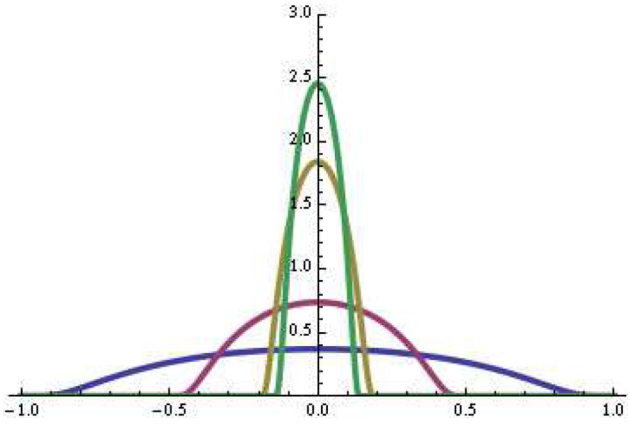

Figure 4: *Bump function for various values of the parameter b:* 0.15 *(green),* 0.2 *(yellow),* 0.5 *(purple) and* 1 *(blue).*

f. Bump functions for various values of b (0.15, 0.2, 0.5 and 1) are illustrated in Figure 4 (in one dimension).

Unlike the Gaussian functions, the bump functions have compact support.

For $\mathbf{p}$ such that $|f(\mathbf{p})| \geq f_c$, $h(\mathbf{p}) = 1$, thus $g = f \star w$. For the choices of w proposed above, g is smooth. Otherwise, the distance function appears in w_h via h and therefore g is not necessarily smooth (it can have discontinuous partial derivatives, same as the distance function).

4.3. Selection of parameters

The parameter f_c controls the distance to the boundary outside of which g is guaranteed to be smooth. It needs to be small enough to exclude points where f is not smooth, or taken as small as possible otherwise.

The parameter a or b, depending on the choice for the kernel function, controls the width of the kernel function. It gives a

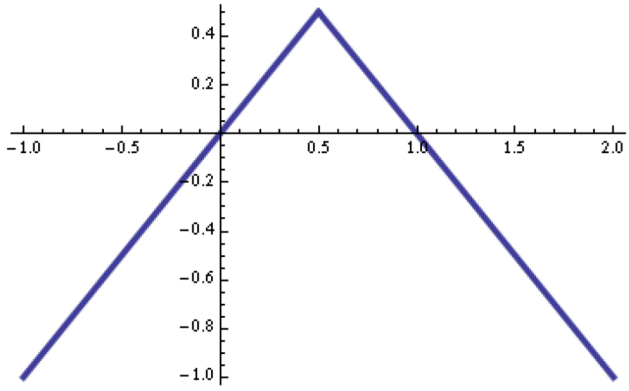

Figure 5: *Plot of the distance to the boundary of the segment* [0, 1].

trade-off between how close g approximates the distance function f and the shape of the level sets of g. It should be selected based on the type of applications.

As an illustration, we consider the distance to the boundary of the segment [0, 1] (illustrated in Figure 5). For the weight function, a Gaussian is used with $a = 50$. At first, we set: $f_c = 1$. Note that in this case the discontinuity of the derivative of f happens at $x = 0.5$; since $0.5 < f_c$, the discontinuity of the derivative will not be smoothed. See Figure 6 for the plot of $g(x)$ corresponding to this case. Note the discontinuity of the derivative of g at $x = 0.5$ in the zoom (right image).

By setting $f_c = 0.25$, we can get a smooth approximation g as illustrated in Figure 7. In this particular case, we can compute the derivative of g analytically and evaluate it at $x = 0.5$ to verify that it is 0.

By varying the parameter a, we affect the shape of the level sets at the expense of the distance approximation. Figure 8 illustrates the results obtained for $a = 10$, 100, 1000 and 10 000 (note that an increasing value of a corresponds to a decreasing width for the kernel function).

While g is continuous, it is possible to create additional extrema, or extra zero level sets for some selection of f_c and a (or b). For example, extra zero level sets can be obtained by picking a small value for f_c (to limit or avoid the domain with discontinuous derivative) and a large Gaussian width. This case is illustrated in Figure 9. This example was produced by considering the distance to the boundary of the segment [0, 1] (see Figure 5) and by setting $f_c = 0.1$ and $a = 2.0$.

It is possible to prevent additional zero level sets by using a modified version of (2):

$$g(\mathbf{p}) = sign(\mathbf{p}; \Omega) \int_{\mathbb{R}^3} |f(\mathbf{p} - \mathbf{s}\, h(\mathbf{p}))|\, w(\mathbf{s}) d\mathbf{s},$$

where we use the unsigned distance in the integrand, and compute the sign of the function at the end. While this approach solves the problem of the extra zero level sets, it does not prevent additional extrema (with respect to the exact signed distance function).

Alternatively, we can achieve a similar result when a bump function (with compact support) is used, by setting its parameters appropriately. We remark that the capping distance f_c and the width of the bump function b are related if we want to prevent additional zero level sets. If b is set, then f_c has a lower bound depending on b in order to prevent the filtering region to cross the surface boundary (where the sign of f changes), and therefore create additional zero level sets. Similarly, a given f_c implies an upper bound on b. Since we want to avoid any non-zero weighted $\mathbf{s}$ to cross the surface, the following inequality must hold for all $\mathbf{p}$:

$$b\, h(\mathbf{p}) \leq |f(\mathbf{p})|$$

using (4), it expands to

$$b\,(3(min(\frac{|f(\mathbf{p})|}{f_c}, 1))^2 - 2(min(\frac{|f(\mathbf{p})|}{f_c}, 1))^3) \leq |f(\mathbf{p})|.$$

Through substitutions and solving, we reach the following results:

$$b \leq \frac{8 f_c}{9}, \tag{7}$$

$$f_c \geq \frac{9b}{8}. \tag{8}$$

5. Numerical Evaluation

In the general case, the integral defined in (2) cannot be evaluated analytically and therefore a numerical approximation is required. The convolution (2) can be approximated by the following finite summation:

$$g(\mathbf{p}) \approx \sum_{i=1}^{n} f(\mathbf{p} - \mathbf{s}_i\, h(\mathbf{p}))\, w_i, \tag{9}$$

where n is the number of samples, $f(\mathbf{p})$ is the distance function being filtered at the point $\mathbf{p}$, $\mathbf{s}_i$ is the ith sample, $w_i = w(\mathbf{s}_i)$ is the weight associated to the sample $\mathbf{s}_i$ and h is the function controlling the size of the kernel. Equation (9) can be used when the samples $\mathbf{s}_i$ are on a regular grid.

It is possible to get better results by sampling, for example, from the distribution with density w (assuming that w is normalized). In this case, an approximation of (2) is obtained by

$$\frac{1}{n} \sum_{i=1}^{n} f(\mathbf{p} - \mathbf{s}_i\, h(\mathbf{p})), \tag{10}$$

where $\mathbf{s}_i$ are sampled from the distribution with density w. This corresponds to the standard Monte-Carlo approximation of integrals.

The quality of the result depends on the number of samples, their distribution and weights. Because most of the samples are likely to be in the same neighbourhood, packet sampling can be used to evaluate efficiently all the samples at once. We discuss below further details for the filter evaluation.

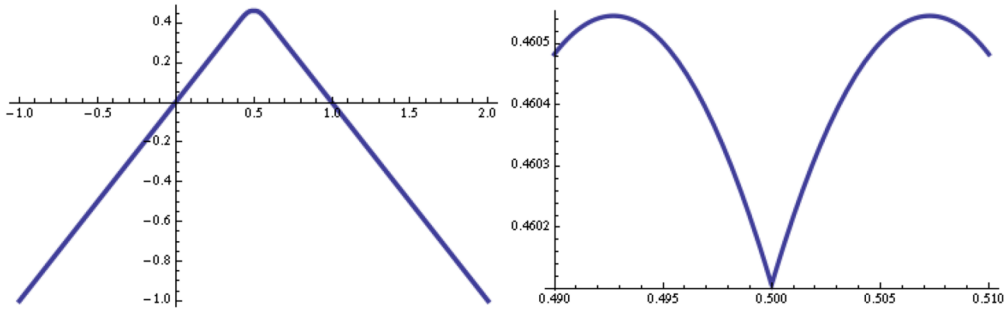

Figure 6: *Plot of g using* $f_c = 1.0$. *Right: zoom near the point* $x = 0.5$.

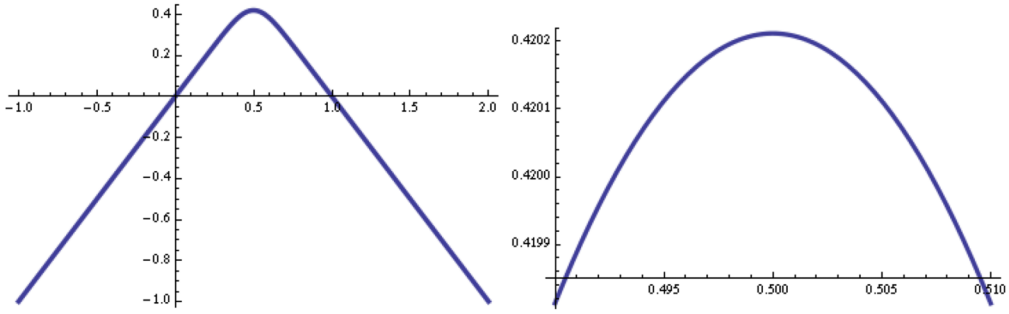

Figure 7: *Plot of g using* $f_c = 0.25$. *Right: zoom near the point* $x = 0.5$.

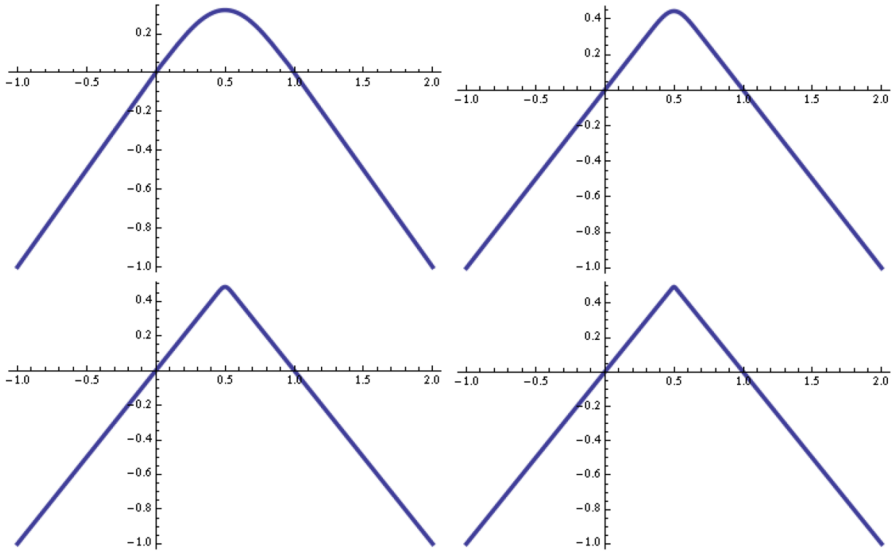

Figure 8: *Influence of the kernel width on the level set shapes. First row: left:* $a = 10$*; right:* $a = 100$. *Second row: left:* $a = 1000$*; right:* $a = 10\,000$.

5.1. Sample distribution and weights

To efficiently and accurately evaluate (2) numerically the samples should be distributed inside some volume. Equation (2) suggests sampling in the entire space, at least for the Gaussian kernel, while for the bump function we only need to sample into a finite volume. For practical reasons, when (9) is used for the approximation, we limit ourselves to a finite volume (called a unit volume) near the evaluation point. Two obvious ways to define a unit volume are: a unit sphere centred at the query point and a unit cube. For efficiency purposes, we used a unit cube in our experiments. Therefore, all the samples are distributed inside a unit cube and defined by their

position and weight. It is clear that the more samples we have, the closer the approximation to the integral is, but at the same time the less efficient the method is, so a balance has to be found. To distribute the samples inside the unit cube, different approaches can be used. The most naive solution, which is largely used in discrete filters, is a regular pattern of a 3^3 grid where each sample point has a weight given by the function w. However, our experiments show that the resulting field still has creases even if we increase the number of samples in the regular grid. Instead we experimentally found out that it is more efficient to draw samples following the function w. In this case, more samples are drawn where the weights are larger.

For this purpose, the approximation in (10) is used, where $\mathbf{s}_i$ are sampled directly from the distribution w. When a Gaussian is selected, the Box–Muller transform is used to sample from it. For the bump function, we use rejection sampling (a uniform distribution over $[-b, b]^3$ can be used as a proposal distribution). Alternative approaches such as adaptive rejection sampling or importance sampling could be used as well. But it would not result in a significant difference, since the samples are only computed once during initialization.

Results obtained with these different approaches are illustrated in Figure 10. The regular uniform filter has visible C^1 discontinuities just like the signed distance field. The Gaussian distributed samples provide visually smoother fields. In Figure 11, we show how the number of samples affects the quality of the result. In practice, numerical filters provide good results while preserving the efficiency of the method.

5.2. Adaptive quality

As mentioned above, the number of samples influence the quality of the result as well as the efficiency. To achieve good quality without sacrificing computational efficiency, the number of samples is adaptively changed across space. From our experiments, a low number of samples is able to approximate the convolution reasonably well far from the surface or the medial axis. Therefore, we suggest the number of samples to be adaptively increased only around the surface and the medial axis.

Computing the medial axis and the distance to the media axis is a difficult task. However, it can be approximated by analysing the difference between the value at a central point in space and the values at points in its neighbourhood as observed in [GS99]. The farther the query point from the medial axis, the closer the neighbourhood average to the central point value will be. Figure 12 shows the number of samples (green for low, red for high) when using the medial axis detection field introduced in [GS99]. If the first samples are on a uniform $3 \times 3 \times 3$ grid, then we can use them as a reliable neighbourhood average. Finally, we use a smooth step as a transfer function to control the number of samples. We use a similar process using the distance and a transfer function to control the number of samples based on the distance to the surface. Both of those functions are mixed together and interpolate between the minimum and maximum number of samples:

$$n_a(\mathbf{p}) = N_{min} + \frac{n_{medial} + n_{distance}}{2} (N_{max} - N_{min}).$$

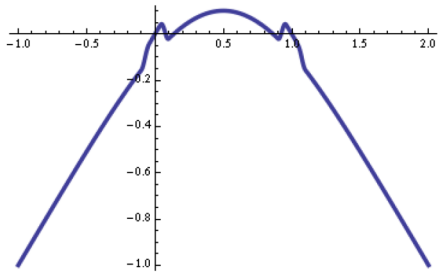

Figure 9: *By using a large width for the Gaussian and a small value for f_c, it is possible to obtain unwanted extra zero level set.*

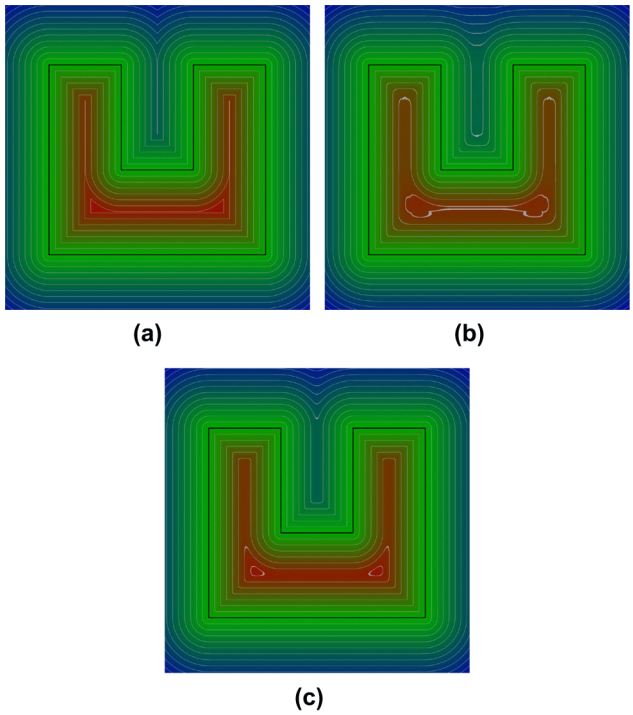

Figure 10: *Filtering the distance field: (a) shows the signed distance field, (b) uses a regular grid of samples used with (9), (c) uses (10) with a Gaussian distribution for w and the samples drawn from this distribution.*

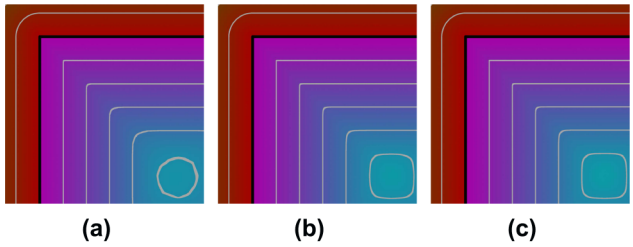

Figure 11: *Filtering of distance fields with (a) 32 samples, (b) 64 samples and (c) 92 samples.*

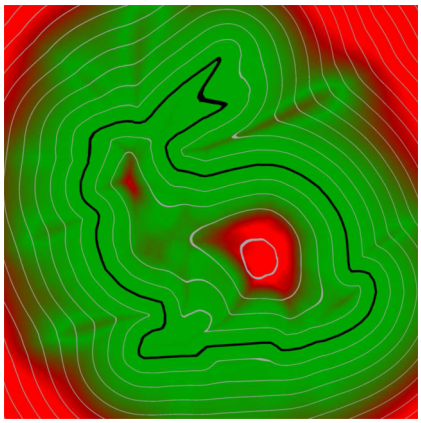

Figure 12: *Medial axis detection using the initial samples. Green indicates low sampling rate while red means high sampling rate.*

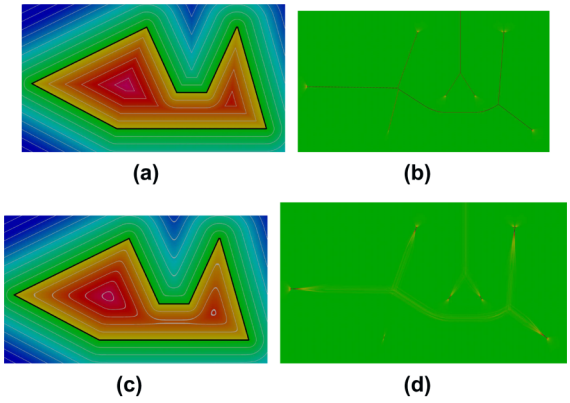

Figure 13: *The pictures (a) and (c) show the iso-values of a scalar field, (a) for the signed distance, (c) our method. The pictures (b) and (d) show the derivative discontinuities using edge detection.*

The function n_a provides a real value to control the number of samples. Using just the integer part of this value is not effective because the change in the number of samples introduces noise in the resulting field due to the discrete jumps. Instead, we use the fractional part to weight the last sample.

6. Results and Applications

The quality of filtering can be evaluated visually by applying operations on these fields. In Figure 13, we show how filtering affects the gradient field. Figures 13(b) and (d) illustrate the results obtained with the exact signed distance function and our method when an edge detection filter is applied to the gradient field. The edge detection filter uses 3^3 samples with the weights all set to -1 except for the central sample which is set to $n-1$, where n is the number of samples. We remark that the distance field contains several clear discontinuity of the gradient, while our method removes most of them.

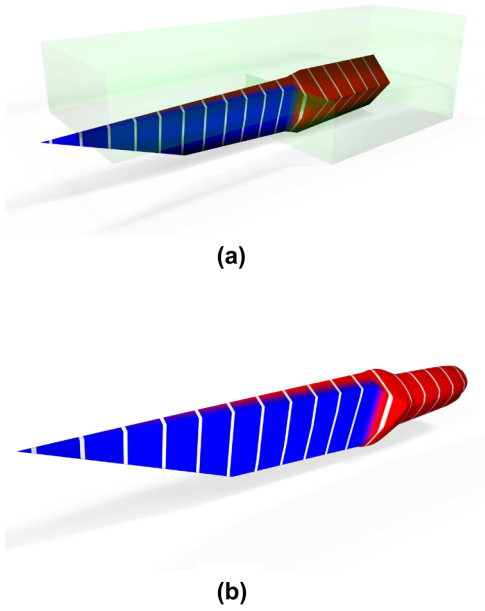

Figure 14: *Localized smoothing: (a) the original model and the smoothing volume (light blue); (b) the resulting shape, using smoothing through convolution filtering.*

Convolution filters have several applications in shape modelling. In Figure 1, our method is applied to compute a smooth offset from the Stanford bunny. An alternative approach for computing an offset could be to adapt the recent method for computing point-set morphology [CB14]. In the following, we describe additional applications of convolution filtering.

6.1. Localized smoothing

Filters can be used to smooth the shape selectively. To achieve localized smoothing, two C^1-continuous functions are needed. The function f_o represents the original object, while the function f_s represents the smoothing volume. The function h in (9) is replaced by a function which converts the values from f_s into a filter size value. Here, we use a smooth step function to remap the values from f_s to a kernel size value. For any point outside the smoothing volume, we do not need any filtering and therefore keep the value $f_o(\mathbf{p})$. If the point is within the volume f_s, then, convolution filtering can be applied by replacing $h(\mathbf{p})$ in (9) by:

$$h_m(\mathbf{p}) = smoothstep(\frac{|f_o(\mathbf{p})|}{s_d})\, s_r,$$

where s_d is the distance from the boundary of the smoothing object at which the kernel size will reach its maximum value and s_r is the smoothing radius (i.e. the maximum kernel size).

Figure 14 shows how a knife model can be made by defining first a shape with sharp edges all around. The smoothing volume is then defined around the handle and the top of the blade. This smoothing volume controls the size of the filter kernel allowing a smooth handle and a blade sharp only on one side. Figure 15 shows

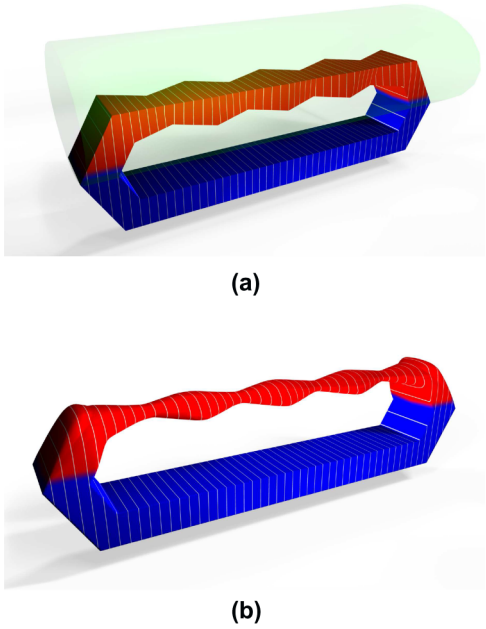

Figure 15: *Modelling a smooth handle with localized smoothing: (a) the original model and the smoothing volume (light blue) (b) the resulting shape using convolution filtering inside the smoothing volume.*

that a smooth handle can be made from a basic shape following the same procedure.

6.2. Blending

Simple blending with addition or subtraction of material for smooth transition between two objects was introduced in [PASS95]. The main idea is to apply an R-function defining a set-theoretic operation between two objects and apply some displacement to the resulting value. The result of the blending operation is the smooth transition between the initial surfaces. C^1 continuity is crucial for blending operations. When applied to exact distance functions, the additive blending shows a sharp edge crossing the otherwise smooth additional material. Figure 16 shows the effect of C^1 discontinuity on blending union (c), and the result obtained using our method (e).

In Figure 17, a gear model is subtracted from a sphere using a blending difference. The C^1 discontinuities of the signed distance fields create visible edges in the model as seen in Figure 17(b). Using convolution filters on signed distance fields, the sharp edges are smoothed out.

6.3. Smooth metamorphosis

Metamorphosis also benefits from C^1-continuous fields. C^1 discontinuities introduce unwanted creases during the transformation. The metamorphosis is a weighted sum of the two fields in its simplest case, but other methods also rely on some summation of the two fields. During intermediate frames, the surface will pass through the C^1 discontinuities present in each field. This causes creases on

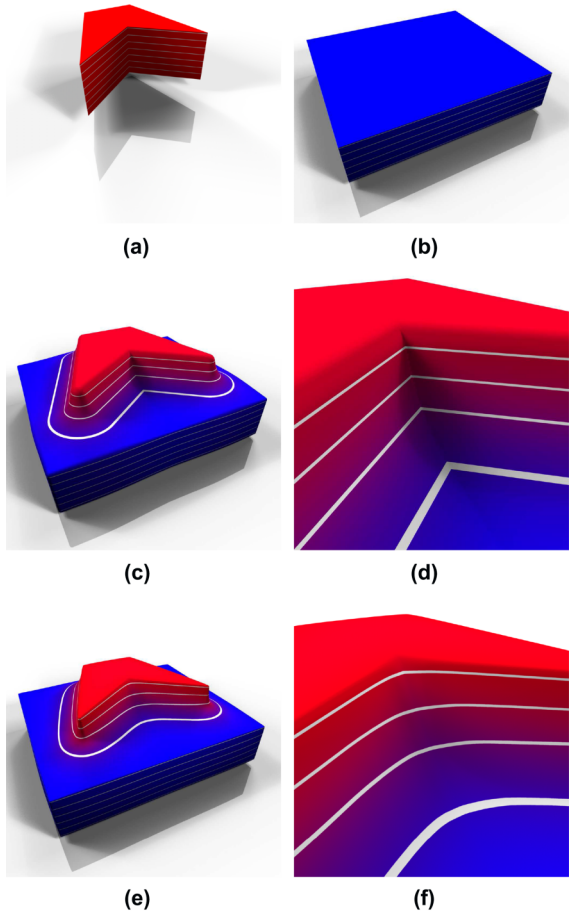

Figure 16: *Blending union operation: (a) and (b) show two initial objects; (c) the blending union with signed distances; (d) zoom to the discontinuity area; (e) and (f) show the same operation with filtered signed distances.*

the surface. Using filtered distance fields, the intermediate shapes look smoother and only have creases as they get closer to one of the interpolated shapes. Figure 18 shows intermediate shapes of the metamorphosis between a fan disk and a mechanical part with sharp features (a and b), using signed distances (c and d) and our method based on filters (e and f). The metamorphosis here is achieved using a simple linear interpolation between the values of each field.

6.4. Heterogeneous modelling

C^1-continuous fields are crucial for heterogeneous multi-material modelling to avoid stress concentrations. We apply transfinite interpolation as described in [RSVT00] using our filtered field to achieve better results than with exact signed distance functions, and with more control and faster computations than L_p-dist fields [BFP13]. The transfinite interpolation uses two (or more) source features defining different materials. The material properties in-between the features are interpolated across space blending both properties based on the distances to the surface boundary of each feature. The formulation relies on distance

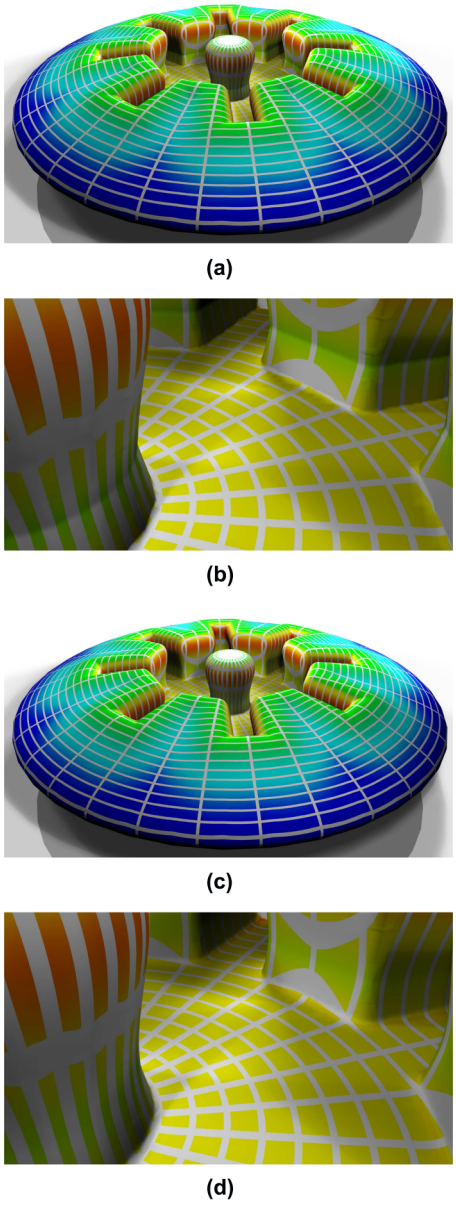

Figure 17: *Blending difference operation: (a) the blending difference between a gear and a sphere with signed distances; (b) zoom to the discontinuity area; (c) and (d) show the same operation with filtered signed distances.*

properties, but the C^1 discontinuities cause stress concentrations and other issues due to the loss of differential properties as discussed in [BST04].

Figure 19 shows the transfinite interpolation using different method for computing an approximation of the distance fields. The white and grey stripes represent the source feature shapes, and the colours define the distribution of each material. Using exact distance fields, Figure 19(a), there are visible C^1 discontinuities which are problematic for the solidity of the final object. Using L_1-dist field

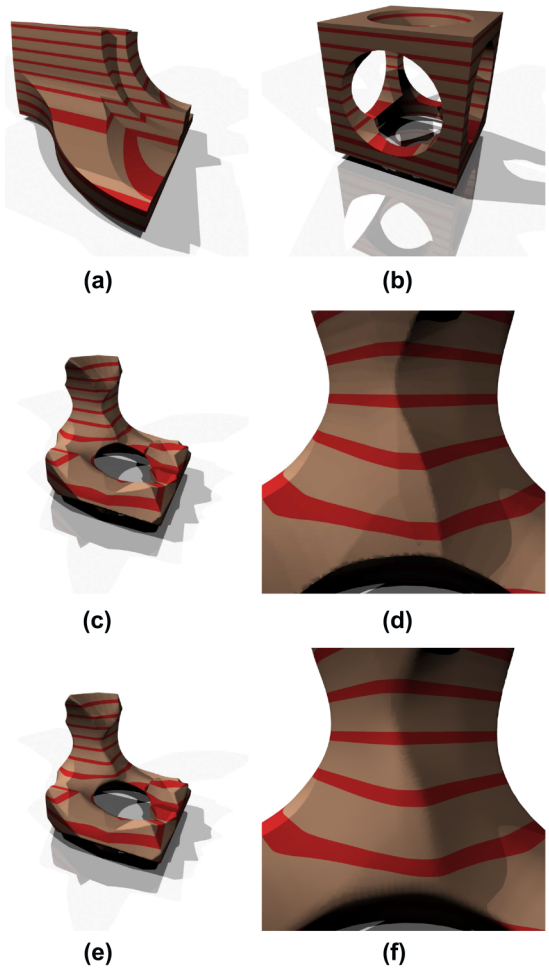

Figure 18: *Metamorphosis between a fan disk (a) and a mechanical part (b). In (c) and (d): signed distances are used. In (e) and (f): smoothed distance fields are used.*

instead of distances, Figure 19(b), is slower, but also poorly approximates the distances far away from the boundary. Figure 19(c) uses convolution filtering applied to both feature shapes with a filter size b of 3.5 reached at capping distance f_c of 5. Figure 19(d) uses $b = 1.5$ and $f_c = 5.0$ and (e) uses $b = 1.5$ and $f_c = 2.0$. These figures illustrate the effect of b as well of f_c on the field. Figure 19(f) uses $b = 0.5$ and $f_c = 2.0$. Such parameters violate the inequality introduced in (7). This results in additional zero level sets and bad behaviour of the field. Overall, the filters succeeded to create C^1-continuous material blending. The maximum filter region can be adjusted and the capping distance f_c lets the user control how close to the surface the smoothing should happen.

7. Conclusion and Discussion

One needs to compute a scalar field to a polygonal mesh in order to apply some specific function-based operations to the scalar field or to use it within a general function-based modelling environment. Computing the signed Euclidean distance to the mesh is one of the

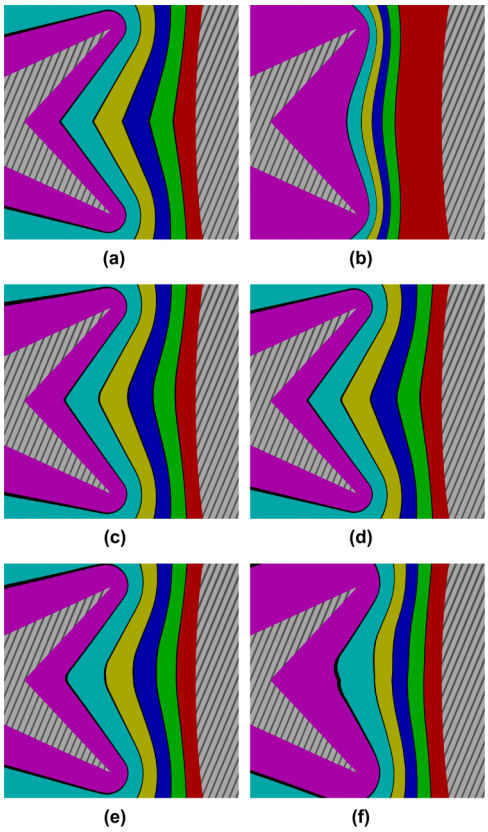

Figure 19: *Transfinite interpolation of material properties between two material features: the top left is the distribution of materials using exact signed distances, top right (b) uses the L_1-dist field. The other examples use filters with various parameters to control the smoothing.*

possible methods, as it can be evaluated very efficiently for any polygonal mesh. The main concern with the Euclidean signed distance field is that it is generally non-differentiable (C^1-discontinuous). We proposed to smooth the distance field by taking its convolution product with a smooth kernel. The integral is numerically evaluated as a finite summation. The introduced filtering procedure for multiple coherent queries was used for smoothing the distance field in order to efficiently compute a smooth field. Several experiments show that the resulting field is smoother and therefore is more suitable for operations such as blending, metamorphosis and other general modelling operations in the context of a function representation modelling system.

The proposed approach relies on the efficient evaluation of the signed Euclidean distance. In our approach, we used a BVH (Bounded Volume Hierarchy) structure to achieve maximum performance. In addition, we benefit from packet sampling as most of the points where the distance should be evaluated lie in the neighbourhood of the query point. However, our approach would benefit from more efficient evaluations of the signed Euclidean distance, which is still an open research as several methods have been introduced recently. We reviewed most of them in [SFP13], but a more detailed survey of the current state of the art has yet to be done. One of the ways to increase the efficiency is the acceleration by using graphics hardware (GPU) implementation. However, it relies on the size of the input mesh and does not handle large input meshes well.

Another direction of future work is related to the convolution itself. Convolution proved to be useful to smooth the field. Convolution with varying kernel size across space have other potential applications which have yet to be investigated. In this paper, we used a numerical method based on a Monte-Carlo approach to approximate the integral. Other numerical techniques can be investigated to evaluate this integration, such as for example the Fast Multipole Method. However, its direct application is not straightforward and requires additional research.

Acknowledgements

This work was partially sponsored by the EU Interreg IVA project 5-023-FR_SHIVA. The Stanford bunny model is courtesy of the Stanford University Computer Graphics Laboratory.

References

[BA05] Baerentzen J. A., Aanaes H.: Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics 11* (May 2005), 243–253.

[BFP13] Belyaev A., Fayolle P.-A., Pasko A.: Signed Lp-distance fields. *Computer Aided Design 45*, 2 (Feb. 2013), 523–528.

[BST04] Biswas A., Shapiro V., Tsukanov I.: Heterogeneous material modeling with distance fields. *Computer Aided Geometric Design 21*, 3 (Mar. 2004), 215–242.

[CB14] Calderon S., Boubekeur T.: Point morphology. *ACM Transactions on Graphics* (2014). (Proc. SIGGRAPH 2014), *33*, 45:1–45:13.

[CBC*01] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C., Evans T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 67–76.

[CDR00] Clarenz U., Diewald U., Rumpf M.: Anisotropic geometric diffusion in surface processing. In *VIS '00: Proceedings of the Conference on Visualization '00* (Los Alamitos, CA, USA, 2000), IEEE Computer Society Press, pp. 397–405.

[COSL98] Cohen-Or D., Solomovic A., Levin D.: Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics 17* (Apr. 1998), 116–141.

[CT11] Calakli F., Taubin G.: SSD: Smooth signed distance surface reconstruction. *Computer Graphics Forum 30*, 7 (2011), 1993–2002.

[CWW13] Crane K., Weischedel C., Wardetzky M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics 32*, 5 (Oct. 2013), 152:1–152:11.

[FPA11] Fryazinov O., Pasko A. A., Adzhiev V.: BSP-fields: An exact representation of polygonal objects by differentiable scalar fields based on binary space partitioning. *Computer-Aided Design 43*, 3 (2011), 265–277.

[FPRJ00] Frisken S. F., Perry R. N., Rockwood A. P., Jones T. R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 249–254.

[FPS08] Fayolle P.-A., Pasko A., Schmitt B.: SARDF: Signed approximate real distance functions in heterogeneous objects modeling. In *Heterogeneous Objects Modelling and Applications*. Springer-Verlag, Berlin, Heidelberg (2008), pp. 118–141.

[GBC*13] Gourmel O., Barthe L., Cani M.-P., Wyvill B., Bernhardt A., Paulin M., Grasberger H.: A gradient-based implicit blend. *ACM Transactions on Graphics 32*, 2 (Apr. 2013), 12:1–12:12.

[GBF03] Guendelman E., Bridson R., Fedkiw R.: Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics 22* (July 2003), 871–878.

[GS99] Gagvani N., Silver D.: Parameter-controlled volume thinning. *Graphical Models and Image Processing 61*, 3 (May 1999), 149–164.

[Har96] Hart J. C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (1996), 527–545.

[JBS06] Jones M. W., Baerentzen J. A., Sramek M.: 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12* (July 2006), 581–599.

[JKSH13] Jacobson A., Kavan L., Sorkine-Hornung O.: Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics 32*, 4 (July 2013), 33:1–33:12.

[JSW05] Ju T. Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics 24* (July 2005), 561–566.

[Ju04] Ju T.: Robust repair of polygonal models. *ACM Transactions on Graphics 23* (Aug. 2004), 888–895.

[LW11] Liu S., Wang C. C.: Fast intersection-free offset surface generation from freeform models with triangular meshes. *IEEE Transactions on Automation Science and Engineering 8*, 2 (2011), 347–360.

[MdGD*10] Mullen P., deGoes F., Desbrun M., Cohen-Steiner D., Alliez P.: Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum 29* (2010), 1733–1741.

[MYC*01] Morse B. S., Yoo T. S., Chen D. T., Rheingans P., Subramanian K. R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 89–98.

[OBA*03] Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics 22* (July 2003), 463–470.

[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.

[PK08] Pavic D., Kobbelt L.: High-resolution volumetric computation of offset surfaces with feature preservation. *Computer Graphics Forum 27*, 2 (2008), 165–174.

[PT92] Payne B. A., Toga A. W.: Distance field manipulation of surface models. *IEEE Computer Graphics and Applications 12* (January 1992), 65–71.

[Req96] Requicha A.: *Geometric Modeling: A First Course*. University of South California. 1996.

[RP66] Rosenfeld A., Pfaltz J. L.: Sequential operations in digital picture processing. *Journal of ACM 13* (Oct. 1966), 471–494.

[RSVT00] Rvachev V., Sheiko T., Shapiro V., Tsukanov I.: Transfinite Interpolation over Implicitly Defined Sets. *Computer-Aided Geometric Design 18*, 3 (2001), 195–220.

[SFP13] Sanchez M., Fryazinov O., Pasko A.: Efficient evaluation of continuous signed distance to a polygonal mesh. In *SCCG '12: Proceedings of the 28th Spring Conference on Computer Graphics* (New York, NY, USA, 2013), ACM, pp. 101–108.

[SOS04] Shen C., O'Brien J. F., Shewchuk J. R.: Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics 23* (Aug. 2004), 896–904.

[VBG*13] Vaillant R., Barthe L., Guennebaud G., Cani M.-P., Rohmer D., Wyvill B., Gourmel O., Paulin M.: Implicit skinning: Real-time skin deformation with contact modeling. *ACM Transactions on Graphics 32*, 4 (July 2013), 125:1–125:12.

[WK03] Wu J., Kobbelt L.: Piecewise linear approximation of signed distance fields. In *Proceedings of Vision, Modeling and Visualization 03* (2003), pp. 513–520.

[YT02] Yngve G., Turk G.: Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics 8* (October 2002), 346–359.

[ZO02] Zhao H., Osher S.: Visualization, analysis and shape reconstruction of unorganized data sets. In *Geometric Level Set Methods in Imaging, Vision and Graphics* (2002), S. Osher and N. Paragios (Eds.). Springer-Verlag, pp. 361–380.