Rounding, Filleting and Smoothing of Implicit Surfaces

Pierre-Alain Fayolle [0000-0003-4723-6208][1], Oleg Fryazinov [0000-0003-2263-7646][2] and Alexander Pasko [0000-0002-4785-7066][3]

[1]The University of Aizu, Japan, fayolle@u-aizu.ac.jp
[2]Bournemouth University, UK, ofryazinov@bournemouth.ac.uk
[3]Bournemouth University, UK, apasko@bournemouth.ac.uk

Corresponding author: Oleg Fryazinov, ofryazinov@bournemouth.ac.uk

ABSTRACT

We describe an approach for performing constant radius offsetting and the related operations of filleting, rounding and smoothing for implicit surfaces. The offsetting operation is used as the basic component for defining the remaining operations. These operations are important operations for any modelling system. While it is known how to perform these operations for parametric representation and polygon meshes, there is limited prior work for implicit surfaces and procedural volumetric objects. The proposed approach is based on repeatedly computing the distance to a given implicit surface and its offset surfaces. We illustrate the results obtained by this approach with several examples, including procedurally defined microstructures and CAD objects.

Keywords: Implicit surfaces, Function Representation, Distance function, Offsetting, Smoothing.

1 INTRODUCTION

Modern developments in geometric modelling allow for using a variety of geometry representations or combinations of them in a wide range of applications. As new representations are introduced, one wants to adapt existing processing techniques and methods used with the other representations.

Geometry representation with implicit surfaces is well-known in modelling as well as in computer graphics applications. The idea is to use an implicit form for the geometry representation, i.e., to use a function (or scalar field) or a predicate for point coordinates, which allows us distinguish points belonging to the interior of the object, to the exterior of the object or to its surface. In recent years, this representation is getting more and more attention because of many useful properties and applications. For example, with geometric objects defined in an implicit form, it is possible to model objects for engineering applications, organic objects, to easily perform animation and metamorphosis, to model procedurally-based multi-level parameterized microstructures, objects ready for digital fabrication and many more.

One of the important subset of continuous scalar fields is signed distance fields, where the value of the defining function gives the Euclidean distance to the surface of the implicitly defined object, i.e., the distance to the zero-level set of the field. One of the reasons signed distance fields are attracting

attention is that they have a number of applications in different areas from solid modelling to real-time computer animation. At the same time, this representation is quite restrictive comparing with arbitrary continuous scalar fields, with some common operations, such as non-uniform scaling, being not supported.

As modelling systems based on implicit surfaces evolve, a need appears to use techniques previously introduced in modelling systems based on parametric curves and surfaces and discrete geometry (e.g., polygonal meshes). One example of such techniques is filleting, which means rounding off a concave sharp edge or a corner of a mechanical part. Any mechanical CAD system needs to support such an operation as it is needed to simulate manufacturing with round-end mills. It is well known how to perform rounding and filleting for parametric representations, but there is limited prior work for implicit surfaces and procedural volumetric objects.

In this work, we discuss possible implementations of filleting, rounding and smoothing operations applied to objects defined in an implicit form by the zero level-set of a continuous scalar field. It is known that in the case of signed distance fields, these operations can be defined in a simple way. In practice, however, the distance property (the property of a scalar field to be a distance field) can be obtained analytically only for a very limited set of primitives and operations and is easily lost as we discussed before. Therefore, our goal here is to implement distance-based operations on general implicit surfaces without distance property.

We propose to use a numerical method to compute a signed distance field from an arbitrary continuous scalar field (arbitrary implicit surface). Offsetting the surface (the zero iso-level of the scalar field) is then performed by considering different iso-level values. By using repeated offsetting operations and re-distancing of the corresponding scalar field (re-computing the distance function from the scalar field), we show how to implement rounding, filleting and smoothing operations for general implicit surfaces. We demonstrate our approach with experimental results and several examples, including CAD models and procedural microstructures defined by real-valued scalar functions.

2 RELATED WORKS

The mathematical basis for offsetting of solids is described by Rossignac and Requicha in [22]. Offset operations can be considered a particular case of Minkowski sums. Given an offset operation, one can define rounding, filleting and smoothing by repeated applications of the offset operation. Minkowski sums and offsetting have been mostly studied for parametric curves and surfaces, polygon meshes and point-clouds, but is less common for general implicit curves and surfaces because of lack of distance property in the general case. We briefly review some of the existing works below.

A survey of the different methods available for the offset of curves and surfaces is provided by Pham in [21]. It was later revisited and extended by Maekawa in [10]. A description of the main tools and methods used for the offset of curves and surfaces is given by Patrikalakis and Maekawa in Chapter 11 of [19]. Offsetting operations for solid models (CSG and BRep) is discussed by Rossignac and Requicha in [22].

For the case of polyhedral models, an algorithm for computing Minkowski sums is described in [6]. It relies on computing a convex decomposition, which is not easy to do for complex objects. The work of Barki et al. [1] introduces a method to compute Minkowski sums of polyhedron with a convex polyhedron. An alternative approach by Campen and Kobbelt [4] presents an efficient computational approach for computing Minkowski sums with arbitrary polyhedral element. These approaches require a defect free polygon mesh model. For the particular case of the Minkowski sum with a sphere (a constant radius offset), recent works rely on computing a signed distance field to the input surface (the polygonal mesh), and then meshing the offset implicit surface. Pavic and Kobbelt [20] propose a volumetric method computing a distance field to the surface. The surface of the offset solid is obtained by a Marching Cubes like algorithm. In [9], grid cells in the neighborhood of the offset surface are identified by using an octree and some simple criteria. The signed distance at each grid node is efficiently computed and the offset surface is meshed by a variant of dual contouring. Chen and Wang propose a method [5], where each face, edge and vertex of a given polygon mesh are offset. The union of these point-sets (polygons) is computed. Then using the LDNI (Layered Depth Normal

Images) representation of these sets, unwanted components are filtered out. The final surface is finally obtained by meshing. A similar approach based on dexel (depth element) is described in [11]. All these approaches rely on the input surface to be defined as a polygonal mesh.

In [8], points sampled on a surface are offsetted in both directions and added to the input point-cloud; Radial Basis Functions are then fitted to this extended point-cloud. Calderon and Boubekeur use a Moving Least Square approach to fit a point-set surface to points projected on the Minkowski sum of an input point-cloud with some shape [3]. In [14], Molchanov et al. fit an approximation of the distance function to samples using a Moving Least Square approach.

For implicit surfaces, there are few methods available to compute an offset. Pasko et al. [18] compute the Minkowski sums with a numerical method involving global minimization. This approach is not very efficient. Additionally, it is unpractical when several offsets need to be applied successively, such as for example in filleting or smoothing. An alternative approach is described in [2], where the authors use the fast-marching method [24] to propagate the distance defined on a narrow band near the surface and compute a distance field. To compute the distance in a narrow band, one needs either to mesh the surface and compute the distance to the triangle mesh, or to compute the distance (or an approximation) by a numerical procedure.

3 DISTANCE FIELD COMPUTATION

The first task is to compute distance and distance offsetting for surfaces defined implicitly. Let the surface of a given object be defined as $\partial S = \{\mathbf{p} \in \Re^k : f(\mathbf{p}) = 0\}$ for some given function f and $k = 2$ (planar implicit curve) or $k = 3$ (an implicit surface in 3-dimensional space). The distance offsetting operation can be done by computing the signed distance function to ∂S and considering a different iso-level than the zero iso-level. Given the implicit definition of the surface, we try to perform such computations without meshing ∂S or sampling points from it. Operations such as rounding, filleting or smoothing are all defined in terms of this elementary offset operation.

In our method, we use normalization to create an initial approximation of the scalar field with distance property (near the surface), then use the re-initialization method to numerically compute the signed distance field.

3.1 Normalization

A function f is normalized to the order m if $\frac{\partial f}{\partial v} = 1$ and $\frac{\partial^k f}{\partial v^k} = 0$, for $k = 2...m$, near the surface ∂S. Here v is the unit normal to the surface. A normalized function behaves like the distance function near its zero level-set. Methods for normalizing functions were introduced by Rvachev and are discussed in detail by Shapiro [25]. Assuming that f has a non-vanishing gradient on its zero level-set, the first order normalization is defined as follows:

$$f_n(\mathbf{x}) = \frac{f(\mathbf{x})}{\sqrt{f^2(\mathbf{x}) + |\nabla f(\mathbf{x})|^2}} \tag{3.1}$$

Since R-functions for set-theoretic operations preserve normalization (see [25] for details), normalized functions can be constructed by applying set-theoretic operations with R-functions to normalized primitives. The Rvachev's normalization is related to the Taubin's distance approximation [27] defined as:

$$f_t(\mathbf{x}) = \frac{f(\mathbf{x})}{|\nabla f(\mathbf{x})|} \tag{3.2}$$

In practice, both approaches produce acceptable approximation of the exact distance only close to the surface, and may not work well if the gradient varies too quickly. We use the first order normalization Eqn. (3.1) to initialize the signed distance function computation.

3.2 Re-initialization

The distance to a surface implicitly defined as $\partial S = \{\mathbf{p} \in \Re^k : f(\mathbf{p}) = 0\}$, $k = 2$ or $k = 3$, can then be obtained by solving numerically the re-initialization equation proposed by Sussman et al. in [26] to steady state:

$$\frac{\partial \phi}{\partial t} = sign(f)(1 - |\nabla \phi|) \tag{3.3}$$

Where $\phi(\mathbf{x}, t = 0) = f(\mathbf{x})$ and $sign(f)$ is the sign function:

$$sign(f) = \begin{cases} -1, f < 0 \\ 0, f = 0 \\ 1, f > 0 \end{cases} \tag{3.4}$$

To increase the efficiency and the numerical stability, we use the following modifications. First, we use a slightly modified version of the sign function:

$$sign(f) = \frac{f}{\sqrt{f^2 + \varepsilon^2}} \tag{3.5}$$

Here ε is a reasonably small number. This modification allows us to avoid numerical issues around the zero-level set because of its continuity.

For the initial value condition of (3.3), we use the first-order normalization of f, as discussed above: $\phi(\mathbf{x}, t = 0) = \frac{f(\mathbf{x})}{\sqrt{f^2(\mathbf{x}) + |\nabla f(\mathbf{x})|^2}}$, which provides a better approximation of the distance, at least near the surface boundary.

Eqn. (3.3) is solved numerically on a regular grid. We use the forward Euler method for time integration, and the first order upwind method for computing the spatial derivatives. Thus, we need to compute a few iterations of:

$$\phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t.sign(f).(H(D_x^+ \phi_{ij}^n, D_x^- \phi_{ij}^n, D_y^+ \phi_{ij}^n, D_y^- \phi_{ij}^n) - 1) \tag{3.6}$$

Where ϕ_{ij}^n is the value at the n-th iteration of ϕ at the node (i, j) and H is the numerical Hamiltonian:

$$H(a,b,c,d) = \begin{cases} \sqrt{\max((a^-)^2, (b^+)^2) + \max((c^-)^2, (d^+)^2)}, sign(f) \geq 0 \\ \sqrt{\max((a^+)^2, (b^-)^2) + \max((c^+)^2, (d^-)^2)}, sign(f) < 0 \end{cases} \tag{3.7}$$

With $a^+ = \max(a, 0), a^- = \min(a, 0)$.

$D_x^{\pm} \phi_{ij}^n$ and $D_y^{\pm} \phi_{ij}^n$ are upwind finite difference approximations of the spatial derivatives given by:

$$D_x^+ \phi_{ij}^n = \frac{\phi_{i+1,j}^n - \phi_{ij}^n}{\Delta x} \tag{3.8}$$

$$D_x^- \phi_{ij}^n = \frac{\phi_{ij}^n - \phi_{i-1,j}^n}{\Delta x} \tag{3.9}$$

and similarly, for $D_y^{\pm}\phi_{ij}^n$. While the presentation above is for the case $k=2$, all these equations extend naturally to the 3D case. For further details on numerically solving Eqn. (3.3), please refer to [26].

Russo and Smereka remarked in [23] that this scheme violates the property that derivatives should be computed according to the direction of the characteristics in cells intersecting ∂S. They propose to use a fix in such cells: for grid edges intersecting ∂S, the adjacent node values are set to the distance (or an approximation of the distance) to ∂S. An approximation can be computed, for example, using the first order Rvachev normalization, which is an accurate approximation close to ∂S. We are using this sub cell fix in our implementation.

When using Forward Euler for time integration, the time step needs to be selected appropriately in order to not violate the Courant-Friedrichs-Lewy (CFL) condition. We use $\Delta t = c\min(\Delta x, \Delta y)$ and respectively $\Delta t = c\min(\Delta x, \Delta y, \Delta z)$ in 3D, where Δx, Δy, and Δz are the grid cell length along each direction and $c = 0.4$ in our experiments. For further details on the numerical solution of such type of problem, one can refer, for example, to Chapters 1 to 7 of [15].

Recent works, such as [13], extend this approach with higher order scheme for the spatial derivatives. In [12], the author compares different methods for time-integration: forward Euler, second-order Runge-Kutta and Gauss-Seidel iteration of the forward Euler method and concludes that the forward Euler method does not introduce numerical instability and that the Gauss-Seidel iteration produces equivalent results to the second order Runge-Kutta method but is slightly faster. Another approach for computing the distance is the Fast Marching Method of Sethian [24], which has the advantage of being non-iterative. In our experiments, we found that solving (3.3) using the combination of the forward Euler method with the first order upwind finite difference produced the best results and was the fastest for the operations described in this paper.

4 DISTANCE-BASED OPERATIONS ON ARBITRARY IMPLICIT SURFACES

Once we are able to approximate the distance field on the base of the given continuous scalar field within some proximity to the zero level-set, we are able to perform distance-based operations such as offsetting as well as operations, which use the local distance property of the scalar field. We discuss such operations below.

4.1 Offsetting

Offsetting is a very simple operation for continuous scalar fields possessing the distance property. For an object defined in an implicit form with the defining function ϕ, an offset can be defined as $\phi_r(\mathbf{x}) = \phi(\mathbf{x}) + r$, where r is an offset value. The offset value is signed and the sign defines the direction of the offset. In this work, we assume the following convention: $\phi > 0$ inside the domain bounded by ∂S and $\phi < 0$ outside. In this case, positive r defines objects expansion and negative r defines objects contraction.

Given an input function f, we compute the distance field ϕ by the normalization and the re-initialization methods as described in sections (3.1) and (3.2). Since we only need the distance field to be accurate up to a distance r to the surface, an upper bound for the number of iterations of the re-initialization method is given by $\lceil r/\Delta t \rceil$, where Δt is the time-step used in the numerical solution.

Indeed, since the front moves with unit speed, it propagates by Δt in one iteration. Thus, in order to have the front propagated up to a distance r, $\lceil r/\Delta t \rceil$ iterations are needed.

Applying level-set methods to compute offsets to a given implicit surface is certainly not novel. See, for example, the work [7], which computes offset curves by contouring the zero iso-level of the solution to the surface evolution equation at unit speed in the normal direction. However, our goal ultimately is to apply multiple offsetting (and thus distance re-computation), which is not directly possible with the precedent method.

One has to carefully select the offset distance *r*, since the zero level-sets of ϕ_r may not necessarily correspond to valid solids (regularized sets) for some values of *r*. In order to avoid any problem, we can restrict the offset radius such that no boundary points after the offset are on the medial axis of the original solid.

4.2 Rounding

Rounding a solid corresponds to smoothing all its convex sharp features (edges and corners) while keeping the rest of the solid's boundary unchanged. The rounding algorithm for an implicitly defined object f can be defined as follows:

1) compute the distance function ϕ to the given implicit surface
2) compute the offset to $\phi = 0$ by *r* in the negative direction: $\phi_{-r}(\mathbf{x}) = \phi(\mathbf{x}) - r$
3) compute the distance function ψ to the surface $\phi_{-r}(\mathbf{x}) = 0$
4) compute the offset to $\psi = 0$ by *r* in the positive direction: $\psi_r(\mathbf{x}) = \psi(\mathbf{x}) + r$

In steps 1 and 3, the distance is computed using the re-initialization method defined in section (3.2). The resulting rounded surface is then defined by the point set $\{\mathbf{x} : \psi_r(\mathbf{x}) = 0\}$

4.3 Filleting

Filleting is the opposite operation to rounding. It corresponds to smoothing all concave sharp features (edges and corners) while keeping the rest of the solid surface unchanged. Filleting is obtained by offsetting by *r* in the positive direction then offsetting the previous solid by *r* in the negative direction, i.e. the following operations have to be performed:

1) compute the distance function ϕ to the given implicit surface
2) compute the offset to $\phi = 0$ by *r* in the positive direction: $\phi_r(\mathbf{x}) = \phi(\mathbf{x}) + r$
3) compute the distance function ψ to the surface $\phi_r(\mathbf{x}) = 0$
4) compute the offset to $\psi = 0$ by *r* in the negative direction: $\psi_{-r}(\mathbf{x}) = \psi(\mathbf{x}) - r$

The resulting filleting surface is then defined by the point set $\{\mathbf{x} : \psi_{-r}(\mathbf{x}) = 0\}$.

4.4 Smoothing

Smoothing a solid requires smoothing each sharp feature (corner or edge). To implement this operation, we can combine rounding and filleting, where the order of these two operations is not important. Assuming that we apply rounding first, smoothing is obtained by the following operations:

1) compute the distance function ϕ to the given implicit surface
2) compute the offset to $\phi = 0$ by *r* in the negative direction: $\phi_{-r}(\mathbf{x}) = \phi(\mathbf{x}) - r$

3) compute the distance function ψ to the surface $\phi_{-r}(\mathbf{x}) = 0$

4) compute the offset to $\psi = 0$ by $2r$ in the positive direction: $\psi_{2r}(\mathbf{x}) = \psi(\mathbf{x}) + 2r$

5) compute the distance function γ to the surface $\psi_{2r}(\mathbf{x}) = 0$

6) compute the offset to $\gamma = 0$ by r in the negative direction: $\gamma_{-r}(\mathbf{x}) = \gamma(\mathbf{x}) - r$

The resulting smoothing surface is then defined by the point set $\{\mathbf{x} : \gamma_{-r}(\mathbf{x}) = 0\}$. Note that in the list of operations above we combined the two offsets in positive direction to make the whole evaluation more efficient. Thus, the smoothing operation requires the computation of the distance function three times.

5 APPLICATIONS AND RESULTS

In this section, we present some examples and potential applications for the approach described in this paper. We also give some experimental results related to the accuracy of the method and its computational efficiency.

5.1 Shelling and microstructures

The first example illustrates offsetting and shelling an implicit surface representing a jaw bone. The jaw bone was implicitly modeled by using convolution surfaces and does not have the distance property initially. See Fig. 1(a). for the bone surface (the meshed zero level-set) and Fig. 1(b). for the corresponding scalar field visualized on a slice.

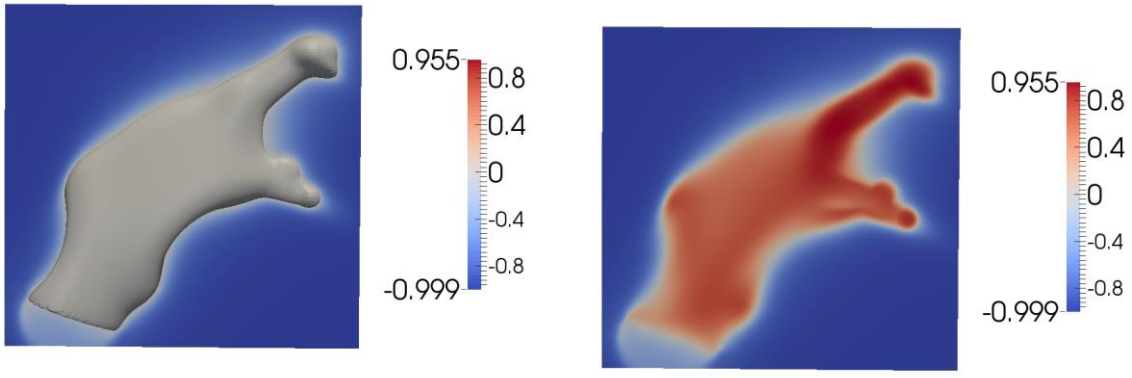

Fig. 1: An implicitly modeled jaw bone. (a) The meshed zero level-set modeled with convolution surfaces, (b) Visualization of the corresponding scalar field on a slice.

The model does not have a distance property (see, for example, Fig. 1(b).) and therefore we use the techniques from section 3.1 and 3.2 to compute it. See Fig. 2(a). for the bone surface (meshed from the zero level-set of the computed distance function), and Fig. 2(b). for the corresponding distance field visualized on a slice.

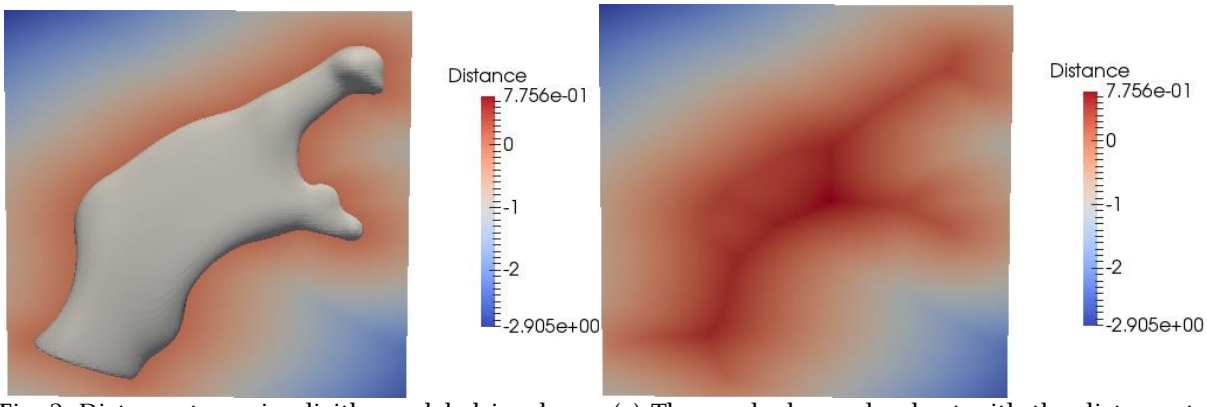

Fig. 2: Distance to an implicitly modeled jaw bone. (a) The meshed zero level-set with the distance to the surface on a slice, (b) Visualization of the distance field to the bone surface on a slice.

Given the distance function to the zero level-set of the defining function for the bone, an offset in the negative direction is computed and applied to obtain a shrunk version of the bone. The original bone is then carved by subtracting its shrunk version. The defining function for the shell is given by: $\phi - \phi_{-r}$, where ϕ is the distance to the bone surface, ϕ_{-r} is the offset and "$-$" is the subtraction implemented by the corresponding R-function [16], [25]. A radius of 0.1 is used for the offset. A slice of the bone shell is shown in Fig. 3(a). with the corresponding distance field shown in Fig. 3(b).

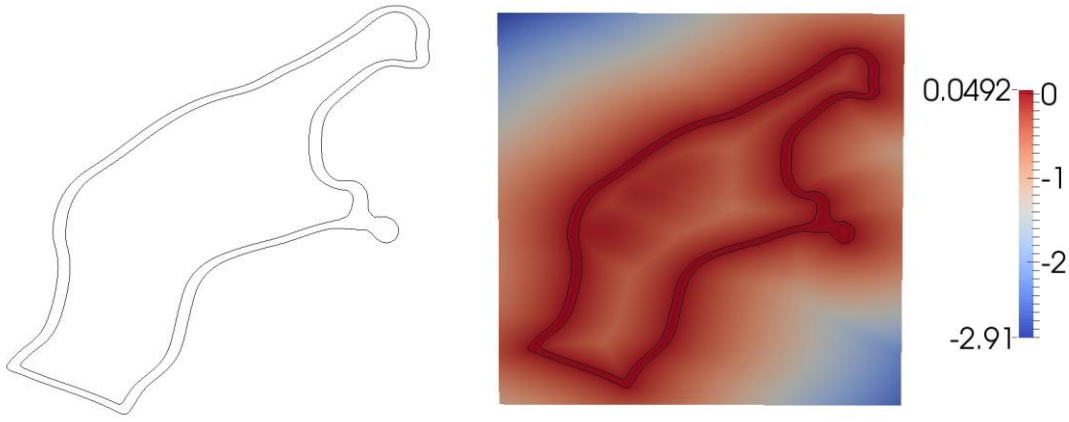

Fig. 3: Offsetting and shelling the jaw model: (a) Jaw shell obtained by carving a shrunk bone corresponding to an offset of 0.1, (b) The jaw shell and the corresponding field on a slice.

The distance property of the object allows us to perform further operations on the shelled object, for example, filling the interior of the object with procedural microstructures as discussed in [17]. For this example, we follow the approach described in section 3.1 of [17] for modelling lattice microstructures. An infinite lattice structure is obtained from the intersection of parallel slabs. A finite lattice bound within the jaw bone is then obtained by the intersection of the infinite lattice with the input jaw bone solid. By defining the frequency of the slabs as a function of the distance to the bone surface, we obtain a lattice scaffold, which is denser near the surface. The resulting shell with the internal lattice microstructures is illustrated in Fig. 4(b).

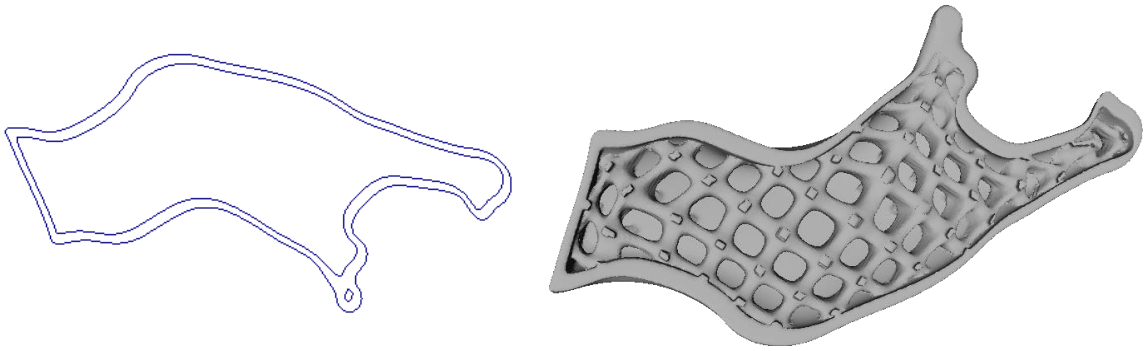

Fig. 4: Offsetting and filling with microstructures the jaw model: (a) Jaw shell obtained by carving a shrunk bone corresponding to an offset of 0.1, (b) The jaw shell with an internal microstructure.

5.2 Smoothing

The final two examples illustrate smoothing CAD objects with sharp features by multiple offsetting as described in section 4.4. The object in Fig. 5(a). is a simple union of two rectangular boxes, where the union is defined by R-functions and the boxes are implicitly defined. The smoothed shape is shown in Fig. 5(b). Note how all edges and corners are smoothed, while the rest of the surface is kept unchanged (see Fig. 5(c). for a closer view on the smoothed object). In this example, a radius of 0.025 is used.

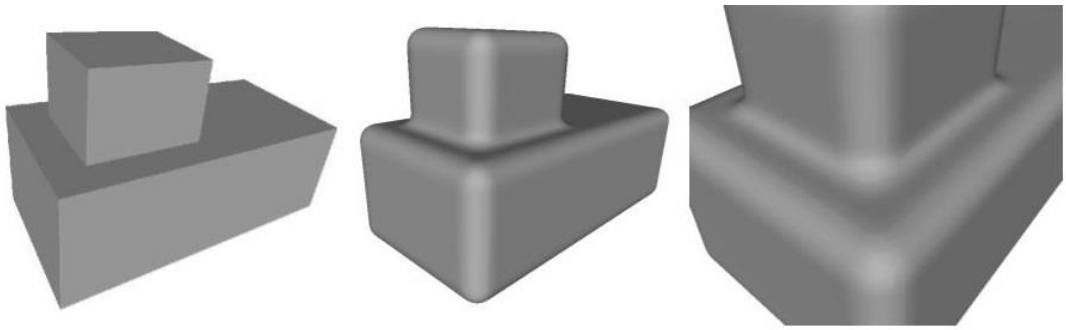

Fig. 5: A simple shape defined as the union of two rectangular boxes represented as implicit surfaces. (a) The input shape. (b) The smoothed shape obtained after multiple offsets (a radius of 0.025 is used). (c) A closer view on the smoothed shape.

The object in Fig. 6(a). is a slightly more complex model. The input model is implicitly defined and created by applying set-theoretic operations defined by R-functions to implicitly defined geometric primitives, such as cuboids or cylinders. The field for this model initially does not have a distance property. See Fig. 6(a). and (b). for a contour plot of the field on a given slice. Smoothing the object is obtained by repeated offsetting as described in section 4.4. A radius of 0.15 is used for the smoothing operation. The result of the smoothed surface is shown in Fig. 7(b)., (d). Compare with the input shape in Fig. 7(a). and a closer view in Fig. 7(c). Note how the original surface is kept while all sharp corners and edges (convex and concave) are rounded in Fig. 7(b). and (d).

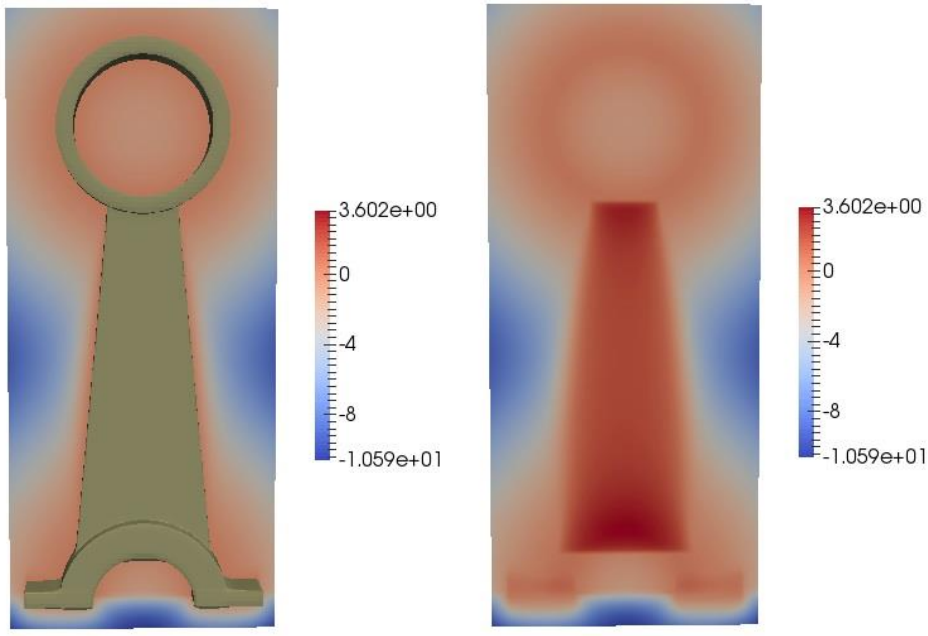

Fig. 6: A mechanical shape implicitly defined. (a) The original shape and its contour plot on a slice, (b) Its contour plot on a slice (interior and exterior).

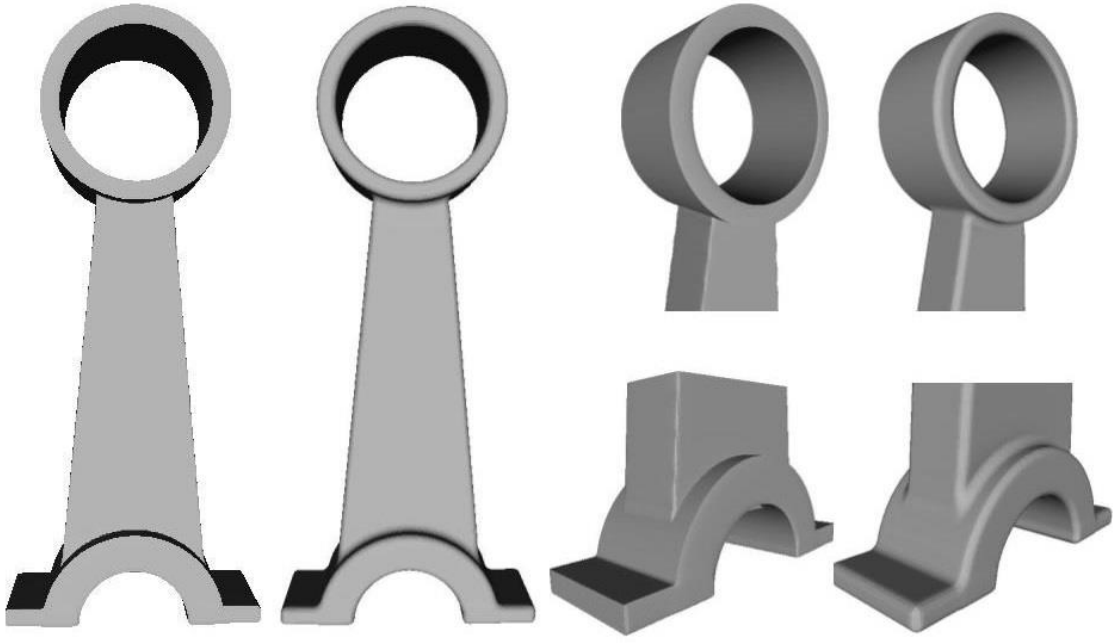

Fig. 7: A mechanical shape implicitly defined and its smoothed version. (a) The original shape, (b) The shape with sharp features smoothed using a radius of 0.15, (c) A zoom on the original shape, and (d) A zoom on the smoothed shape.

5.3 Approximation quality

In order to assess the error of the distance computation near the zero level-set, we sampled 10,000 points on the surface of a unit sphere, and evaluated the maximum and average deviation of the computed distance function at these samples. Initially, we start from the following defining function for the implicit unit sphere: $f(\mathbf{x}) = 1 - x^2 - y^2 - z^2$. The computed distance function is evaluated inside a grid cell by linear interpolation of the values at the grid nodes. Tab. 1. gives the maximum and average deviation for different grid resolutions.

	32^3	64^3	128^3
Maximum	0.004	0.001	0.0002
Average	0.003	0.0007	0.0002

Tab. 1: Maximum and average deviation of the computed distance near the surface.

To evaluate the quality of approximation, we compute the relative error: $\|d - \phi\|_2 / \|d\|_2$, where d is the exact distance to the unit sphere, ϕ is the computed distance and $\|.\|_2$ is the L^2 norm. Fig. 8(a). shows the evolution, as a function of the number of iterations, of the relative error over the domain $[-2,2]^3$ for a grid with 128 subdivisions along each dimension. Fig. 8(b). shows the relative error over the domain near the surface obtained by a +/- 0.1 offset from the surface of the sphere.

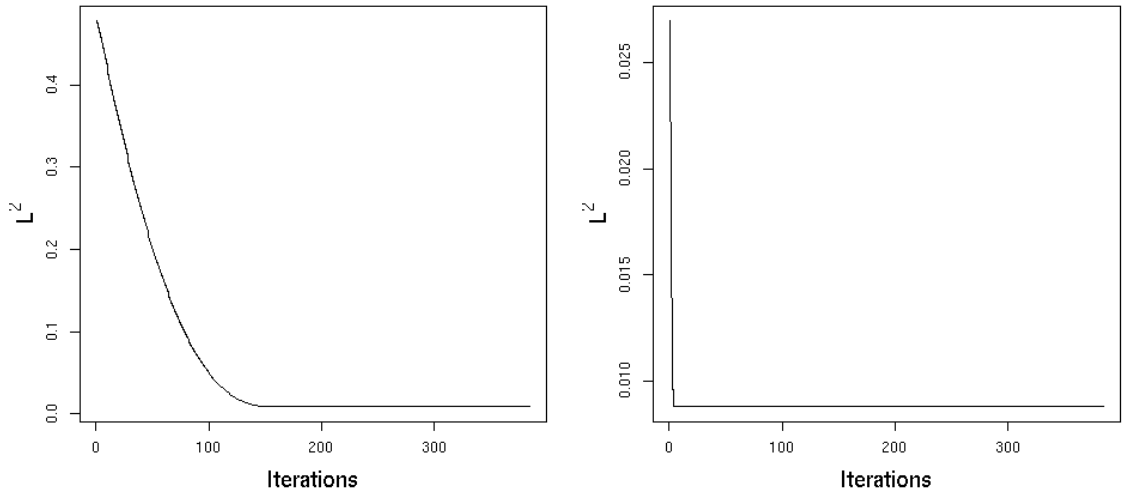

Fig. 6: The L^2 relative error for a unit sphere as a function of the number of iterations. (a) The error over the domain $[-2,2]^3$, (b) The relative error near the surface.

5.4 Computation time

In this section, we provide the computation time for the methods described in sections 3 and 4. The different methods have been implemented in C++ and run on a low-end desktop PC (3 GHz CPU and 4 GB of RAM). The code is not particularly optimized, and contains lots of room for improvement. Tab. 2. gives the time taken in seconds for a single iteration with different grid resolutions of the distance computation step (section 3.2). The computations are done for the different shapes used in this paper. Only one thread is used for these computations.

	32^3	64^3	128^3
Sphere	0.0017	0.014	0.11
Jaw bone	0.0018	0.014	0.11
Boxes union	0.0018	0.014	0.11
Mechanical part	0.0018	0.014	0.11

Tab. 2: Time (in seconds) for one iteration of the distance computation for the different shapes used in this work.

Tab. 3. summarizes the time taken for creating the examples shown in sections 5.1 and 5.2. All the computations are on a grid with a resolution of 128 by 128 by 128. The number of iterations of Eqn. (3.6) depends on the size of the grid cells and the radius for the offset. For all the examples presented here, 10 iterations were sufficient. This number of iterations was obtained from the expression $\lceil r/\Delta t \rceil$ (for an offset by r) and using the appropriate value for r and Δt for each example. The measured time corresponds to the time taken for initially sampling the implicit on the regular grid, computing the normalized value (section 3.1), applying one of the operations described in section 4.1 (offset) or 4.4 (smoothing), and applying additional operations (if any). It is easy to distribute the computation over multiple threads. The second column of Tab. 3. shows the time taken when multiple threads are used.

	1 thread	*4 threads*
Jaw microstructure	3.85	1.91
Boxes union	2.52	1.52
Mechanical part	2.95	1.67

Tab. 3: Time (in seconds) for computing the different examples shown in this paper.

The first example (the jaw bone with microstructures) only needs the computation of one offset, which requires only one distance re-initialization. Only the distance to the shape of the jaw bone surface is needed to compute the offset. In addition, the distance to the shape is also used to control the rod shapes. The microstructures are added at the end. In this example, the bottleneck is the sampling of the original field (representing the jaw bone surface) on the regular grid.

The last two examples (the union of boxes and the mechanical part) illustrate the smoothing operation, which requires computing three offsets. Each offset requires computing the distance to the updated zero level-set. The difference in time between these two examples is due to the sampling of the original field on the regular grid, the expression defining the mechanical part being more complicated than the other example.

6 CONCLUSIONS

Offsetting, filleting, rounding and smoothing are important operations in any CAD system, especially when dealing with mechanical parts. In this paper, we have proposed an approach for computing these operations on geometry defined implicitly by continuous scalar fields (or implicit surfaces). Filleting, rounding and smoothing are defined in terms of repeated offsets to implicit surfaces. The offset operation is based on computing the distance to a given implicit surface, which is done by solving the re-initialization equation given an initial approximation obtained with normalization.

The approach was shown to produce convincing results in a modelling framework dealing with implicit surfaces, yet it has to be implemented in a more complex modelling system in order to

evaluate its full efficiency. The main limitation of the discussed approach is that the computations are based on a numerical procedure, which requires sampling the function defining the model on a regular grid. Consequently, some approximation occurs. It is not clear, however, how to avoid such approximation if multiple offsets need to be applied, using an efficient computational procedure.

In our applications, we did not use many nested re-initialization operations, while during the modelling process a designer might want to use many nested offsets. Experimenting with that is one obvious future research direction.

The radius, used for the offset and the related operations, is constant. The operation (filleting, rounding, smoothing) is therefore global. A possible direction of future work is to investigate the possibility to have an operation defined only locally.

Another potential further extension of this work includes extension of the set of operations and primitives that require distance property of the defining function, for example, multi-scale modelling with fractal-based functions. As the distance property is very useful for fast direct rendering of implicit surfaces, the application of our method for these purposes is also an interesting topic to further research.

Pierre-Alain Fayolle, http://orcid.org/0000-0003-4723-6208
Oleg Fryazinov, http://orcid.org/0000-0003-2263-7646
Alexander Pasko, http://orcid.org/0000-0002-4785-7066

REFERENCES

[1] Barki, H.; Denis, F.; Dupont, F.: Contributing vertices-based Minkowski sum of a nonconvex – convex pair of polyhedral, ACM Transactions on Graphics, 30(1), 2011, 1-16. https://doi.org/10.1145/1899404.1899407

[2] Breen, D.; Mauch, S.; Whitaker, R.: 3D scan-conversion of CSG models into distance, closest-point and colour volumes, Volume Graphics, 2000, 135-158. https://doi.org/10.1007/978-1-4471-0737-8_8

[3] Calderon, S.; Boubekeur, T.: Point morphology. ACM Transactions on Graphics, 33(4), 2014, 1-13. https://doi.org/10.1145/2601097.2601130

[4] Campen, M.; Kobbelt, L.: Polygonal boundary evaluation of Minkowski sums and swept volumes, Computer Graphics Forum, 29(5), 2010, 1613-1622. https://doi.org/10.1111/j.1467-8659.2010.01770.x

[5] Chen, Y.; Wang, C. C.: Uniform offsetting of polygonal model based on layered depth-normal images, Computer-Aided Design, 43(1), 2011, 31-46. https://doi.org/10.1016/j.cad.2010.09.002

[6] Hachenberger, P.: Exact Minkowksi sums of polyhedra and exact and efficient decomposition of polyhedra into convex pieces, Algorithmica, 55(2), 2009, 329-345. https://doi.org/10.1007/s00453-008-9219-6

[7] Kimmel, R.; Bruckstein, A. M.: Shape offsets via level sets, Computer-Aided Design, 25(3), 1993, 154-162. https://doi.org/10.1016/0010-4485(93)90040-U

[8] Liu, S.; Wang, C. C.: Duplex fitting of zero-level and offset surfaces, Computer-Aided Design, 41(4), 2009, 268-281. https://doi.org/10.1016/j.cad.2008.10.008

[9] 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. https://doi.org/10.1109/TASE.2010.2066563

[10] Maekawa, T.: An overview of offset curves and surfaces, Computer-Aided Design, 31(3), 1999, 165-173. https://doi.org/10.1016/S0010-4485(99)00013-5

[11] Martinez, J; Hornus, S.; Claux, F.; Lefebvre, S.: Chained segment offsetting for ray-based solid representations, Computers & Graphics, 46, 2015, 36-47. https://doi.org/10.1016/j.cag.2014.09.017
[12] Min, C.: On reinitializing level set functions. Journal of Computational Physics, 229(8), 2010, 2764-2772. https://doi.org/10.1016/j.jcp.2009.12.032
[13] Min, C; Gibou, F.: A second order accurate level set method on non-graded adaptive Cartesian grids, Journal of Computational Physics, 225(1), 2007, 300-321. https://doi.org/10.1016/j.jcp.2006.11.034
[14] Molchanov, V.; Rosenthal, P.; Linsen, L.: Non-iterative second-order approximation of signed distance functions for any isosurface representation, Computer Graphics Forum, 29(3), 2010, 1211-1220. https://doi.org/10.1111/j.1467-8659.2009.01699.x
[15] Osher, S.; Fedkiw, R.: Level set methods and dynamic implicit surfaces, Springer, 2003, 1-273. https://doi.org/10.1007/b98879
[16] Pasko, A.; Adzhiev, V.; Savchenko, V.; Sourin, A.: Function representation in geometric modelling: concepts, implementation and applications, The Visual Computer, 11(8), 1995, 429-446. https://doi.org/10.1007/BF02464333
[17] Pasko A.; Fryazinov O.; Vilbrandt T.; Fayolle P.-A.; Adzhiev V.: Procedural function-based modelling of volumetric microstructures, Graphical Models, 73(5), 2011, 165-81. https://doi.org/10.1016/j.gmod.2011.03.001
[18] Pasko, A.; Okunev, O.; Savchenko, V.: Minkowski sums of point sets defined by inequalities, Computers & Mathematics with Applications, 45(10), 2003, 1479-1487. https://doi.org/10.1016/S0898-1221(03)00131-7
[19] Patrikalakis, N. M.; Maekawa, T.: Shape interrogation for computer aided design and manufacturing, Springer, 2002, 1-408. https://doi.org/10.1007/978-3-642-04074-0
[20] Pavic, D.; Kobbelt, L.: High-resolution volumetric computation of offset surfaces with feature preservation, Computer Graphics Forum, 27(2), 2008, 165-174. https://doi.org/10.1111/j.1467-8659.2008.01113.x
[21] Pham, B.: Offset curves and surfaces: a brief survey, Computer-Aided Design, 24(4), 1992, 223-229. https://doi.org/10.1016/0010-4485(92)90059-J
[22] Rossignac, J.; Requicha, A.: Offsetting operations in solid modelling, Computer-Aided Design, 3(2), 1986, 129-148. https://doi.org/10.1016/0167-8396(86)90017-8
[23] Russo, G.; Smereka, P.: A remark on computing distance functions, Journal of Computational Physics, 163(1), 2000, 51-67. https://doi.org/10.1006/jcph.2000.6553
[24] Sethian, J.: A fast marching level-set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences of the United States of America, 93(4), 1996, 1591-1595. https://doi.org/10.1073/pnas.93.4.1591
[25] Shapiro, V.: Semi-analytic geometry with R-functions, Acta Numerica, 16, 2007, 239-303. https://doi.org/10.1017/S096249290631001X
[26] Sussman M.; Smereka P.; Osher S.: A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, Journal of Computational Physics, 14, 1994. 146–159. https://doi.org/10.1006/jcph.1994.1155
[27] Taubin, G.: Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(11), 1991, 1115-1138. https://doi.org/10.1109/34.103273