Title:
Rounding, Filleting and Smoothing of Implicit Surfaces

Authors:
Pierre-Alain Fayolle, fayolle@u-aizu.ac.jp, The University of Aizu, Japan
Oleg Fryazinov, ofryazinov@bournemouth.ac.uk, Bournemouth University, UK
Alexander Pasko, apasko@bournemouth.ac.uk, Bournemouth University, UK

Keywords:
Implicit Surfaces, Function Representation, Distance Function, Offsetting, Filleting

DOI: 10.14733/cadconfP.2017.278-282

Introduction:
Modern developments in geometric modelling allow for using a variety of geometry representations 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. One example of such techniques is filleting. 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 zero level-sets of continuous scalar fields (usually called implicit surfaces). This representation is useful to define a wide range of primitives and operations, including complex geometry such as procedural microstructures [3]. It is possible to simply define these operations, if the scalar field corresponds to the distance to the surface of the object of interest. In practice, however, the distance property can be obtained analytically only for a very limited set of primitives and operations and is easily lost by some common operations in shape modelling, such as, for example, non-uniform scaling. In this work, we propose to use a numerical method to compute a signed distance field on the basis of an arbitrary continuous scalar field. Offsetting the surface (an iso-level set of the scalar field) is then performed by considering different iso-levels. 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 provide several examples, including CAD models and procedural microstructures, defined by a real-valued scalar function.

Main idea:
Rounding, filleting and smoothing operations have offsetting with a constant radius in their core. The mathematical basis for offsetting of solids is described by Rossignac and Requicha in [4]. Offset operations can be considered a particular case of Minkowski sums. Minkowski sums and offsetting have been mostly studied for polygonal meshes and point-clouds, but are less common for general implicit surfaces because of the lacking distance property in the general case.

Our goal is to compute constant radius offsets for surfaces defined implicitly as $\partial S = \{\mathbf{p} \in \Re^k : f(\mathbf{p}) = 0\}$ for some given function f and $k = 2$ or 3. This 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.

Normalization and re-initialization

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 $k=2...m$ where 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 are discussed in detail by Shapiro [5]. Here, to initialize the computation of the signed distance function, we use the first order normalization defined as follows: $f_1(\mathbf{x})=\frac{f(\mathbf{x})}{\sqrt{f^2(\mathbf{x})+|\nabla f(\mathbf{x})|^2}}$.

Distance to a surface implicitly defined as $\partial S=\{\mathbf{p}\in\Re^k : f(\mathbf{p})=0\}$ can then be obtained by solving numerically the re-initialization equation proposed by Sussman et al. [6]: $\frac{\partial\phi}{\partial t}=sign(f)(1-|\nabla\phi|)$, where $\phi(\mathbf{x},t=0)=f(\mathbf{x})$ and $sign(f)$ is a sign function. To increase stability of the numerical solution we use $sign(f)=\frac{f}{\sqrt{f^2+\varepsilon^2}}$ with reasonably small ε and, as discussed above, we use the first order normalization of f as the initial condition, i.e. $\phi(\mathbf{x},t=0)=\frac{f(\mathbf{x})}{\sqrt{f^2(\mathbf{x})+|\nabla f(\mathbf{x})|^2}}$.

The forward Euler method is used for the time integration, and a first order upwind method is used to compute the spatial derivatives, giving the following iterative scheme:

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

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:

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

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 $D_x^+\phi_{ij}^n=\frac{\phi_{i+1,j}^n-\phi_{ij}^n}{\Delta x}$, $D_x^-\phi_{ij}^n=\frac{\phi_{ij}^n-\phi_{i-1,j}^n}{\Delta x}$ and similarly for $D_y^{\pm}\phi_{ij}^n$. All these equations extend naturally to the 3D case.

Offsetting

For an object defined in an implicit form with the defining function ϕ, an offset in positive direction 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. To make the offset a constant radius offset (corners are replaced by

spherical patches and edges by cylindrical patches), we need the signed distance function, which is computed by the re-initialization method as described above. In practice, we only need the distance function to be accurate up to the distance r from the surface. This gives us an upper bound for the number of iterations of the re-initialization method.

Applying level-set methods to compute offsets to a given implicit surface is certainly not novel. See, for example, the work [2], 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.

As an example of this offsetting technique, we compute a shell from a jaw bone model, which was implicitly modeled by using convolution surfaces. The defining function for this model does not have a distance property and therefore we use our method to obtain it. See Fig. 1. for the bone surface, and the corresponding distance field visualized on a slice. The shell, obtained by subtracting an offset of the original bone to itself, is shown in Fig. 2(a).

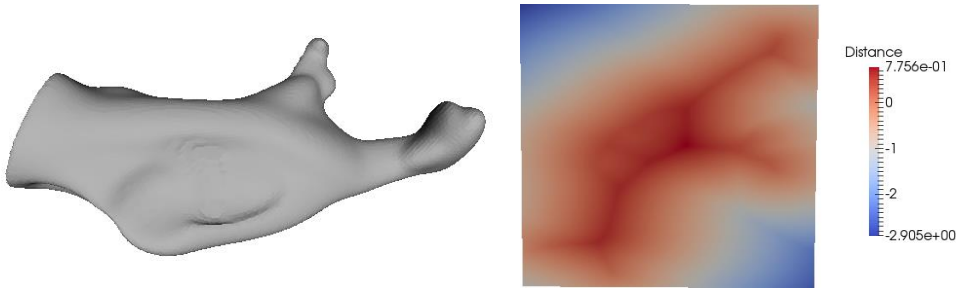

Fig. 1: An implicitly modeled jaw bone: (a) The meshed zero level-set, (b) Visualization of the distance field to the bone surface on a slice.

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 [3]. Here, the size of the rods is parameterized by the distance to the bone surface. The resulting shell with the internal microstructures is illustrated in Fig. 2(b).

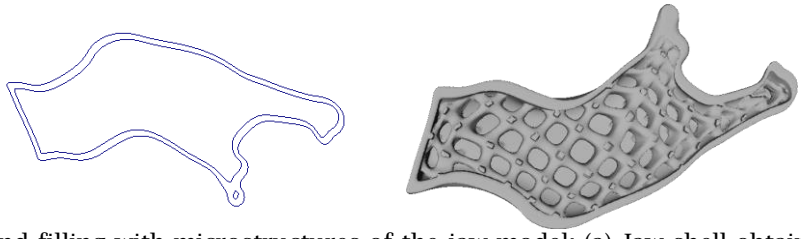

Fig. 2: Offsetting and filling with microstructures of the jaw model: (a) Jaw shell obtained by carving a shrunk bone corresponding to an offset of 0.1, (b) The jaw shell with an internal microstructure.

Rounding, filleting and smoothing

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 implicitly defined object can be defined as follows:

1) compute the distance function ϕ to the surface $f(\mathbf{x}) = 0$
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 by the re-initialization method defined above. The resulting rounded surface is defined by the point set $\{\mathbf{x} : \psi_r(\mathbf{x}) = 0\}$

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. Note that the value for the offset in filleting should reflect the geometry of the implicitly defined object, otherwise removing the material by negative offsetting can result in a non-valid solid object.

Smoothing a solid requires smoothing each sharp feature (corner or edge). It can be implemented as a combination of rounding and filleting, where the order of these two operations is not important. Assuming that we apply the rounding first, smoothing a solid is obtained by offsetting in the positive direction by r then in the negative direction by r (this corresponds to the rounding operation), then again offsetting in the negative direction by r and finally offsetting in the positive direction by r (this corresponds to the filleting operation). We can combine the two offsets in the negative direction by r into one offset in the negative direction by $2r$. Note that smoothing requires the computation of the distance function (at most) three times.

An alternative approach to the techniques above consists in replacing the first two steps (distance computation and value offset) by computing the minimum of the function f in the closed ball centered at $\mathbf{x}$ and of radius r. The constrained minimization can be done by the Alternating Direction of Method of Multipliers (ADMM) [1]. While the function f is in general not convex, the method seems to converge to the global minimum in practice.

The example in Fig. 3 illustrates the smoothing of a CAD object with sharp features as described in this paragraph. The model was created by using set-theoretic operations with R-functions [5] applied to implicitly defined geometric primitives such as cylinders and boxes. The field for the model does not have a distance property. The input object (with a zoom on its bottom part) is shown in the left images. The result of the smoothed implicit is shown in the right images. Note how the original surface is kept while all sharp corners and edges (convex and concave) are rounded.

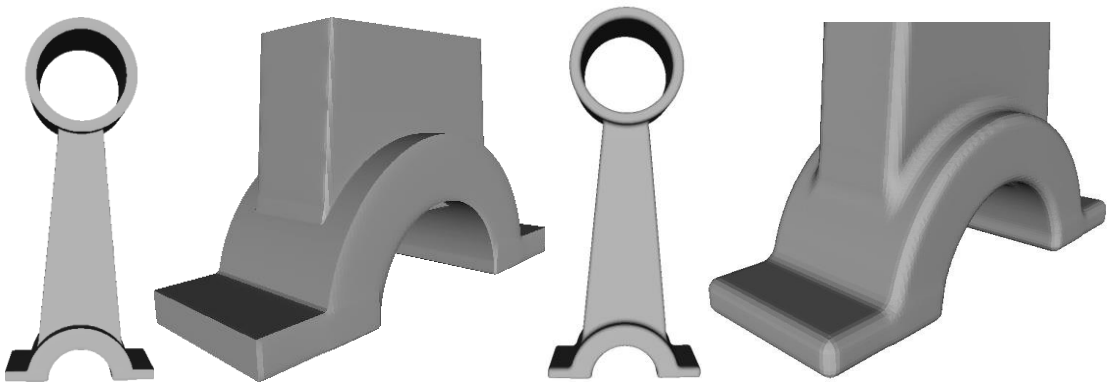

Fig. 3: A mechanical shape implicitly defined: (a) The original shape with a zoom of the bottom part, (b) The shape with sharp features smoothed using a radius of 0.15.

Computation Time

Tab. 1 summarizes the time taken for creating the examples shown in Fig. 2 and Fig. 3. The implementation is in C++ and was 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. All the computations were done on a grid with the resolution of 128×128×128. For all these examples, 10 iterations of (1) were sufficient. The measured time corresponds to the time taken for initially sampling the scalar field on the regular grid, computing the normalized value, applying one of the operations: offset or smoothing, and applying additional operations (if any). It is easy to distribute the computation over multiple threads. The second column of Tab. 1 shows the times taken when multiple threads are used.

	1 thread	*4 threads*
Jaw microstructure	3.85	1.91
Mechanical part	2.95	1.67

Tab. 1: 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 (for the distance to the shape of the jaw bone surface). 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 second example illustrates the smoothing operation, which requires computing three offsets. Each offset requires computing the distance to the updated zero level-set.

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 modeling framework dealing with implicit surfaces and can be extended with further set of operations and primitives that require distance property of the defining function.

References:

[1] Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J.: Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers, Foundations and Trends in Machine Learning, 3(1), 2011, 1-122. https://doi.org/10.1561/2200000016

[2] 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

[3] 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

[4] Rossignac, J.; Requicha, A.: Offsetting in solid modelling, Computer-Aided Design, 3(2), 1986, 129-148. https://doi.org/10.1016/0167-8396(86)90017-8

[5] Shapiro, V.: Semi-analytic geometry with R-functions, Acta Numerica, 16, 2007, 239-303. https://doi.org/10.1017/S096249290631001X

[6] 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