Technical Report 2009-001

Distance to set operations in constructive modeling of solids

Pierre-Alain Fayolle

August 19, 2009

Computer Graphics Laboratory
The University of Aizu
Tsuruga, Ikki-Machi, Aizu-Wakamatsu City
Fukushima, 965-8580 Japan

Technical Report 2009-001

Title:
Distance to set operations in constructive modeling of solids

Authors:
Pierre-Alain Fayolle

Key Words and Phrases:
implicit surfaces, distance function, constructive modeling, set operations

Abstract:
We propose in this paper methods to compute the signed distance to surface obtained by the intersection (respectively union, difference) of two solids (in two and three dimensions). These implementations can replace "min/max" or "R-functions" used to model set operations on implicit surfaces.

Report Date:
8/19/2009

Written Language:
English

Any Other Identifying Information of this Report:

Distribution Statement:
First Issue: 10 copies

Supplementary Notes:

Computer Graphics Laboratory
The University of Aizu
Aizu-Wakamatsu
Fukushima 965-8580
Japan

Distance to set operations in constructive modeling of solids

Pierre-Alain Fayolle

08/04/2009

Contents

Abstract

We propose in this paper methods to compute the signed distance to surface obtained by the intersection (respectively union, difference) of two solids (in two and three dimensions). These implementations can replace “min/max” or “R-functions” used to model set operations on implicit surfaces.

1 Introduction

In geometric modeling, a solid can be defined by the sign of a function: the set $\{\mathbf{p} : f(\mathbf{p}) \geq 0\}$ defines the interior and the boundary, and the set $\{\mathbf{p} : f(\mathbf{p}) < 0\}$ the exterior (see [4] on implicit surfaces, [1] on Function Representation and the references therein). The case where f is the signed Euclidean distance to the boundary of the solid $f = 0$ is of special interest. Distance based models are extremely useful in many applications such as: constant-radius offsetting and blending operations [16], surface metamorphosis and smoothing [13], object reconstruction from a set of cross-sections [10], rendering with sphere tracing [6], generation of skeletal shape representation [23], heterogeneous object modeling [3], and others. We propose in this paper methods for calculating the signed Euclidean distance from a point to the surface of a solid constructed by applying set-theoretic operations (union, intersection, difference) to primitives defined by distance functions. That is: if d_1 and d_2 represent the distance to the surface of two primitives S_1 and S_2, we propose algorithms to calculate the distance d to the union (respectively intersection, difference) of S_1 and S_2.

1.1 Related works

1.1.1 The distance function

Let $d(\mathbf{p})$, $\mathbf{p} \in R^3$ be the signed distance function to an oriented closed surface M. The function d is the vanishing viscosity solution of the Eikonal equation [22, 21, 19]:

$$\|\nabla d\|_2 = 1, d|_M = 0 \tag{1}$$

where $\|.\|_2$ is the Euclidean norm. Let $\mathbf{c}$ be the closest point of $\mathbf{p}$ in the surface M, the signed distance is then $\epsilon\|\mathbf{p} - c\|_2$, where $\epsilon = -1$ if $\mathbf{p}$ is outside M. If the surface is smooth, then $\mathbf{p} - \mathbf{c}$ is orthogonal to the surface. The signed Euclidean distance function is at least C^0, and may be not differentiable at some points.

Expressions for the distance function to most of the classic surfaces of a CSG system (sphere, cylinder, cone) are known analytically [6] and distance to general quadrics and ellipsoids can be computed by a numerical procedure [7].

In general, if the surface M is available as an oriented point-set or a mesh of polygons, it is possible to solve the Eikonal equation (1) on a finite grid. There exists various optimal numerical algorithms such as the fast marching method [19], the fast sweeping method [21, 22], or the characteristics / scan conversion algorithm [11]. Algorithms, that exploit the GPU, have also been designed in order to compute efficiently the Euclidean distance function [8, 20]. After a grid is obtained with the signed Euclidean distance to M

in each of its nodes, it is always possible to apply spline interpolation / approximation, to get an analytical expression [15].

1.1.2 Constructive geometry with real valued functions

In constructive geometry, complex solids are built by applying successively set-theoretic operations (union, intersection, difference) to primitives. When solids are described by real valued functions, like in implicit surfaces or F-Rep, expressions for set-theoretic operations have been proposed by Sabin [18], Ricci [14] and Rvachev [17]. Sabin [18] and Ricci [14], independently proposed the use of “min/max”. If d_1 and d_2 are the functions defining two solids, then the union is defined by the function $max(d_1, d_2$ and the intersection by $min(d_1, d_2)$ (the difference is obtained from the intersection by replacing d_2 by $-d_2$). Rvachev proposed the “R-functions” [17]:

$$\begin{aligned} d_1 \vee_\alpha d_2 &= \tfrac{1}{1+\alpha}(d_1 + d_2 + \sqrt{d_1^2 + d_2^2 - 2\alpha d_1 d_2}) \\ d_1 \wedge_\alpha d_2 &= \tfrac{1}{1+\alpha}(d_1 + d_2 - \sqrt{d_1^2 + d_2^2 - 2\alpha d_1 d_2}) \end{aligned} \tag{2}$$

Neither ”min/max” nor the R-functions keep the distance to the constructed solid. This is illustrated in Fig. 1, where the approximate distance to the surface made by the intersection of the halfspaces: $x \geq 0$ and $y \geq 0$ is computed with an ”R-function” on the left and ”min” on the right.

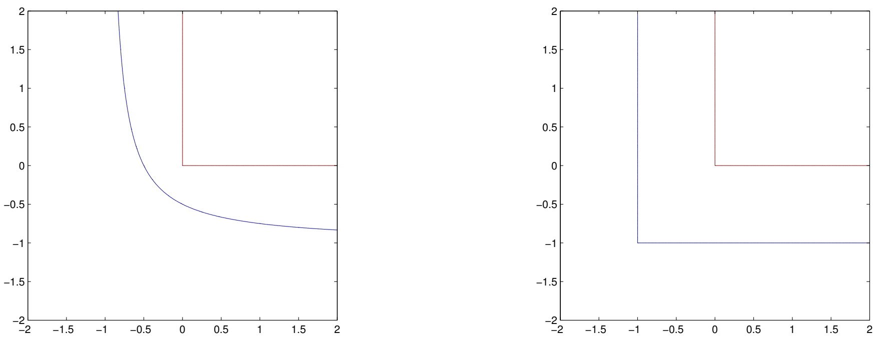

Figure 1: Approximate distance to the intersection of the halfspaces $x \geq 0$ and $y \geq 0$. Left: R-function is used to implement the intersection. Right: Min is used.

”Min/max” tend however to keep a better approximation of the distance than the ”R-functions”. This can be illustrated by computing the union of a disk with itself and looking at the value of the function (union of the disks) at the center. The distance to a circle of radius 1 and center $[0, 0]$ is: $d(\mathbf{p}) = 1.0 - \sqrt{\mathbf{p}^2}$, and the union of the disk with itself is defined by:

$max(d(\mathbf{p}), d(\mathbf{p}))$ or: $d(\mathbf{p}) \vee_0 d(\mathbf{p})$. In the former case, the distance at the center is: 1.0 while in the latter it is: 3.41421.

The functions “min/max” are not differentiable on the set of points corresponding to the equality of their arguments. Because of this, “R-functions” are sometimes preferred; ”R-functions” are not differentiable only on the set of points where their arguments are both equal to 0. This property of the “R-functions” is used for example when implementing a blending effect [12].

Some works tried to address this issue by modifying the contour lines of the functions $min(x, y)$ and $max(x, y)$: functions proposed in the work [9, 2] were designed for blending, while [5] were designed for keeping the distance approximation of ”min/max” while removing the points where the function is not C^1.

1.2 Overview and main contributions

The main contributions of this paper are methods to compute in two and three dimensions the distance to solids defined by the union (respectively intersection, difference) of solids defined by distance functions. That is: if S_1 and S_2 are solids defined by the distance functions d_1 and d_2, we describe methods to compute d the signed distance to $S = S_1 \cup S_2$ (respectively $S_1 \cap S_2$ and $S_1 \backslash S_2$). These expressions for the set-theoretic operations can be used instead of ”min/max” or the ”R-functions” in implicit surfaces or F-Rep modeling systems.

Union and intersection are dual, and the difference is obtained from the intersection, so we will limit the discussion to the construction of an expression for the intersection. We first describe the method in two dimensions, when d_1 and d_2 are functions in R^2. Then, we explain how to modify the method for the three dimensional case. Finally, we illustrate through examples in two and three dimensions the behavior of these functions and how they compare against ”min/max” or ”R-functions”.

2 Construction of the intersection in two dimensions

In this section, we describe how to get an expression for computing the intersection of two shapes defined by signed distance functions. We start by an example: the intersection of two orthogonal halfspaces $x \geq 0$ and $y \geq 0$ (as illustrated in Fig. 1), and explain how the scalar field constructed by $min(x, y)$ differs from the signed distance to the intersection. We generalize the method to the intersection of any objects defined by their signed distance fields: $d_1(x, y)$ and $d_2(x, y)$.

2.1 Intersection to two orthogonal halfspaces

Let us consider the example used for Fig. 1: two halfspaces are defined by $x \geq 0$ and $y \geq 0$. The set of points satisfying $min(x, y) \geq 0$ forms the intersection of the two halfspaces, and the points such that $min(x, y) = 0$ defines the boundary (which looks like an 'L'). Fig. 2 (left) illustrates some contour lines of the distance field to the previous boundary. The distance field is signed such that points inside the intersection of the halfspaces are positive and points outside are negative. For comparison, Fig. 2 (right) shows some contour lines of $min(x, y)$.

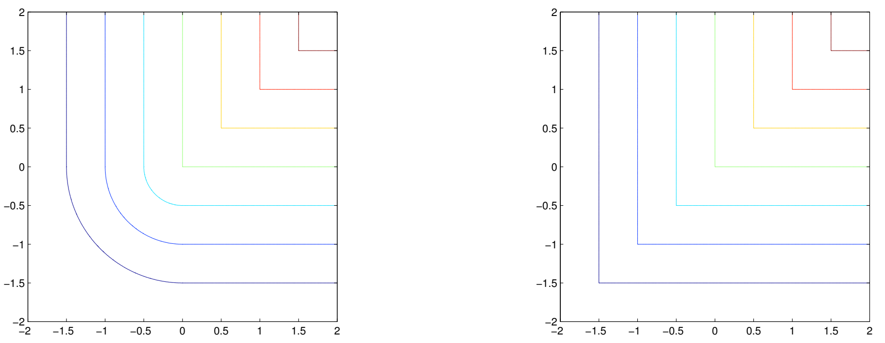

Figure 2: Left: contour lines for the exact distance to the boundary of the intersection of two halfspaces $x \geq 0$ and $y \geq 0$. Right: contour lines of the scalar field $min(x, y)$. Contour lines are from -1.5 to 1.5 with step of 0.5.

As it can be seen in Fig. 2, min gives the exact distance field everywhere except the space defined by: $x < 0$ and $y < 0$ (let us call that space Q). In that space, the closest point on the boundary is the point defined by the intersection of the two lines $x = 0$ and $y = 0$, and the contour lines are circular arcs centered at this intersection point.

We need now to define Q for any general shapes to be intersected and then explain how to compute the closest point on the surface in that space.

2.2 Determination of the space where *min* does not give the exact distance

Fig. 3 illustrates the case of the local intersection of two general (non orthogonal) curves intersecting at the point **O**. The boundary ($d = 0$), one contour line ($d = -1.0$) and the normals to each curve at O are plotted.

If the closest intersection point is known, then the region of interest can be identified using the normal to each curve at the intersection point as seen Fig. 3. However, it requires to first compute the set of all intersection points, which is not easy.

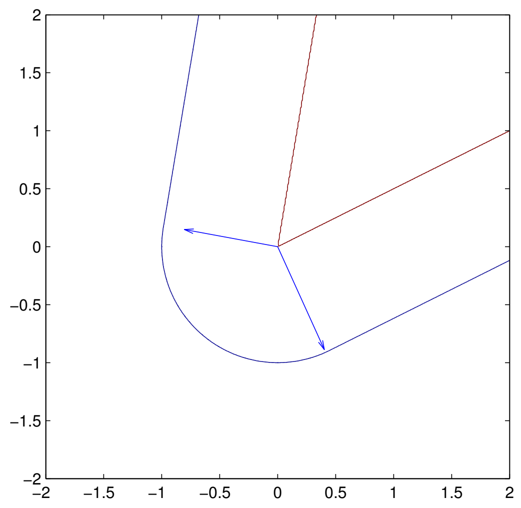

Figure 3: Local intersection of two general curves. Curve resulting of the intersection, one contour line at the distance −1.0 and the normals to each curve at the intersection point.

We will proceed in the reverse way by first determining the space of interest and then computing the closest intersection point in that space.

Given a point $\mathbf{p} \in R^2$ (see as an illustration Fig. 4), we first calculates the projection $\mathbf{p}_1$ of $\mathbf{p}$ on the first curve c_1. Using the signed distance field d_1 to the curve and its gradient, the projection is given by:

$$\mathbf{p_1} \leftarrow \mathbf{p} - d_1(\mathbf{p})\nabla d_1(\mathbf{p}) \tag{3}$$

Similarly, we calculate $\mathbf{p_2}$ the projection of $\mathbf{p}$ on the second curve c_2.

$\mathbf{p}$ belongs to the zone of interest Q, if $d_1(\mathbf{p_2}) < 0$ and $d_2(\mathbf{p_1}) < 0$.

2.3 Determination of the closest point

If the current point of evaluation $\mathbf{p_0}$ is in Q, then the next step is to compute its closest point $\mathbf{p}$ on the boundary of the intersection. The closest point is at the intersection of the two curves:

$$L(\mathbf{p}) = \begin{bmatrix} d_1(\mathbf{p}) = 0 \\ d_2(\mathbf{p}) = 0 \end{bmatrix} \tag{4}$$

This system is solved for $\mathbf{p} = (x, y)$ by using the damped Newton method with the initial guess $\mathbf{p_0}$. Eq. 5 is iterated until the residual $L(\mathbf{p_k})$ is small, and $\mathbf{p}$ is set to $\mathbf{p_k}$.

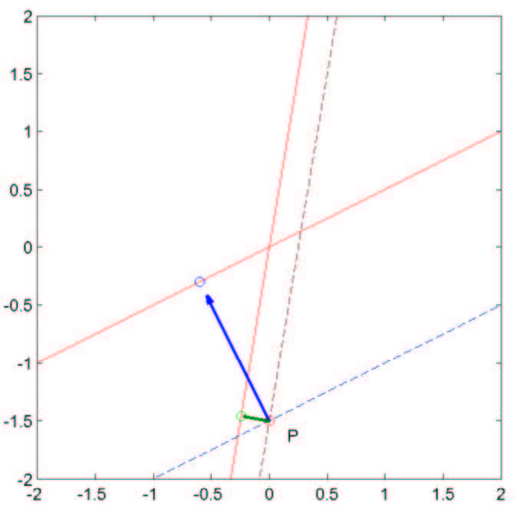

Figure 4: Two curves c_1 and c_2 (solid lines) defined by the distance fields d_1 and d_2. A given point $\mathbf{p}$ and the contour lines of d_1 and d_2 passing through $\mathbf{p}$. $\mathbf{p_1}$ and $\mathbf{p_2}$ the projections of $\mathbf{p}$ on c_1 and c_2.

$$\mathbf{p_{k+1}} = \mathbf{p_k} - \alpha J^{-1}(\mathbf{p_k})L(\mathbf{p_k}) \tag{5}$$

where α is the damping factor, and J is the Jacobian matrix of L.

$$J(\mathbf{p}) = \begin{pmatrix} \frac{\partial d_1}{\partial x}(\mathbf{p}) & \frac{\partial d_1}{\partial y}(\mathbf{p}) \\ \frac{\partial d_2}{\partial x}(\mathbf{p}) & \frac{\partial d_2}{\partial y}(\mathbf{p}) \end{pmatrix} \tag{6}$$

Because $\mathbf{p_0}$ is close to $\mathbf{p}$, $\mathbf{p_k}$ converges relatively fast.

2.4 Distance to the intersection

Putting together the results of the previous subsections, a procedure for calculating the distance to the intersection of two curves defined by the signed distance functions is:

distInter Given a point $\mathbf{p} \in R^2$, and two (2D) solids defined by the signed distance functions $d_1 \geq 0$ and $d_2 \geq 0$, compute the distance to the curve, boundary of the intersection of the two solids.

1. [distance to curve 1] $d_1 = d_1(\mathbf{p})$
2. [distance to curve 2] $d_2 = d_2(\mathbf{p})$
3. [normal to d_1] $\mathbf{n_1} = \nabla d_1(\mathbf{p})$
4. [projection on curve 1] $\mathbf{p_1} = \mathbf{p} - d_1\mathbf{n_1}$

```
5. [normal to $d_2$] $\mathbf{n_2} = \nabla d_2(\mathbf{p})$
6. [projection on curve 2] $\mathbf{p_2} = \mathbf{p} - d_2\mathbf{n_2}$
7. [in Q?] $inQ? = d_1(\mathbf{p_2}) < 0$ AND $d_2(\mathbf{p_1}) < 0$
8. if (inQ? is true) then:
   (a) [closest point] $\mathbf{p_c} = closestInter(\mathbf{p}, d_1(.), d_2(.))$ (see section 2.3)
   (b) [distance] $d = -\|\mathbf{p} - \mathbf{p_c}\|_2$
9. else:
   (a) [distance] $d = min(d_1, d_2)$
10. [return] return $d$
```

2.5 Extension to union and difference

The distance to the union is similar except for:

- the section 2.2 (step 7 of the algorithm distInter in section 2.4), where the condition $d_1(\mathbf{p_2}) < 0$ AND $d_2(\mathbf{p_1}) < 0$ should be replaced by the condition $d_1(\mathbf{p_2}) > 0$ AND $d_2(\mathbf{p_1}) > 0$,
- the step 8.b) of the algorithm distInter in section 2.4, which should be replaced by: $d = \|\mathbf{p} - \mathbf{p_c}\|_2$.

The distance to the set-theoretic difference is obtained by replacing the function $d_2(.)$ by the function $-d_2(.)$ in the algorithm distInter.

In the following section, we extend the method for working with three dimensional objects.

3 Construction of the intersection in three dimensions

The construction of the distance to the intersection in three dimensions is similar to the two dimensional case. The only difference is in determining the closest intersection point.

3.1 Determination of the closest point

In three dimensions, the system in eq. 4 has three unknowns (x, y and z) but only two equations. The third equation is obtained by observing that the closest intersection point is in the plane containing the gradient of each function at the current point of evaluation ($\mathbf{p_0}$). Let $\mathbf{n_1} = \nabla d_1(\mathbf{p_0})$ and $\mathbf{n_2} = \nabla d_2(\mathbf{p_0})$ be the gradients of d_1 and d_2 at $\mathbf{p_0}$. The third equation becomes: $\mathbf{n}(\mathbf{p} - \mathbf{p_0}) = 0$, where $\mathbf{n} = \mathbf{n_1} \wedge \mathbf{n_2}$. We have to solve the following system:

$$L(\mathbf{p}) = \begin{bmatrix} d_1(\mathbf{p}) = 0 \\ d_2(\mathbf{p}) = 0 \\ \mathbf{n}(\mathbf{p} - \mathbf{p_0}) = 0 \end{bmatrix} \quad (7)$$

This is solved with the damped Newton method as in section 2.3. The Jacobian of the system eq. 7 is:

$$J(\mathbf{p}) = \begin{pmatrix} \frac{\partial d_1}{\partial x}(\mathbf{p}) & \frac{\partial d_1}{\partial y}(\mathbf{p}) & \frac{\partial d_1}{\partial z}(\mathbf{p}) \\ \frac{\partial d_2}{\partial x}(\mathbf{p}) & \frac{\partial d_2}{\partial y}(\mathbf{p}) & \frac{\partial d_2}{\partial z}(\mathbf{p}) \\ n_x & n_y & n_z \end{pmatrix} \tag{8}$$

where n_x, n_y and n_z are the components of n.

3.2 Distance to the intersection

Putting together the results of the previous subsections, a procedure for calculating the distance to the intersection of two surfaces defined by signed distance functions is:

distInter3D Given a point $\mathbf{p} \in R^3$, and two (3D) solids defined by the signed distance functions $d_1 \geq 0$ and $d_2 \geq 0$, compute the distance to the surface, boundary of the intersection of the two solids.

```
1. [distance to surface 1] $d_1 = d_1(\mathbf{p})$
2. [distance to surface 2] $d_2 = d_2(\mathbf{p})$
3. [normal to $d_1$] $\mathbf{n_1} = \nabla d_1(\mathbf{p})$
4. [projection on surface 1] $\mathbf{p_1} = \mathbf{p} - d_1\mathbf{n_1}$
5. [normal to $d_2$] $\mathbf{n_2} = \nabla d_2(\mathbf{p})$
6. [projection on surface 2] $\mathbf{p_2} = \mathbf{p} - d_2\mathbf{n_2}$
7. [in Q?] $inQ? = d_1(\mathbf{p_2}) < 0$ AND $d_2(\mathbf{p_1}) < 0$
8. if (inQ? is true) then:
    (a) [closest point] $\mathbf{p_c} = closestInter3D(\mathbf{p}, d_1(.), d_2(.))$ (see section 3.1)
    (b) [distance] $d = -\|\mathbf{p} - \mathbf{p_c}\|_2$
9. else:
    (a) [distance] $d = min(d_1, d_2)$
10. [return] return $d$
```

4 Results and discussions

4.1 Visualization of distance fields

The distance field to the intersection and union of two solids is illustrated in two and three dimensions in the following examples.

4.1.1 Contour field in two dimensions

Fig. 5 illustrates the contour field of the signed distance functions resulting in the intersection of two non-orthogonal planes and two disks. Intersection was implemented using: "R-functions" (left), "min/max" (middle) and the method proposed in this paper. The functions built using "R-functions" do not respect the Euclidean metric. The functions built using "min/max" failed to do so only in the space identified in section 2.2. Compare the results with the method proposed here and the circular arcs in the contour lines outside of the constructed solids.

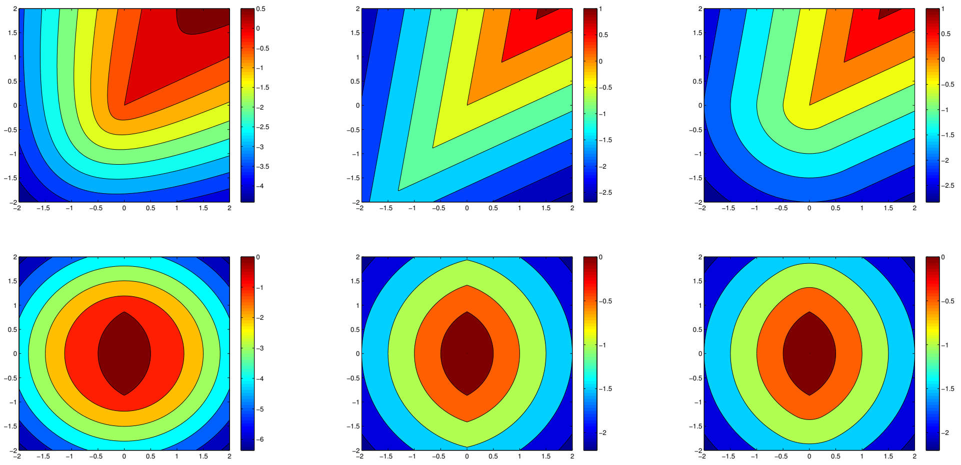

Figure 5: First row: Intersection of two halfplanes using for the intersection (from left to right): "R-functions", "min", the method proposed in this work. Second row: Intersection of two disks.

Fig. 6 illustrates the contour field of the distance to the union of two halfplanes (left) and two disks (right) using the method introduced here.

4.1.2 Contour field on planar section in three dimensions

Visualizing scalar fields in three dimensions is more difficult. We visualize instead the fields in a planar section. Fig. 7 (right) illustrates the proposed method applied in three dimensions to compute the distance to the intersection of two spheres. This should be compared with the sections in fig. 7 left and middle obtained with respectively "R-functions" and "min". Additionally, fig. 8 illustrates the result of the proposed method to compute the distance to the union of two spheres by showing the distance field on a section by the plane $y = 0$.

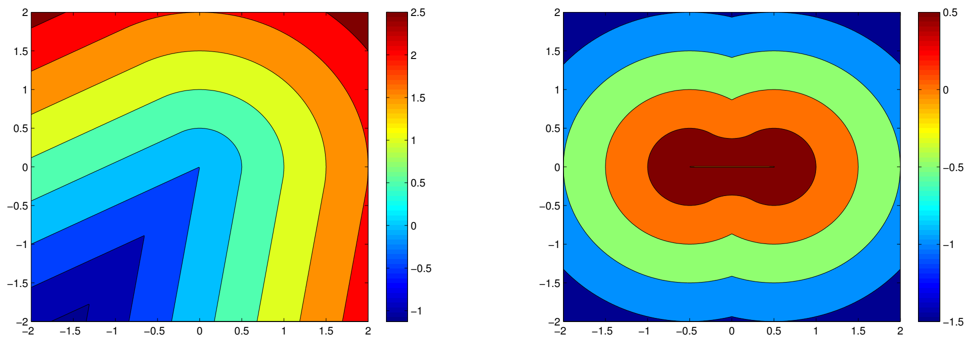
Figure 6: Left: distance to the union of two halfplanes with our method. Right: distance to the union of two disks.

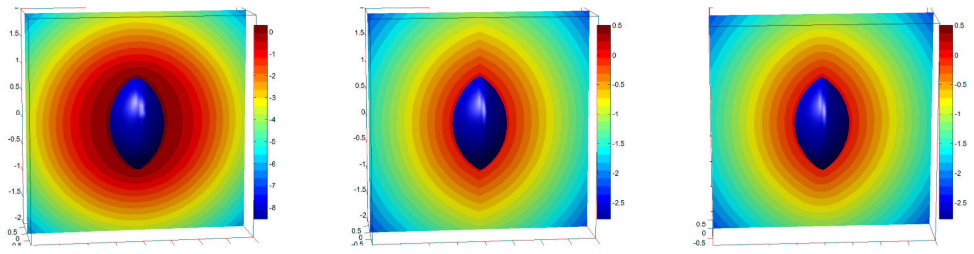
Figure 7: Distance to the intersection of two spheres using for the intersection (from left to right): “R-functions”, “min”, the method proposed in this work. Second row: Intersection of two disks.

4.1.3 Discussion

As mentioned earlier, the distance function is at least C^0 but not everywhere differentiable; for example, the sphere is not differentiable at its center. Our algorithms, as described above, make use of the gradient of the functions passed as input. The question is what value of the gradient to use in these cases. The points where the distance function is not differentiable have more than one closest point on the surface of the solid, and therefore more than one gradient at that point. The easiest solution is to consistently pick one gradient from the set of possible gradients. This is the solution that was used for the intersection / union of the disks and spheres in the examples above.

An alternative solution is to use slightly modified versions of “min” and “max” that are C^1. In the case of “min”, it is done by rewriting it as: $min(d_1, d_2) = \frac{1}{2}(d_1 + d_2 + \frac{(d_1-d_2)^2}{\sqrt{(d_1-d_2)^2}})$ and adding a small perturbation ϵ:

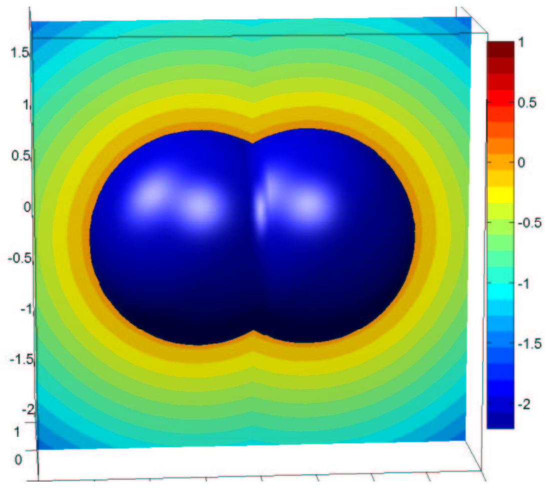

Figure 8: Distance to the union of two spheres with our method.

$\tilde{min}(d_1, d_2) = \frac{1}{2}(d_1 + d_2 + \frac{(d_1-d_2)^2}{\sqrt{(d_1-d_2)^2+\epsilon}})$. This function has the advantage of being C^1 but it is also slightly displacing each iso-contour; for example, if $d_1 = 0$ and $d_2 > 0$, then $\tilde{min}(d_1, d_2) = \frac{1}{2}(d_2 + \frac{(-d_2)^2}{\sqrt{(-d_2)^2+\epsilon}}) < 0$ instead of $min(d_1, d_2) = 0$.

Successive applications of set operations It is possible to successively apply the proposed implementation of intersection, union or difference to solids built with these operations. The gradient of the arguments is needed, which means the gradient of the function $\mathbf{p} \rightarrow intersection(d_1(\mathbf{p}), d2(\mathbf{p}))$ (respectively union, difference) needs to be calculated. The expression of the intersection is not differentiable for the point-set: $\{\mathbf{p} : \mathbf{p} \notin Q \wedge d_1(\mathbf{p}) = d_2(\mathbf{p})\}$. This set corresponds to the points with more than one gradient. As discussed previously, it is possible to select consistently one of them and use it as the gradient at that point for the purpose of our algorithm.

The expressions of the set operations proposed here are slower than "min/max" or "R-functions". The bottleneck is the loop computing the closest intersection point in Q and the distance to it (step 8 in distInter and distInter3D). All the functions have been implemented in Matlab®. With this implementation, evaluation of the (2D) intersection on a 400×400 grid takes 0.4 seconds against 0.03 for the "r-function". For a $40 \times 40 \times 40$ grid, the (3D) intersection takes the same time (0.4 seconds) against 0.015 for the "r-function".

5 Conclusion

We have presented in this work functions in two and three dimensions that implement the distance to set operations (intersection, union or difference). Contrary to the existing implementations ("min/max", "R-functions"), the proposed implementations correspond to the exact distance function to the resulting shape. The use of these functions should allow to implement easily rolling blend.

References

[1] Pasko A., Adzhiev V., Sourin A., and Savchenko V. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 8(11):429–446, 1995.

[2] L. Barthe, N. A. Dodgson, M. A. Sabin, B. Wyvill, and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum*, 22(1):23–33, 2003.

[3] Arpan Biswas, Vadim Shapiro, and Igor Tsukanov. Heterogeneous material modeling with distance fields. *Comput. Aided Geom. Des.*, 21(3):215–242, 2004.

[4] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces.* Morgan-Kaufmann, 1997.

[5] P.-A. Fayolle, A. Pasko, B. Schmitt, and N. Mirenkov. Constructive heterogeneous object modeling using signed approximate real distance functions. *Journal of Computing and Information Science in Engineering*, 6(3):221–229, 2006.

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

[7] John C. Hart. Distance to an ellipsoid. In Paul Heckbert, editor, *Graphics Gems IV*, pages 113–119. Academic Press, Boston, 1994.

[8] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of ACM SIGGRAPH*, pages 277–286. ACM, 1999.

[9] P.-C. Hsu and C. Lee. The scale method for blending operations in functionally-based constructive geometry. *Computer Graphics Forum*, 22(2):143–158, 2003.

[10] M. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):75–84, 1994.

[11] S. Mauch. *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology, 2003.

[12] A. Pasko and V. Savchenko. Blending operations for the functionally based constructive geometry. In *set-theoretic Solid Modeling: Techniques and Applications, CSG 94 Conference Proceedings*, pages 151–161. Information Geometers, 1994.

[13] B. Payne and A. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, 1992.

[14] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[15] C. Roessl, F. Zeilfelder, G. Nurnberger, and H.-P. Seidel. Spline approximation of general volumetric data. ACM Solid Modeling 2004, 2004.

[16] J. Rossignac and A. Requicha. Constant-radius blending in solid modeling. *Computers in Mechanical Engineering*, 3(1):65–73, 1984.

[17] V. Rvachev. *Theory of R-functions and Some Applications*. Naukova Dumka, Kiev, 1982. In Russian.

[18] M. Sabin. The use of potential surfaces for numerical geometry. Technical Report VTO/MS/153, 1968.

[19] J. Sethian. *Level-Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[20] A. Sud, A. Otaduy, and D. Manocha. Difi: Fast 3d distance field computation using graphics hardware. *Computer Graphics Forum*, 23(3):557–566, 2004. Proceedings of Eurographics 2004.

[21] Y.R. Tsai. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys.*, 178(1):175–195, 2002.

[22] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 2004.

[23] Y. Zhou, A. Kaufman, and A. Toga. 3d skeleton and centerline generation based on an approximate minimum distance field. *The Visual Computer*, 14(7):303–314, 1998.