Distance to Objects Built with Set Operations in Constructive Solid Modeling

Pierre-Alain Fayolle The University of Aizu
Fukushima, Japan
fayolle@u-aizu.ac.jp

Alexander Pasko Bournemouth University
Bournemouth, UK
apasko@bournemouth.ac.uk

ABSTRACT

We present in this paper methods to compute the signed Euclidean distance to surfaces obtained by the intersection (respectively union or difference) of two solids (in two or three dimensions). These implementations can replace min/max or R-functions traditionally used to model set operations used with implicit surfaces.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling – Boundary representations, Constructive Solid Geometry (CSG), Curve, surface, solid, and object representations

General Terms

Algorithms, Experimentation

Keywords

Shape and solid modeling, constructive modeling

1. INTRODUCTION

In shape modeling, a solid can be defined by the sign of a continuous 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 (the so called implicit surfaces or F-Rep, see [3], [11] and the references therein). The particular case where f is the signed Euclidean distance to the boundary of the solid 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 [9], rendering with sphere tracing [5], generation of skeletal shape representation [24], heterogeneous object modeling [2] and others. We present 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 distances to the surfaces of two primitives S_1 and S_2, we present an algorithm for calculating the distance d to the surface of the union (respectively intersection, difference) of S_1 and S_2.

1.1 Related works

1.1.1 *The distance function*

Let $d(\mathbf{p})$, $\mathbf{p} \in R^2$ or 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 [23, 21, 19]:

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

where $\|.\|_2$ is the Euclidean norm. Let $\mathbf{c}$ be the closest point to $\mathbf{p}$ in the surface M, the signed distance is then $\epsilon\|\mathbf{p} - \mathbf{c}\|_2$, where $\epsilon = -1$ if $\mathbf{p}$ is outside the solid delimited by the surface M, 1 otherwise. If the surface is smooth, then $\mathbf{p} - \mathbf{c}$ is orthogonal to the surface. The signed Euclidean distance function is continuous but not everywhere differentiable.

Expressions for the distance function to most of the typical surfaces of a CSG system (sphere, cylinder, cone) are known analytically [5] and distance to general quadrics and ellipsoids can be computed by a numerical procedure [6].

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 (Eq. (1)) on a finite grid. There exist various numerical algorithms such as the fast marching method [19], the fast sweeping method [21, 23], or the characteristics / scan conversion algorithm [10]. Algorithms exploiting the GPU have also been designed in order to compute efficiently the Euclidean distance function on a grid [7, 20]. Once a grid is obtained, with the signed Euclidean distance to M sampled at each grid node, it is possible to apply spline interpolation or approximation to get a function approximating the distance [15].

1.1.2 *Constructive geometry with real valued functions*

In constructive geometry, complex solids are built by applying successively set operations (union, intersection, difference) to primitives. For solids described using real valued functions, like in implicit surfaces or F-Rep modeling systems, expressions for these operations have been proposed by Sabin [18], Ricci [14] and Rvachev [17].

Sabin [18] and Ricci [14], independently proposed the use of the functions min and max to implement the intersection and union of implicit surfaces. If d_1 and d_2 are the functions

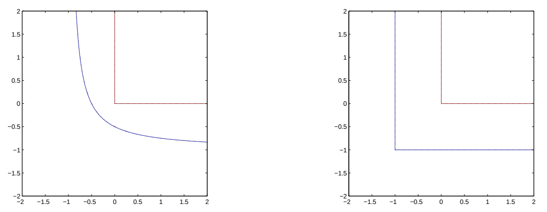

Figure 1: Approximate distance to the intersection of the halfspaces $x \geq 0$ and $y \geq 0$. The red curve corresponds to the boundary of the halfspaces intersection and the blue curve corresponds to the set of points at a value -1 outside of the intersection. Left: R-function is used to implement the intersection. Right: min is used.

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 operation is obtained by $min(d_1, -d_2)$, i.e. by replacing d_2 by $-d_2$). Rvachev proposed the R-functions [17]:

$$d_1 \vee_\alpha d_2 = \frac{1}{1+\alpha}(d_1 + d_2 + \sqrt{d_1^2 + d_2^2 - 2\alpha d_1 d_2}) \quad (2)$$

$$d_1 \wedge_\alpha d_2 = \frac{1}{1+\alpha}(d_1 + d_2 - \sqrt{d_1^2 + d_2^2 - 2\alpha d_1 d_2}) \quad (3)$$

$$d_1 \setminus_\alpha d_2 = d_1 \wedge_\alpha (-d_2) \quad (4)$$

Neither the scalar field obtained by using min/max nor the R-functions correspond to the Euclidean distance to the surface of the constructed solid. This is illustrated in Fig. 1, where the approximate distance field to the surface made by the intersection of the halfspaces: $x \geq 0$ and $y \geq 0$ is computed using an R-function (left) and min (right). The red curve corresponds to the solid boundary (i.e. corresponds to the distance 0) and the blue curve to the set of points at a value of -1 outside of the intersection.

Scalar fields constructed using the min/max functions keeps a better approximation of the distance field than when the R-functions are used. This can be illustrated by computing the union of a disk with itself and inspecting the value of the function (representing the distance to the union of the disks) at the center of the disk. 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: $max(d(\mathbf{p}), d(\mathbf{p}))$ (when using min/max) or: $d(\mathbf{p}) \vee_0 d(\mathbf{p})$ (when using the R-functions). 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 (i.e. for all points $\mathbf{p}$ where $d1(\mathbf{p}) = d2(\mathbf{p})$). Because of this, R-functions are sometimes preferred in modeling systems; R-functions are not differentiable only on the set of points where their arguments are both equal to 0. Smoothness of the R-functions is used in defining some solid modeling operations such as for example when implementing blending between shapes (see for example the description proposed in [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 works [8, 1] were designed for defining blending operations, while functions proposed in [4] 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 Euclidean distance to the surface of 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 for computing d the signed distance to the boundary of $S = S_1 \cup S_2$ (respectively $S_1 \cap S_2$ and $S_1 \setminus S_2$). These expressions for the set operations can be used instead of min/max or the R-functions in implicit surfaces or F-Rep modeling systems.

Computation for the union and intersection operations are symmetric and the difference operation can be obtained from the expression of the intersection, so we limit the discussion to the construction of an expression for the intersection. First, we investigate in section 2 the case of the intersection of two orthogonal halfspaces. Then in section 3, we describe how to compute the distance to the intersection of two general solids. In section 4, modifications for obtaining the distance for the union and difference operations are given. Finally, two and three dimensional examples are used to illustrate the behavior of these functions and to compare them with min/max and R-functions.

2. DISTANCE TO THE INTERSECTION OF TWO ORTHOGONAL HALFSPACES

Before describing the computation of the distance to the intersection of two solids defined by arbitrary functions, we start by a simple example: the distance to the intersection of two orthogonal halfspaces given by: $x \geq 0$ and $y \geq 0$ (as illustrated in Fig. 1 and 2). In particular, we explain how the scalar field constructed by $min(x, y)$ differs from the exact distance to the intersection.

2.1 Intersection of two orthogonal halfspaces

Let us consider the example used in Fig. 1: two halfspaces are defined by $x \geq 0$ and $y \geq 0$. Their intersection is the domain given by $\{(x, y) : x \geq 0 \,\&\, y \geq 0\}$.

Figure 2 (left) illustrates some contour lines of the distance field to the boundary of the domain previously defined. The distance field is signed such that points inside the intersection of the halfspaces are positive and points outside are negative. Fig. 2 (right) shows some contour lines of the scalar field defined by $min(x, y)$.

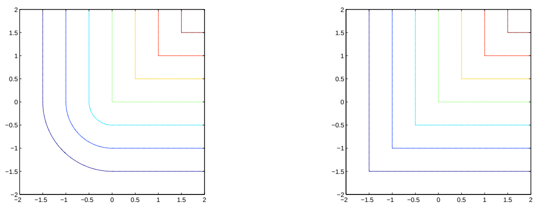

Figure 2: Left: contour lines of the Euclidean distance to the boundary of the intersection of two halfspaces $x \geq 0$ and $y \geq 0$. Right: contour lines of the scalar field $min(x, y)$. For both cases, contour lines vary from -1.5 to 1.5 with step of 0.5.

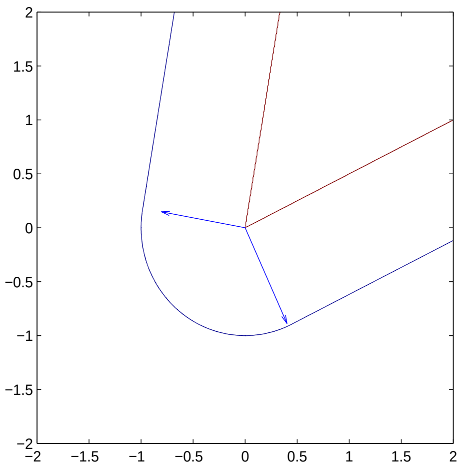

Figure 3: Local intersection of two general curves. Boundary curve (red) resulting from the intersection, the contour line at the distance -1.0 (blue) and the normals to each curve at the intersection point.

As illustrated by Fig. 2, the scalar field obtained by *min* corresponds to the Euclidean distance field everywhere except in the domain defined by: $Q := \{(x, y) : x < 0 \,\&\, y < 0$. For each point in that domain, the closest point on the boundary is the point obtained by the intersection of the two lines $x = 0$ and $y = 0$. Contour lines in that domain are circular arcs centered at this point.

The expression of Q for two orthogonal halfspaces is simple. We need now to define it for any general shapes to be intersected and then explain how to compute the closest point on the surface in that domain.

3. DISTANCE TO THE INTERSECTION OF TWO ARBITRARY SOLIDS

3.1 Determination of the domain where min differs from the Euclidean distance

Figure 3 illustrates the case of the local intersection of two general (non orthogonal) curves intersecting at a point $\mathbf{O}$. The boundary ($d = 0$), one contour line ($d = -1.0$) and the normals to each curve at $\mathbf{O}$ are drawn.

If the closest point is known, then the region of interest can be identified using the normal to each curve at the intersection point as seen in Fig. 3. However, since it requires to know the closest point, this method is not feasible. Instead, we proceed in the reverse way by first determining the domain of interest and then computing the closest point in that domain.

Given a point $\mathbf{p} \in R^2$ or R^3 (see Fig. 4 for an illustration in two dimensions), we first compute 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 $\nabla d_1(\mathbf{p})$, the projection is given by:

$$\mathbf{p_1} \leftarrow \mathbf{p} - d_1(\mathbf{p})\nabla d_1(\mathbf{p}) \tag{5}$$

Similarly, we compute $\mathbf{p_2}$ the projection of $\mathbf{p}$ on the second curve c_2. $\mathbf{p}$ belongs to the zone of interest Q, if $d_1(\mathbf{p_2}) < 0$

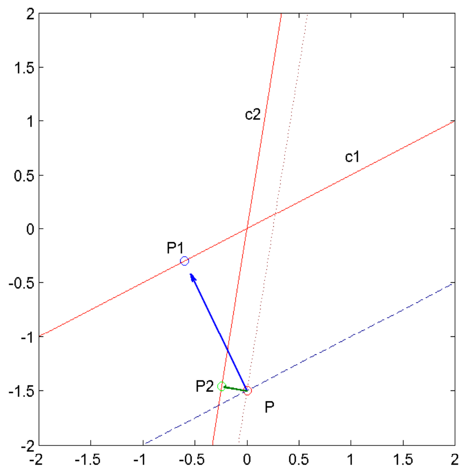

Figure 4: Two curves c_1 and c_2 (solid lines) defined by the iso-lines of the distance fields $d_1 = 0$ and $d_2 = 0$. A given point p and the contour lines of d_1 and d_2 passing through p. $\mathbf{p_1}$ and $\mathbf{p_2}$ the projections of p on c_1 and c_2.

and $d_2(\mathbf{p_1}) < 0$.

After determining if a point is in the domain where *min* applied to the distance fields differs from the Euclidean distance field, we need to determine the closest point on the boundary of the solid obtained by the intersection of the two solids.

3.2 Determination of the closest point

In all the above examples, the closest point on the boundary of the intersection from a point in the domain Q happened to be at the intersection of the two curves. This is not always the case. The closest point (x, y, z) on the boundary of the intersection of S_1 and S_2 from a point (x_0, y_0, z_0) in Q is the solution to the following constrained optimization problem:

$$\begin{aligned} \textbf{Minimize: } & \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} && (6)\\ \textbf{such that: } & d_1(x, y, z) \geq 0 && (7)\\ \textbf{and: } & d_2(x, y, z) \geq 0 && (8)\end{aligned}$$

Where: d_1 and d_2 are the distance fields to the boundary of the solids S_1 and S_2. In *Mathematica*^{tm} [22], such a constrained minimization problem can be solved with the command *NMinimize*. The point (x_0, y_0, z_0) can be used as a starting point. Since this point is in general close to the solution, the convergence is relatively fast.

3.3 Distance to the intersection

Regrouping all the results obtained so far, a procedure for computing the distance to the boundary of the intersection of two solids defined by signed distance functions is:

```
Require: a point $\mathbf{p}$ in $R^2$ or $R^3$ and two solids defined by
    the signed distance functions $d_1() \geq 0$ and $d_2() \geq 0$.
Ensure: $vd$ is the distance to the curve or surface, boundary of the intersection of the two solids.
1: $vd_1 = d_1(\mathbf{p})$
```

```
 2: $vd_2 = d_2(\mathbf{p})$
 3: $\mathbf{n_1} = \nabla d_1(\mathbf{p})$
 4: $\mathbf{p_1} = \mathbf{p} - vd_1\mathbf{n_1}$
 5: $\mathbf{n_2} = \nabla d_2(\mathbf{p})$
 6: $\mathbf{p_2} = \mathbf{p} - vd_2\mathbf{n_2}$
 7: $inQ = d_1(\mathbf{p_2}) < 0$ AND $d_2(\mathbf{p_1}) < 0$
 8: if inQ is true then
 9:     $\mathbf{p_c}$ = solution of (6) subject to constraints (7) and (8)
10:     $vd = -\|\mathbf{p} - \mathbf{p_c}\|_2$
11: else
12:     $vd = min(vd_1, vd_2)$
13: end if
14: return $vd$
```

4. MODIFICATIONS FOR THE UNION AND THE DIFFERENCE

Modification of the previous algorithm for obtaining the distance to the difference of two solids consists in replacing the function $d_2(.)$ by the function $-d_2(.)$ everywhere.

For the union, the following changes need to be made to the previous algorithm for the intersection:

- Line 7 needs to be changed to: $inQ = d_1(\mathbf{p_2}) > 0$ AND $d_2(\mathbf{p_1}) > 0$,
- Line 10 needs to be replaced by: $vd = \|\mathbf{p} - \mathbf{p_c}\|_2$

Finally, the closest point computed in line 9 is obtained by solving the following constrained minimization problem:

$$\textbf{Minimize: } \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} \quad (9)$$
$$\textbf{such that: } d_1(x,y,z) \le 0 \quad (10)$$
$$\textbf{and: } d_2(x,y,z) \le 0 \quad (11)$$

5. RESULTS AND DISCUSSIONS

5.1 Visualization of the resulting distance fields

Scalar fields corresponding to the intersection and union of two solids is illustrated in two and three dimensions in the following examples. Min/max, R-functions and the methods described in this paper are used to compute these scalar fields.

5.1.1 *Contour plots in two dimensions*

Figure 5 illustrates the contour plot of signed distance functions obtained by the intersection of two non-orthogonal planes and by the intersection of two disks. Intersection was implemented using: R-functions (left), min/max (middle) and the method proposed in this paper (right). The functions built using R-functions clearly differ from the Euclidean distance to the boundary of the constructed solid. The functions built using min/max fail also to do so but only in a subset of R^2. In comparison, the method described in this paper allows the construction of an accurate signed distance field to the boundary of the constructed solids as illustrated by the circular arcs in the contour lines outside of the constructed solids.

Figure 6 illustrates contour plots of the distance fields to the boundary of the union of two halfplanes (left) and two disks (right) using the method introduced here.

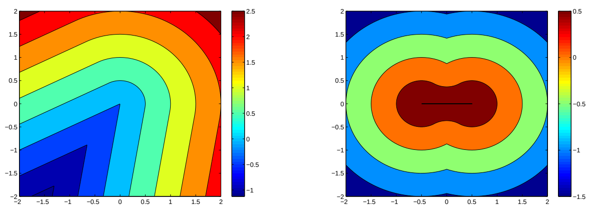

Figure 6: Left: contour plot of the distance to the union of two halfplanes with our method. Right: contour plot of the distance to the union of two disks.

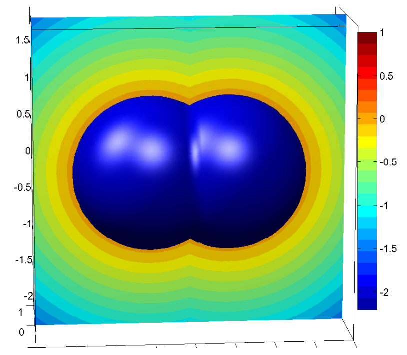

Figure 8: Distance to the union of two spheres computed with the presented method.

5.1.2 *Contour plots in three dimensions*

Visualizing scalar fields in three dimensions is more difficult. We visualize instead contour plots of 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. Contour plots need to be compared with the contour plots shown in fig. 7 (left) and (middle) obtained with respectively R-functions and min.

Figure 8 illustrates the result of the proposed method to compute the distance to the union of two spheres by showing a contour plot of the constructed distance field on a section by the plane $y = 0$.

5.2 Discussion

5.2.1 *Differentiability*

The distance function is continuous but not everywhere differentiable; for example: the function $d(p) = 1 - \|\mathbf{p} - \mathbf{p_c}\|_2$ is not differentiable at the point $\mathbf{p} = \mathbf{p_c}$. The algorithms described above use the gradient of the functions passed as arguments. Points at which the gradients can not be computed need to be handled differently. These points are on the medial axis of the solid and have more than one closest point on the surface of the solid. A possible solution is to select one of the closest point on the surface and define the gradient as the unit vector ending at this point.

5.2.2 *Successive applications of set operations*

It is possible to successively apply the proposed methods for intersection, union or difference to solids built using these operations. The gradient of the function constructed by the methods described in this paper needs to be computed. In

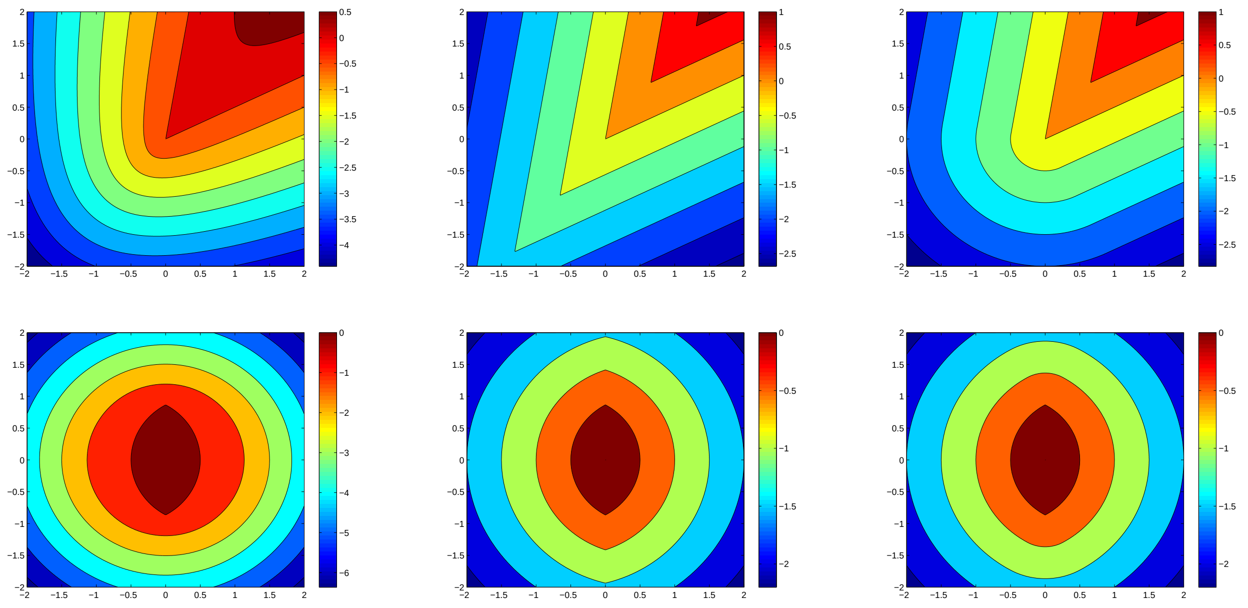
Figure 5: First row: contour plots of the scalar fields for the intersection of two halfplanes built using(from left to right): R-functions, min and the method described in this work. Second row: Result for the intersection of two disks.

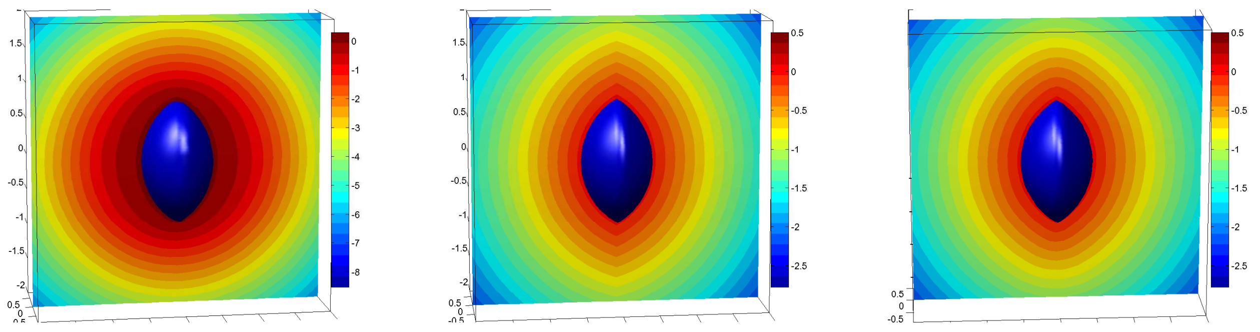
Figure 7: Distance to the intersection of two spheres using for the intersection operation (from left to right): R-functions, min and the method proposed in this work.

the example of the intersection, the function is not differentiable at the points: $\{\mathbf{p} : \mathbf{p} \notin Q$ and $d_1(\mathbf{p}) = d_2(\mathbf{p})\}$. This point-set corresponds to the medial axis of the resulting solid. It is possible to use the same method as described above. An alternative solution is to use modified versions of min and max. In the case of min, it is possible to write: $min(d_1, d_2) = \frac{1}{2}(d_1+d_2+\frac{(d_1-d_2)^2}{\sqrt{(d_1-d_2)^2}})$ and to add a small perturbation ϵ: $\tilde{min}(d_1, d_2) = \frac{1}{2}(d_1 + d_2 + \frac{(d_1-d_2)^2}{\sqrt{(d_1-d_2)^2+\epsilon}})$. This function is C^1 however it has the inconvenient of displacing each iso-contour; for example: assuming that $d_1 = 0$ 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 0.

5.3 Complexity

The proposed expressions for the set operations are in general slower than min/max or the corresponding R-functions. This is due to the necessity to solve a constrained minimization problem in some cases. We have only done experiments with the *NMinimize* function of $Mathematica^{tm}$ and some simple algorithms to solve this minimization problem. Since the objective function is not too complicated, there should be some more efficient methods. In addition, it is possible to provide a good approximate guess for the starting point, making the convergence relatively fast.

6. CONCLUSION

We have presented in this paper algorithms for computing in two and three dimensions the distance to a solid obtained by applying set operations (intersection, union or difference) to primitives. Contrary to the existing implementations of these operations (min/max and R-functions), the proposed methods allow to compute the signed Euclidean distance field to the boundary of the resulting shape.

We are considering extending this work by experimenting with the iterative application of these operations in order to construct complex solids. The use of these functions should also allow to easily implement rolling ball blend and double offsetting.

7. REFERENCES

[1] 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.

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

[3] J. Bloomenthal, editor. *Introduction to Implicit Surfaces.* Morgan-Kaufmann, 1997.

[4] 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.

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

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

[7] 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.

[8] 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.

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

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

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

[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. In *Proceedings of ACM Solid Modeling.* 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, British Aircraft Corporation, 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. Tsai. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys.*, 178(1):175–195, 2002.

[22] Wolfram Research Inc. *Mathematica Edition: Version 7.0.* Wolfram Research, Inc., 2008.

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

[24] 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.