Hierarchical Spline Approximation of the Signed Distance Function

Xinghua Song[1], Bert Jüttler[2], Adrien Poteaux[2]
[1]Projet GALAAD, INRIA Sophia Antipolis, France, [2]Institute of Applied Geometry, JKU Linz, Austria

***Abstract*—We present a method to approximate the signed distance function of a smooth curve or surface by using polynomial splines over hierarchical T-meshes (PHT splines). In particular, we focus on closed parametric curves in the plane and implicitly defined surfaces in space.**

***Keywords*—signed distance function, hierarchical T-spline, trimmed offsets**

1. INTRODUCTION

The signed distance function of a closed curve or surface Γ in the plane or in three-dimensional space assigns to any point $\mathbf{x}$ the shortest distance between $\mathbf{x}$ and Γ, with a positive sign if $\mathbf{x}$ is inside Γ and a negative one otherwise. It is a highly useful representation in a large number of problems in geometric computing. For instance, several approaches to surface reconstruction use the signed distance function, see for example [3], [4], [17].

The signed distance function is also closely related to the concept of level set methods, which were introduced in [11]. For instance, the distance function preservation has several advantages from the geometric and numerical point of view [7], [12], [18]. Kimmel and Bruckstein [9] used level sets for shape offsetting in the plane, in particular for computing trimmed offsets.

In the present paper, we propose a new method to compute a hierarchical approximation of the signed distance function. While algorithms developed in the context of level set methods use a computational grid, we use spline approximation, which enable us to get smooth approximations. More precisely, we interpolate the signed distance function of the boundary of the given shape by using a spline function defined over a hierarchical T-mesh [6], [14] level by level.

A simple but potentially useful application of our method is to generate the trimmed offsets of the boundary of a given shape. Since the signed distance function of the given boundary curve/surface can be approximated with any desired precision, we can obtain an implicit representation for any offset of the given boundary. We illustrate this observation by several examples in Section 5.

The computation of trimmed offset surfaces is still a challenging geometrical problem. The issue of self-intersection detection and elimination in general freeform curves and surfaces has been discussed by Pekerman et al. [13]. Seong et al. [15] used distance maps for trimming local and global self-intersections in offset curves/surfaces, see also [10].

The remainder if this paper is organized as follows. After a short introduction to polynomial splines over hierarchical T-meshes and a summary of properties of the signed distance function, we describe the computation of the approximate signed distance function, and provide some examples for trimmed offset computation.

2. PHT-SPLINES

Spline functions over T-meshes have recently been analyzed in [6]. For our application, we are mainly interested in the special case of C^1-smooth bi- and tricubic polynomial splines over hierarchical T-meshes, which is summarized in this section. We will call these spline functions *PHT-splines* for short.

A two-dimensional *T-mesh* is a partition of an axis-aligned box (e.g., the unit square) into smaller axis–aligned boxes. The edges of the boxes form a rectangular grid that may possess T-junctions. If a grid point of a two-dimensional T-mesh is a crossing vertex (i.e., it possesses valency 4), or belongs to the boundary of the domain, then we call it a *base vertex*.

A three-dimensional T-mesh is the extension of this concept to the three-dimensional space. It is a partition of an axis-aligned box such that each cell is another axis-aligned box. A vertex of a 3D T-mesh is called a *base vertex* if it either

- belongs to a boundary edge,
- is a crossing vertex on a boundary facet, or
- possesses valency 6.

In this paper, we consider *hierarchical* meshes. More precisely, we assume that the T-mesh has been obtained by repeatedly applying a splitting step, where an axis-aligned box is subdivided into two smaller axis-aligned boxes, to the original axis-aligned box representing the entire domain of the spline functions and to the boxes obtained by subdividing it.

For a given hierarchical T-mesh $\mathscr{T}$, a C^1-smooth bi- or tricubic PHT-spline $f(\mathbf{x})$ is a function which is a bicubic (2D) or tricubic (3D) polynomial within each cell of $\mathscr{T}$, such that the collection of these polynomial segments forms a globally C^1-smooth function. The space of these functions admits a simple local construction, as follows.

At each base vertex, we specify the value, the first derivatives, the mixed second derivative(s), and (in the

3D-case) the mixed third derivative of the function. Thus, we specify 4 or 8 (depending on the dimension) value and derivative data at each base vertex. Then, using Hermite interpolation by cubic polynomials, we can compute these data of $f(\mathbf{x})$ at the other vertices of the hierarchical T-mesh $\mathscr{T}$. For any point $\mathbf{x}$ in any cell C of $\mathscr{T}$, the value of $f(\mathbf{x})$ is determined by the value and derivative data at the 4 (resp. 8) vertices of the cell C.

More precisely, we obtain a bicubic or tricubic polynomial for each cell, which matches the value and derivative data at the vertices. Thus, a PHT-spline defined over a given T-mesh $\mathscr{T}$ can be determined by the value and derivative data at the base vertices of $\mathscr{T}$. This is in agreement with the dimension formulas of PHT-spline given in [5].

3. The signed distance function (SDF)

Consider a simple closed curve or surface Γ in the Euclidean space $\mathbf{R}^n$, where $n = 2$ or 3. (In the remainder of this paper, we call Γ a surface and assume that this includes the case of curves.) The surface Γ divides $\mathbf{R}^n$ into three parts: the interior Γ^+, the exterior Γ^- and Γ. The *signed distance function* of Γ is a scalar function which is defined in $\mathbf{R}^n$ as:

$$d(\mathbf{x}) = \begin{cases} \|\mathbf{x}-\mathbf{p}\| & \text{if } \mathbf{x} \in \Gamma^+ \\ -\|\mathbf{x}-\mathbf{p}\| & \text{if } \mathbf{x} \in \Gamma^- \\ 0 & \text{if } \mathbf{x} \in \Gamma \end{cases}, \tag{1}$$

where $\mathbf{p} = (p_1, ..., p_n)$ is the closest point of $\mathbf{x} = (x_1, ..., x_n)$ on the surface Γ.

3.1 Derivatives

For later reference we compute the derivatives of the signed distance function. We assume that the point $\mathbf{x}$ lies in Γ^+, i.e.,

$$d(\mathbf{x}) = \|\mathbf{x}-\mathbf{p}\|, \tag{2}$$

and that it possesses a unique closest point on Γ. Thus, $\mathbf{x}$ is not on the medial axis. (Points in Γ^- can be dealt with analogously.) The computation of the derivatives of the signed distance function (2) leads to the following formulas:

$$\begin{aligned} d_i(\mathbf{x}) &= \frac{\partial d}{\partial x_i}(\mathbf{x}) = \\ &= \frac{\mathbf{x}-\mathbf{p}}{\|\mathbf{x}-\mathbf{p}\|} \cdot \left(\frac{\partial \mathbf{x}}{\partial x_i} - \frac{\partial \mathbf{p}}{\partial x_i} \right) = \frac{x_i - p_i}{\|\mathbf{x}-\mathbf{p}\|}, \end{aligned} \tag{3}$$

$$\begin{aligned} d_{ij}(\mathbf{x}) &= \frac{\partial^2 d}{\partial x_i \partial x_j}(\mathbf{x}) = \\ &= \frac{-1}{\|\mathbf{x}-\mathbf{p}\|} \frac{\partial p_i}{\partial x_j} - \frac{(x_i - p_i)(x_j - p_j)}{\|\mathbf{x}-\mathbf{p}\|^3}, \end{aligned} \tag{4}$$

and

$$\begin{aligned} d_{ijk}(\mathbf{x}) &= \frac{\partial^3 d}{\partial x_i \partial x_j \partial x_k}(\mathbf{x}) = \\ &= \frac{-\frac{\partial^2 p_i}{\partial x_j \partial x_k}}{\|\mathbf{x}-\mathbf{p}\|} + \\ &+ \frac{\frac{\partial p_i}{\partial x_j}(x_k - p_k) + \frac{\partial p_i}{\partial x_k}(x_j - p_j) + \frac{\partial p_j}{\partial x_k}(x_i - p_i)}{\|\mathbf{x}-\mathbf{p}\|^3} + \\ &+ \frac{3(x_i - p_i)(x_j - p_j)(x_k - p_k)}{\|\mathbf{x}-\mathbf{p}\|^5}. \end{aligned} \tag{5}$$

Therefore, if we want to compute the required derivatives of the signed distance function at a point $\mathbf{x}$, we only need to compute $\frac{\partial p_i}{\partial x_j}$ and $\frac{\partial^2 p_i}{\partial x_j \partial x_k}$, where $i \neq j$, $j \neq k$ and $k \neq i$.

In this paper we consider two representations of the curve or surface Γ. In the plane, we consider a *closed planar parametric curve*. In the three-dimensional space we consider an *implicitly defined surface*, which is given as the zero set of scalar field. In both cases, we assume that the curve and surface is sufficiently smooth (C^2 for the planar case and C^3 for the surface case), possibly except for a few singular points. Thus, the derivatives of the signed distance function are well defined for almost all points.

The purpose of considering two different representations (implicitly defined surfaces vs. parametric curves) is to cover in this paper the evaluation of the derivatives of the signed distance function for both types of representations. Clearly, our method can be extended to other representations of smooth curves and surfaces, such as parametric free-form surfaces and implicitly defined planar curves.

3.2 The case of planar parametric curves

Given a planar parametric curve $\mathbf{c}(u)$, if $\mathbf{p} = \mathbf{c}(u^*)$ denote the closest point of $\mathbf{x} = (x_1, x_2)$ in $\mathbf{c}(u)$, then we have:

$$\frac{\partial \mathbf{p}}{\partial x_j} = \frac{d\mathbf{c}}{du}\frac{\partial u}{\partial x_j}(u^*). \tag{6}$$

Computing the derivative of

$$(\mathbf{c}(u^*) - \mathbf{x}) \cdot \frac{d\mathbf{c}}{du}(u^*) = 0$$

with respect to x_1 gives

$$\begin{aligned} \left(\frac{d\mathbf{c}}{du}\frac{\partial u}{\partial x_1}(u^*) - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \cdot \frac{d\mathbf{c}}{du}(u^*) + \\ + (\mathbf{c}(u^*) - \mathbf{x}) \cdot \frac{d^2\mathbf{c}}{du^2}\frac{\partial u}{\partial x_1}(u^*) = 0. \end{aligned} \tag{7}$$

Consequently, we can obtain $\frac{\partial u}{\partial x_1}(u^*)$ by solving this linear equation. Similarly, we can obtain $\frac{\partial u}{\partial x_2}(u^*)$. Using additional differentiations we may evaluate the mixed second order derivative of the signed distance function. Finally we may evaluate the required derivatives of the signed distance function of $\mathbf{c}(u)$ at point $\mathbf{x}$ according to equations (3) and (4).

3.3 The case of implicitly defined surfaces

Now we consider an implicitly defined surface $\mathcal{Z}(g) = \{\mathbf{x} \in \Omega \subseteq \mathbf{R}^3 |\ g(\mathbf{x}) = 0\}$ in $\mathbf{R}^3$ which is defined by a scalar function $g(\mathbf{x})$. We assume that the surface does not contain singular points. In this case, the closest point satisfies

$$(\nabla g)(\mathbf{p}) \times (\mathbf{p} - \mathbf{x}) = \mathbf{0}. \tag{8}$$

Differentiating this equation with respect to x_1 gives

$$\left(\mathbf{H} \cdot \frac{\partial \mathbf{p}}{\partial x_1}\right) \times (\mathbf{p} - \mathbf{x}) + \\ + (\nabla g)(\mathbf{p}) \times \left(\frac{\partial \mathbf{p}}{\partial x_1} - \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\right) = \mathbf{0}, \tag{9}$$

where $\mathbf{H}$ is the Hessian of g. Thus, we obtain $\frac{\partial \mathbf{p}}{\partial x_1}$ by solving this system of linear equations. The remaining two first derivative vectors $\frac{\partial \mathbf{p}}{\partial x_2}$ and $\frac{\partial \mathbf{p}}{\partial x_3}$ can be computed similarly. Moreover, if we compute the derivative of (9) with respect to x_2, and then substitute $\frac{\partial \mathbf{p}}{\partial x_1}$, $\frac{\partial \mathbf{p}}{\partial x_2}$ and $\frac{\partial \mathbf{p}}{\partial x_3}$, we obtain $\frac{\partial^2 \mathbf{p}}{\partial x_1 \partial x_2}$; in the same way, we can compute $\frac{\partial^2 \mathbf{p}}{\partial x_2 \partial x_3}$ and $\frac{\partial^2 \mathbf{p}}{\partial x_3 \partial x_1}$. Finally, we get all required derivatives of the signed distance function of $\mathcal{Z}(g)$ at a point $\mathbf{x}$ from the three equations (3), (4) and (5).

4. Fitting the SDF using PHT-splines

In this section, we describe how we approximate the signed distance function d of a given surface by interpolating the value and derivative data of d using a PHT-spline.

4.1 Outline

The algorithm consists of three steps.

1) We first choose a bounding box of the given surface that we will use as the initial T-mesh $\mathscr{T}^0$. Then, we compute the closest point of every base vertex in $\mathscr{T}^0$ using Newton's method, and compute the first order derivatives at the base vertices according to equations (2) to (5). This generates an initial bicubic (2D) or tricubic (3D) PHT-spline $f^0(\mathbf{x})$. We put $k = 0$.
2) We compute the fitting errors of the PHT-spline $f^k(\mathbf{x})$ over the T-mesh $\mathscr{T}^k$ at level k. If the fitting errors of all cells in $\mathscr{T}^k$ are less than a given threshold ϵ, or if k has reached the maximum allowed subdivision level $k_{\max}$, then we output the result. Otherwise, we subdivide the cells whose fitting errors are greater than ϵ to form a new T-mesh $\mathscr{T}^{k+1}$ at level $k+1$.
3) We compute the closest point of every new base vertex in $\mathscr{T}^{k+1}$ by using a Newton-type method, and then compute the associated first order derivatives according to equations (2) to (5). This gives us a new PHT-spline $f^{k+1}(\mathbf{x})$ over the T-mesh $\mathscr{T}^{k+1}$ at level $k+1$, which approximates the signed distance function better than $f^k(\mathbf{x})$. Let $k \leftarrow k+1$ and continue with step 2.

We will now describe the key steps of our method in more detail.

4.2 Closest point computation

Given a base vertex $\mathbf{x}$ and an integer N (specified by the user), we first sample N data points $\{\mathbf{x}_i\}_{i=1}^N$. Then, we choose the point which is closest to $\mathbf{x}$ as the initial point and compute the closest point of the base point $\mathbf{x}$ using a Newton-type method. In order to make this paper self-contained, we summarize the method below.

In the case of a parametric curve $\mathbf{c}(u)$, we compute the closest point of $\mathbf{x}$ as follows (τ_0 denotes a positive threshold and k a small number which specifies the number of initial points, typically $k \leq 3$):

1) Choice of the initial point: We sample N parameters from the domain of $\mathbf{c}(u)$ uniformly, and we take the associated points of these parameters on the curve $\mathbf{c}(u)$ as the sampled data points $\{\mathbf{x}_i\}_{i=1}^N$. Let us denote with $\{\mathbf{x}_0^j\}_{j=1}^k$ the k closest points of $\mathbf{x}$ in $\{\mathbf{x}_i\}_{i=1}^N$; we will use them as initial points.
   The next two steps are applied to all k closest points, but only the point which gives the shortest distance among the improved points is kept. (Considering more than one initial point improves the robustness of the closest point computation.)
2) Initialization of the iteration: Let u_0^j denote the parameter of the closest point of $\mathbf{x}_0^j$ on the curve, and
$$h(u) = (\mathbf{c}(u) - \mathbf{x}) \cdot \mathbf{c}'(u), \tag{10}$$
   be a scalar function, where $'$ is the derivative with respect to the curve parameter u. We put $n = 0$.
3) Iterative improvement of the parameter value: Compute $u_{n+1}^j = u_n^j - \frac{h(u_n^j)}{h'(u_n^j)}$. If $|h(u_{n+1}^j)| < \tau_0$, output $\mathbf{p} = \mathbf{c}(u_{n+1}^j)$; otherwise, let $n \leftarrow n+1$ and return to step 2.

In the case of an implicitly defined 3D surface $\mathcal{Z}(g)$, we proceed as follows (now we need two thresholds $\tau_1 > 0$ and $\tau_2 > 0$):

1) Choice of the initial points: First, we get a polygon mesh of $\mathcal{Z}(g)$ by using an existing method, such as the marching triangulation algorithm (see e.g. [8]). Then, we identify the k closest points $\{\mathbf{x}_0^j\}_{j=1}^k$ to $\mathbf{x}$ among the vertices of the polygon mesh.
   The following steps are applied to all of them. As in the planar case, we keep only the improved point with the shortest distance to $\mathbf{x}$.
   Let $n = 0$.
2) Projection into the tangent plane: If
$$\frac{\|(\mathbf{x} - \mathbf{x}_n^j) \times \nabla g\|}{\|\nabla g\|} < \tau_1,$$
   then we output $\mathbf{p} = \mathbf{x}_n^j$; otherwise, we compute the tangent plane $\mathscr{N}_x$ of the surface $\mathcal{Z}(g)$ at the

point $\mathbf{x}_n^j$ and denote with $\overline{\mathbf{x}}_n^j$ the projected point of $\mathbf{x}$ into the tangent plane $\mathscr{N}_x$.

3) Projection onto the surface: We compute

$$\overline{\mathbf{x}}_{n+1}^j = \overline{\mathbf{x}}_n^j - \frac{g(\overline{\mathbf{x}}_n^j)}{(\nabla)g(\overline{\mathbf{x}}_n^j)\cdot(\mathbf{x}-\overline{\mathbf{x}}_n^j)}(\mathbf{x}-\overline{\mathbf{x}}_n^j).$$

If $|g(\overline{\mathbf{x}}_{n+1}^j)| < \tau_2$, then we define $\mathbf{x}_{n+1}^j = \overline{\mathbf{x}}_{n+1}^j$, put $n \leftarrow n+1$ and return to Step 2; otherwise, we put $n \leftarrow n+1$ and start again the step 3.

Clearly, the convergence of these Newton-type methods can be improved by a step-size control (which may guarantee the convergence towards a locally closest point) and it also depends on the number of sample points which are used to define the initial solution. We do not touch these issues in this paper. See e.g. [1], [2].

4.3 Estimation of fitting errors

To estimate the fitting error of a cell C_j of the T-mesh $\mathscr{T}^k$ at level k, we select m random points $\{\mathbf{q}_i\}_{i=1}^m$ (m is given by the user), and compute the associated closest point $\overline{\mathbf{q}}_i$ on the surface $\mathcal{Z}(f)$ for every point $\mathbf{q}_i$. Then, the fitting error of the PHT-spline $f^k(\mathbf{x})$ at every sampled point is $\epsilon_i = |f^k(\mathbf{q}_i) - \|\mathbf{q}_i - \overline{\mathbf{q}}_i\|\,|$, and the fitting error of C_i is the biggest fitting error among all sampled points.

In addition to this deviation of the value, one might also consider the difference between the gradients of the spline function and the gradients of the signed distance function or the deviation from a unit gradient field. In our test examples, using the deviation from the correct value produced fairly reasonable results.

4.4 Examples

We demonstrate the performance of the algorithm by three examples, see Fig. 1. The figure shows the given curves (top row), the given T-meshes obtained by the adaptive refinement procedure (center row) and the graph surfaces of the signed distance functions (bottom row). The graph surfaces are smooth everywhere, except for the points on the medial axis of the interior and exterior regions Γ^+ and Γ^-. The PHT spline fitting algorithm correctly identifies these regions and refines the T-mesh accordingly.

The computation time for the three examples was in the order of less than 10 seconds on a standard PC, see Section 6.2 for details.

Finally we present experimental results relating the number of cells in the T-mesh to the fitting error, see Fig. 2. We computed approximations of the signed distance function for four different fitting errors. The error and the number of cells are plotted on a doubly logarithmic scale.

We compare these results with the number of cells needed for approximating a C^4 smooth function on a uniform grid. Due to the high approximation order (which is equal to 4) of bicubic splines, the slope of the graph relating the number of cells to the fitting error is

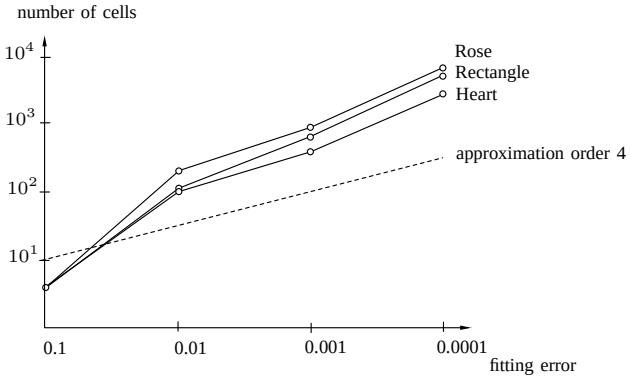

Fig. 2. Number of cells in the T-mesh vs. fitting error. The bounding box of all three examples is the unit square.

$\frac{1}{2}$, since the number of cells for cell size $h = 1/n$ is n^2. The number of cells needed for approximating the non-C^4-smooth signed distance function grows faster than that, but the rate of growth is still relatively close to $\frac{1}{2}$. This indicates that by using T-splines we can efficiently represent the non-smooth regions of the signed distance function.

5. Computing approximate trimmed offsets

Recall that an *offset curve* or *surface* is obtained by adding a scaled multiple of the unit normal vector field to the points of the curve or surface. Offsetting is a fundamental operation in geometric design, and offset surfaces are required for various applications, such as NC milling.

It is well known that offsetting may produce curves and surfaces with self-intersections, which have to be detected and eliminated in applications (e.g., the so-called "swallow-tails" which occur in offsets of planar curves). After eliminating these unwanted segments of the offset curves or surface, one obtains the *trimmed offset curve* or *surface*. Computing and eliminating local and global self-intersections is a challenging computational problem.

If an approximation of the signed distance function is available, we can easily compute an implicit representation of approximate trimmed offset surfaces. More precisely, let $\overline{f}(\mathbf{x})$ be the final PHT-spline. We note that it fits the signed distance function of the given surface $\mathcal{Z}(g)$. Therefore, the trimmed offset surface of $\mathcal{Z}(g)$ which has the offsetting distance δ can be approximately represented as a level set of the scalar function $\overline{f}(\mathbf{x})$,

$$\mathcal{Z}(g_\delta) = \{\mathbf{x} \in \mathbf{R}^3 |\ \overline{f}(\mathbf{x}) = \delta\}.$$

We illustrate this simple observation by a two surface examples. Figure 3 shows two objects and the associated families of trimmed offset surfaces. In order to visualize the offsets, we used a cutting plane and show only a part of these surfaces, which are contained in the interior of the object.

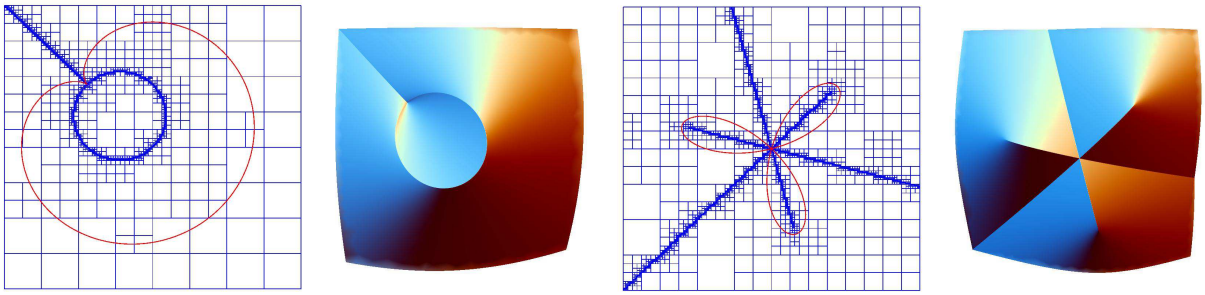

Fig. 1. Approximation of the signed distance function by PHT splines for two curve examples. The figures show the curves, the T-meshes and the graph surfaces.

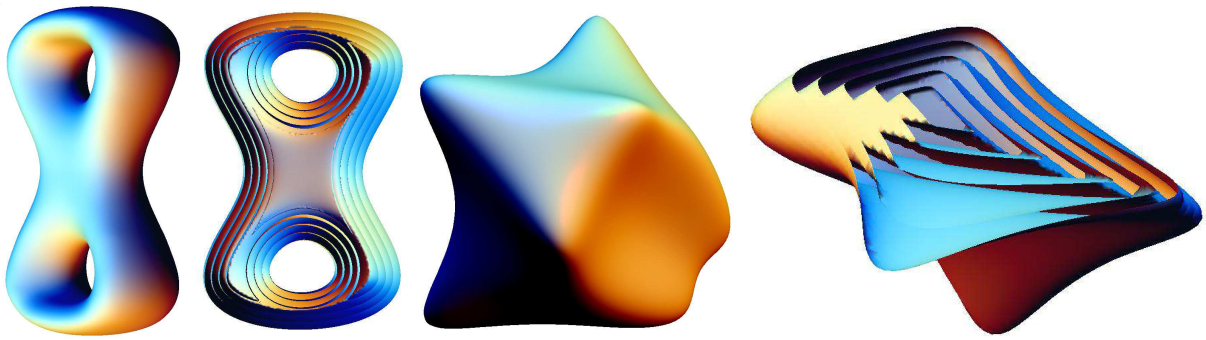

Fig. 3. Two closed surfaces and their trimmed inner offsets.

6. Conclusion

We presented a new method to fit the signed distance function of a given boundary curve of surface by using a simple but efficient PHT-spline (bi- and tricubic piecewise polynomial functions over hierarchical T-meshes) Hermite interpolation method. We then use this fitting to compute an implicit approximation of the trimmed offset curves and surfaces of the given boundary. The report [16] shows how to generate an approximation of the medial axis of the given domain.

Acknowledgments. The author acknowledge the support of the Austrian Science Fund, project S9202, and of the European Union (PITN-GA-2008-214584 SAGA).

References

[1] M. Aigner and B. Jüttler. Robust computation of foot points on implicitly defined curves. In M. Dæhlen, K. Mørken, and L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces: Tromsø 2004*, pages 1–10. Nashboro Press, Brentwood, 2005.

[2] I.J. Anderson, M.G. Cox, A.B. Forbes, J.C. Mason, and D.A. Turner. An efficient and robust algorithm for solving the foot point problem. In *Mathematical methods for curves and surfaces II*, pages 9–16, Nashville, TN, USA, 1998. Vanderbilt University Press.

[3] C.L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Proc. SIGGRAPH*, pages 109–118, New York, NY, USA, 1995. ACM.

[4] J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proc SoCG*, pages 223–232, New York, NY, USA, 2000. ACM.

[5] J. Deng, F. Chen, and Y. Feng. Dimensions of spline spaces over T-meshes. *J. Comput. Appl. Math.*, 194(2):267–283, 2006.

[6] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng. Polynomial splines over hierarchical T-meshes. *Graph. Models*, 70(4):76–86, 2008.

[7] J. Gomes and O. D. Faugeras. Reconciling distance functions and level sets. In *Proc. SCALE-SPACE*, pages 70–81, London, 1999. Springer.

[8] E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.

[9] R. Kimmel and A.M. Bruckstein. Shape offsets via level sets. *Computer-Aided Design*, 25:154–162, 1993.

[10] T. Maekawa, W. Cho, and N.M. Patrikalakis. Computation of self-intersections of offsets of Bézier surface patches. *ASME J. Mech. Design*, 119:275–283, 1997.

[11] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[12] S.J. Osher and R.P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[13] D. Pekerman, G. Elber, and M.-S. Kim. Self-intersection detection and elimination in freeform curves and surfaces. *Computer-Aided Design*, 40:150–159, 2008.

[14] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *Proc. SIGGRAPH*, pages 477–484, New York, 2003. ACM.

[15] J.-K. Seong, G. Elber, and M.-S. Kim. Trimming local and global self-intersections in offset curves/surfaces using distance maps. *Computer-Aided Design*, 38:183–193, 2006.

[16] X. Song, B. Jüttler, and A. Poteaux. Hierarchical spline approximation of the signed distance function and applications to trimmed offsets and medial axis computation. Technical report, NFN Industrial Geometry, 2010. available from www.industrial-geometry.at/techrep.php.

[17] H. Yang, M. Fuchs, B. Jüttler, and O. Scherzer. Evolution of T-spline level sets with distance field constraints for geometry reconstruction and image segmentation. In *Shape Modeling International*, pages 247–252. IEEE Press, 2006.

[18] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127(1):179–195, 1996.