Evolution of T-spline Level Sets with Distance Field Constraints for Geometry Reconstruction and Image Segmentation

Huaiping Yang[†], Matthias Fuchs[‡], Bert Jüttler[†] and Otmar Scherzer[‡]
[†]Johannes Kepler University Linz, [‡]University of Innsbruck
{*yang.huaiping,bert.juettler*}*@jku.at*, {*matz.fuchs,otmar.scherzer*}*@uibk.ac.at*

Abstract

We study the evolution of T-spline level sets (i.e, implicitly defined T-spline curves and surfaces). The use of T-splines leads to a sparse representation of the geometry and allows for an adaptation to the given data, which can be unorganized points or images. The evolution process is governed by a combination of prescribed, data-driven normal velocities, and additional distance field constraints. By incorporating the distance field constraints we are able to avoid additional branches and singularities of the T-spline level sets without having to use re-initialization steps. Experimental examples are presented to demonstrate the effectiveness of our approach.

Keywords: T-spline, level sets, unorganized points, image segmentation

1 Introduction

Implicitly defined curves and surfaces, i.e., curves and surfaces which are described as the zero set of a scalar field, have found numerous applications in Shape Modeling and Geometric Computing. They have been used for geometric modeling [4, 12], for object reconstruction from unorganized points [3, 8, 16, 17] and for improving the robustness of algorithms for computing surface-surface intersections [6]. Depending on the area of the application, different representations of the underlying scalar fields have emerged. These include functions obtained by hierarchically combining simpler ones [4], representations based on radial basis functions [3], spline functions [8, 6, 12, 16], to grid–based discretizations [17].

For various problems in image processing, many approaches are based on the *evolution processes* generating time–dependent families of curves (and similarly for surfaces) by an implicit velocity field in the direction of its normals. One example of this evolution is used for segmentation. Kass et al. [9] proposed 'snakes', or active contours, for boundary detection. They compute the boundary curve of a given 2D object by minimizing an energy functional in a space of admissible curves. Caselles et al. [5] proved that this problem can be transformed to the problem of computing a geodesic curve in a Riemannian space with a metric determined by the image data. Solving this problem using the steepest-descent method leads to an curve evolution equation.

For *implicitly defined curves and surfaces*, one may formulate *evolution processes* using the *level set approach* of Osher and Sethian [11]. As a major advantage in certain applications, where the topology is not known a priori, the level-set representation is parameter–free and it intrinsically adapts to topological changes during the evolution. Consequently, one can detect complex topological structures, such as objects consisting of multiple components, without using prior knowledge.

The problem of geometry reconstruction from point data clouds involves similar equations. Zhao et al. [17] present a convection model to compute an implicit surface S that minimizes a global distance function to the input data set. Allegre et al. [2] translates the convection scheme into Computational Geometry terms.

While typical implementations of level set evolutions rely on grid-based discretizations of the domain, this paper proposes to represent the function f by a bivariate or trivariate *T-spline* function (see [13]). On the one hand, due to the use of a piecewise rational scalar field, the resulting zero level sets are algebraic spline curves and surfaces, which can be pieced together with any desired level of differentiability. On the other hand, the use of T-splines leads to a sparse representation of the geometry, which can, however, be refined locally, adapting the numbers of degrees of freedom to the particular data.

The remainder of the paper is organized as follows. The next section defines implicit T-spline curves and surfaces. Section 3 formulates the evolution process for these geometry representation. In particular, it is shown how to incorporate a distance field constraint, which makes it possible to avoid (possibly time–consuming) renormalization steps.

The fourth section applies evolution of T-spline curves and surfaces to the problem of geometry reconstruction, both from unorganized point data and images. After presenting some experimental results in Section 5, we conclude this paper and discuss future work.

2 T-spline Level Sets

Sederberg et al. [13] generalized non-uniform B-spline surfaces to so–called T-splines. In this section, we introduce *implicitly defined T-spline curves and surfaces*. Let $f(x, y)$ be a bivariate T-spline function defined over some domain D,

$$f(x,y) = \frac{\sum_{i=1}^{n} c_i B_i(x,y)}{\sum_{i=1}^{n} B_i(x,y)}, \quad (x,y) \in D \subset \mathbb{R}^2 \qquad (1)$$

with the real coefficients (control points) c_i, $i = 1, 2, ..., n$, where n is the number of control points. For cubic T-splines, the basis functions are $B_i(x, y) = N^3_{i0}(x)N^3_{i0}(y)$ where $N^3_{i0}(x)$ and $N^3_{i0}(y)$ are certain cubic B-splines, whose are determined by the T-spline control grid (T-mesh).

The zero set of the function f is defined by

$$\mathcal{C}(f) = \{\ (x,y) \in D \subset \mathbb{R}^2 \mid f(x,y) = 0\ \}, \qquad (2)$$

and it is called an *implicit T-spline curve*.

This definition can be easily generalized to *implicit T-spline surfaces* in 3D. Both implicit T-spline curves and surfaces are called *T-spline level sets* in our paper.

In order to simplify the notation, we use $\mathbf{x}$ to uniformly represent the point $\mathbf{x} = (x, y)$ resp. $\mathbf{x} = (x, y, z)$, and gather the control coefficients (in a suitable ordering) in a column vector $\mathbf{c}$. The T-spline basis functions form another column vector $\mathbf{b} = [b_1, b_2, ..., b_n]^\top$, where

$$b_i = B_i(\mathbf{x}) / \sum_{i=1}^{n} B_i(\mathbf{x}), \quad i = 1, 2, ..., n.$$

The *T-spline level set* $\Gamma(f)$ is defined as the zero set of the function $f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}$. For a fixed set of basis functions $\mathbf{b}$, the T-spline level set is determined by the control coefficients $\mathbf{c}$.

Since a T-spline function is piecewise rational, the T-spline level sets are piecewise algebraic curves and surfaces. Moreover, if no singular points are present, they inherit the order of differentiability of the basis functions, i.e., they are C^2 in the cubic case. Derivatives of f, which will be needed for formulating the evolution, can easily be evaluated.

3 T-spline Level Set Evolution

We describe the evolution process of the T-spline level set, which is driven by normal velocities, combined with an additional signed distance field constraint.

3.1 Evolution with Normal Velocity

Consider a T-spline level set $\Gamma(f)$ defined as the zero set of a time-dependent function $f(\mathbf{x}, \tau)$, where

$$f(\mathbf{x}, \tau) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}(\tau), \qquad (3)$$

with some time parameter τ. It will be subject to the evolution process

$$\partial \mathbf{x} / \partial \tau = v(\mathbf{x}, \tau)\vec{\mathbf{n}}, \quad \mathbf{x} \in \Gamma(f), \qquad (4)$$

where v is a scalar-valued *velocity function* (or *speed function*) along the normal direction $\vec{\mathbf{n}} = \nabla f / |\nabla f|$ of Γ. A short computation gives the evolution equation of T-spline level sets under the normal velocity v,

$$\partial f / \partial \tau = -v(\mathbf{x}, \tau)|\nabla f|, \quad \mathbf{x} \in \Gamma(f). \qquad (5)$$

In our case, however, f is a linear combination of the time-dependent coefficients $\mathbf{c}$, see (3). In order to translate (5) into an evolution equation for the coefficients, we use a least–squares approach. More precisely, we choose the time derivative of the T-spline f by solving

$$E_0 = \int_{\mathbf{x} \in \Gamma(f)} \left(\frac{\partial f(\mathbf{x}, \tau)}{\partial \tau} + v(\mathbf{x}, \tau)\, |\nabla f(\mathbf{x}, \tau)|\ \right)^2 \mathrm{d}s \to \text{Min},$$

where s represents the arc length or surface area of the T-spline level set. For the actual computation, a discretized version (which can be seen as a numerical integration) is more appropriate, i.e., we replace E_0 with

$$E = \sum_{j=1}^{N_0} \left(\frac{\partial f(\mathbf{x}_j, \tau)}{\partial \tau} + v(\mathbf{x}_j, \tau)\, |\nabla f(\mathbf{x}_j, \tau)|\ \right)^2, \qquad (6)$$

where $\mathbf{x}_j$, $j = 1, \ldots, N_0$ ($N_0 \gg n$) is a sequence of sampling points, which are uniformly distributed along the T-spline level set. Finally, the substitution (cf. Eq. (3))

$$\partial f(\mathbf{x}, \tau) / \partial \tau = \mathbf{b}(\mathbf{x})^\top \dot{\mathbf{c}}(\tau), \qquad (7)$$

where the dot $\dot{\mathbf{c}}$ indicates differentiation with respect to τ, leads to the *evolution term* E of the T-spline level set,

$$E = \sum_{j=1}^{N_0} \left(\mathbf{b}(\mathbf{x})^\top \dot{\mathbf{c}}(\tau) + v(\mathbf{x}_j, \tau)\, |\nabla f(\mathbf{x}_j, \tau)|\ \right)^2. \qquad (8)$$

The evolution term E is a non–negative definite quadratic function of the derivatives $\dot{\mathbf{c}}$,

$$E = \dot{\mathbf{c}}^\top\, Q_E(\mathbf{c})\, \dot{\mathbf{c}} + \mathbf{l}_E(\mathbf{c})^\top\, \dot{\mathbf{c}} + k_E(\mathbf{c}). \qquad (9)$$

The coefficients of this function, which are collected in the symmetric non–negative definite matrix Q_E, the vector $\mathbf{l}_E$ and the scalar k_E, depend on the coefficients $\mathbf{c}$ and can be found from (8). It should be noted that the matrix $Q_E(\mathbf{c})$ is likely to be singular. In particular, this is the case if the support of at least one T-spline basis function and the T-spline level set are disjoint.

3.2 Distance Field Constraint

For most existing level set evolutions, the initial function f is chosen as an approximation to the *signed distance function* of its zero level set. But during the evolution, f will drift away from this signed distance property, and cause some flat and/or steep regions, which may dramatically decrease the accuracy of the computed solution [1]. This motivates the use of *level set reinitialization* to restore the *signed distance* property. However, the re-initialization procedure is usually relatively expensive (especially in 3D) and has to be applied frequently.

We will avoid the use of re-initialization by introducing an additional *distance field constraint*. Recently, similar techniques have been proposed in the literature [10, 15].

Since an ideal signed distance function ϕ satisfies $|\nabla\phi| = 1$ everywhere in the domain, we propose the constraint term

$$S_0 = \int_D \Big(\frac{\partial|\nabla f(\mathbf{x},\tau)|}{\partial\tau} + |\nabla f(\mathbf{x},\tau)| - 1 \Big)^2 \, \mathrm{d}\mathbf{x} \to \mathrm{Min}$$

as a penalty function which penalizes the deviation of f from a signed distance function. If – for some value of the time parameter τ – the gradient length at some point is less (resp. greater) than 1, then the time derivative of this length will be forced to be positive (resp. negative), in order to restore the unit gradient property.

Once again, the actual computation is based on a discretized version. We uniformly sample N_1 points $\mathbf{y}_j$, $j = 1, \ldots, N_1$ ($N_1 \gg n$) in the domain of level set function and use them to derive a discretized version of S_0,

$$S = \frac{A(D)}{N_1} \sum_{j=1}^{N_1} \Big(\frac{\partial|\nabla f(\mathbf{y}_j,\tau)|}{\partial\tau} + |\nabla f(\mathbf{y}_j,\tau)| - 1 \Big)^2, \quad (10)$$

where $A(D)$ is the area/volume of the domain D.

In our case, the level set function f has the form (3), hence the time derivative of the gradient length

$$\frac{\partial|\nabla f(\mathbf{y}_j,\tau)|}{\partial\tau} = \frac{2\nabla f(\mathbf{y}_j,\tau)}{|\nabla f(\mathbf{y}_j,\tau)|} \, (\nabla \mathbf{b}(\mathbf{y}_j)^\top \dot{\mathbf{c}}(\tau)) \quad (11)$$

depends linearly on $\dot{\mathbf{c}}(\tau)$.

By combining (10) and (11), we may represent the signed distance field constraint term as a non–negative definite quadratic function of the derivatives $\dot{\mathbf{c}}$,

$$S = \dot{\mathbf{c}}^\top \, Q_S(\mathbf{c}) \, \dot{\mathbf{c}} + \mathbf{l}_S(\mathbf{c})^\top \, \dot{\mathbf{c}} + k_S(\mathbf{c}). \quad (12)$$

The coefficients of this function, which are collected in the symmetric non–negative definite matrix Q_S, the vector $\mathbf{l}_S$ and the scalar k_S, depend on the coefficients $\mathbf{c}$ and can be found from (10) and (11). According to our numerical experiments, the matrix $Q_S(\mathbf{c})$ is generally positive definite, i.e., non–singular, except for very rare special cases (such as a T-spline f which represents the signed distance function with respect to a straight line).

3.3 Solving the Evolution Equation

For each evolution step of T-spline level sets, the time derivatives $\dot{\mathbf{c}}(\tau)$ are computed by minimizing the weighted linear combination

$$F(\dot{\mathbf{c}}) = E(\dot{\mathbf{c}}) + \omega \, S(\dot{\mathbf{c}}) \to \mathrm{min}, \quad (13)$$

see (8) and (10), with a certain positive weight ω. This leads to a quadratic objective function of the unknown time derivatives $\dot{\mathbf{c}} = (\dot{c}_i)_{i=1,2,\ldots,n}$. The solution $\dot{\mathbf{c}}(\tau)$ is found by solving a sparse linear system of equations, $\nabla F = 0$.

We then generate the updated control coefficients

$$\mathbf{c}(\tau + \Delta\tau) = \mathbf{c}(\tau) + \dot{\mathbf{c}}\Delta\tau. \quad (14)$$

simply by using an explicit Euler step $\Delta\tau$. The step size is chosen as $\min(1, \{C/v(\mathbf{x}_j,\tau)\}_{j=1,\ldots,N_0})$, where C is a user-defined value. The traveling distance (approximately $\Delta\tau \cdot v(\mathbf{x}_j,\tau)$) of each point $\mathbf{x}_j$ on the T-spline level set is constrained to be (approximately) less than the constant C.

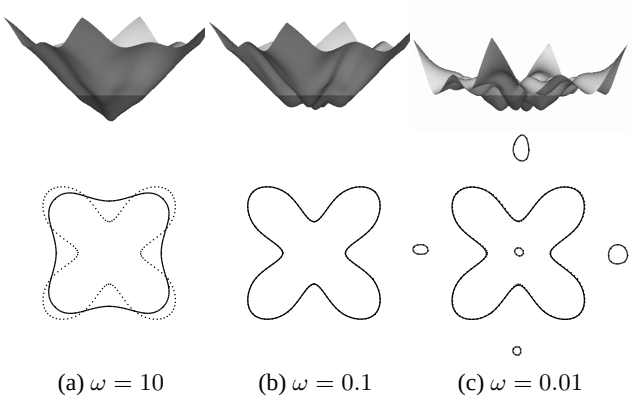
The combination of evolution term E and the signed distance field constraint term S helps to maintain the signed distance property of the level set function during its evolution, without any additional re-initialization steps. Figure 1 illustrates the effects which can be achieved by using various weight values of ω. A large value of the weight leads to a T-spline function which is almost the signed distance function of a circle (left). On the other hand, a very small value produces a T-spline level set with additional branches (right). In between these two extreme situations, a proper choice of the weight gives the desired result (center).

4 Geometry Reconstruction through Evolution of T-spline Level Sets

We describe a unified algorithm for solving two problems: image segmentation (*Problem 1*), shape reconstruction from *unorganized point data* (*Problem 2*). The reconstructed geometry (2D curves or 3D surfaces) may have complex topology, which is unknown a priori. The algorithm takes as input an image data or a set of unorganized points (possibly with noise), and produces a T-spline level set approximating the given image contour or unorganized points with an appropriate number of control coefficients (control points).

4.1 Outline of the Algorithm

The algorithm can be divided into three stages: initialization, evolution, and refinement. Figure 2 shows the flow



(a) $\omega = 10$ (b) $\omega = 0.1$ (c) $\omega = 0.01$

Figure 1. Influence of the weights ω. The figures show the graphs of the T-spline function (top row), the T-spline level set (bottom row, solid) and the target shape (dotted).

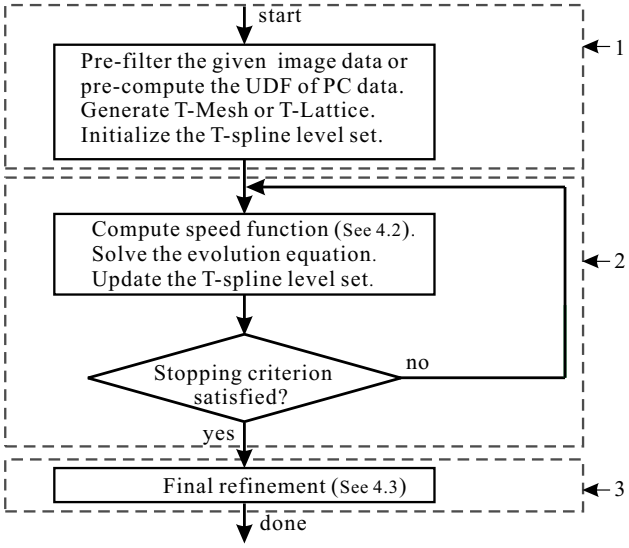


Figure 2. Flow chart of the algorithm.

chart. In the initialization stage, the given image data (Problem 1) is pre-filtered or the *unsigned distance field* (UDF) of the given unorganized points (Problem 2) is pre-computed, e.g., by using the fast marching method [14]. In the 2D case, we use graphics hardware acceleration [7].

The T-mesh (T-lattice) is automatically generated through binary-tree (octree) subdivision of those cells containing data points or high gradient image pixels, in order to adapt it to the input data. Then the T-spline level set is initialized to be a circle-shaped curve, or sphere shaped surface, containing all data points, or the interesting parts of the image.

During the evolution stage, the T-spline level set is evolved towards the desired result step by step, guided by an intelligent data-driven speed function, until some stopping criteria is satisfied.

4.2 Speed Function

For image segmentation (Problem 1), we use a similar speed function as that proposed by Caselles et al. [5],

$$v = g(I)(\gamma + \kappa) - (1 - g(I))(\nabla g(I) \cdot \vec{\mathbf{n}}) \qquad (15)$$

where γ is a constant velocity (which is also known as a *balloon force*), $\kappa = \mathrm{div}(\nabla f/|\nabla f|)$ is the curvature (resp. mean curvature in the surface case) of the level sets of f, and $g(I) = \exp(-\eta|\nabla I|^2)$ is an *edge detector* function, where $\eta > 0$ is a constant.

This speed function can be easily extended for shape recovery from unorganized points (Problem 2):

$$v = e(d)(\gamma + \kappa) - (1 - e(d))(\nabla d \cdot \vec{\mathbf{n}}) \qquad (16)$$

where d is the *unsigned distance* function of the unorganized points, and $e(d) = 1 - \mathrm{e}^{-\eta d^2}$ is another edge detector function. Note that a discretized version of the *unsigned distance* function d is already pre-computed in the initialization stage, thus $d(\mathbf{x})$ (and $\nabla d(\mathbf{x})$) can be efficiently acquired by bi-linear or tri-linear interpolation from the neighboring grid points of $\mathbf{x}$.

In our experiments, all data points are contained in the bounding box ($-1 \le x, y, z \le 1$), and we chose $\eta = 1$.

4.3 Final Refinement

In the case of given unorganized data points (Problem 2), the final refinement is achieved by replacing the evolution term E in (6) with

$$\tilde{E} = \sum_{k=1}^{M} \left(\frac{\partial f(\mathbf{x}_k, \tau)}{\partial \tau} + \widetilde{v}_k |\nabla f(\mathbf{x}_k, \tau)|\right)^2 \to \min, \qquad (17)$$

where $\mathbf{x}_k$ are the closest points on the T-spline level set to the given data points $\mathbf{p}_k$, and $\widetilde{v}_k = (\mathbf{x}_k - \mathbf{p}_k) \cdot \vec{\mathbf{n}_k}$.

Again, $\tilde{E}$ is combined with the signed distance field constraint S, and the updated T-spline level set can be obtained (see Section 3.3). Then the closest points $\mathbf{x}_k$ are recomputed, and Eq. (17) is reconfigured for the next iteration. The above procedure is repeated until the approximation error (maximum distance between the T-spline level set and the data points $\mathbf{p}_k$) cannot be reduced further.

For the given image data (Problem 1), the evolution result can be improved in a similar way. A set of sharp edge points are detected within a narrow band region of the T-spline level set, and then served as the target data points to be approximated by the final refinement.

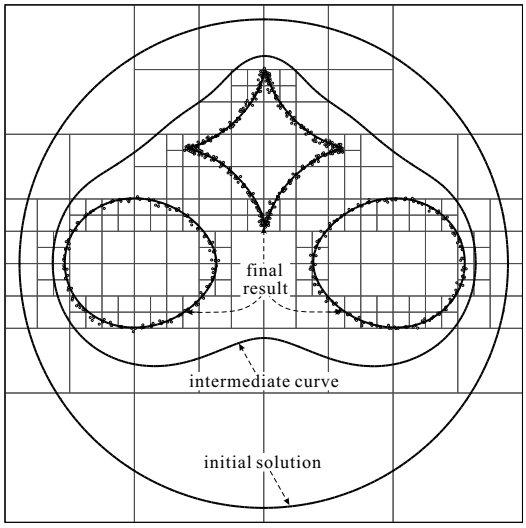


Figure 3. 2D Geometry reconstruction. The data points, the T-mesh, and 3 T-spline level sets are shown.

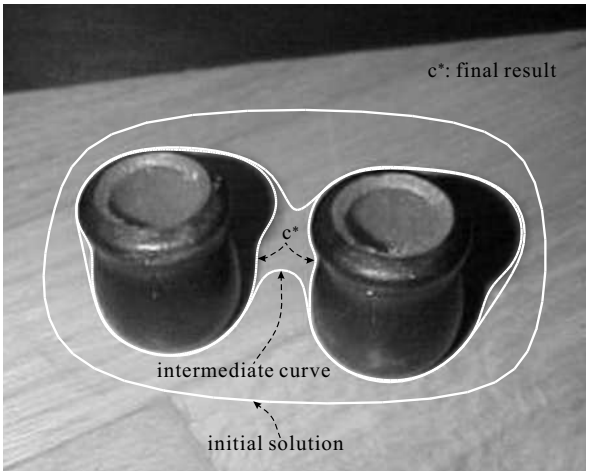


Figure 4. 2D Image segmentation. 3 T-spline level sets (initial solution, intermediate result and final solution) are shown.

5 Experimental results

We present some examples to demonstrate the effectiveness of our method. All the experiments are run on a PC with AMD Opteron(tm) 2.20GHz CPU and 3.25G RAM. All the given image or data points are contained in a square or cubic domain ($-1 \leq x, y, z \leq 1$).

Example 1: 2D geometry reconstruction. In the first example (see Figure 3), the data set consists of 940 points in 2D, and the approximating T-spline level set is described by 272 coefficients. We start with a circumscribed level set and apply the evolution. The level set splits into three components which approximate the given data. The entire computation took about 8 seconds.

Example 2: 2D image segmentation. The second example (Figure 4) demonstrates the use of a T-spline level set for image segmentation. Again, we start with a circumscribed level set and apply the evolution. The level set splits into two components which identify the two objects in the figure, along with the shadows. The entire computation took about 10 seconds.

Example 3: 3D geometry reconstruction. The third example (Figure 5) deals with the reconstruction of 3D objects from unorganized point data. In this simple case, the data are taken from four ellipsoids. The level set evolution starts with a circumscribed sphere, and the result correctly identifies the four components.

6 Conclusion and future work

We have shown how to formulate evolution processes for T-spline level sets that can be used to address problems of shape reconstruction from image data (Problem 1) and from unorganized point clouds (Problem 2). These processes are based on a least–squares approximation of the velocity fields, which are derived from the given data. We also propose the use of an additional *distance field constraint*, which is combined into the evolution equation to avoid time-consuming re-initialization steps.

The T-spline representation of the level set function is sparse and piecewise rational. For both geometry reconstruction and image segmentation, we are able to generate the T-mesh according to the distribution of the points or edges respectively. This means that – in the ideal case – the number of degrees of freedom increases only linearly with the length (area) of the curve (surface) which is to be reconstructed.

Since implicitly defined curves and surfaces cannot be used directly in many applications such as Computer Aided Design, we plan to *couple* the evolution of *T-spline level sets* with *parametric curves and surfaces*. More precisely, the evolving T-spline level set will guide the evolution of the parametric representation. This is expected to lead to approximation algorithms for self–adapting parametric representations, which may automatically determine the correct topology.

Acknowledgment. The authors were supported by the Austrian Science Fund (FWF) through the Joint Research Programme (FSP) S92 “Industrial Geometry”, subproject 2.

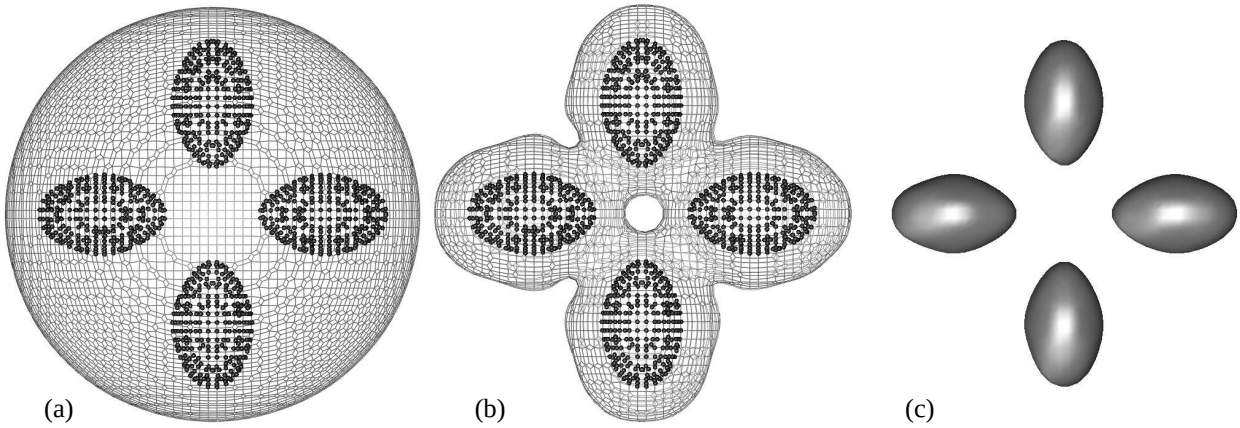


Figure 5. 3D Geometry reconstruction. The figure shows the initial level set (a), an intermediate level set during the evolution (b), and the final result after refinement (c).

References

[1] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Physics*, 148(1):2–22, 1999.

[2] R. Allègre, R. Chaine, and S. Akkouche. Convection-driven dynamic surface reconstruction. In M. Spagnuolo, A. Pasko, and A. Belyaev, editors, *Proc. Shape Modeling International*, pages 33–42, 2005.

[3] J. C. Carr et al. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH'01*, pages 67–76, 2001.

[4] R. Cartwright et al. Web-based shape modeling with HyperFun. *IEEE Computer Graphics and Applications*, 25:60–69, 2005.

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Int. J. Comp. Vision*, 22(1):61–79, 1997.

[6] T. Dokken et al. Intersection algorithms for geometry-based IT applications using approximate algebraic methods, EU project IST 2001–35512. http://www.sintef.no/IST_GAIA, 2002–2005.

[7] K. Hoff et al. Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proc. SIGGRAPH'99*, pages 277–286, 1999.

[8] B. Jüttler and A. Felis. Least–squares fitting of algebraic spline surfaces. *Adv. Comput. Math.*, 17:135–152, 2002.

[9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Int. J. Comp. Vision*, 1(4):321–331, 1988.

[10] C. Li, C. Xu, C. Gui, and M. Fox. Level set evolution without re-initialization: a new variational formulation. In *Proc. Comp. Vision and Pattern Recognition*, volume 1, pages 430–436. IEEE, 2005.

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

[12] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proc. 5th ACM Symposium on Solid Modeling and Applications*, pages 246–257, 1999.

[13] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics*, 22(3):477–484, 2003.

[14] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proc. National Academy of Sciences*, volume 93, pages 1591–1595, 1996.

[15] K. van den Doel and U. Ascher. On level set regularization for highly ill-posed distributed parameter estimation problems. manuscript available at http://www.cs.ubc.ca/˜kvdoel/pubs.html.

[16] Z. W. Yang, J. S. Deng, and F. L. Chen. Fitting unorganized point clouds with active implicit B-spline curves. *The Visual Computer*, 21(8-10):831–839, 2005.

[17] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proc. 1st IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201, Vancouver, 2001.