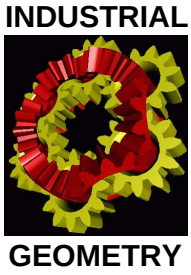
Forschungsschwerpunkt S92

Industrial Geometry

http://www.ig.jku.at



FSP Report No. 1

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

H. Yang, M. Fuchs, B. Jüttler, O. Scherzer

December 2005

FWF
Der Wissenschaftsfonds.

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 [6], for object reconstruction from unorganized points [5, 13, 22] and for improving the robustness of algorithms for computing surface-surface intersections [10]. 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 [6], representations based on radial basis functions [5], spline functions [10, 13], to grid–based discretizations [22].

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. For instance, a family of (closed) parameterized curves $\mathbf{x}_\tau(u)$ may evolve according to

$$\frac{\partial \mathbf{x}_\tau(u)}{\partial \tau} \cdot \vec{\mathbf{n}}_\tau(u) = v(\mathbf{x}_\tau(u), \tau), \tag{1.1}$$

where the parameter τ represents the time, $v(\mathbf{x}, \tau)$ is some (possibly time-dependent) speed function and $\vec{\mathbf{n}}_\tau$ the outer unit normal of the curve $\mathbf{x}_\tau$.

One example of an evolution of this type is used for segmentation. Kass et al. [14] 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. [8] 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 of the type (1.1).

For *implicitly defined curves and surfaces*, one may formulate *evolution processes* as in Eq. (1.1) using the *level set approach* of Osher and Sethian [16]. Assume that an image is given by a map $I : D \to \mathbb{R}$, where D is a two–dimensional domain. Then we can represent any curve $\mathbf{x}$ in I as the zero level-set of a so–called level-set function $f : D \to \mathbb{R}$. The evolution (1.1) can be reformulated as

$$\frac{\partial f(x,y)}{\partial \tau} = -v(x, y; \tau)|\nabla f(x, y)|\,. \tag{1.2}$$

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. [22] present a convection model to compute an implicit surface S that minimizes a global distance function to the input data set. Chaine et al. [2, 9] translates the convection scheme into Computational Geometry terms.

While typical implementations of level set evolutions rely on grid-based discretizations of the domain, this pa-

per proposes to represent the function f by a bivariate or trivariate *T-spline* function (see [17]). 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 provides some background information about T-splines and 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. [17] generalized non-uniform B-spline surfaces to so–called T-splines. After recalling the definition, we introduce implicitly defined T–spline curves and surfaces.

2.1. T-splines

As the most characteristic feature of T-splines, the control grids permit T-junctions. See Figure 1, which shows the pre-image of a T-mesh in (x, y) parameter space. The control grid of a T-spline is called a T-mesh. The pre-image of each edge in a T-mesh is a line segment of constant x or y, which is called an x–edge or a y–edge. If a T-mesh is simply a rectangular grid without T-junctions, the T-spline reduces to a B-spline.

In this paper we restrict our discussion to the cubic case. If no multiple knots are present, then cubic T-splines are C^2. The equation for a cubic T-spline function is

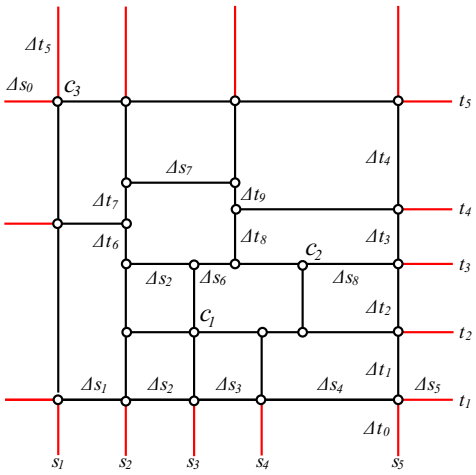
$$f(x,y) = \frac{\sum_{i=1}^{n} c_i B_i(x,y)}{\sum_{i=1}^{n} B_i(x,y)}, \quad (x,y) \in D \qquad (2.1)$$

where the c_i are control points (in our case: coefficients) and n is the number of control points. The basis functions $B_i(x, y)$ are

$$B_i(x,y) = N^3_{i0}(x) N^3_{i0}(y) \qquad (2.2)$$

where $N^3_{i0}(x)$ and $N^3_{i0}(y)$ are the cubic B-splines associated with certain knot vectors

$$[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}] \text{ and } [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}],$$



control point	x–knots y-knots
c_1	$[s_1, s_2, s_3, s_4, s_5 - \Delta s_8]$ $[t_1 - \Delta t_0, t_1, t_2, t_3, t_3 + \Delta t_9]$
c_2	$[s_3, s_3 + \Delta s_6, s_5 - \Delta s_8, s_5, s_5 + \Delta s_5]$ $[t_1, t_2, t_3, t_4, t_5]$
c_3	$[s_1 - \Delta s_0, s_1 - \Delta s_0, s_1, s_2, s_2 + \Delta s_7]$ $[t_1, t_5 - \Delta t_4 + \Delta t_9 - \Delta t_7, t_5, t_5 + \Delta t_5, t_5 + \Delta t_5]$
We use multiple knots at the boundaries, i.e., $\Delta s_0 = \Delta s_5 = \Delta t_0 = \Delta t_5 = 0$.	

Figure 1. Pre-image of a T-mesh

respectively. The knot vectors s_i and t_i of an individual control point c_i associated with $(s_i, t_i) = (s_{i2}, t_{i2})$ are decided by the T-mesh in the following way. Consider a ray $R(\Delta s) = (s_{i2} + \Delta s, t_{i2}), \Delta s > 0$ in the (x, y) parameter space. The knots s_{i3} and s_{i4} are the x coordinates of the first two x–edges intersected by the ray (not including the initial edge of $s = s_{i2}$), see Figure 1. The other knots are found in a similar manner. In order to control the boundary of the domain more conveniently, one usually uses multiple knots at the boundaries.

Once these knot vectors are determined for each basis function, the T-spline is defined by Equation (2.1). The support of a basis function is $D_i = (s_{i0}, s_{i4}) \times t_{i0}, t_{i4})$. The set D is the domain of the T-spline function, $D \subset \{D_1 \cup D_2 \cup ... \cup D_n\}$.

2.2. Implicit T-spline Curves and Surfaces

Let $f(x, y)$ be a bivariate T-spline function defined over some domain D,

$$f(x,y) = \frac{\sum_{i=1}^{n} c_i B_i(x,y)}{\sum_{i=1}^{n} B_i(x,y)}, \quad (x,y) \in D \subset \mathbb{R}^2 \qquad (2.3)$$

with the real coefficients (control points) c_i, $i = 1, 2, ..., n$, where n is the number of control points. The basis functions $B_i(x, y)$ are given in Equation (2.2). The zero set of

the function f is defined by

$$\mathcal{C}(f) = \{\ (x,y) \in D \subset \mathbb{R}^2 \mid f(x,y) = 0\ \}, \tag{2.4}$$

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

The definition of *implicit T-spline curves* in 2D can be easily generalized to *implicit T-spline surfaces* in 3D:

$$\mathcal{S}(f) = \{\ (x,y,z) \in D \subset \mathbb{R}^3 \mid f(x,y,z) = 0\ \}, \tag{2.5}$$

where $f(x,y,z)$ is a trivariate T-spline function,

$$f(x,y,z) = \frac{\sum_{i=1}^{n} c_i B_i(x,y,z)}{\sum_{i=1}^{n} B_i(x,y,z)}, \quad (x,y,z) \in D \subset \mathbb{R}^3. \tag{2.6}$$

The definition of the basis functions in the 3D case naturally generalizes the definition in the plane,

$$B_i(x,y,z) = N_{i0}^3(x) N_{i0}^3(y) N_{i0}^3(z), \tag{2.7}$$

where $N_{i0}^3(x)$, $N_{i0}^3(y)$ and $N_{i0}^3(z)$ are the cubic B-spline basis functions associated with the knot vectors

$$\xi_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}]$$

for $\xi = x, y, z$, respectively. The influence domain of an individual real coefficient c_i is $D_i = (x_{i0}, x_{i4}) \times (y_{i0}, y_{i4}) \times (z_{i0}, z_{i4})$. The knot vectors of c_i are decided by the 3D T-mesh (or T-lattice) in a similar way as that described for 2D parameter space in Section 2.1.

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

$$\mathbf{x} = (x,y) \ \text{ resp. } \ \mathbf{x} = (x,y,z), \tag{2.8}$$

and gather the control coefficients (in a suitable ordering) in a column vector $\mathbf{c}$. The T-spline basis functions form another column vector $\mathbf{b}$,

$$\mathbf{b} = [b_1, b_2, ..., b_n]^\top,$$

and

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

The *T-spline level set* $\Gamma(f)$ is defined as the zero set of the function

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^\top \mathbf{c} \tag{2.9}$$

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.

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

$$f(\mathbf{x}, \tau) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}(\tau), \tag{3.1}$$

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

$$\frac{\partial \mathbf{x}}{\partial \tau} = v(\mathbf{x}, \tau)\vec{\mathbf{n}}, \quad \mathbf{x} \in \Gamma(f), \tag{3.2}$$

where v is a scalar-valued *velocity function* (or *speed function*) along the normal direction $\vec{\mathbf{n}}$ of Γ,

$$\vec{\mathbf{n}} = \frac{\nabla f}{|\nabla f|}. \tag{3.3}$$

During the evolution, the definition of the level sets,

$$f(\mathbf{x}, \tau) \equiv 0, \quad \mathbf{x} \in \Gamma(f), \tag{3.4}$$

implies

$$\frac{\partial f}{\partial \tau} + \nabla f \cdot \frac{\partial \mathbf{x}}{\partial \tau} = 0, \quad \mathbf{x} \in \Gamma(f). \tag{3.5}$$

Combining (3.2), (3.3) and (3.5), we get the evolution equation of T-spline level sets under the normal velocity v,

$$\frac{\partial f}{\partial \tau} = -v(\mathbf{x}, \tau)|\nabla f|, \quad \mathbf{x} \in \Gamma(f). \tag{3.6}$$

In our case, however, f is a linear combination of the time-dependent coefficients $\mathbf{c}$, see (3.1). In order to translate (3.6) 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 is more appropriate, i.e., we replace E_0 with

$$E = \sum_{j=1}^{N_0} \left(\frac{\partial f(\mathbf{x}_j, \tau)}{\partial \tau} + v(\mathbf{x}_j, \tau)\, |\nabla f(\mathbf{x}_j, \tau)|\ \right)^2, \tag{3.7}$$

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. (3.1))

$$\frac{\partial f(\mathbf{x}, \tau)}{\partial \tau} = \mathbf{b}(\mathbf{x})^\top \dot{\mathbf{c}}(\tau), \tag{3.8}$$

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

$$E = \sum_{j=1}^{N_0} (\mathbf{b}(\mathbf{x})^\top \dot{\mathbf{c}}(\tau) + v(\mathbf{x}_j, \tau) \, |\nabla f(\mathbf{x}_j, \tau)| \,)^2. \quad (3.9)$$

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

$$E = \dot{\mathbf{c}}^\top \, Q_E(\mathbf{c}) \, \dot{\mathbf{c}} + \mathbf{l}_E(\mathbf{c})^\top \, \dot{\mathbf{c}} + k_E(\mathbf{c}). \quad (3.10)$$

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 (3.9). 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. Reinitialization

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. However, during the evolution, f will drift away from this signed distance property. Although the definition of f off its zero level set can be arbitrary in the continuous formulation, flat and/or steep regions that develop in the level set function can dramatically decrease the accuracy of the computed solution [1].

This motivates the use of *level set reinitialization* which restores the *signed distance* property. It can be done either by applying a Fast Marching technique [18] or by considering the steady state solution to the PDE

$$\frac{\partial f}{\partial \tau} + \text{sign}(f_0)(|\nabla f| - 1) = 0, \quad (3.11)$$

where f_0 is the current level set function to be reinitialized [19].

However, the re-initialization procedure is usually relatively expensive (especially in the 3D case) and has to be applied frequently. Moreover, in some cases – when a large time step is used – the level set function may greatly deviate from a signed distance function after only one or several iteration steps, which will cause difficulties.

In our case, where the level set is a (piecewise) algebraic curve or surface, there is an additional difficulty. It is well known that the implicit form of a rational parametric curve (or surface) segment may have singularities, even in cases where the parametric representation is regular. For instance, the cubic Bézier curve in Fig. 2 is regular, while its implicit form has a double point in the region of interest. Consequently, if the target shape is such a (regular) cubic curve, its level set representation will have singularities, unless the original evolution equation – which pulls the level set towards the target shape – is modified. Such a modification is described below.

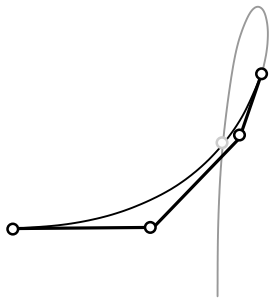


Figure 2. Planar cubic with double point.

3.3. Distance Field Constraint

We will avoid the use of re-initialization by introducing an additional *distance field constraint*. Recently, similar techniques have been proposed in the literature [15, 20].

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

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

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,

$$S = \frac{A(D)}{N} \sum_{j=1}^{N} (\frac{\partial |\nabla f(\mathbf{y}_j, \tau)|}{\partial \tau} + |\nabla f(\mathbf{y}_j, \tau)| - 1)^2, \quad (3.12)$$

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

As an obvious generalization, one may modify S_0 by including an additional positive weight function under the integral. In (3.12), this can be taken into account by sampling the points $\mathbf{y}_j$ according to the density specified by the weight function.

In our case, the level set function f has the form (3.1), hence the time derivative of the gradient length

$$\frac{\partial |\nabla f(\mathbf{y}_j, \tau)|}{\partial \tau} = \frac{2 \nabla f(\mathbf{y}_j, \tau)}{|\nabla f(\mathbf{y}_j, \tau)|} \, (\nabla \mathbf{b}(\mathbf{y}_j)^\top \dot{\mathbf{c}}(\tau)) \quad (3.13)$$

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

By combining (3.12) and (3.13), we may represent the signed distance field constraint term as a non–negative definite quadratic function of the derivatives $\dot{\mathbf{c}}$,

$$S = \dot{\mathbf{c}}^{\top} Q_S(\mathbf{c})\, \dot{\mathbf{c}} + \mathbf{l}_S(\mathbf{c})^{\top}\, \dot{\mathbf{c}} + k_S(\mathbf{c}). \tag{3.14}$$

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 (3.12) and (3.13). 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).

3.4. 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

$$F(\dot{\mathbf{c}}) = E(\dot{\mathbf{c}}) + \omega_1\, S(\dot{\mathbf{c}}) \to \min, \tag{3.15}$$

see (3.9) and (3.12), with a certain positive weight ω_1. 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,

$$\frac{\partial}{\partial \dot{c}_i} F(\dot{\mathbf{c}}) = 0, \quad i = 1, \ldots, n. \tag{3.16}$$

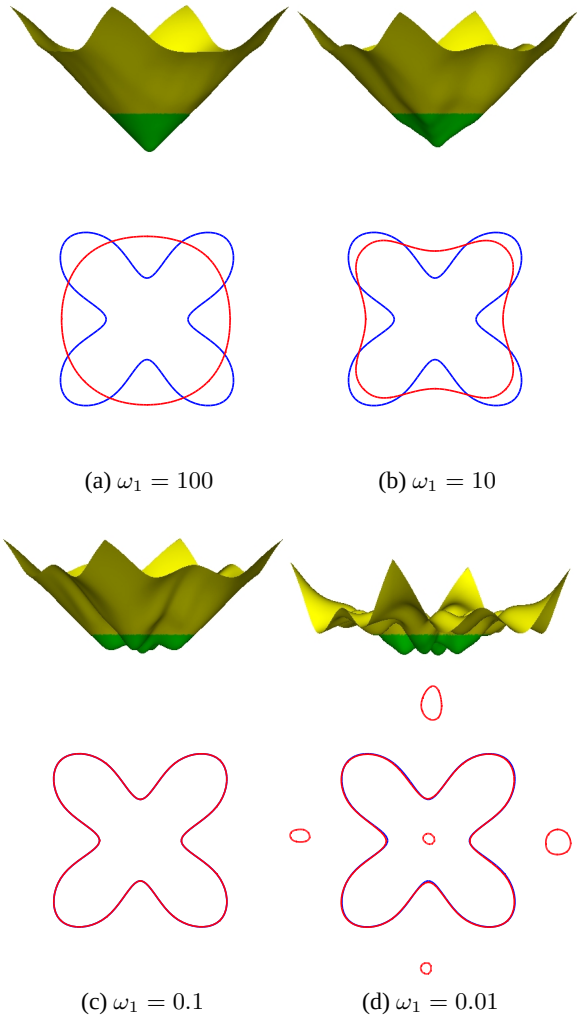
Very efficient algorithms for solving systems of this type exist [4].

We then generate the updated control coefficients

$$\mathbf{c}(\tau + \Delta\tau) = \mathbf{c}(\tau) + \dot{\mathbf{c}}\Delta\tau. \tag{3.17}$$

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 3 illustrates the effects which can be achieved by using various weight values of ω_1. A large value of the weight (top, let) leads to a T-spline function which is almost the signed distance function of a circle (i.e., its graph is a circular cone). On the other hand, a very small value produces a T-spline level set with additional branches (bottom, right). In between these two extreme situations, a proper choice of the weight gives the desired result (bottom, left).



(a) $\omega_1 = 100$ (b) $\omega_1 = 10$

(c) $\omega_1 = 0.1$ (d) $\omega_1 = 0.01$

Figure 3. Influence of the weights ω_1. The figures show the graphs of the T-spline function (green), the T-spline level set (red) and the target shape (blue).

4. Geometry Reconstruction through Evolution of T-spline Level Sets

In this section, we give an unified algorithm for solving two problems through evolution of T-spline level sets. For *Problem 1*, image segmentation, we assume that *image data* (i.e., a scalar field of, e.g., grey values) is given. On the other hand, *Problem 2* is shape reconstruction from *unorganized point data*.

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 4 shows the flow chart of the presented algorithm.

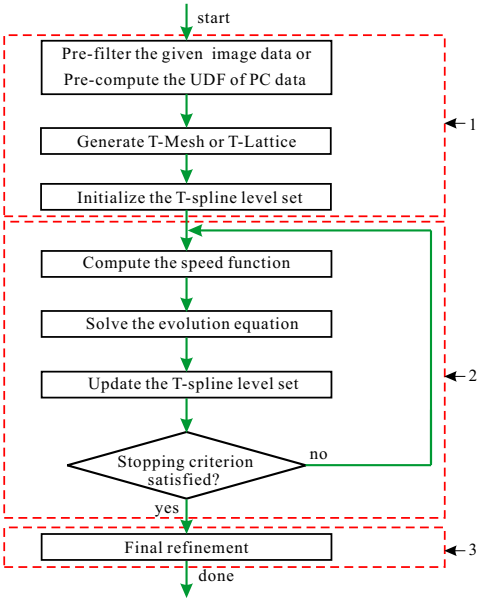


Figure 4. Algorithm for geometry reconstruction using T-spline level sets.

In the initialization stage, the given image data (Problem 1) is pre-filtered or the *unsigned distance field* of the given unorganized points (Problem 2) is pre-computed, e.g., by using the fast marching method [18]. In the 2D case, we use graphics hardware acceleration [12].

The T-mesh (or T-lattice) is generated according to the given data (image or points), see below. The T-spline level set is then initialized to be a circle-shaped curve, or a 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. Finally, for the last refinement stage, the result of T-spline level set is further improved by solving a non-linear least squares problem.

4.2. T-mesh / T-lattice Generation

In the case of given 2D (3D) point cloud data (Problem 2), the T-mesh (T-lattice) can be automatically generated through binary-tree (octree) subdivision (cf. Fig. 5), as follows.

1. Set the initial T-mesh (T-lattice) to be an axis-aligned bounding box containing all the data points.
2. For each cell containing more than n_0 data points (n_0 is a user-specified constant value), subdivide it by applying the binary-tree (octree) subdivision.
3. Repeat step (2) until a user-specified threshold (e.g., a maximum level of subdivision) is reached.

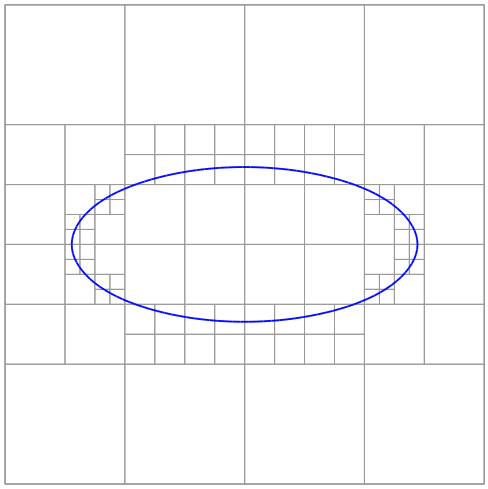


Figure 5. T-mesh generated by binary-tree subdivision.

For the given 2D (3D) image data (Problem 1), the T-mesh (T-lattice) can be generated in a similar way. The only difference is in step (2): instead of checking the number of data points inside the cell D_i, we check the function value

$$h(D_i) = \int_{D_i} |\nabla I| dD \tag{4.1}$$

for the given image intensity field I. If $h(D_i) \geq \widetilde{h}$ for some user-specified constant $\widetilde{h}$, then subdivide D_i into smaller cells. Figure 5 gives an example for T-mesh generation.

The theoretic motivation for this is the following. The integral (4.1) corresponds to the total length of all edges in the intersection of I and D_i. The above condition means that the total length of the edges within each tile of the subdivision is bounded by $\widetilde{h}$. Thus, we use a finer T-mesh in areas of high boundary variation. This makes sense as we are interested in the reconstruction of the boundary.

4.3. Speed Function

The speed function v (Ref. Equation (3.6)) plays a key role in the algorithm, since it decides the evolution process as well as the final result of T-spline level set. Caselles et al. [8] propose a geodesic active contour model based on the following evolution equation

$$\frac{\partial f}{\partial \tau} = g(I)(\gamma + \kappa)|\nabla f| + \nabla f \cdot \nabla g(I), \qquad (4.2)$$

which means that the level sets move according to

$$\frac{\partial \mathbf{x}_\tau}{\partial \tau} \cdot \vec{\mathbf{n}}_\tau = g(I)(\gamma + \kappa) - (\nabla g(I) \cdot \vec{\mathbf{n}}), \qquad (4.3)$$

where γ is a constant velocity (which is also known as a *balloon force*), κ is the curvature on the level sets of f,

$$\kappa = \mathrm{div}(\frac{\nabla f}{|\nabla f|}), \qquad (4.4)$$

and $g(I)$ is some *edge detector* function. In [7], Caselles et al. propose to use

$$g(I) = \frac{1}{1 + |\nabla I|^p} \quad (p = 1 \text{ or } 2). \qquad (4.5)$$

The original motivation for the use of the equations (4.2) and (4.3), respectively, stems from "Snakes" or active contour models as proposed by Kass et al. [14]. Caselles et al. [8] showed that this approach is connected to computing geodesics or minimal distance curves in a Riemannian space, where the metric in this space is determined by the edge detector function of the image data. This model is called *Geodesic Active Contours*. They propose to solve the resulting minimization problem using the steepest-descent method and derive the curve evolution (4.3) with $\gamma = 0$. This evolution process deforms an initial curve $\mathbf{x}_0$ towards the object boundary. The final boundary is given by the steady state solution of (4.3). As mentioned before, the balloon force γ does not naturally appear when deriving (4.3) from the Snake model. In [8] the authors propose setting $\gamma > 0$ in order to increase the speed of the evolution.

For the evolution of our T-spline level sets for image segmentation (Problem 1), we use a similar speed function as that in (4.3) with a slight modification,

$$v = g(I)(\gamma + \kappa) - (1 - g(I))(\nabla g(I) \cdot \vec{\mathbf{n}}) \qquad (4.6)$$

Generalizing (4.5), Caselles et al. [8] mention that any strictly decreasing function $g : [0, \infty] \rightarrow \mathbb{R}$ such that $g(r) \rightarrow 0$ as $r \rightarrow \infty$ qualifies as an edge detector. In order to get satisfactory results for more complex topologies, we choose

$$g(I) = \mathrm{e}^{-\eta |\nabla I|^2}, \qquad (4.7)$$

where $\eta > 0$ is a constant parameter. The edge detector is more sensitive to high gradients in the image, if we choose η to be large.

One can see that the speed (4.6) function is a linear combination of two parts: the first part makes the level set smooth by curvature flow, while the second part attracts the level set to the detected edges (even if some noise or small gaps may exist). The new term of $(1-g(I))$ is to weaken the influence of the second part when the level set is far from the edges ($0 \ll g(I) < 1$), which is a natural choice in practical applications. When the level set is close to the edges ($g(I) \simeq 0$), the second part again plays the leading role in the evolution.

There are two important reasons for us to choose the speed function in (4.6). Firstly, the smoothness of the moving level set is provided by using the curvature flow, instead of using a tension term (e.g., thin plate regularization term), where the latter would easily flatten the implicit field and cause a trivial result. Secondly, the convergence result is insensitive to the choice of balloon force γ. Actually, it is possible to choose $\gamma = 0$, and the model still converges (in a slower motion) [8].

Furthermore, this speed function can be easily extended, in order to deal with recovering shape from unorganized points (Problem 2):

$$v = e(d)(\gamma + \kappa) - (1 - e(d))(\nabla d \cdot \vec{\mathbf{n}}) \qquad (4.8)$$

where d is the *unsigned distance* function of the unorganized points, $e(d)$ is another edge detector function

$$e(d) = 1 - \mathrm{e}^{-\eta d^2}. \qquad (4.9)$$

Again, η is a pre-defined, and its value is affected by the size of the data range. In our experimental setting, since all data points are contained in the same bounding box ($-1 \leq x, y, z \leq 1$), then we can usually set $\eta = 1$. 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 $\mathbf{x}$.

4.4. Final Refinement

The T-spline level set continues evolving until a maximum number of iteration steps is reached or some user-specified stopping criterion is satisfied, e.g., the maximum distance between the zero level set and the given data points is smaller than a certain threshold value.

In the case of given unorganized data points (Problem 2), we now choose the points $\mathbf{x}_j$ in (3.7) to be the closest points of the given data on the T-spline level set. In addition, we derive the velocities (4.8) no longer from the (approximation to the) unsigned distance field. Instead, we derive them

from the distance to the closest points. Consequently, we replace the evolution term E with

$$\tilde{E} = \sum_{k=1}^{M} \left(\frac{\partial f(\mathbf{x}_k, \tau)}{\partial \tau} + \widetilde{v}_k |\nabla f(\mathbf{x}_k, \tau)| \right)^2 \rightarrow \min, \quad (4.10)$$

where

$$\widetilde{v}_k = (\mathbf{x}_k - \mathbf{p}_k) \cdot \vec{\mathbf{n}}_k. \quad (4.11)$$

Again, this term is combined with the signed distance field constraint S.

The updated T-spline level set can be obtained in the same way as that for the evolution process (Ref Section 3.4). Then the closest points $\mathbf{x}_k$ are recomputed, and the Equation (4.10) 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 also can be further improved in a similar way. One may first detect a set of sharp edge points within a narrow band region of the T-spline level set, using some edge detector function as shown in (4.5). Those detected edge points then serve as the target data points to be approximated by the T-spline level set, in order to guide the final refinement of the segmentation result.

Remark 1 This technique is closely related to the method of minimization of the normal distance [3], which has recently been called tangent distance minimization (TDM) [21], for parametric curves and surfaces. Indeed, the evolution defined by (4.10) with step-size $\Delta\tau = 1$ can be shown to be equivalent to these earlier methods. As a major difference, our method is using implicitly defined T-spline level sets and is able to deal with complex topological changes in a natural way.

5. Experimental results

In this section, 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 6), the data set consists of 940 points in the plane, and the approximating T-spline level set (a curve) is described by 272 coefficients. We start with a level set that represents a circumscribed circle 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 7) demonstrates the use of a T-spline level set for image segmentation. Again, we start with a circumscribed level set and apply the evolution which is driven by the velocities derived from the data. 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 in this section (Figure 8) deals with the reconstruction of 3D objects from unorganized point data. In this simple case, the data are taken from four ellipsoids. Similar to the 2D case, we start with a circumscribed sphere and apply the evolution. The T-spline level set correctly identifies the four components.

6. Discussion

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. In this section we discuss the potential benefits of using T–spline level sets.

1. All models for geometry reconstruction and image segmentation have to ensure that the resulting curve satisfies some regularity conditions. Without any regularization, the solution would be very sensitive to noisy image data and essentially become ill-posed. E.g., in the snake model, Kass et al. [14] minimize the first and second derivative of the curve to ensure regularity. Caselles et al. [7], [8] additionally pre-filter the image data before starting their evolution. The same holds for Frick and Scherzer [11].

   T-spline functions are piecewise rational, and the T-spline level sets are algebraic spline curves and surfaces. This naturally implies certain regularity properties of the function f and, as a consequence, of its zero level set. Thus we are able to get satisfying reconstruction results – even for noisy data – without any preprocessing.

2. Frequently, the the evolution equation (1.2) is numerically solved on a per-pixel bases, i.e. the number of degrees of freedom equals the number of pixels in the picture. The T-spline representation of the level set function is sparse and drastically reduces the number of degrees of freedom. In addition, for both geometry reconstruction and segmentation, we are able to generate the T-mesh according to the distribution of the edges or points respectively. This means that – in

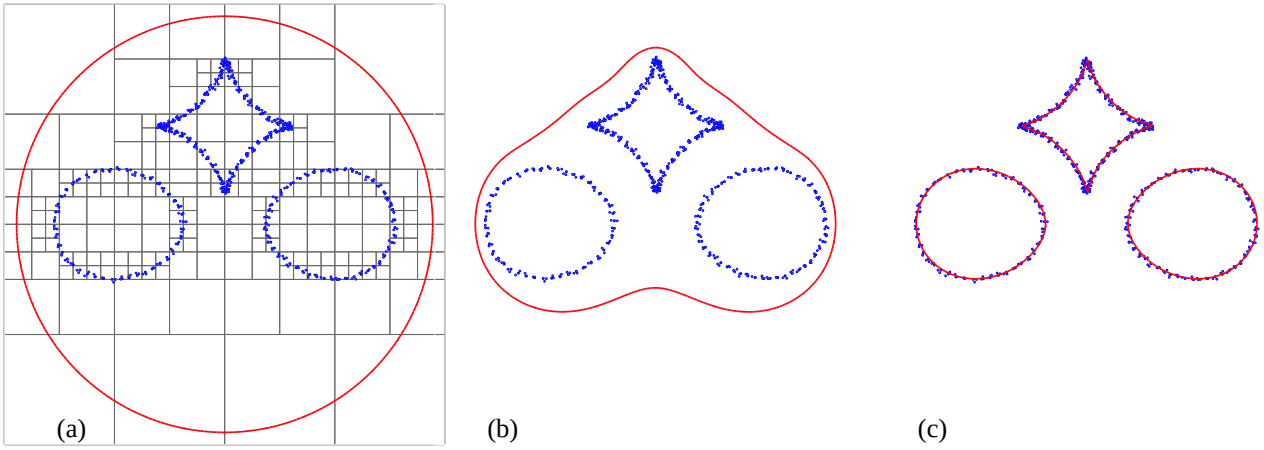


Figure 6. Geometry reconstruction from 2D unorganized points using a a T-spline level set. The figure shows the initial level set with the generated T-mesh (a), an intermediate level set during the evolution (b), and the final result after refinement (c).

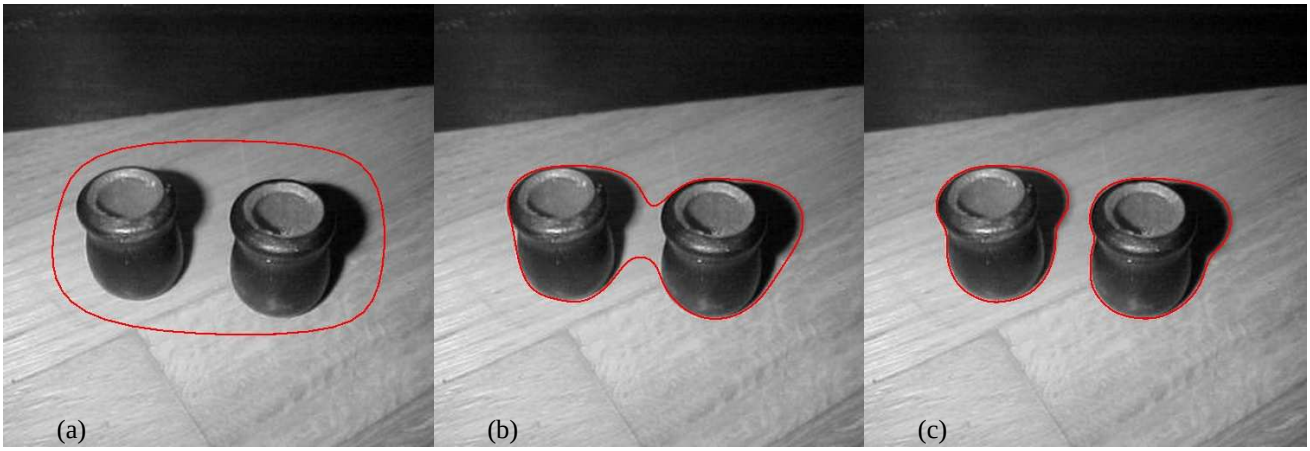


Figure 7. Image segmentation using a T-spline level set. The figure shows the initial level set (a), an intermediate level set during the evolution (b), and the final result after refinement (c).

the ideal case – the number of degrees of freedom increases only linearly with the length of the curve which is to be reconstructed.

As a straightforward modification of our algorithm, one might adapt the structure of the T-mesh / T-lattice to the data during the evolution, in order to introduce additional degrees of freedom, where needed.

3. Note that the first two properties complement one another. If the T-mesh is refined (thus the number of degrees of freedom increased), the regularization property of the T-splines decreases. i.e. in the ideal case the loss of accuracy by using a coarser than pixel-sized grid is actually required to regularize the evolution problem. That means that by using the correctly refined T-mesh, we hope to reconstruct exactly as much level of detail as the noise level of the data allows for.
4. Frequent re-initialization steps are often needed for existing level set methods, since otherwise numerical instabilities and additional branches may happen during the level set evolution. Instead of using these potentially time-consuming re-initialization steps, we propose the use of an additional *distance field constraint*, which is combined into the evolution equation to intrinsically maintain the distance field property of the level set function. Our experimental results show that the *distance field constraint* can greatly increase the

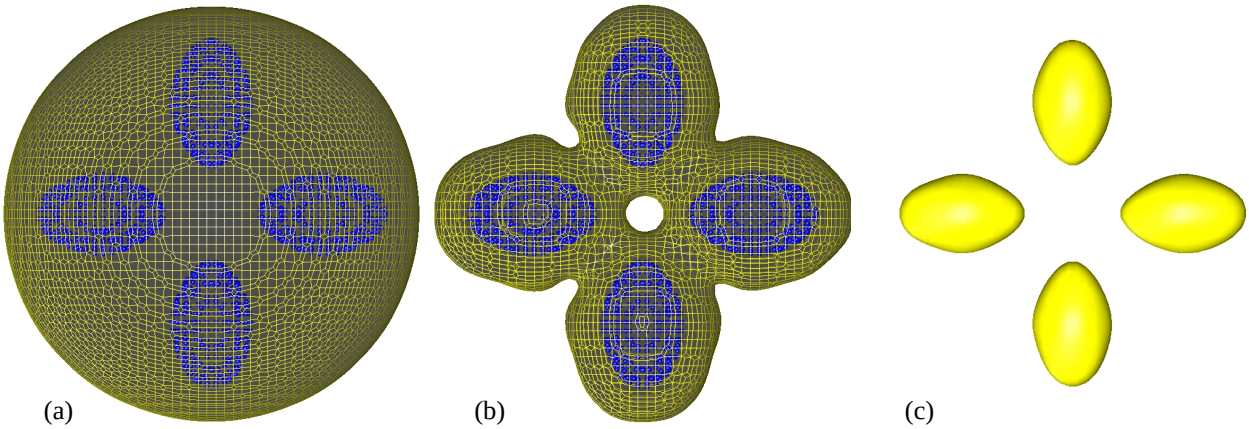


Figure 8. Geometry reconstruction from 3D unorganized points using a T-spline level set. The figure shows the initial level set (a), an intermediate level set during the evolution (b), and the final result after refinement (c).

stability of the evolution process, and thus improve the computed solution.

7. Future work

Frick and Scherzer [11] implicitly computed the curve evolution proposed by Caselles et al. [7, 8] by solving a variational problem in every time step. This involves heavy computational effort. It is possible to do the same implicit computation with T-spline functions. In this case the advantage of less degrees of freedom would have even more impact on computation times. This may be a subject of further investigation.

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.

A first example is shown in Figure 9. In this example, we generate a T-spline level set (a curve) which approximates 360 data points. Again we start with a circumscribed circle and apply the distance-driven evolution process.

While evolving the level set, we simultaneously evolve a closed parametric B-spline curve, which is made to follow the implicitly defined T-spline curve. The control polygon of the curve is shown; the curve itself cannot be distinguished from the zero contour of the T-spline function.

The parametric curve is synchronized with the T-spline level set, and its topology is adapted whenever the topology of the T-spline level set changes. As the result, we obtain both an implicit and a parametric representation of the same object.

We plan to extend this to the 3D case. On the one hand, this will facilitate the computation of the evolving T-spline level set, since the generation of sample points (for the numerical integration of the normal velocity contribution to the objective function) becomes simpler. On the other hand, the T-spline level set has some difficulties to capture fine details of an object. The parametric representation will be used to capture these details in the final refinement stage. The implicit representation serves to guide the parametric representation to develop 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.

References

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

[2] R. Allègre, R. Chaine, and S. Akkouche. Convection-driven dynamic surface reconstruction. In *Proc. Shape Modeling International*, pages 33–42, Cambridge, MA, USA, June 2005.

[3] A. Blake and M. Isard. *Active Contours*. Springer, New York, 2000.

[4] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. In R. Martin et al., editors, *The Mathematics of Surfaces XI*, volume 3604 of *LNCS*, pages 62–83. Springer, 2005.

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

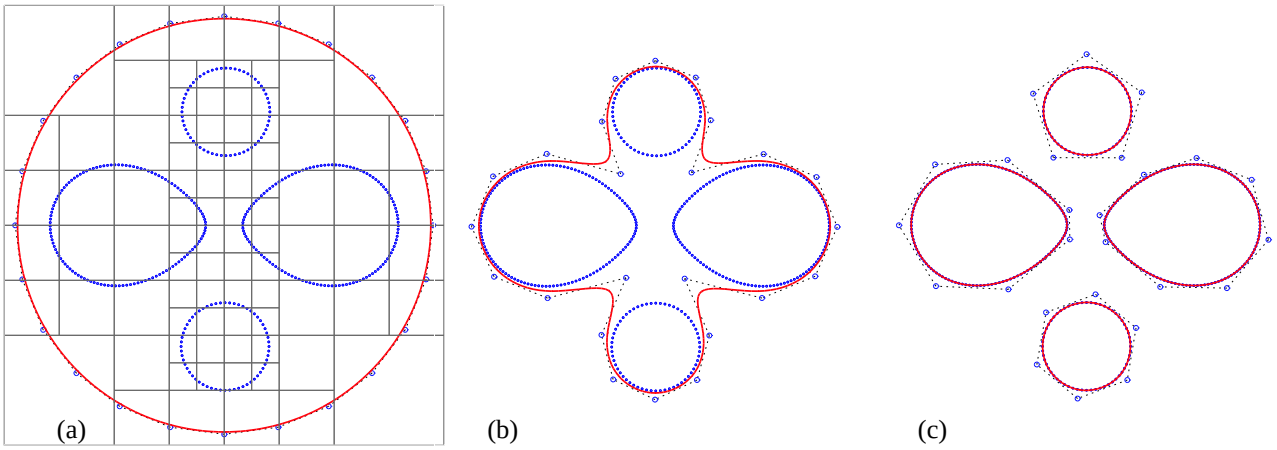


Figure 9. Geometry reconstruction from 2D unorganized points by coupling the evolution of T-spline level sets with parametric B-spline curves. The figure shows the initial B-spline curve with its control points (a), the intermediate B-spline curve during the evolution (b), and the final result (c).

[6] R. Cartwright, V. Adzhiev, A. Pasko, Y. Goto, and T.L. Kunii. Web-based shape modeling with HyperFun. *IEEE Computer Graphics and Applications*, 25:60–69, 2005.

[7] V. Caselles, F. Catté, T. Coll, and C. Sbert. A geometric model for active contours in image processing. *Numerische Mathematik*, 66:1–31, 1993.

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

[9] R. Chaine. A geometric convection approach of 3-d reconstruction. In *Proc. Symposium on Geometry Processing*, pages 218–229, 2003.

[10] 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.

[11] K. Frick and O. Scherzer. Application of non-convex BV regularization for image segmentation. In *Proc. International conference on PDE-Based Image Processing and Related Inverse Problems*. CMA, University of Oslo, to appear.

[12] K. E. Hoff, T. Culver, J. Keyser, Ming Lin, and D. Manocha. Fast computation of generalized Voronoi diagrams using graphics hardware. *SIGGRAPH'99*, pages 277–286, 1999.

[13] B. Jüttler and A. Felis. Least–squares fitting of algebraic spline surfaces. *Advances in Computational Mathematics*, 17:135–152, 2002.

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

[15] Chunming Li, Chenyang Xu, Changfeng Gui, and M. D. Fox. Level set evolution without re-initialization: a new variational formulation. In *Proc. Computer Vision and Pattern Recognition*, volume 1, pages 430–436. IEEE Computer Society, 2005.

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

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

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

[19] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. of Computational Physics*, 1(114):146–159, 1994.

[20] 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.

[21] W. Wang, H. Pottmann, and Y. Liu. Fitting b-spline curves to point clouds by squared distance minimization. *ACM Transactions on Graphics*, page accepted, 2005.

[22] 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.