Particle–based T-Spline Level Set Evolution for 3D object reconstruction with Range and Volume Constraints

Robert Feichtinger
Johannes Kepler University Linz, Austria
robert.feichtinger@jku.at

Huaiping Yang
Johannes Kepler University Linz, Austria
yang.huaiping@jku.at

Bert Jüttler
Johannes Kepler University Linz, Austria
bert.juettler@jku.at

ABSTRACT

We consider an evolution process for implicitly defined surfaces, which are represented as the zero–levels of T-spline functions. The paper presents two novel contributions. First, we will use particles on the evolving surface in order to discretize the evolution equation. In particular we describe criteria for local and global resampling, which are needed in order to maintain a sufficiently uniform distribution of the particles. Second, we discuss volume and range constraints which can be added to the framework. More precisely, it is possible to specify a fixed volume (volume constraint) or to define a region which should or should not be contained in the final object (range constraint). These constraints can also be regarded as a priori knowledge of the data.

Keywords: T-splines, particles, range constraint, volume constraints

1 INTRODUCTION

This paper addresses the problem of evolution of implicitly defined surfaces, with applications to geometry reconstruction from discrete data. The evolution process generates time-dependent families of surfaces converging to the target shape defined by the data, which are guided by an implicit velocity field in the direction of surface normals. For example this kind of evolution is used for segmentation in image processing. Kass et al. [16] proposed "snakes" or active contours for boundary detection, which is based on deforming an initial contour towards the boundary to be detected. Caselles et al. [5] proved that the classic active contour model in 2D is equivalent to finding a geodesic curve in a Riemannian space with a metric derived from the image content. For implicitly defined surfaces, one may formulate evolution processes by using the level set approach of Osher and Sethian [19]. Similar evolution processes have also been used for geometry reconstruction from unorganized data points [18, 28].

Surface reconstruction from scattered data points (possibly noisy and incomplete) is an important problem in geometric modeling. Depending on the type of the application, different representations have been used, such as triangular meshes [7, 17], subdivision surfaces [6, 22], parametric spline surfaces [8], discretized level sets [18], scalar spline functions [14, 23, 28], radial basis functions [4], and point set surfaces [1, 2, 13, 20, 21]. In comparison with parametric representations, implicit representations offer advantages such as the non-existence of the parametrization problem, repairing capabilities of incomplete data, and simple operations of shape editing [29]. As a major advantage in evolution-based approaches, where the topology is not known a priori, the implicit representation intrinsically adapts to topological changes during the evolution.

In many applications of surface reconstruction, there may be a priori knowledge concerning the geometric properties of the object to be reconstructed. In particular, this a priori knowledge could be formulated by certain shape constraints to be combined into the evolution equation of the surface. These additional constraints may help to obtain a desired reconstruction result, especially when the given data contains holes. For example, a suitable volume constraint can be used to stop the evolving surface from entering the holes left by the data. The idea of volume constraint has also been used in shape deformation to achieve better quality and realism, e.g. the swirling-sweepers proposed by Angelidis et al. [3] and the divergence-free vector-field based method by Funck et al. [25, 26].

Our method is different from those in that we use zero contour scalar functions instead of meshes to represent surfaces, where the volume constraint is formulated with the help of the time derivatives of the coefficients. Another constraint to be addressed is the so-called range constraint [9, 10], which allows us to specify regions lying inside or outside of the reconstructed surface.

Particles have been used for different applications in mathematics and computer graphics. One can use them for numerically solving differential equations or to sample and to control implicitly defined surfaces [27].

WSCG'2008, February 4 – February 7, 2008 .

They have also found attention for simulation of fur, grass and other fuzzy textures, also special effects like fire and explosions are created by particles in this applications collision detection is important [15].

In this paper we use particles in order to efficiently deal with the generation of sample points on evolving surfaces. Special attention is paid to local and global resampling, which is needed in order to maintain a sufficiently uniform distribution of the sample points. In addition we show how to apply constraints during the evolution process. These constraints represent a priori knowledge of the data. The user can specify a fixed volume (volume constraint) or a region which should not lie inside the final object (range constraint).

The paper is organized as follows. In the next section we start with a description of the T-spline level set evolution. Afterwards we go into more detail for the particle generation and handling of particle resampling. In section 4 we treat range and volume constraints. Finally we conclude with some examples.

2 EVOLUTION

First we recall the idea of the evolution of an implicitly defined surface. Such a surface is also denoted as an active surface. In particular we consider implicitly defined surfaces represented as the zero–level set of a T-spline function.

2.1 Problem specification

Throughout the paper we assume that we have some given data, which describes one or more objects in 3D. This data can be given in various forms, e.g. as a triangular mesh, an implicitly or parametrically defined surface or an unorganized point cloud. Here we will focus on point clouds (e.g. data from a 3D scanner). The given data will be referred to as the "target".

In order to detect the topology of the target and to reconstruct it, we consider an evolution process which drives the surface towards the target. More precisely we will use a time dependent family of surfaces $C = C_t$. In our case each surface from this family will be defined as the zero–level set of a scalar field, whose coefficients depend on the time variable τ. During the evolution we will modify these coefficients such that the surface moves towards the given data, see Fig. 1. In particular we want the zero–level set to move in the direction of its normals in dependence of a speed function v.

2.2 Speed functions

As mentioned before the evolution of the active surface will be guided by a speed (or velocity) function v. This function depends on the target, the implicitly defined surface and some geometric properties (curvature κ and normals $\vec{\mathbf{n}}$) of the active surface.

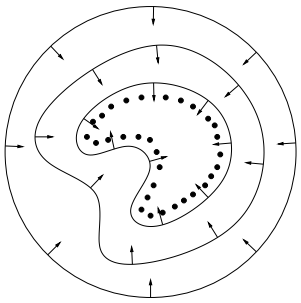


Figure 1: 3 time steps of an active curve moving towards a point cloud data

We use the following speed function for point cloud data

$$v = e(d)\,(\lambda + \kappa) - (1 - e(d))\,(\vec{\mathbf{n}}^T \nabla d), \qquad (1)$$

where e is called the edge detector function,

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

In this function $\vec{\mathbf{n}}$ and κ are geometric quantities derived from the surface, while λ is a constant velocity (also known as the balloon force). η is a pre-described constant which depends on the range of the data. For additional information regarding the choice of these constants we refer to the extended version of [28]. The last variable d is the unsigned distance function which depends on the target.

Both the unsigned distance field and the edge detector functions will be pre–computed. We use graphics hardware acceleration [12] to determine the unsigned distance function $d(\mathbf{x})$. The gradient $\nabla d(\mathbf{x})$ can be efficiently acquired by trilinear interpolation of the neighbouring grid points.

2.3 Evolution of T-spline level sets

We use the zero–level set of a T-spline function (see [24]) to represent the implicitly defined surface. Such a function has the form

$$f(\mathbf{x}, \tau) = \sum_{i=1}^{n} T_i(\mathbf{x})\, c_i(\tau) \;\; \mathbf{x} = (x_1, x_2, x_3) \in \Omega \subset \mathbb{R}^3, \quad (3)$$

where T_i are the trivariate T-spline basis functions and the real coefficients c_i depend on the time variable τ. The axis–aligned bounding box $\Omega = [-1, 1]^3$ contains the region of interest. The given data is scaled such that it lies inside Ω. The basis functions

$$T_i(\mathbf{x}) = \frac{B^3_{r_i}(x_1) B^3_{s_i}(x_2) B^3_{t_i}(x_3)}{\sum_{i=1}^{n} B^3_{r_i}(x_1) B^3_{s_i}(x_2) B^3_{t_i}(x_3)}$$

are rational functions which are defined with cubic B-splines over certain knot vectors $r_i = (r_{i0}, r_{i1}, r_{i2}, r_{i3}, r_{i4})$, s_i and t_i, which are determined with the help of the so–called T-spline

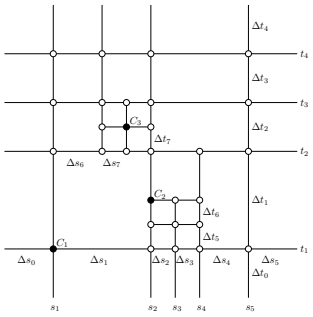


Figure 2: A T-spline grid. We use 4–fold knots at boundaries.

control points	x-knots y-knots
C_1	$[s_1-\Delta s_0, s_1-\Delta s_0, s_1, s_2, s_3]$ $[t_1-\Delta t_0, t_1-\Delta t_0, t_1, t_2, t_3]$
C_2	$[s_1-\Delta s_0, s_1, s_2, s_3, s_4]$ $[t_1, t_1+\Delta t_5, t_1+\Delta t_5+\Delta t_6, t_2, t_2+\Delta t_7]$
C_3	$[s_1, s_1+\Delta s_6, s_1+\Delta s_6+\Delta s_7, s_2, s_5]$ $[t_1, t_2, t_2+\Delta t_7, t_3, t_4]$

Figure 3: The knot vectors for some selected control points.

grid. This is a generalization of the knot vectors of tensor–product splines. See Fig. 2 and Fig. 3 for an illustration where we use a 2D graphic which is more convenient to illustrate the idea of T-splines. See [24] for more information.

As the main advantage of using T-splines, they support T-junctions in the grid. Therefore we can refine the grid locally which is not the case for tensor product B-splines. If there are no T-junctions in the grid, then the T-spline simplifies to a tensor–product spline.

The zero–level set of such a T-spline function f defines a time–dependent surface.

$$\Gamma(f,\tau) = \{\, \mathbf{x} \in \Omega \subset \mathbb{R}^3 \mid f(\mathbf{x},\tau) = 0 \,\}. \tag{4}$$

To describe the evolution we use the approach presented in [28]. We consider $\Gamma(f,\tau)$ the zero–level set of the T-spline f at a given time τ. It will be subject to the evolution process

$$\frac{\partial \mathbf{x}}{\partial \tau} = v\vec{\mathbf{n}}, \;\; \mathbf{x} \in \Gamma(f,\tau). \tag{5}$$

Here v is the speed function which was introduced in Section 2.2. The value of v depends on the point $\mathbf{x} \in \Gamma$ and on the first and second derivative of the T-spline f at $\mathbf{x}$. $\vec{\mathbf{n}}$ is the unit normal direction given by $\vec{\mathbf{n}} = \nabla f(\mathbf{x},\tau)/|\nabla f(\mathbf{x},\tau)|$. Since

$$f(\mathbf{x},\tau) = 0, \;\; \mathbf{x} \in \Gamma(f,\tau), \tag{6}$$

has to hold during the evolution, it follows that

$$\dot{f} + \nabla f(\mathbf{x},\tau)\,\frac{\partial \mathbf{x}}{\partial \tau} = 0, \;\; \mathbf{x} \in \Gamma(f,\tau), \tag{7}$$

where $\dot{f} = \frac{\partial f(\mathbf{x},\tau)}{\partial \tau}$. Combining (5) and (7) we get the following equation for the evolution of the zero–level set

$$\frac{\partial f(\mathbf{x},\tau)}{\partial \tau} = -v|\nabla f(\mathbf{x},\tau)|, \;\; \mathbf{x} \in \Gamma(f,\tau). \tag{8}$$

To translate this equation into an equation for the time dependent coefficients $c_i(\tau)$ we use a least-squares approach

$$E_0(\dot{\mathbf{c}}) = \int_{\mathbf{x}\in\Gamma(f)} \left(\dot{f}(\mathbf{x},\tau) + v\,|\nabla f(\mathbf{x},\tau)|\right)^2 \mathrm{d}A \to \min. \tag{9}$$

where A represents the area element of the T-spline level set, and $\mathbf{c} = (c_1,\ldots,c_n)$. In order to solve this equation we use numerical integration to get a discretized version

$$E(\dot{\mathbf{c}}) = \sum_{j=1}^{N_0} \left(\dot{f}(\mathbf{x}_j,\tau) + v(\mathbf{x}_j,\tau)\,\left|\nabla f(\mathbf{x}_j,\tau)\right|\right)^2 \to \min. \tag{10}$$

Here $\mathbf{x}_j,\ j = 1\ldots N_0$ are sample points on the active surface. In the present work we will use particles, instead of considering uniformly distributed sample points as in [28]. This sampling method will be described in the following section.

Before we continue with the description of the particle generation we summarize the algorithm used for the evolution process.

Algorithm 1

1. *Initialization:* Pre-compute the evolution speed function and choose the initial position of the implicitly defined surface.
2. *Initial particles:* Compute a set of particles on the initial active surface.
3. *Evolution:* Apply one time step of the evolution to the implicitly defined surface.
4. *Projection:* Project the particles in normal direction onto the new implicitly defined surface.
5. *Consistency check:* Check if a redistribution of the particles is necessary.
6. *Termination:* Check whether the stopping criterion is satisfied. Continue with step 3 (no) or finish (yes).

We choose the initial implicitly defined surface as a ball (Step 1) such that all data points lie inside. We use uniformly distributed points on this ball (Step 2) as a starting set of particles. To generate this set we chose a random point close to the ball and use the method described in section 3.4.

For step 3, we have to solve the Eqn. (10). We first linearize the quadratic problem for the $\dot{c}_i$ and then use an explicit Euler step to compute the new coefficients c_i.

Note that we use the distance field constraint to avoid the reinitialization of the T-spline function, see [28].

3 PARTICLES

For the evolution of an implicit level set surface we need sample points on the current zero–level set. It is necessary to have efficient techniques for the computation and the update of the sample points in each time step.

In our previous work [28], these sample points are uniformly distributed and computed by using a fine grid. So one has to check in which grid cells the zero–level set lies after each step, which is rather time consuming.

We will use particles which do not need to be recomputed each time step. Instead they move with the active surface. This movement is achieved by using a projection along the normal direction. During the evolution it can happen that the distance between two neighbouring particles gets too big. Therefore we need to define a criterion when a redistribution is necessary. In most situations the redistribution is done locally instead of recomputing all particles.

Since we also want to gather information about the number of components in the target, we need a way to generate one set of particles per component. This is achieved by splitting the set of particles according to the current state of the implicitly defined surface. So when the topology of the surface changes during the evolution, this change should be reflected in the particles.

3.1 Particles data structure

The data structure used for the particles is similar to the data structure for a triangular mesh. For each particle we store the coordinates and a list pointing to the neighbouring particles, but we do not use the face information. These particles are then used as sample points for the T-spline level set evolution.

To compute this triangulation we use Hartmann's algorithm for marching triangulation [11]. This algorithm allows us to compute a triangulation either for a whole object or for a bounded domain on a surface.

To apply Hartmann's algorithm we have to provide the possibility to compute or at least to estimate the normal direction at any point of the surface, which is of course possible for the zero–level set of a T-spline function f. The unit normals are given by $\vec{\mathbf{n}} = \nabla f/|\nabla f|$.

As an initial set we take the vertices of an uniformly distributed triangulation of the sphere, since the initial level set is approximately a sphere. The distance between these points is denoted by ρ. We refer to ρ as the feature size.

3.2 Criteria for particle resampling

During the evolution we use two criteria to decide if a resampling is necessary. Afterwards we check if the resampling can be done locally or globally. See section 3.3 and 3.4 for the resampling methods.

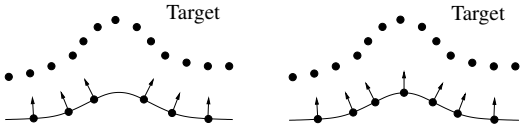


Figure 4: Target with a concave feature and an active curve with particles

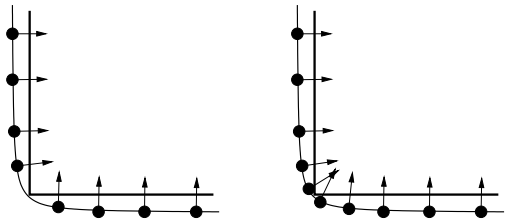


Figure 5: Target with a sharp feature

The marching triangulation can be applied either to a whole object (global resampling) or to a bounded domain (local resampling).

We use the following criteria:

1. The distance between two neighbouring sample points P_i and P_j becomes too large. This means we get areas with too few sample points. As a criterion we use
$$|P_i - P_j| > 2\rho \tag{11}$$
For example this occurs if the target contains a concave region. We will demonstrate this in a 2D example, see Fig. 4. On the left-hand side, the distance between the two particles in the center is growing. On the right-hand side, the result after resampling is shown. The arrows indicate the normal direction at the particles.

2. The normals of neighbouring points vary too much. Here the criterion can be written as
$$\mathbf{n}_i \cdot \mathbf{n}_j < \varepsilon \tag{12}$$
where $\mathbf{n}_i$ is the unit normal of the implicitly defined surface at the sample point P_i. ε is chosen such that too big angles are detected e.g. to filter out angles greater than 80°. This criterion helps to produce additional sample points at sharp corners and therefore to increase the accuracy. For an example see Fig. 5. Again the left-hand side shows the particles before resampling and the normal directions. The right-hand side displays the result after resampling with a modified step size.

If one of the two criteria is satisfied we first use a local resampling. In case that criterion 2 holds we additionally modify the step size for the marching triangulation, to ensure that we produce additional sample points in the region around the critical points. This gives us a new set of particles S_1. Afterwards we check if a global resampling is necessary as well. Therefore we take the old particles $\widehat{S}$ and test if they are contained in the new generated set of particles or at least close enough to this set,

$$\forall_{P_i \in \widehat{S}} \exists_{P_j \in S_1} \left| P_i - P_j \right| \leq \rho. \tag{13}$$

The global resampling is used only when the implicitly defined surface has split during the evolution step, where the condition (13) is not fulfilled. Most of the time we will just use the local method. Note that holes which occur during the evolution in the implicitly defined surface are also handled by the local method.

We summarize the algorithm for particle resampling.

Algorithm 2

1. Apply the two criteria to test if resampling is necessary. If none of them is satisfied, then stop.
2. Apply local resampling.
3. Test if the old particles are close enough to the newly generated ones. If one of them is too far away, then apply global resampling; a splitting event may have happened.

3.3 Local resampling

We start with two sample points P_i and P_j, which are neighbours in the old set of particles. These points have been detected by one of the two previously described criteria. First we create a local region Ω. This region contains P_i, P_j and the neighbouring sample points (Note that this information is stored in our data structure). Then we check, if additional sample points lie inside this region. If this is the case add these points. Afterwards we apply the marching triangulation to Ω.

Note that the two criteria (11) and (12) might be satisfied for more than one pair of sample points. If we get more than one pair of sample points we create the local region Ω for one pair. For the remaining pairs we check whether they are separated from already generated regions or not. In the first case we create a new region $\widehat{\Omega}$ in the second case we have to extend the already existing region. These two cases are demonstrated in the following example.

Example 1 We use data taken from a torus. The top left picture in Fig. 6 shows the sample points on the initial surface. The top right picture visualizes the corresponding T-spline grid. The second row shows the first resampling, which involves two disjoint regions.

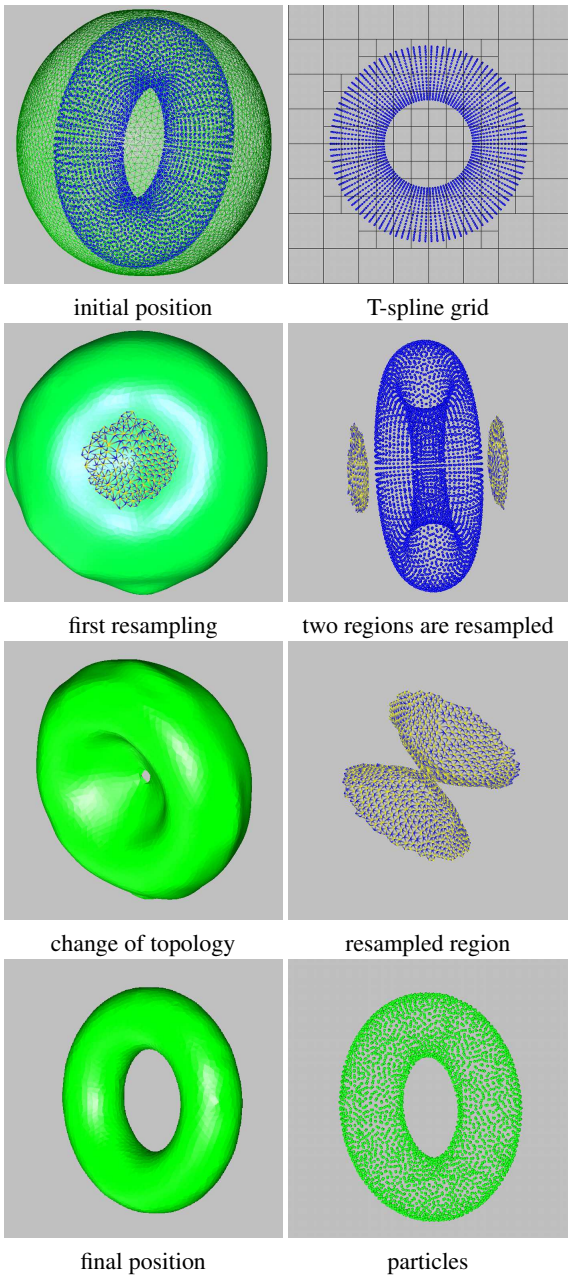


Figure 6: Evolution with local resampling

In the next row a change of topology takes place; a larger region has to be resampled in order to reflect this change. The pictures in the bottom row show the final result (left) and the final set of particles (right).

3.4 Global resampling

In this case we start with all particles $\widehat{S}' \subseteq \widehat{S}$ which do not satisfy the condition (13). We pick one of these points $P \in \widehat{S}'$ as the starting point for a global marching triangulation. This gives us a set S_2. Afterwards we check if the remaining particles in $S' \setminus \{P\}$ lie close to S_2. Otherwise we repeat the procedure. So we end up with one set of particles for each part of the implicitly defined surface at the current stage.

In practice we will not check for all sample points in $\widehat{S}$ instead we randomly select points such that $1/10$th of the points is tested. This is used to speed up the process.

4 CONSTRAINTS

In this section we will present two constraints which can be added to our evolution–based frame work. The first constraint C_R that we will present allows us to define regions where we force the active surface to lie inside or outside this region. This constraint will lead again to a quadratic minimization problem in $\dot{\mathbf{c}}$. So it has the same form as our original problem $E(\dot{\mathbf{c}})$, see Eqn. (10).

With help of the second constraint C_V we can specify a volume which the active surface has to enclose. This gives us a linear constraint which can be solved by using Lagrangian multipliers.

4.1 Range Constraints

This constraint allows us to specify regions such that the surface either contains this region or does not pass trough it. Since the zero–level set of our T-spline function has the property that

$$\begin{array}{rcll} f(\mathbf{x}) & > & 0 & \mathbf{x} \text{ outside } \Gamma \\ f(\mathbf{x}) & = & 0 & \mathbf{x} \in \Gamma \\ f(\mathbf{x}) & < & 0 & \mathbf{x} \text{ inside } \Gamma \end{array} \tag{14}$$

it divides the domain in two parts. (Note that one of these parts could be empty, but that would not make much sense in our framework.) Considering a set of points $\{\mathbf{x}_i\}_{i=1\ldots N_0}$ which should lie inside the zero–level set, we have to ensure that

$$f(\mathbf{x}_i) \leq 0. \tag{15}$$

In the case that $f(\mathbf{x}) > 0$ for some $\mathbf{x} \in \{\mathbf{x}_i\}$ the function value has to be pushed downward at this point. This is achieved by forcing $\dot{f}(\mathbf{x}) < 0$. Therefore we propose to add the following therm to our framework

$$C_R(\dot{\mathbf{c}}) = \sum_{j=1}^{N_0} \left(\dot{f}(\mathbf{x}_j,\tau) + f(\mathbf{x}_j,\tau) + \delta\right)^2 \alpha_\varepsilon(f(\mathbf{x}_j,\tau)) \tag{16}$$

where δ is an user–defined constant. For our examples we use $\delta = 0.2$. This constant controls how steep the T-spline becomes. The'activator' function α_ε controls the influence of the term $C_R(\dot{\mathbf{c}})$,

$$\alpha_\varepsilon(f) = \begin{cases} 1 & f > -\varepsilon \\ 0 & f < -2\varepsilon \\ C^2 - \text{blend} & \text{in between} \end{cases} \tag{17}$$

where ε is an user–defined positive constant (e.g., the feature size ρ can again be used). The optimization problem

$$\widehat{F}(\dot{\mathbf{c}}) = E(\dot{\mathbf{c}}) + \omega_c r\, C_R(\dot{\mathbf{c}}) \rightarrow \min. \tag{18}$$

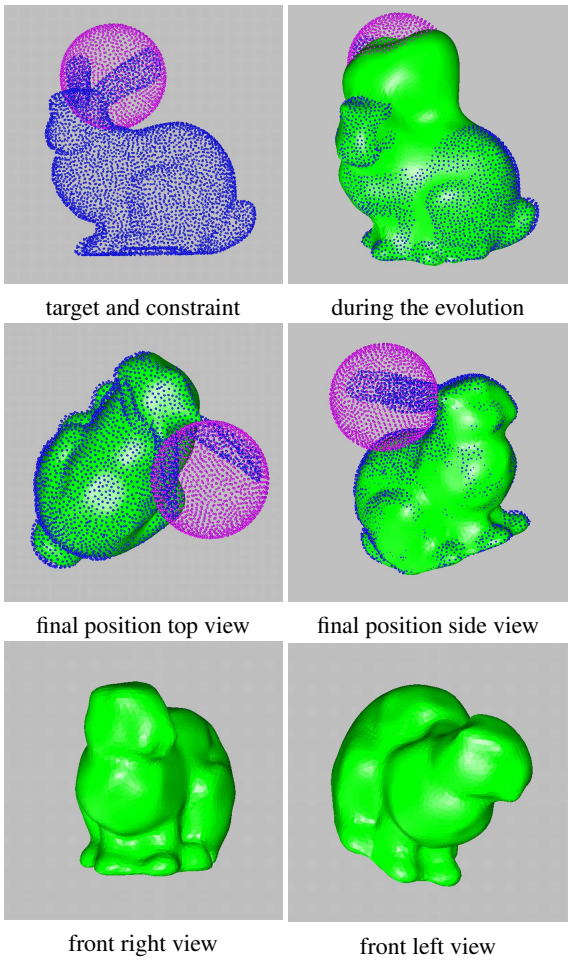


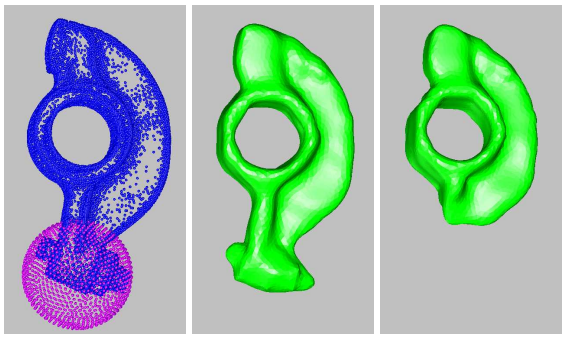
Figure 7: Evolution with range constraints

leads again to a sparse linear system of equations with a symmetric positive definite matrix, which can be dealt with efficiently.

Note that we can treat the case, where a given set of points $\{\mathbf{y}_i\}_{i=1\ldots N_1}$ should lie outside of our active surface, in a similar way.

We can also use this method for producing offsets of the target. Therefore we use all target points as sample points for the constraint. In this case δ defines the offset distance.

Example 2 Range constraints can be used to define regions which should not lie inside the target. In this example we will use a sphere to cut away the ears of the bunny model. The top left picture in Fig. 7 shows the target points and the constraint set. As the initial implicitly defined surface we take a sphere, this is not displayed. The top right picture visualizes the T-spline zero–level set after some time steps. The middle row shows two views on the final position. The ears have been cut off while the rest is recovered. In the bottom row, the final mesh is visualized from two different view points.



target and constraint | result without constraint | result with constraint

Figure 8: Rocker arm reconstruction

Example 3 The left picture in Fig. 8 shows the rocker arm data and a constraint region (ball). The data consists of 10044 points, and we used 1423 T-spline coefficients to reconstruct it. In the middle the result without using the constraint is shown, 4330 particles are used. The right picture visualizes the result using the constraint to cut away the lower part of the rocker arm, therefore 3622 particles are needed.

4.2 Volume Constraints

The volume constraint allows the surface to maintain a specified volume change during the evolution. Suppose $V(\tau)$ is the specified volume function with respect to the time τ, then the volume constraint can be represented as

$$\int_{\Gamma} v_n(\mathbf{x},\tau)dA = \dot{V}(\tau), \tag{19}$$

where A is the area element of the surface Γ, and

$$v_n(\mathbf{x},\tau) = -\frac{\dot{\mathbf{f}}(\mathbf{x},\tau)}{|\nabla \mathbf{f}(\mathbf{x},\tau)|}$$

is the normal velocity of the T-spline level set.

The volume function V can be an arbitrary function of τ, as long as the time derivative $\dot{V}(\tau)$ is well defined. The volume constraint defined in (19) is a volume-increase, volume-preserving or volume-decrease constraint, when $\dot{V}(\tau) > 0$, $\dot{V}(\tau) = 0$ or $\dot{V}(\tau) < 0$, respectively.

The volume constraint (19) is linear in the time derivatives of the T-spline control coefficients. Again we use numerical integration to obtain a linear constraint, which is to be considered along with the quadratic objective function. This leads to a quadratic optimization problem with linear constraints, which is solved using Lagrangian multipliers.

Example 4 The volume constraint is very helpful when dealing with noisy data containing holes. One may wish to specify the volume of the target object. By defining an appropriate function $V(\tau)$ such that $V(0) = V_0$ and $V(\tau) \to V_\infty$ as $\tau \to \infty$, the volume of the surface (initially equal to V_0) will converge to the desired value V_∞. Fig. 9 shows an example to demonstrate how this constraint works. The given data points are sampled from a cube after removing the top face. The volume of the cube equals 0.5^3. The figure shows the different results by specifying different values of the target volume V_∞. The original shape of the cube is recovered when $V_\infty = 0.125$.

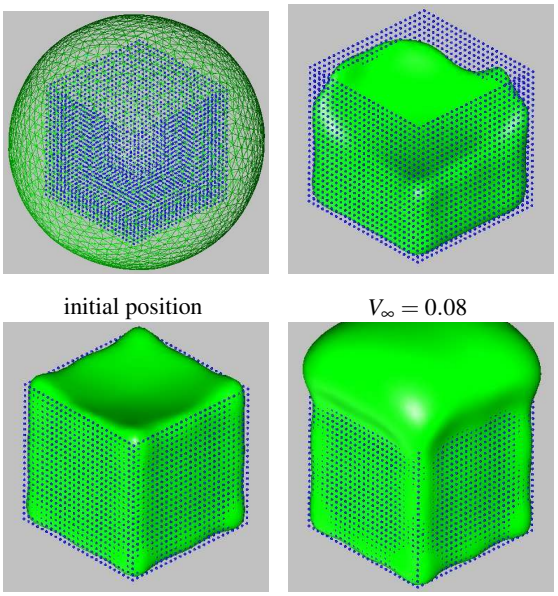


Figure 9: Evolution with volume constraints

5 CONCLUDING REMARKS

We presented a particle-based approach to T-spline level set evolution. The set of particles is updated during the evolution, in order to avoid the frequent recomputation in each time step. In addition we can apply range and volume constraints to the evolving surface. The range constraints can be formulated as linear inequalities, which have been dealt with by a penalty method. The volume constraints lead to linear conditions, which can be incorporated directly into the framework.

In order to capture finer details of the target by T-spline level sets, a large number of T-spline coefficients is needed, slowing down the computations. Instead, one should better use a post–processing step, which is based on a displacement map and bilateral filtering, see [29]. That paper also presents computing times for the reconstruction of realistic 3D models via T-spline Level sets.

Acknowledgments.

The financial support of the Austrian Science fund through the Joint Research Program S92 'Industrial Geometry', subproject 2, and by the European Union through the Marie Curie IIF Fellowship for Huaiping Yang (project 22073 ISIS) is gratefully acknowledged.

REFERENCES

[1] M. Alexa and A. Adamson. Interpolatory point set surfaces - Convexity and Hermite data. *ACM Trans. Graph.* accepted.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, 2001.

[3] A. Angelidis, M.-P. Cani, G. Wyvill, and S. King. Swirling-sweepers: constant volume modeling. *Graphical Models (Special issue on PG'04)*, 68(4):324–332, 2006.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proc. SIGGRAPH'01*, pages 67–76, 2001.

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

[6] K.-S. D. Cheng, W. Wang, H. Qin, K.-Y. K. Wong, H. Yang, and Y. Liu. Fitting subdivision surfaces to unorganized point data using SDM. In *Pacific Conference on Computer Graphics and Applications 2004*, pages 16–24, 2004.

[7] T. K. Dey and S. Goswami. Tight cocone: a watertight surface reconstructor. In *Shape Modeling International*, pages 127–134, 2003.

[8] M. Eck and H. Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Proc. SIGGRAPH'96*, pages 325–334, 1996.

[9] R. Feichtinger, H. Yang, M. Fuchs, B. Jüttler, and O. Scherzer. Dual evolution of planar parametric spline curves and T-spline level sets. *Computer–Aided Design*, 2007. in press.

[10] S. Flöry and M. Hofer. Constrained curve fitting on manifolds. *Computer–Aided Design*, 2007. in press.

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

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

[13] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, and W. Strasser. Bayesian point cloud reconstruction. In *Proc. Eurographics '06*, 2006.

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

[15] E.-A. Karabassi, G. Papaioannou, T. Theoharis, and A. Boehm. Intersection test for collision detection in particle systems. *J. Graph. Tools*, 4(1):25–37, 1999.

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

[17] Y. Ohtake, A. Belyaev, and H.-P. Seidel. A composite approach to meshing scattered data. *Graph. Models*, 68(3):255–267, 2006.

[18] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, New York, 2002.

[19] 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.

[20] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *Proc. SIGGRAPH'03*, pages 641–650, 2003.

[21] M. Pauly, L. P. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Trans. Graph.*, 25(2):177–193, 2006.

[22] H. Qin, C. Mandal, and B. C. Vemuri. Dynamic catmull-clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, 1998.

[23] 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.

[24] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *Proc. SIGGRAPH '03*, pages 477–484, 2003.

[25] W. v. Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. In *Proc. SIGGRAPH'06*, pages 1118–1125, 2006.

[26] W. v. Funck, H. Theisel, and H.-P. Seidel. Explicit control of vector field based shape deformations. In *Pacific Conference on Computer Graphics and Applications 2007*, pages 291–300, 2007.

[27] A. P. Witkin and P. S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. SIGGRAPH '94*, pages 269–277, 1994.

[28] 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. Extended version available at http://www.ig.jku.at.

[29] H. Yang and B. Jüttler. Meshing non-uniformly sampled and incomplete data based on displaced T-spline level sets. In *Shape Modeling International*, pages 251–260. IEEE Press, 2007.