The Visual Computer manuscript No.
(will be inserted by the editor)

Huaiping Yang · Bert Jüttler

Evolution of T-spline Level Sets for Meshing Non-uniformly Sampled and Incomplete Data

Abstract Given a large set of unorganized point sample data, we propose a new framework for computing a triangular mesh representing an approximating piecewise smooth surface. The data may be non–uniformly distributed, noisy, and they may contain holes. This framework is based on the combination of two types of surface representations: triangular meshes, and T-spline level sets, which are implicit surfaces defined by refinable spline functions allowing T-junctions. Our method contains three main steps. Firstly, we construct an implicit representation of a smooth (C^2 in our case) surface, by using an evolution process of T-spline level sets, such that the implicit surface captures the topology and outline of the object to be reconstructed. The initial mesh with high quality is obtained through the marching triangulation of the implicit surface. Secondly, we project each data point to the initial mesh, and get a scalar displacement field. Detailed features will be captured by the displaced mesh. Finally, we present an additional evolution process, which combines data-driven velocities and feature-preserving bilateral filters, in order to reproduce sharp features. We also show that various shape constraints, such as distance field constraints, range constraints and volume constraints can be naturally added to our framework, which is helpful to obtain a desired reconstruction result, especially when the given data contains noise and inaccuracies.

Keywords Mesh reconstruction · Point cloud · Displacement maps · T-spline · Level sets

1 Introduction

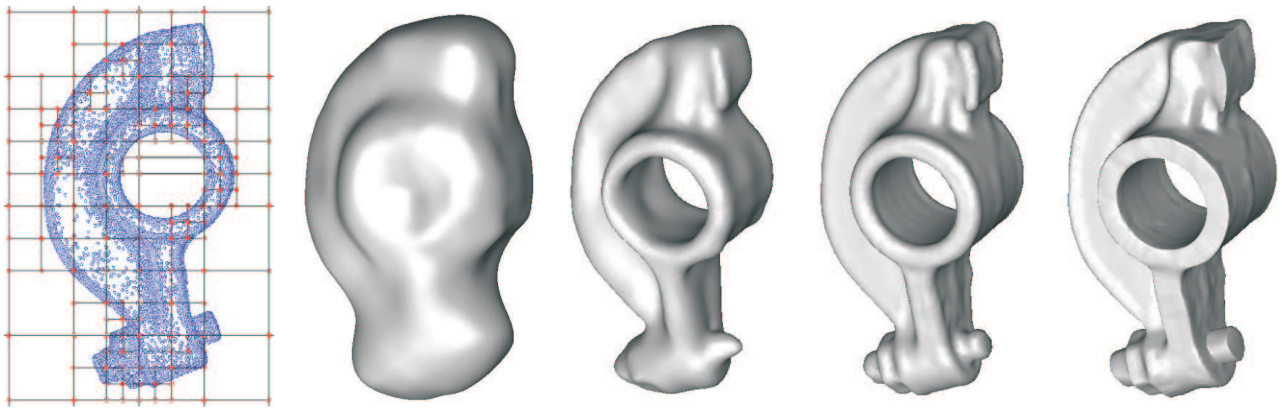
We consider the problem of surface reconstruction and approximation from unorganized data points. Due to the widespread use of 3D scanning devices for shape acquisition, this problem has an increasing number of applications in computer graphics, computer aided design, computer vision and image processing. Depending on the area of the application, the reconstructed surface may have a explicit representation (e.g. meshes) or an implicit representation (e.g. level sets). Among the various approaches, these two representations may complement each other. On the one hand, the implicit representations [51] offer advantages such as the non-existence of the parametrization problem, repairing capabilities of incomplete data and simple operations of shape editing, but they are hard to model sharp features [39]. On the other hand, the explicit representations can easily handle sharp features, but have difficulties when processing topology changes.

H. Yang, B. Jüttler
Institute of Applied Geometry, Johannes Kepler University
4040 Linz, Austria
E-mail: yang.huaiping@jku.at, bert.juettler@jku.at

1.1 Our Work

We propose a hybrid model for surface reconstruction by combining two types of representations: an implicit T-spline level set and a mesh. Given a set of unorganized and noisy data points without normals as input, we want to reconstruct a mesh surface which approximates the data. We develop a three-phase algorithm (cf. Fig. 1) to perform this reconstruction:

1. *Initial mesh generation* (Fig. 1 (c)). In the first phase, we use an evolution process (Fig. 1 (a) and (b)) to create an implicit representation, which is defined as the zero level set of a C^2 T-spline scalar function. The obtained T-spline level set (with correct topology) is to serve as a smooth base surface S_0 for the displacement mapping. A high-quality initial mesh (with accurate normals) is generated from the implicit function by using the marching triangulation [26] method.
2. *Displacement mapping* (Fig. 1 (d)). In the second phase, we produce a smooth scalar displacement field, which is computed by projecting data points to the initial mesh. Small geometric features are then constructed by the displacement mapping of the mesh



(a) (b) (c) (d) (e)

Fig. 1 Mesh reconstruction of the Rocker-Arm model. The figure shows the data points and the T-mesh in (a), an intermediate T-spline level set during the evolution in (b), the final T-spline level set (the initial mesh) in (c), the displaced mesh in (d), and the final mesh with sharp features in (e).

along the normal direction, which is guided by the smooth gradient vector field of the implicit function.

3. *Recovering sharp features* (Fig. 1 (e)). In the third phase, we use an additional evolution process, which combines data-driven velocities and feature-preserving bilateral filters, in order to better represent sharp features.

This paper is an extension to our work [55] presented at Shape Modeling International 2007. In this extended version, we concentrate more on the process generating the base surface, i.e., we show how to formulate the evolution of T-spline level sets. More specifically, we discuss how to combine different shape constraints, such as distance field constraints, range constraints and volume constraints, into the framework such that a more robust and effective evolution process can be established that produces the desired result. The distance field constraints help us to avoid additional branches and singularities of the level sets, without having to use costly re-initialization steps. The range constraints allow us to specify regions lying inside or outside of the reconstructed surface. A suitable volume constraint can be used to stop the level sets from entering the holes in the data. We note that these constraints have been used in our previous work [18] in 2D for dual evolution of planar B-spline curves and T-spline level sets.

Our method combines two types of representations: the implicit T-spline level set and the mesh. This combination strategy makes our method benefit from the advantages of both representations. On the one hand, the evolution of T-spline level sets is able to capture the complex topology of noisy data. On the other hand, the mesh representation helps to produce detailed and sharp features. Compared with other existing approaches for similar purposes, our method also shows the following advantages:

1) The use of two complementary representations can improve and speed up some geometric computations in the algorithm. For example, with the help of implicit representation, a high-quality initial mesh can be obtained, and the projection of the data points to the base surface can be efficiently computed by Newton iteration.

2) We use an evolution process to recover the sharp features. The evolution is governed by a combination of two terms: a data-driven velocity and a bilateral filtering. This evolution process can produce sharp features, which are faithful to the given data.

3) By incorporating the range constraints and volume constraints, we are able to conveniently exploit the a priori knowledge about the geometric properties of the object to be reconstructed. This is especially helpful when the given data contains noise and inaccuracies.

1.2 Related Work

Numerous approaches have been proposed to compute a surface approximation of a given set of unorganized points. Depending on the area of the application, different representations have been used, such as triangular meshes [3,7,15,38,2,41,47], subdivision surfaces [28,44, 12], parametric spline surfaces [16,23], discretized level sets [42,57,8], scalar spline functions [45,33], radial basis functions [10,40] and point set surfaces [1,43,19]. It is beyond the scope of this paper to give a detailed overview of all the existing work. [29] gives an excellent survey of the previous work on surface extraction from point clouds.

In this paper, we suggest a new reconstruction algorithm combining two types of representations: an implicit T-spline level set and a mesh. The proposed method relies on two main tools: displacement mapping [14] and bilateral filtering [48,50]. In the remainder of this section, we will describe some related work about these two tools.

Given a smooth base surface S_0, a displaced surface S can be generated by a scalar field (a displacement map), which specifies the displacement values along the normal directions of S_0. The use of displacement maps is quite popular for geometric modeling purposes. They are used in high end rendering systems [14,5], to capture the fine detail of a 3D photography model [36], for geometric simplification with appearance-preserving [13], for building semi-regular multiresolution meshes from an arbitrary connectivity input mesh[25] and for multiresolution mesh deformations [35]. In order to avoid cracks between adjacent triangles of a mesh, the interpolated normal is used [24] to displace the surface, B-spline surfaces are fitted [36] to the mesh before the displacement mapping, displaced subdivision surfaces [37] are suggested which is based on the butterfly subdivision scheme. There are also existing approaches for reconstructing a displaced subdivision surface directly from a given set of points [31]. Most recently, the author in [56] presents a displaced surface representation based on a manifold structure.

Extracting sharp features from 3D data is important [49], but difficult due to the feature-insensitive sampling and the noise of the given data. Many approaches have been proposed to address this problem [27,30,53, 6]. As a non-iterative scheme for edge-preserving smoothing, the bilateral filter is used in [20] and [32] to denoise a given surface. The author in [52] presents a robust general approach conducting bilateral filters to govern the sharping of triangular meshes. Also, the authors in [4] conduct the bilateral filter in the reconstruction of surfaces from scattered data.

The remainder of the paper is organized as follows. The next section describes how to formulate the evolution process of T-spline level sets, in order to generate the smooth base surface for the object reconstruction. Section 3 discusses the combination of different constraints into the evolution process. In particular, it is shown how to incorporate distance field constraints, range constraints and volume constraints such that the T-spline level sets evolution will be more robust and effective when dealing with non-uniformly sampled and incomplete data. Section 4 presents how to reconstruct geometric details and sharp features of the object by deriving a mesh representation from the smooth implicit surface. After presenting some experimental results in Section 5, we conclude this paper and discuss future work.

2 Evolution of T-spline Level Set for Base Surface Generation

In this section, we describe how to generate the base surface through the evolution of T-spline level sets. The input of our algorithm is a set of unorganized (maybe noisy and defected) data points $(\mathbf{p}_k)_{k=1,2,...,n}$, which are scattered over an unknown piecewise smooth surface $S_{\mathbf{p}}$. The base surface represented S_0 by T-spline level sets will capture the topology and the outline of the surface $S_{\mathbf{p}}$.

2.1 Definition of T-spline Level Sets

The *T-spline level set* $\Gamma(f)$ is defined as the zero set of a trivariate T-spline function f over some domain D,

$$\Gamma(f) = \{\ (x,y,z) \in D \subset \mathbb{R}^3 \mid f(x,y,z) = 0\ \}, \qquad (1)$$

where

$$f(x,y,z) = \frac{\sum_{i=1}^{n} c_i B_i(x,y,z)}{\sum_{i=1}^{n} B_i(x,y,z)} \qquad (2)$$

are T-spline [46] functions in 3D, with the real coefficients (control points) c_i, $i = 1,2,...,n$. For cubic T-splines, the basis functions are

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

where $N_{i0}^3(x)$, $N_{i0}^3(y)$ and $N_{i0}^3(z)$ are certain cubic B-splines, whose associated knot vectors are determined by the T-spline control grids (T-mesh).

In order to simplify the notation, we use $\mathbf{x}$ to represent the point $\mathbf{x} = (x,y,z)$ in 3D, 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

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

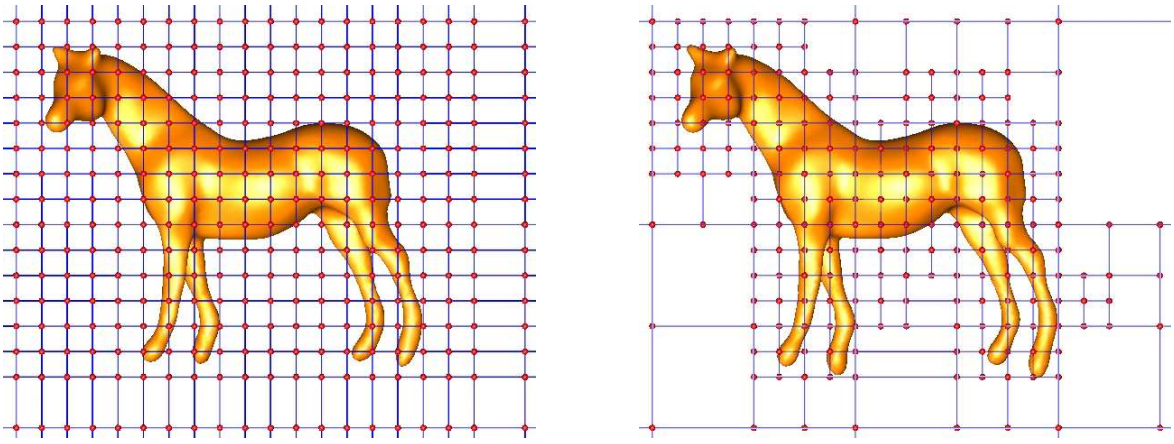
Now the T-spline function $f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}$.

T-splines [46] are generalizations of tensor product B-splines. T-spline control grids permit T-junctions, which provides a valuable property of local refinement. As shown in Figure 2, much less (decreased by 71%) control points are needed to represent the same level set (a horse) by trivariate T-splines than that by trivariate tensor product B-splines. Since a T-spline function is piecewise rational, the T-spline level sets are piecewise algebraic surfaces (C^2 in our case).

The T-spline control grid (T-mesh) (cf. Fig. 1 (a)) is constructed through octree subdivision of the function domain D, as follows.

1. Set the initial T-mesh to be an axis-aligned bounding box of the cubic domain which contains 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 octree subdivision.
3. Repeat step 2 until a user-specified threshold (e.g., a maximum level of subdivision) is reached or no cell contains more than n_0 data points any more.

In this way, the distribution of T-spline control points is made adaptive to the geometry of the object to be reconstructed, which usually leads to a sparse representation.



(a) B-splines with 5148 control points (b) T-splines with 1484 control points

Fig. 2 A horse implicitly defined by trivariate B-spline (T-spline) scalar functions.

2.2 Evolution of T-spline level sets

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

$$f(\mathbf{x}, \tau) = \mathbf{b}(\mathbf{x})^\top \mathbf{c}(\tau), \tag{4}$$

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

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

which is driven by a (possibly time-dependent) vector field $\mathbf{v}$. Note that we only consider the normal component $(\mathbf{v} \cdot \mathbf{n})$ of the velocity along the level set, since the tangent velocity does not change the shape at all. The unit normal vector $\mathbf{n}$ is given by

$$\mathbf{n}(\mathbf{x}, \tau) = \frac{\nabla f(\mathbf{x}, \tau)}{|\nabla f(\mathbf{x}, \tau)|}. \tag{6}$$

During the evolution, the definition of T-spline level sets

$$f(\mathbf{x}, \tau) \equiv 0, \quad \mathbf{x} \in \Gamma(f), \tag{7}$$

implies

$$\frac{\partial f(\mathbf{x}, \tau)}{\partial \tau} + \nabla f(\mathbf{x}, \tau) \cdot \frac{\partial \mathbf{x}}{\partial \tau} = 0, \quad \mathbf{x} \in \Gamma(f). \tag{8}$$

Combining (5), (6) and (8), we get the evolution equation of T-spline level sets under the vector field $\mathbf{v}$,

$$\frac{\partial f(\mathbf{x}, \tau)}{\partial \tau} = -\mathbf{v}(\mathbf{x}, \tau) \cdot \nabla f(\mathbf{x}, \tau), \quad \mathbf{x} \in \Gamma(f). \tag{9}$$

In our method, we always start the evolution of T-spline level sets from an initial level set, which contains all data points inside. The initial level set function f can be found by approximating it to the signed distance field of a bounding sphere or a rough offset of the surface to be reconstructed [57].

2.3 Evolution Speed Function

The speed function $\mathbf{v}$ plays a a key role in the evolution of T-spline level sets. In order to capture the base surface of the given data points, we use a similar speed function as that proposed by Caselles et al. [11],

$$\mathbf{v} = e(d)(\gamma + \kappa)\mathbf{n} - (1 - e(d))(\nabla d \cdot \mathbf{n})\mathbf{n} \tag{10}$$

where γ is a constant velocity (which is also known as a *balloon force*), $\kappa = \mathrm{div}(\nabla f/|\nabla f|)$ is the mean curvature of the level set surface, d is the *unsigned distance* function of the data points, and $e(d) = 1 - \mathrm{e}^{-\eta d^2}$ is the edge detector function. η is pre-defined, and its value is affected by the size of the data range. In our experimental setting, all data points are contained in a unit bounding box $(-1 \le x, y, z \le 1)$, and we usually set $\eta = 1$.

The speed function (10) is a linear combination of two parts. In the first part, the mean curvature term κ makes the level set smooth, and the balloon force γ is used to increase the speed of the evolution (especially for capturing narrow concave boundaries). The second part attracts the level set to the detected boundary edges. The edge detector function $e(d)$ is used as weighting coefficients to balance the influence of the two parts.

2.4 Discretization of the Evolution Equation

The evolution equation (9) under the speed function (10) is discretized by uniformly sampling a set of points $\mathbf{x}_j$, $j = 1, \ldots, N_0$ on the T-spline level set. We use the marching triangulation method to generate the uniform samples, which will be described later in Section 4.1. Since usually the number of sample points N_0 is much larger than the number of T-spline control coefficients n, the equation can not be exactly satisfied. We use a least-squares approach to choose the time derivative of

the function f by solving

$$E = \sum_{j=1}^{N_0} (\dot{f}(\mathbf{x}_j, \tau) + \mathbf{v}(\mathbf{x}_j, \tau) \cdot \nabla f(\mathbf{x}_j, \tau))^2 \rightarrow \text{Min} \quad (11)$$

where $\dot{f}(\mathbf{x}_j, \tau) = \partial f(\mathbf{x}_j, \tau)/\partial \tau$ is the time derivative of f.

In order to prevent the problem from being ill-posed, we add a regularization term to the above function E to be minimized,

$$E + \lambda_c |\dot{\mathbf{c}}|^2 \rightarrow \text{Min}, \quad (12)$$

where $\dot{\mathbf{c}}$ is the time derivative of the T-spline control coefficients, and λ_c is a pre-scribed small constant [17].

Since in our case $\dot{f}(\mathbf{x}, \tau) = \mathbf{b}(\mathbf{x})^\top \dot{\mathbf{c}}(\tau)$ (cf. Equation (4)), the solution to (12) can be found by solving a sparse linear system of equations.

Then we generate the updated control coefficients

$$\mathbf{c}(\tau + \Delta\tau) = \mathbf{c}(\tau) + \dot{\mathbf{c}} \Delta\tau. \quad (13)$$

by using an explicit Euler step $\Delta\tau$. $\Delta\tau$ is chosen as

$$\Delta\tau = \min(1, \{ \frac{h}{|\mathbf{v}(\mathbf{x}_j, \tau) \cdot \mathbf{n}(\mathbf{x}_j, \tau)|} \}_{0 \le j \le N_0}) \quad (14)$$

where h is a user-defined constant to indicate the maximum allowed evolution step size for each sample point on the T-spline level set.

3 Constraints for T-spline level sets evolution

In this section, we present three different constraints including distance field constraints, range constraints, and volume constraints, which can be applied to improve the robustness and effectiveness of T-spline level sets evolution.

3.1 Distance Field Constraint

The distance field constraint is used to restore the *signed distance* property of the T-spline level set function during the evolution, without having to use the costly *level set reinitialization* procedure [54].

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

$$S_0 = \int_D (\frac{\partial |\nabla f(\mathbf{x}, \tau)|}{\partial \tau} + |\nabla f(\mathbf{x}, \tau)| - 1)^2 \mathrm{d}V \rightarrow \text{Min}, \quad (15)$$

where $\mathrm{d}V$ is the volume element. This constraint acts as a penalty function to penalize the deviation of f from a signed distance function.

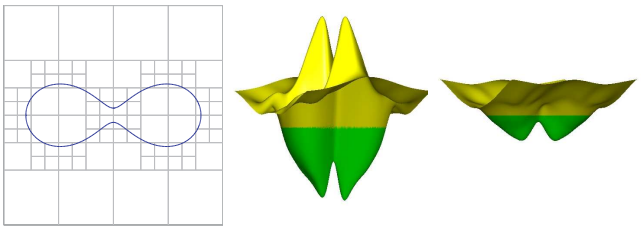
In practice, we realize that it is usually sufficient to maintain the *signed distance* property of f only in a small neighborhood of the zero level set (like the narrow-band level set method). Thus we propose to use a much more efficient version of the distance field constraint,

$$S = \sum_{j=1}^{N_0} (\frac{\partial |\nabla f(\mathbf{x}_j, \tau)|}{\partial \tau} + |\nabla f(\mathbf{x}_j, \tau)| - 1)^2, \quad (16)$$

by only considering the sample points $\mathbf{x}_j$, $j = 1, \ldots, N_0$ on the zero level set. Combining the distance field constraint into the evolution equation, we get the new objective function to be minimized

$$E + \lambda_c |\dot{\mathbf{c}}|^2 + \lambda_s S \rightarrow \text{Min}. \quad (17)$$

We usually choose the weight $\lambda_s = 0.1$ in our experiments. More details about the influence and choice of λ_s are described in [54].



(a) (b) (c)

Fig. 3 The distance field constraint (DFC) for T-spline level sets in 2D. The figure shows the data points with T-mesh in (a), the resulting level set function without DFC in (b), and the resulting level set function with DFC in (c).

Fig. 3 illustrates an example of using the distance field constraint. As shown in (b), without the using of distance field constraints, the level set function becomes very steep (or flat) in some regions, which may cause difficulties in maintaining the numerical stability during the evolution. This situation can be cured by applying the distance field constraint, as shown in (c).

3.2 Range Constraint

One typical advantage of the implicit representation of a surface is fast inside/outside distinction. In our case, we assume that the T-spline level set function $f(\mathbf{x}) > 0$ represents the region outside the surface, and $f(\mathbf{x}) < 0$ inside the surface. The range constraints are used to specify regions which should lie inside/outside the surface to be reconstructed. More specifically, we consider a set of points $\{\mathbf{x}_k\}_{k=1,\ldots,N_1}$ which should not lie outside the zero level set, and another set of points $\{\mathbf{y}_k\}_{k=1,\ldots,N_2}$ which should not lie inside the zero level set,

$$\begin{cases} f(\mathbf{x}_k) \le 0, & k = 1, \ldots, N_1. \\ f(\mathbf{y}_k) \ge 0, & k = 1, \ldots, N_2. \end{cases} \quad (18)$$

The above range constraints can be achieved by adding corresponding penalty terms to the objective function (17) such that the time derivative satisfies $\dot{f}(\mathbf{x}_k) < 0$ if $f(\mathbf{x}_k) >$

0, and $\dot{f}(\mathbf{y}_k) > 0$ if $f(\mathbf{y}_k) < 0$, respectively. For example, to the first set of points $\mathbf{x}_k$, we propose to use

$$R = \sum_{k=1}^{N_1} (\dot{f}(\mathbf{x}_k, \tau) - f(\mathbf{x}_k, \tau))^2 \alpha(f(\mathbf{x}_k, \tau)) \tag{19}$$

where the 'activator' function α is defined as

$$\alpha(f) = \begin{cases} 1, & f > 0. \\ 0, & f \leq 0. \end{cases} \tag{20}$$

The second set of points $\mathbf{y}_k$ can be dealt with similarly.

The range constraints are very useful in practical applications. As noticed in [22], the presence of the balloon force in the evolution speed function (10) may lead to problems when a large time step size is used for the evolution, because the level set may skip over and miss object boundaries defined by incomplete data. This problem can be handled by treating all the data points as the set of points $\mathbf{x}_k$ to be constrained not outside the surface, which is easily done by combining the range constraint term (19) into the objective function to be minimized

$$E + \lambda_c |\dot{\mathbf{c}}|^2 + \lambda_s S + \lambda_r R \rightarrow \text{Min}. \tag{21}$$

Usually we use a large value for the weight λ_r (e.g. $\lambda_r = 100$) of the range constraint. Figure 4(b) and (c) illustrate different results before and after applying range constraints. Another example is given in Figure 6.

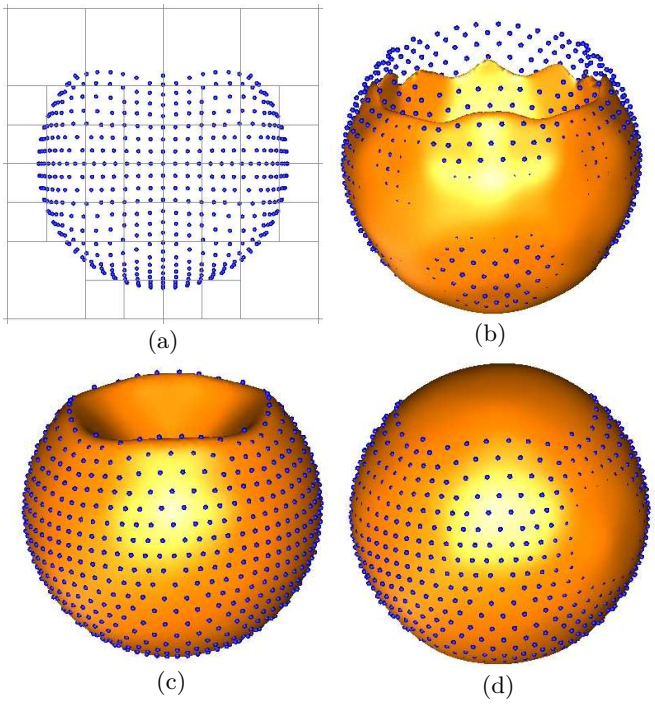


Fig. 4 The range constraint and volume constraint. The figure shows the data points with T-mesh in (a), the resulting T-spline level set without range and volume constraints in (b), the result with range constraint in (c), and the result with volume constraint in (d).

3.3 Volume Constraint

In some applications of surface reconstruction, the volume of the object to be reconstructed may be known a priori. In order to utilize this geometric information, we propose to formulate the volume constraint for the evolution of surfaces.

Let V_0 be the volume of the initial surface (T-spline level set), and V_∞ be the volume of the final surface. In our case $V_0 > V_\infty$, since the initial surface contains the final surface. By defining a smooth monotonic function $V(\tau)$ with respect to the time τ such that $V(0) = V_0$ and $V(\tau) \rightarrow V_\infty$ as $\tau \rightarrow \infty$, the volume of the surface will continuously converge to the desired value V_∞. Then the volume constraint is represented as

$$\int_\Gamma \mathbf{v}_n(\mathbf{x}, \tau) dA = \dot{V}(\tau), \tag{22}$$

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

$$\mathbf{v}_n(\mathbf{x}, \tau) = -\frac{\dot{f}(\mathbf{x}, \tau)}{|\nabla f(\mathbf{x}, \tau)|} \tag{23}$$

is the normal velocity of the T-spline level set. We use the exponential function to define the curve of volume changes

$$V(\tau) = (V_0 - V_\infty) e^{-\tau} + V_\infty, \tag{24}$$

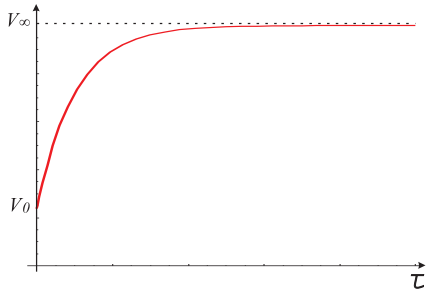


Fig. 5 The volume function $V(\tau)$.

which is illustrated in Fig. 5. It becomes a volume preserving constraint when $V_\infty = V_0$.

By combining (22) and (23), the volume constraint is linear in the time derivatives of the T-spline control coefficients. Now the evolution of T-spline level sets is transformed into a least-squares problem (21) subject to the linear volume constraint. By using the method of Lagrange multipliers, the solution can be obtained by solving a sparse linear system of equations.

The volume constraint is very helpful to handle incomplete data with large holes. Without the volume constraint, the level set may easily shrink into the holes during the evolution (cf. Fig. 4(b) and (c)). With the volume constraint, the level set surface will keep an ap-

propriate volume and the holes will be properly closed (cf. Fig. 4(d)).

4 Mesh Reconstruction with Geometric Details and Sharp Features

4.1 Initial Mesh Generation through Marching Triangulation

After the smooth base surface S_0 is obtained, we use the Marching Triangulation [26] method to generate the mesh representation of S_0. We choose the Marching Triangulation since it is able to produce a high-quality triangular mesh. Other polygonization methods [34, 21] can be also considered.

The requirement for applying the Marching Triangulation method is that the function value $f(\mathbf{x})$ and the gradient $\nabla f(\mathbf{x})$ are available for any point $\mathbf{x}$ in the function domain. This is satisfied by our T-spline function f since f is C^2 continuous in the domain of interest.

A key procedure of the Marching Triangulation is the choice of seed points on the implicit surface. If the implicit surface contains multiple components, then at least one seed point must be chosen for one component, otherwise those components without seed points will not be triangulated. In our case, since the implicit function f defines a good base surface S_0 for the data points $(\mathbf{p}_k)_{k=1,2,\ldots,n}$, we solve this problem as follows:

1. Project each data point $(\mathbf{p}_k)_{k=1,2,\ldots,n}$ to S_0, get the corresponding closest point $(\mathbf{q}_k)_{k=1,2,\ldots,n}$. (More details will be described later in Section 4.2.1.)
2. Initialize the set of potential seed points $\mathcal{P} = \{\mathbf{p}_k\}_{k=1,2,\ldots,n}$.
3. Initialize the set of generated triangles $\mathcal{M} = \emptyset$.
4. Choose an arbitrary seed point $\mathbf{s}_i$ from $\mathcal{P}$, and apply the Marching Triangulation to get a new mesh M_i. Add M_i to $\mathcal{M}$.
5. For each potential seed point $(\mathbf{p}_k)_{\mathbf{p}_k \in \mathcal{P}}$, compute the distance from its closest point $\mathbf{q}_k$ to the new mesh M_i. If the distance is sufficiently small, then remove $(\mathbf{p}_k)$ from $\mathcal{P}$.
6. Repeat steps $4 \sim 5$ until $\mathcal{P} = \emptyset$.
7. Output the generated triangular meshes $\mathcal{M}$.

A similar strategy is also used to generate uniform samples on the T-spline level set during the evolution. The idea is to treat the sample points on the current surface as the data points in the above procedure when applying the marching triangulation for the next surface. Since the triangulation of the initial surface with only one component is easily obtained, all the other surfaces can be handled subsequently.

The Marching Triangulation method is very fast and also simple to implement. As shown in Section 5, usually we can generate over $10,000$ triangles within seconds. The generated mesh is semi-regular, and the vast majority of the mesh vertices have valence 6. The edge length of the triangles is approximately indicated by the marching step length δ_t, which can be determined by the feature size of the reconstructed surface. We usually choose $\delta_t = 0.2 l_T$, where l_T is the diameter of the cells at the finest level of the T-mesh.

4.2 Displacement Mapping

After the base surface S_0 is triangulated by the initial mesh M_0, the topology and parametrization of the surface to be reconstructed is now defined by M_0. Fine geometric details are to be captured through a displacement mapping,

$$M = M_0 + \mathcal{D}, \tag{25}$$

where the displacement field $\mathcal{D}$ is generated from the data points $(\mathbf{p}_k)_{k=1,2,\ldots,n}$.

4.2.1 Data Points Projection

In order to get the displacement field $\mathcal{D}$, we project all data points to the initial mesh M_0. Since M_0 is a discretization of the smooth base surface S_0, the projection process can be transformed into the computation of closest points on the surface S_0, which is implicitly defined by the T-spline function f. Thus, for each data point $\mathbf{p}_k$, one can compute its closest point $\mathbf{q}_k$ efficiently by Newton iteration. Then we associate $\mathbf{q}_k$ to its closest triangle $\mathcal{T}_k$, which will be needed later for the Gaussian filtering of the displacement field. The whole projection procedure is given as follows:

1. For each triangle $\mathcal{T}_i \in M_0$, initialize the array of indices of its associated data points $\mathcal{I}_i = \emptyset$.
2. Initialize the displacement array $(d_k = 0)_{k=1,2,\ldots,n}$.
3. For each data point $(\mathbf{p}_k)_{k=1,2,\ldots,n}$,
   (a) Initialize the closest point $\mathbf{q}_{k,0} = \mathbf{p}_k$.
   (b) Using Newton's method to get the updated closest point $\mathbf{q}_{k,i+1} = \mathbf{q}_{k,i} - \frac{f(\mathbf{q}_{k,i})}{\|\nabla f(\mathbf{q}_{k,i})\|^2} \nabla f(\mathbf{q}_{k,i})$.
   (c) Repeat step 3.2 until $\|\mathbf{q}_{k,i+1} - \mathbf{q}_{k,i}\|$ is sufficiently small. Set $\mathbf{q}_k = \mathbf{q}_{k,i+1}$.
   (d) Set the signed displacement value $d_k = \mathrm{sign}(\mathrm{f}(\mathbf{p}_\mathrm{k})) \cdot \|\mathbf{p}_\mathrm{k} - \mathbf{q}_\mathrm{k}\|$.
   (e) Find the closest vertex $\mathbf{v}_{k_v}$ of M_0 to the point $\mathbf{q}_k$.
   (f) From the one-ring neighborhood of $\mathbf{v}_{k_v}$, get the closest triangle(s) $\mathcal{T}_{k_t}$ to $\mathbf{q}_k$.
   (g) Add k to the array of indices $\mathcal{I}_{k_t}$.
4. Output $(\mathcal{I}_i)_{\mathcal{T}_i \in M_0}$ and the displacement array $(d_k)_{k=1,2,\ldots,n}$.

Please note that we do not compute any ray-triangle intersections for the data points projection. This improvement of efficiency has been achieved by exploiting the advantages of the implicit representation of the base surface.

4.2.2 Displaced Mesh

After the scalar displacement field $\mathcal{D}$ is already sampled at each closest point $(\mathbf{q}_k)_{k=1,2,\ldots,n}$, we then want to map it to each vertex $\mathbf{v}$ of the initial mesh M_0,

$$\hat{\mathbf{v}} = \mathbf{v} + d(\mathbf{v}) \cdot \mathbf{n}. \tag{26}$$

Note that the normals $\mathbf{n}$ are computed from the implicitly defined T-spline surface, $\mathbf{n} = \frac{\nabla f}{\|\nabla f\|}|_{\mathbf{v}}$, instead of the discretized mesh.

Since the given data points are often noisy, the sampled field is therefore not smooth. In order to avoid cracks between adjacent triangles of the displaced mesh, we use Gaussian filtering to smooth the displacement field (we usually choose $\sigma = 2\delta_t$, where δ_t is the marching step length in Section 4.1). Of course, the use of Gaussian filtering will also smooth some desired sharp features. The next section will describe how to recover these sharp features.

After the displacement mapping, the quality of the displaced mesh may be degraded. In the worst case, flips or self-intersections may happen to the displaced triangles, where the displacement values are too large for some deep convex or concave parts of the object surface. The allowed displacement can be bounded with the help of the principal curvature radii, which can be estimated from the implicit T-spline function.

One way to prevent this problem is to find a better base surface by using more degrees of freedom (T-spline control coefficients) and applying the 'final refinement' step [54] to make the T-spline level set more close to the data points. In our algorithm, we use the principal curvature radii as an indicator. If the displacement value is close to or larger than the corresponding curvature radii, we check if the displaced triangle is flipped. If some flips happen, we refine the T-spline level set to make sure that the displacement mapping is intersection free.

4.3 Recovering Sharp Features

After the displacement mapping, the displaced mesh approximates the data points far better than the initial mesh. Most parts of the object to be reconstructed are already well fitted, except near sharp features. In this section, we introduce a data-driven bilateral filtering method to reproduce sharp features of the reconstructed mesh.

4.3.1 Bilateral Filter

The bilateral filter, which was originally proposed in image processing [48,50], is a nonlinear filter derived from Gaussian blur, with a feature-preserving term that decreases the weights of pixels as a function of intensity differences. Following the formulation in [48], the bilateral filtering for image $I(\mathbf{p})$ at the pixel $\mathbf{p}^*$ is defined as

$$\hat{I(\mathbf{p}^*)} = \sum_{\mathbf{p}_j \in N(\mathbf{p}^*)} \frac{W_c(\|\mathbf{p}^* - \mathbf{p}_j\|) W_s(|I(\mathbf{p}^*) - I(\mathbf{p}_j)|)}{W} I(\mathbf{p}_j), \tag{27}$$

where $N(\mathbf{p}^*)$ is the neighborhood of $\mathbf{p}$. W_c is the standard Gaussian filter with parameter σ_c: $W_c(x) = e^{-x^2/2\sigma_c^2}$, and W_s is a similarity weight function for feature-preserving with parameter σ_s: $W_s(x) = e^{-x^2/2\sigma_s^2}$. W is a normalization factor

$$W = \sum_{\mathbf{p}_j \in N(\mathbf{p}^*)} W_c(\|\mathbf{p}^* - \mathbf{p}_j\|) W_s(\|I(\mathbf{p}^*) - I(\mathbf{p}_j)\|).$$

Recently, the bilateral filter has been applied to mesh denoising while preserving sharp features [20,32]. The author in [52] uses the bilateral filtering for recovering of sharp edges on feature-insensitive sampled edges. In [4], the bilateral filter in used for data denoising such that the filtered data points can be later connected into a mesh structure. Unlike these previous works, we combine the bilateral filtering term into a data-driven evolution process, such that the produced sharp features are faithful to the given data points.

4.3.2 Data-Driven Bilateral Evolution

Recall that our bilateral evolution is to obtain a mesh that meets two goals:

1. It provides a good fit to the point set $(\mathbf{p}_k)_{k=1,2,\ldots,n}$.
2. It recovers sharp features by conducting bilateral filters.

Consider a mesh M with time-dependent vertices $\mathcal{V}(\tau) = (\mathbf{v}_i(\tau))_{i=1,2,\ldots,m}$ (m is the number of vertices), whose evolution process is governed by minimizing the following energy function

$$F(\mathcal{V}(\tau)) = E_{dist}(\mathcal{V}(\tau)) + \omega E_{bila}(\mathcal{V}(\tau)) \rightarrow \min, \tag{28}$$

where the two terms E_{dist} and E_{bila} correspond to the two goals listed above, and $\omega > 0$ is a constant weighting coefficient.

The distance energy E_{dist} is defined as

$$E_{dist}(\mathcal{V}(\tau)) = \sum_{k=1}^{n} (\dot{\mathbf{q}}_k + \mathbf{q}_k - \mathbf{p}_k)^2, \tag{29}$$

where $\mathbf{q}_k$ on the mesh is the closest point of $\mathbf{p}_k$, and its time derivative $\dot{\mathbf{q}}_k = \partial \mathbf{q}_k / \partial \tau$ can be represented as a linear combination of related vertex velocities $\dot{\mathbf{v}}_j$

$$\dot{\mathbf{q}}_k = \sum_{\mathbf{v}_j \in \phi(\mathbf{p}_k)} \lambda_j \dot{\mathbf{v}}_j, \tag{30}$$

where $\phi(\mathbf{p}_k)$ contains 3 vertices of the corresponding triangle, and λ_j are the barycentric coordinates of $\mathbf{q}_k$.

Dataset	n_p	n_t	n_v	e_{avr} (10^{-3}) M_I	M_D	M_S	e_{max} (10^{-3}) M_I	M_D	M_S	Run time (s) TL	MT	D-Map	B-Evl
Rocker-Arm	10044	992	8577	4.34	0.88	0.69	24.44	14.12	11.30	10.50	2.19	0.52	6.64
Horse	48485	1484	9137	5.63	0.36	–	43.78	7.69	–	36.13	2.40	3.11	–
Fandisk(cut)	5817	1019	12856	18.27	1.27	0.21	68.86	19.90	7.93	12.30	2.94	0.39	7.78
Ball-joint	137062	988	11831	33.42	0.83	–	60.33	8.05	–	7.50	2.41	6.32	–
Foot	25845	871	4517	35.86	0.62	0.55	98.47	9.07	8.22	7.94	0.81	1.19	0.63
Sculpture	25386	2551	14026	3.93	0.74	0.63	20.72	6.33	5.41	56.33	5.03	1.67	6.16

Table 1 The approximation errors and the execution time of the given examples. n_p: number of data points; n_t: number of T-spline control points; n_v: number of mesh vertices; M_I: initial mesh; M_D: displaced mesh; M_S: sharpened mesh after bilateral evolution; TL: T-spline level set evolution; MT: marching triangulation; D-Map: displacement mapping; B-Evl: bilateral evolution. The right three columns show the run time for the T-spline level set evolution, the marching triangulation, the displacement mapping and the bilateral evolution, respectively. The left several columns show the number of data points, the number of T-spline control points, and the number of mesh vertices. The middle columns give both the approximation errors (e_{avr}) and the maximum errors (e_{max}) for the initial meshes, the displaced meshes and the final meshes, respectively.

The bilateral energy E_{bila} is defined as

$$E_{bila}(\mathcal{V}(\tau)) = \sum_{i=1}^{m} (\dot{\mathbf{v}}_i + \mathbf{v}_i - \mathbf{v}_i')^2, \tag{31}$$

where $\mathbf{v}_i'$ is the updated position of $\mathbf{v}_i$ according to the bilateral filter [52]

$$\mathbf{v}' = \sum_{\mathcal{T}_j \in N(\mathbf{v})} \frac{W_c(\|\mathbf{v} - \mathbf{c}_j\|) W_s(\|\mathbf{v} - \mathbf{v}_j^*\|)}{W} \alpha_j \mathbf{v}_j^*, \tag{32}$$

where $\mathbf{v}$ is any vertex of the mesh M, $\mathbf{v}_j^*$ is the projection point of $\mathbf{v}$ on the plane of the triangle $\mathcal{T}_j$, α_j is the area of $\mathcal{T}_j$, $\mathbf{c}_j$ is the center of $\mathcal{T}_j$, and $N(\mathbf{v})$ is the set of neighboring triangles contributing to the position of $\mathbf{v}$. W_c and W_s are the Gaussian filters as used in (27). See [52] for more details.

Combining (28), (29), (30) and (31), the minimizer of (28) leads to a quadratic objective function of the unknown time derivatives $\dot{\mathcal{V}} = (\dot{\mathbf{v}}_i)_{i=1,2,\ldots,m}$. The solution $\dot{\mathcal{V}}$ is found by solving a sparse linear system of equations, $\nabla F = 0$. Using explicit Euler steps $\mathbf{v}_i \rightarrow \mathbf{v}_i + \Delta\tau \dot{\mathbf{v}}_i$, with a suitable step-size $\Delta\tau$, one can trace the evolving mesh.

Actually, in practice, we do not need to update all vertices during the data-driven bilateral evolution, since the non-sharp region of the surface is already well fitted by the displacement mapping. Instead, we only need to update those sharp-vertices with both a high dihedral angle and a large approximation error, while keeping $\dot{\mathbf{v}} = 0$ for the remaining vertices. Here, we use the same method as indicated in [52] to detect potential sharp vertices with a high dihedral angle. If the one-ring neighborhood region of a potential sharp vertex is already well fitted (the approximation error is small), then we discard it from the list of real sharp vertices. After that, to construct E_{dist} and E_{bila}, we only consider those terms related with real sharp-vertices, such that the size of the linear system is greatly reduced.

The bilateral evolution continues until the approximation error can not be reduced any more. In our method, the approximation error e_{avr} is measured as the average distance of the data points to the mesh

$$e_{avr} = \frac{1}{n} \sum_{k=1}^{n} \|\mathbf{q}_k - \mathbf{p}_k\|. \tag{33}$$

4.3.3 Discussion

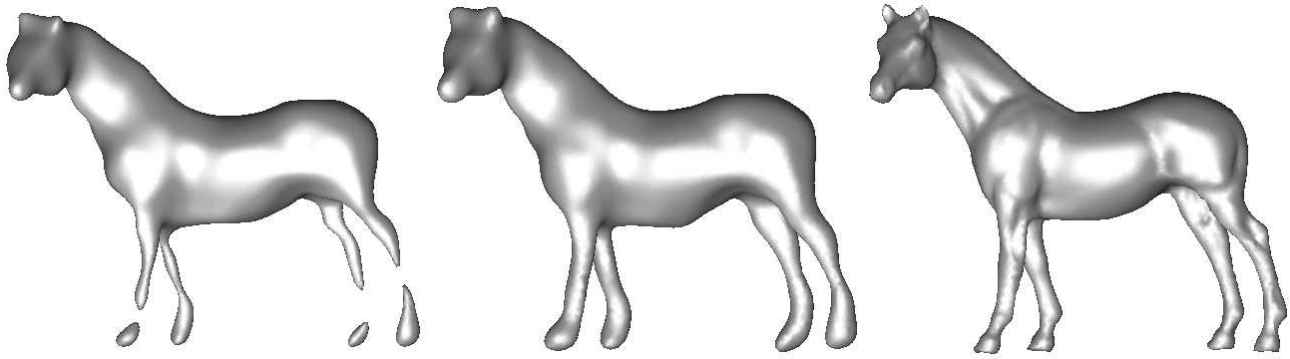
The parameters σ_c and σ_s of the bilateral filter are important to get a satisfying reconstruction result. On the one hand, if σ_c and σ_s are set too large, the sharp features will be smoothed. On the other hand, if they are set too small, the reconstruction result will be very sensitive to the noise of the given data set. In our experimental settings, we usually choose $\sigma_s = \sigma_c = 2\delta_t$ (δ_t is the marching step length in Section 4.1).

Usually within 10 iterations, the sharp features can be well reconstructed. But because most sharp-vertices are attracted towards their corresponding sharp edges, many degenerate triangles will be produced afterwards. Therefore, in order to improve the regularity of the reconstructed mesh, a *MeshSlicing* [9] operation is used to remove degenerate triangles after the bilateral evolution stops (See Figures 7 (d1), 8 (c1) and 10 (d1)).

5 Experimental results

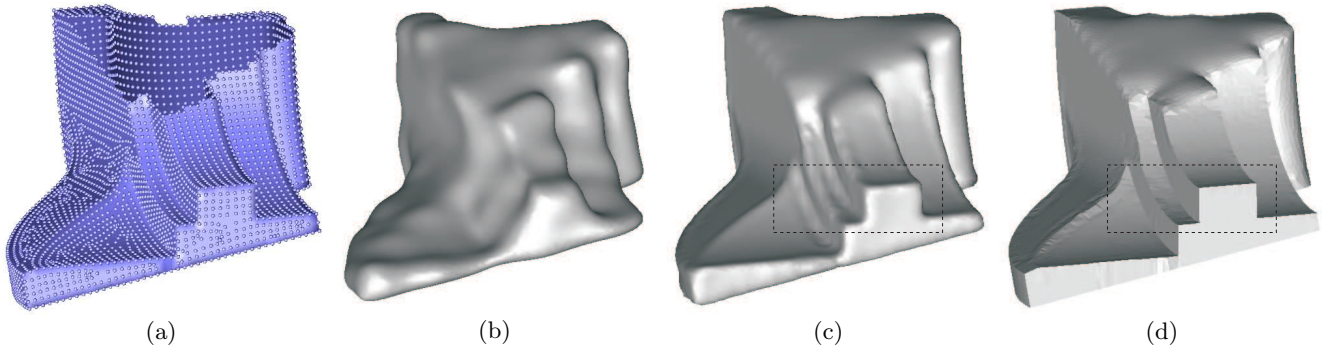
In this section, we present some examples to demonstrate the effectiveness of our method. All the given data points are normalized to be contained in a cubic domain ($[-1,1] \times [-1,1] \times [-1,1]$). The maximum error e_{max} mentioned below is the maximum distance of the data points to the mesh. The average error e_{avr} is the approximation error defined by Eq. (33). All the experiments are run on a PC with AMD Opteron(tm) 2.20GHz CPU and 3.25G RAM. The approximation errors and the execution time are reported in Table 1.

Example 1. The first example is the Rocker-Arm model, which contains non-uniform samples with holes and sharp



(a) (b) (c)

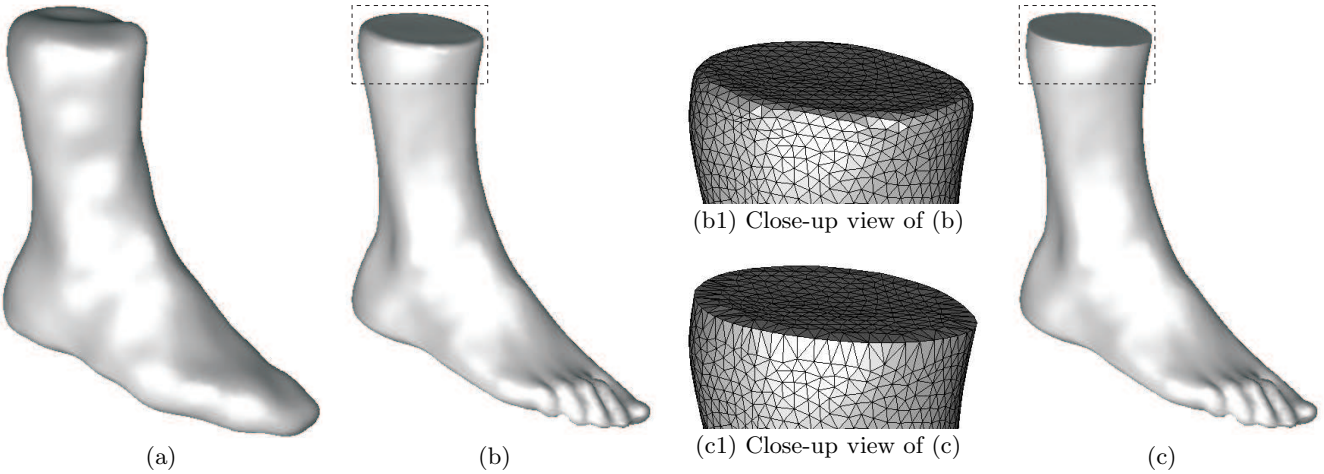
Fig. 6 Mesh reconstruction of the Horse model. The figure shows the disconnected base surface without range constraints in (a), the correct base surface with range constraints in (b), and the final surface in (c).



(a) (b) (c) (d)

(c1) Close-up view of (c) (d1) Close-up view of (d)

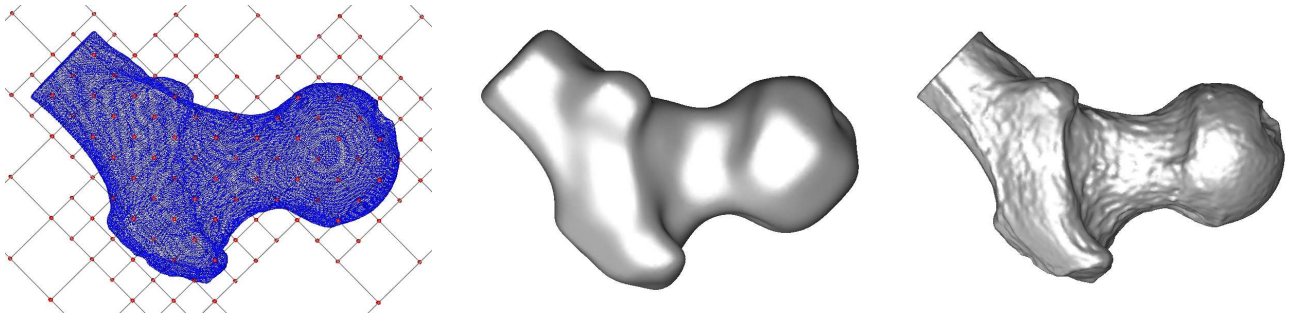
Fig. 7 Mesh reconstruction of part of the Fandisk model. We modified the original data set by cutting it with a plane. The figure shows the data points in (a), the initial mesh in (b), the displaced mesh in (c), and the final mesh with sharp features in (d). Flat shading is used in (d), (c1) and (d1).



(b1) Close-up view of (b)

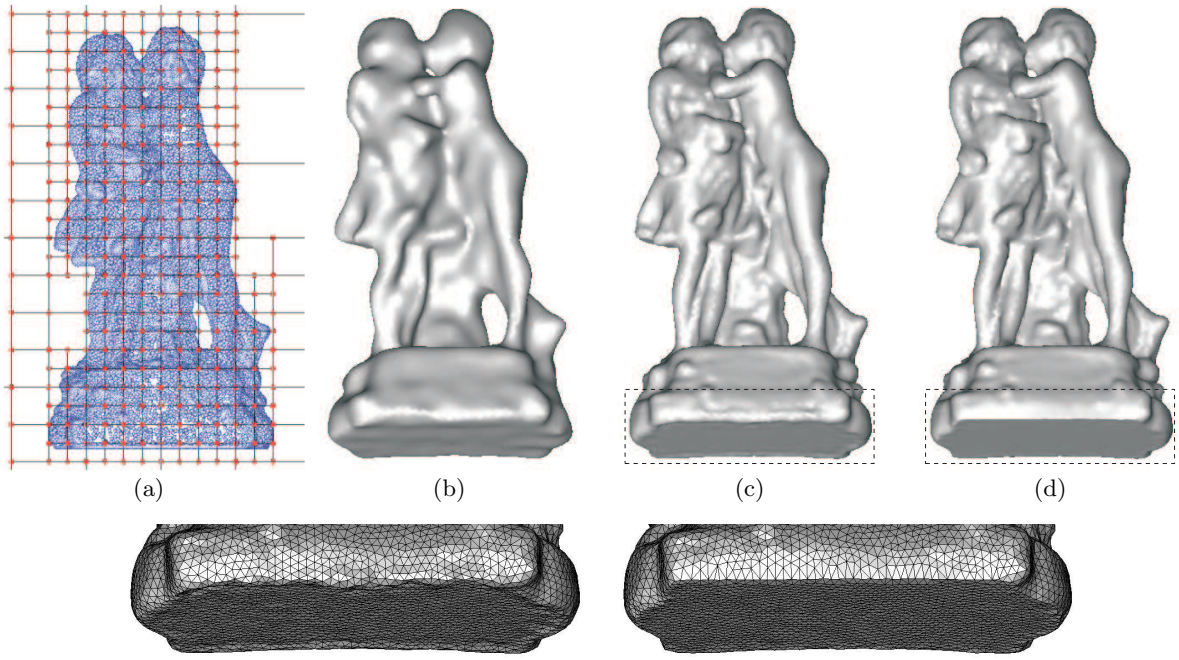
(a) (b) (c1) Close-up view of (c) (c)

Fig. 8 Mesh reconstruction of the Foot model. The figure shows the initial mesh in (a), the displaced mesh in (b), and the final mesh (after bilateral filtering and mesh slicing) with sharp features in (c). Flat shading is used in (b1) and (c1).



(a) (b) (c)

Fig. 9 Mesh reconstruction of the Ball-joint model. The figure shows the data points and the T-mesh in (a), the initial mesh in (b), and the displaced mesh in (c).



(a) (b) (c) (d)

(c1) Close-up view of (c) (d1) Close-up view of (d)

Fig. 10 Mesh reconstruction of the Sculpture model. The figure shows the data points and the T-mesh in (a), the initial mesh in (b), the displaced mesh in (c), and the final mesh with sharp features in (d). Flat shading is used in (c1) and (d1).

features, as shown in Fig. 1. The T-spline level set adapts its topology from genus-0 in (b) to genus-1 in (c), where the initial mesh is constructed. The displaced mesh is shown in (d). By using the data-driven bilateral evolution, the approximation error is further reduced by 21.6% (cf. Table 1), and the sharp edges are recovered in (e).

Example 2. The second example is a horse model, as shown in Fig. 6. The narrow and thin legs of the horse can easily split into several disconnected components in (a) without using the range constraint. After combining the range constraint (cf. Section 3.2), the base surface is correctly constructed in (b), and the final mesh with geometric details is shown in (c).

Example 3. The third example is part of the Fandisk model, which was cut by a plane, creating a large hole in the data set, as shown in Fig. 7. By using our method, the cutting hole can be filled by the T-spline level set representation, and the sharp edges can be recovered by the bilateral evolution of the mesh, as shown in (d). The approximation error is reduced by 83.5%, and the maximum error is reduced by 60.2% during the bilateral evolution (cf. Table 1).

Example 4. The fourth example is the Ball-joint model, as shown in Fig. 9. Since the data points are already well approximated by the displacement mesh, the last phase of our algorithm (recovering sharp features) is discarded.

Example 5. The fifth example is the Foot model, as shown in Fig. 8. After the displacement mapping in (b), the sharp edges can be recovered by the data-driven bilateral evolution, as shown in (c).

Example 6. The last example is a complex sculpture model, as shown in Fig. 10. Through the evolution of T-spline level sets, the base surface with a correct topology is obtained in (b). The updated mesh after displacement mapping is given in (c). Finally, the sharp edges are produced in (d) by using the bilateral evolution.

6 Conclusions

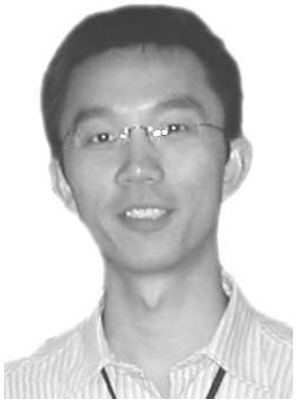
We have introduced a new framework for surface reconstruction from unorganized data points. We use the displacement mapping of a smooth base surface, which is implicitly represented by scalar T-spline functions. We have shown that, with the help of different shape constraints, even non-uniformly sampled and incomplete data can be handled by the implicitly defined base surface. Geometric details can efficiently be dealt with with the help of the displacement mapping. The sharp features of the mesh surface are produced by using a data-driven bilateral evolution. Possible future work includes comparing results by using different forms of functions for the implicit representation of the base surface, and exploiting other kinds of a priori knowledge about the geometric properties of the surface to be reconstructed.

Acknowledgment. The first author was supported by a Marie Curie Incoming International Fellowship of the European commission (project 022073 ISIS). The second author was supported by the Austrian Science Fund through the national research network on Industrial Geometry (S9202). The data sets used for our experiments are courtesy of Stanford computer graphics laboratory and UCI (University of California, Irvine) computer graphics laboratory.

References

1. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Point set surfaces. In *Proc. VIS'01*, pages 21–28, 2001.
2. R. Allègre, R. Chaine, and S. Akkouche. Convection-driven dynamic surface reconstruction. In *Proc. SMI'05*, pages 33–42, 2005.
3. N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 213–222, 2000.
4. A.Miropolsky and A.Fischer. Reconstruction with 3D geometric bilateral filter. In *9th ACM Symposium on Solid Modeling and Applications*, pages 225–231, Genoa, Italy, 2004.
5. A. Apodaca and L. Gritz. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, San Francisco, CA, 1999.
6. M. Attene, B. Falcidieno, J. Rossignac, and M. Spagnuolo. Sharpen & Bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):181–192, 2005.
7. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.
8. J. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Comput. Geom. Theory Appl.*, 22(1):185–203, 2002.
9. M. Botsch and L. Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *Proc. VMV'01*, pages 283–290, 2001.
10. 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.
11. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Int. J. Comput. Vision*, 22(1):61–79, 1997.
12. 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 *Proc. PG'04*, pages 16–24, 2004.
13. J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proc. SIGGRAPH'98*, pages 115–122, 1998.
14. R. Cook. Shade trees. In *Proc. SIGGRAPH'84*, pages 223–231, 1984.
15. T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *Proc. SMI'03*, pages 127–134, 2003.
16. M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Proc. SIGGRAPH'96*, pages 325–334, 1996.
17. H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, Dordrecht, 1996.
18. R. Feichtinger, M. Fuchs, B. Jüttler, O. Scherzer, and H. Yang. Dual evolution of planar parametric spline curves and T-spline level sets. *Computer-Aided Design*, 2007, to appear.
19. S. Fleishman, D. Cohen-Or, and C. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. (Proc. SIGGRAPH'05)*, 24(3):544–552, 2005.
20. S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):950–953, 2003.
21. M. Gavriliu, J. Carranza, D. Breen, and A. Barr. Fast extraction of adaptive multiresolution meshes with guaranteed properties from volumetric data. In *Proc. VIS'01*, pages 295–303, 2001.
22. R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *IEEE Trans. on Image Processing*, 10:1467–1475, 2001.
23. X. Gu, Y. He, and H. Qin. Manifold splines. *Graphical Models*, 68(3):23–254, 2006.
24. S. Gumhold and T. Hüttner. Multiresolution rendering with displacement mapping. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 55–66, 1999.
25. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proc. SIGGRAPH'00*, pages 95–102, 2000.
26. E. Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, 1998.
27. H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH'96*, pages 99–108, 1996.

28. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proc. SIGGRAPH'94*, pages 295–302, 1994.
29. A. Hornung and L. Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 41–50, 2006.
30. A. Hubeli and M. H. Gross. Multiresolution feature extraction from unstructured meshes. In *Proc. of IEEE Visualization'01*, pages 16–25, 2001.
31. W.-K. Jeong and C.-H. Kim. Direct reconstruction of a displaced subdivision surface from unorganized points. *Graphical Models*, 64(2):78–93, 2002.
32. T. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):943–949, 2003.
33. B. Jüttler and A. Felis. Least squares fitting of algebraic spline surfaces. *Adv. Comput. Math.*, 17:135–152, 2002.
34. T. Karkanis and A. J. Stewart. Curvature-dependent triangulation of implicit surfaces. *IEEE Computer Graphics and Applications*, 21(2):60–69, 2001.
35. L. Kobbelt, T. Bareuther, and H. P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum (Proc. Eurographics'00)*, 19(3):249–260, 2000.
36. V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. SIGGRAPH'96*, pages 313–324, 1996.
37. A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. SIGGRAPH'00*, pages 85–94, 2000.
38. B. Mederos, N. Amenta, L. Velho, and L. H. de Figueiredo. Surface reconstruction for noisy point clouds. In *Symposium on Geometry Processing*, pages 53–62, 2005.
39. Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):463–470, 2003.
40. Y. Ohtake, A. Belyaev, and H.-P. Seidel. 3d scattered data approximation with adaptive compactly supported radial basis functions. In *Proc. SMI'04*, pages 31–39, 2004.
41. Y. Ohtake, A. Belyaev, and H.-P. Seidel. A composite approach to meshing scattered data. *Graph. Models*, 68(3):255–267, 2006.
42. S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, New York, 2002.
43. M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):641–650, 2003.
44. 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.
45. 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.
46. T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):477–484, 2003.
47. A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, and D. Cohen-Or. Competing fronts for coarse–to–fine surface reconstruction. In *Proc. Eurographics'06*, pages 389–398, 2006.
48. S. M. Smith and J. M. Brady. SUSAN–a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, 1997.
49. W. B. Thompson, J. C. Owen, H. J. de St. Germain, S. R. Stark, and T. C. Henderson. Feature-based reverse engineering of mechanical parts. *IEEE Transactions on Robotics and Automation*, 12(1):57–66, 1999.
50. C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. ICCV'98*, pages 839–846. IEEE Computer Society, 1998.
51. L. Velho, J. Gomes, and L. H. Figueiredo. *Implicit Objects in Computer Graphics*. Springer Verlag, New York, 2002.
52. C. Wang. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):629–639, 2006.
53. K. Watanabe and A.G. Belyaev. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum (Proc. Eurographics'01)*, 20(3):385–392, 2001.
54. 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 *Proc. SMI'06*, pages 247–252, 2006.
55. H. Yang and B. Jüttler. Meshing non-uniformly sampled and incomplete data based on displaced T-spline level sets. In *Proc. SMI'07*, pages 251–260, 2007.
56. S.-H. Yoon. A surface displaced from a manifold. In *Proc. GMP'06*, pages 677–686, 2006.
57. H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods*, pages 194–201, 2001.



HUAPING YANG received the BE (1998) degree in hydraulic engineering and the PhD (2004) degree in computer science from Tsinghua University, China. In 2004, he did a one-year postdoc in the computer graphics group at the University of Hong Kong. In 2005, he starts a postdoc at JKU Linz, Austria, in the field of applied geometry, funded by a Marie Curie incoming international fellowship. His research interests include geometric modeling, computer graphics, and scientific visualization.

BERT JUETTLER is professor of Mathematics at Johannes Kepler University of Linz, Austria. He did his PhD studies (1992-94) at Darmstadt University of Technology under the supervision of the late Professor Josef Hoschek. His research interests include various branches of applied geometry, such as Computer Aided Geometric Design, Kinematics and Robotics. Bert Juettler is member of the Editorial Boards of Computer Aided Geometric Design (Elsevier) and the Int. J. of Shape Modeling (World Scientific) and serves on the program committees of various international conference (e.g., the SIAM conference on Geometric Design and Computing 2007).