Meshing Non-uniformly Sampled and Incomplete Data Based on Displaced T-spline Level Sets

Paper ID: 32

Abstract

We propose a new method for constructing a piecewise smooth mesh from a set of unorganized data points, which may be non-uniformly sampled, noisy, and even containing holes. The method is based on the construction of an implicit representation of the surface, by using smooth (C^2 in our case) T-spline scalar functions. We first generate the T-spline control grid, and use an evolution process such that the resulting T-spline level sets capture the topology and outline of the object to be reconstructed. The initial mesh with high quality is obtained from the implicit T-spline function through the marching triangulation method. Then we project each data point to the initial mesh, and get a scalar displacement field. Detailed features will be captured by the displaced mesh. We also propose an additional evolution process, which combines data-driven velocities and feature-preserving bilateral filters, in order to reproduce sharp features.

Keywords: mesh reconstruction, point cloud, displacement maps, T-spline, level sets

1 Introduction

1.1 Background and Previous Work

Surface reconstruction from scattered data points has many applications in computer graphics, computer aided design, computer vision and image processing. There is a large body of literature dealing with this problem. It is beyond the scope of this paper to give a detailed overview of all the existing work. [24] gives an excellent survey of the previous work on surface extraction from point clouds.

Typically scanned scattered data is noisy and may even contain holes. The object surface to be reconstructed may have a complex topology, which is not known a priori. There have been many approaches trying to handle this difficulty [41]. Depending on the area of the application, different representations have been used, such as triangular meshes [6, 3, 14, 37, 46, 48, 41], subdivision surfaces [23, 44, 11], parametric spline surfaces [15, 18], discretized level sets [42, 57], scalar spline functions [45, 30], radial basis functions [8, 40] and point set surfaces [2, 43, 16]. These different representations can be classified into two types: parametric representations and implicit representations. Among the various approaches, these two representations may complement each other. On the one hand, the implicit representations [52] 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 [38]. On the other hand, the parametric representations can easily handle sharp features, but have difficulties when processing topology changes.

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 [13] and Bilateral filtering [49, 51]. In the rest part 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 filed (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 [13, 5], to capture the fine detail of a 3D photography model [32], for geometric simplification with appearance-preserving [12], for building semi-regular multiresolution meshes from an arbitrary connectivity input mesh[20] and for multiresolution mesh deformations [31]. While most displacement maps are along the surface normals, there are also vector-valued displacement maps [10, 35] along an arbitrary direction. In order to avoid cracks between adjacent triangles of a mesh, the interpolated normal is used [19] to displace the surface, B-spline surfaces are fitted [32] to the mesh before the displacement mapping, displaced subdivision surfaces [33] 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 [26, 27]. Most recently, the author in [56]

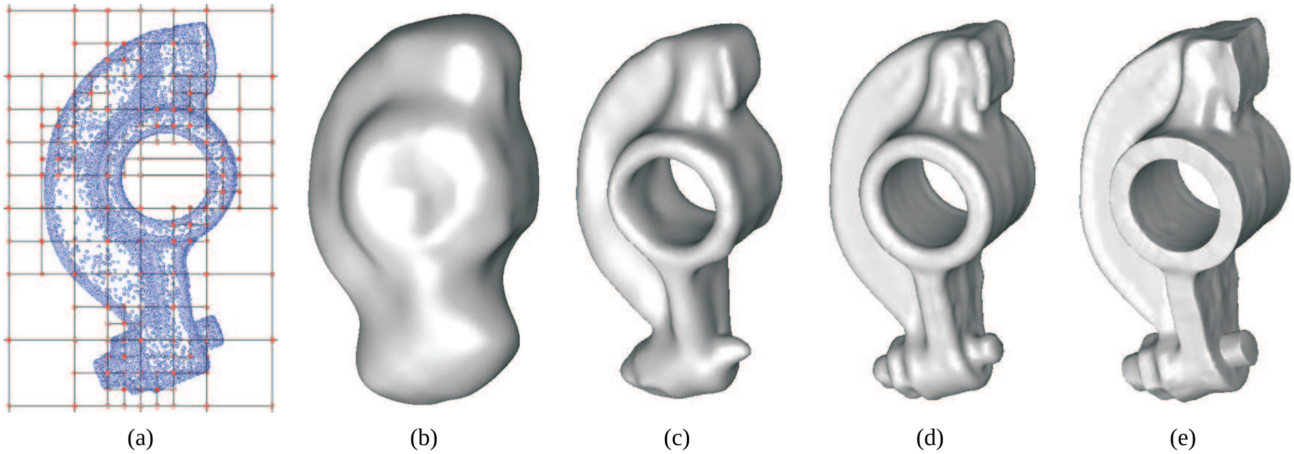

Figure 1. Mesh reconstruction of the Rocker-Arm model. The figure shows the data points and the T-mesh in (a), an intermediate T-spline level set during the evolution in (b), the final T-spline level set (the initial mesh) in (c), the displaced mesh in (d), and the final mesh (after bilateral filtering)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

presents a displaced surface representation based on a manifold structure.

Extracting sharp features from 3D data is important [50], but difficult due to the feature-insensitive sampling and the noise of the given data. Many approaches have been proposed to address this problem [22, 25, 54, 36]. As a non-iterative scheme for edge-preserving smoothing, the Bilateral filter is used in [17] and [28] to denoise a given surface. The author in [53] 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.

1.2 Proposed Approach and Contribution

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 propose a three-phase process to perform this reconstruction:

1. *Initial mesh generation*. In the first phase, we use an evolution process 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 for the displacement mapping. A high-quality initial mesh (with accurate normals) is generated from the implicit function by using the marching triangulation [21] method.

2. *Displacement mapping*. In the second phase, we produce a smooth scalar displacement field, which is computed by projecting each data point to the initial mesh and using Gaussian filtering. Small geometric features are then constructed by the displacement mapping of the mesh along the normal direction, which is guided by the smooth gradient vector field of the implicit function.

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

Our method combines two types of representations: the implicit T-spline level set and the mesh. This combination strategy makes our method benefit from both representations. The evolution of the T-spline level set is able to capture the complex topology of the noisy data, where some existing holes also can be filled. The mesh helps to produce detailed sharp features by using a data-driven displacement mapping, and a feature-preserving geometric filtering. Compared with other existing approaches for similar purposes, our method also shows the following advantages:

1) The use of two representations can improve and speed up some geometric computations in the algorithm. For example, with the help of implicit representation, topological changes are efficiently dealt with during the evolution, and a high-quality initial mesh (with accurate normals) can be obtained. Also, the projection of the data points to the initial mesh can be efficiently computed from the implicit function 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.

2 Initial Mesh Generation

In this section, we describe how to generate the initial mesh. The input of our algorithm is a set of unorganized (maybe noisy and defected) data points $(\mathbf{p}_k)_{k=1,2,\ldots,n}$, which are scattered over an unknown piecewise smooth surface $S_{\mathbf{p}}$. The initial mesh is generated from a smooth base surface S_0, which should capture the complex topology and the outline of the surface $S_{\mathbf{p}}$.

2.1 Base Surface Generation through T-spline Level Set Evolution

In our case, the smooth base surface S_0 is obtained as the zero set of a trivaviate T-spline scalar function f_0. We use T-splines [47] since, on the one hand, the T-spline function is piecewise rational, the implicitly defined surface (T-spline level set) is piecewise algebraic, and its segments can be pieced together with any desired level of differentiability. In this paper, we use cubic T-splines, and the resulted T-spline level set is C^2 continuous in the absence of multiple knots. On the other hand, the use of T-splines leads to a sparse representation of the geometry. The T-spline can be refined locally, by adapting the number and distribution of the degrees of freedom to the particular data.

We use the evolution process described in [55] to find the T-spline function f_0, which defines the base surface S_0. Figure 3 (a) (c) shows an example of this process. First, the T-spline control grid (or T-mesh) is generated according to the distribution of data points through the octree subdivision (see Figure 3 (a)). In order to find the T-spline control coefficients, the method applies an evolution process to an initial level set, which contains all data points inside (see Figure 3 (a)). The evolution is governed by a combination of prescribed, data-driven normal velocities (which are motivated by earlier work in the field of image processing [9], where they have been successfully applied to contour detection and segmentation in images) with additional distance field constraints. The constraints help to avoid additional branches and singularities of the implicit surface, without having to use costly re-initialization steps. Figure 3 (b) shows an intermediate level set during the evolution.

As the result, we obtain the smooth base surface S_0 which captures the topology and the outline of the surface $S_{\mathbf{p}}$ to be reconstructed, except for some detailed geometric features (see Figure 3 (c)). More details can be found in [55]. In particular, that paper also discusses a 'final refinement' step, which can be used to improve the result, especially for noisy data.

In order to speed up the computation of T-spline functions, we use an octree to store the information of associated T-spline control coefficients for any point within the domain of interest. In our experiments, the maximum subdivision depth for T-mesh generation is always set to be no more than 6.

2.2 Initial Mesh Generation through Marching Triangulation

After the smooth base surface S_0 is obtained, we use the Marching Triangulation [21] 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 (such as the Marching Cube [34] algorithm, the dynamic mesh approach [39] and the Dual Contouring [29] method) can be also considered. Figure 3 (c) gives an example of our Triangulation result.

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_0 since f_0 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. Actually it is a non-trivial task to ensure the sufficiency of the seed points such that all components are covered. However, in our case, since the implicit function f_0 defines a good base surface S_0 for the data points $(\mathbf{p}_k)_{k=1,2,\ldots,n}$, we can solve this problem as follows:

1. Project each data point $(\mathbf{p}_k)_{k=1,2,\ldots,n}$ to S_0, get the corresponding foot point $(\mathbf{q}_k)_{k=1,2,\ldots,n}$. (More details will be described later in Section 3.1.)

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 foot point $\mathbf{q}_k$ to the new mesh M_i. If the distance is sufficiently small, then remove $(\mathbf{p}_k)$ from $\mathcal{P}$. (See Section 3.1 for how to compute this distance efficiently.)

6. Repeat steps $4 \sim 5$ until $\mathcal{P} = \emptyset$.

7. Output the generated triangular meshes $\mathcal{M}$.

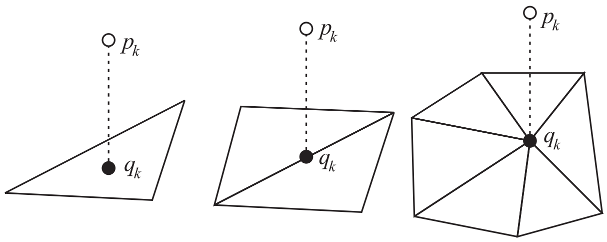

Figure 2. Data points projection to the mesh.

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.

3 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 of M_0,

$$M = M_0 + \mathcal{D} \tag{1}$$

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

3.1 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 turned into the computation of foot points on the surface S_0, which is implicitly defined by the T-spline function f_0. Thus, for each data point $\mathbf{p}_k$, one can compute its foot 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 smooth filtering (smooth parametrization) 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 $(\mathbf{p}_k)_{k=1,2,\ldots,n}$,

   3.1. Initialize the foot point $\mathbf{q}_{k,0} = \mathbf{p}_k$.

   3.2. Using Newton's method to get the updated foot point $\mathbf{q}_{k,i+1} = \mathbf{q}_{k,i} - \frac{f_0(\mathbf{q}_{k,i})}{\|\nabla f_0(\mathbf{q}_{k,i})\|^2} \nabla f_0(\mathbf{q}_{k,i})$.

   3.3. Repeat step 3.2 until $\|\mathbf{q}_{k,i+1} - \mathbf{q}_{k,i}\|$ is sufficiently small. Set $\mathbf{q}_k = \mathbf{q}_{k,i+1}$.

   3.4. Set the signed displacement value $d_k = \mathrm{sign}(\mathrm{f}_0(\mathbf{p}_\mathrm{k})) \cdot \|\mathbf{p}_\mathrm{k} - \mathbf{q}_\mathrm{k}\|$.

   3.5. Find the closest vertex $\mathbf{v}_{k_v}$ of M_0 to the point $\mathbf{q}_k$.

   3.6. From the one-ring neighborhood of $\mathbf{v}_{k_v}$, get the closest triangle(s) $\mathcal{T}_{k_t}$ to $\mathbf{q}_k$.

   3.7. 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 profitted from the implicit representation of the base surface. The searching of closest triangles is now restricted in the one-ring neighborhood of the corresponding closest vertex, as shown in step 3.6. Sometimes the closest point may be lying on an edge or a vertex of the triangular mesh, which means there are more than one closest triangles corresponding to certain data point. In that case, the index of the data point will be associated with each closest triangle. Figure 2 illustrates all of the three cases for data points projection.

3.2 Displaced Mesh

After the scalar displacement field $\mathcal{D}$ is already sampled at each foot 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,

$$\hat{\mathbf{v}} = \mathbf{v} + d(\mathbf{v}) \cdot \mathbf{n}, \tag{2}$$

such that the updated mesh M better approximates the given data points. Note that the normals $\mathbf{n}$ are computed from the implicitly defined T-spline surface 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. 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.

One may ask that why not directly apply an anisotropic filtering method to the displacement field? The displacement mapping acts only along the normals of the implicitly defined base surface, and can not ensure that the updated mesh edges align with the corresponding sharp features.

Actually, we tried to use the (anisotropic) bilateral filtering for displacement mapping, but did not get satisfying sharp edges.

After the displacement mapping, the quality of the displaced mesh may be degraded, although the initial mesh has a high quality through the marching triangulation of T-spline level sets. 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 on the mesh, which can be computed 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 [55] 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 Recovering of 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.

4.1 Bilateral Filter

The Bilateral filter, which was originally proposed in image processing [49, 51], 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 [49], the bilateral filtering for image $I(\mathbf{p})$ at the pixel $\mathbf{p}^*$ is defined as

$$I(\hat{\mathbf{p}}^*) = \sum_{\mathbf{p}_j \in N(\mathbf{p}^*)} \frac{W_c(\|\mathbf{p}^* - \mathbf{p}_j\|) W_s(|I(\mathbf{p}^*) - I(\mathbf{p}_j)|)}{W} I(\mathbf{p}_j), \quad (3)$$

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 [17, 28]. The author in [53] 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 faithfully represent the given data points.

4.2 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

$$F(\mathcal{V}(\tau)) = E_{dist}(\mathcal{V}(\tau)) + \omega E_{bila}(\mathcal{V}(\tau)) \rightarrow \min, \quad (4)$$

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

$$E_{dist}(\mathcal{V}(\tau)) = \sum_{k=1}^{n} (\dot{\mathbf{q}}_k + \mathbf{q}_k - \mathbf{p}_k)^2, \quad (5)$$

where $\mathbf{q}_k$ on the mesh is the foot 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 $\dot{\mathbf{v}}_j$

$$\dot{\mathbf{q}}_k = \sum_{\mathbf{v}_j \in \phi(\mathbf{p}_k)} \lambda_j \dot{\mathbf{v}}_j, \quad (6)$$

where $\phi(\mathbf{p}_k)$ contains 1, 2, or 3 vertices, corresponding to the three cases illustrated in Figure 2 (a), (b) and (c) respectively, and λ_j are corresponding coefficients of them.

The bilateral energy E_{bila} is defined as

$$E_{bila}(\mathcal{V}(\tau)) = \sum_{i=1}^{m} (\dot{\mathbf{v}}_i + \mathbf{v}_i - \mathbf{v}_i')^2, \quad (7)$$

where $\mathbf{v}_i'$ is the updated position of $\mathbf{v}_i$ according to the bilateral filter [53]

$$\mathbf{v}' = \sum_{\mathcal{T}_j \in N(\mathbf{v})} \frac{W_c(\|\mathbf{v} - \mathbf{c}_j\|) W_s(\|\mathbf{v} - \mathbf{v}_j^*\|)}{W} \alpha_j \mathbf{v}_j^*, \quad (8)$$

Dataset	n_p	n_t	n_v	M_I	e_{avr} (10^{-3}) M_D	M_S	M_I	e_{max} (10^{-3}) M_D	M_S	TL	Run time (s) 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
Bunny	6078	1638	8265	2.03	0.43	–	16.72	7.74	–	37.88	2.32	0.33	–
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
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.

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 (3). See [53] for more details.

Combining (4), (5), (6) and (7), the minimizer of (4) 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 bihedral angle and a large approximation error, while keeping $\dot{\mathbf{v}} = 0$ for other vertices. Here, we use the same method as indicated in [53] 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 evolution process can be fast computed.

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

$$e_{avr} = \frac{1}{n}\sum_{k=1}^{n} \|\mathbf{q}_k - \mathbf{p}_k)\|. \tag{9}$$

4.3 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 2.2).

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* [7] operation can be used to remove degenerate triangles after the bilateral evolution stops.

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. (9). 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 features, and has been shown in Fig. 1. The T-spline level

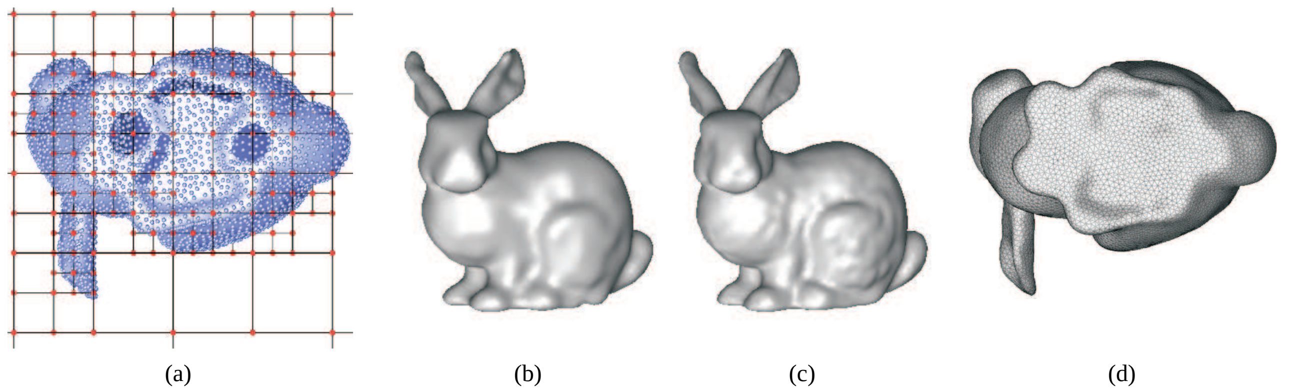

Figure 3. Mesh reconstruction of the Bunny model. The figure shows the data points and the T-mesh in (a), the initial mesh in (b), the displaced mesh in (c), and the bottom view of the mesh in (d).

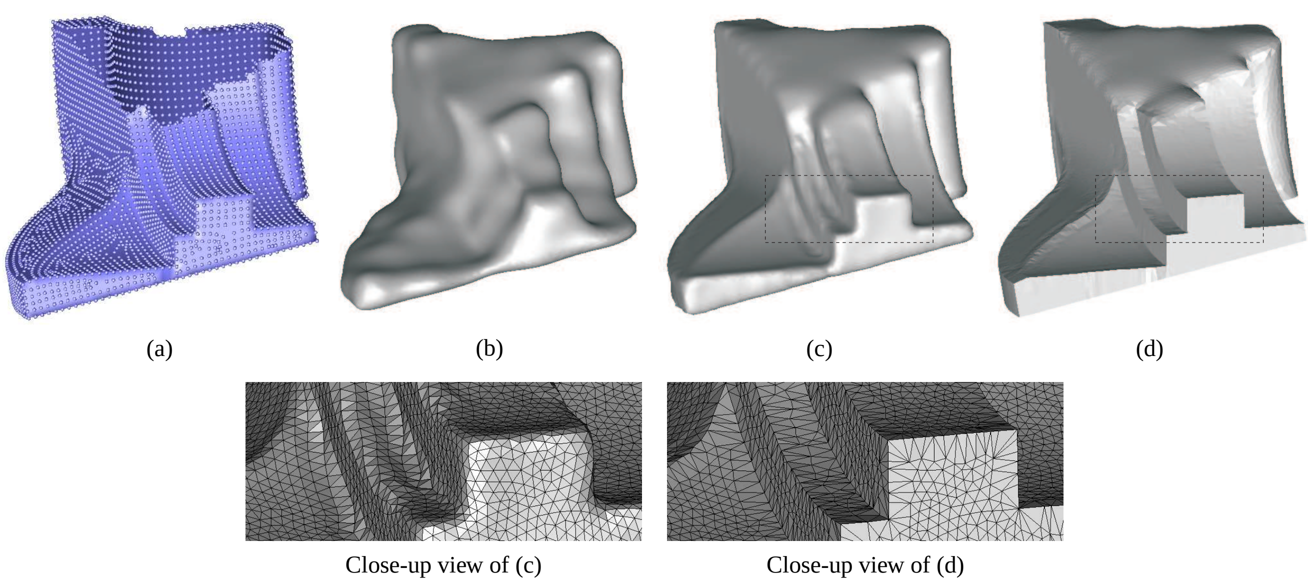

Figure 4. Mesh reconstruction of part of the Fandisk model. The figure shows the data points in (a), the initial mesh in (b), the displaced mesh in (c), and the final mesh (after bilateral filtering) with sharp features in (d).

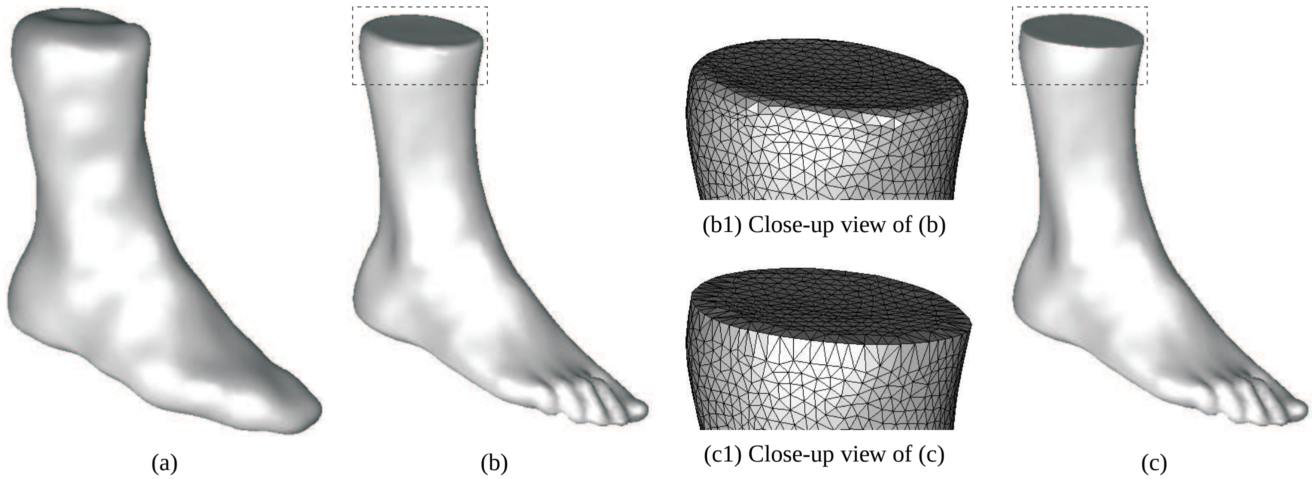

Figure 5. Mesh reconstruction of the Foot model. The figure shows the initial mesh in (a), the displaced mesh in (b), and the final mesh (after bilateral filtering) with sharp features in (c).

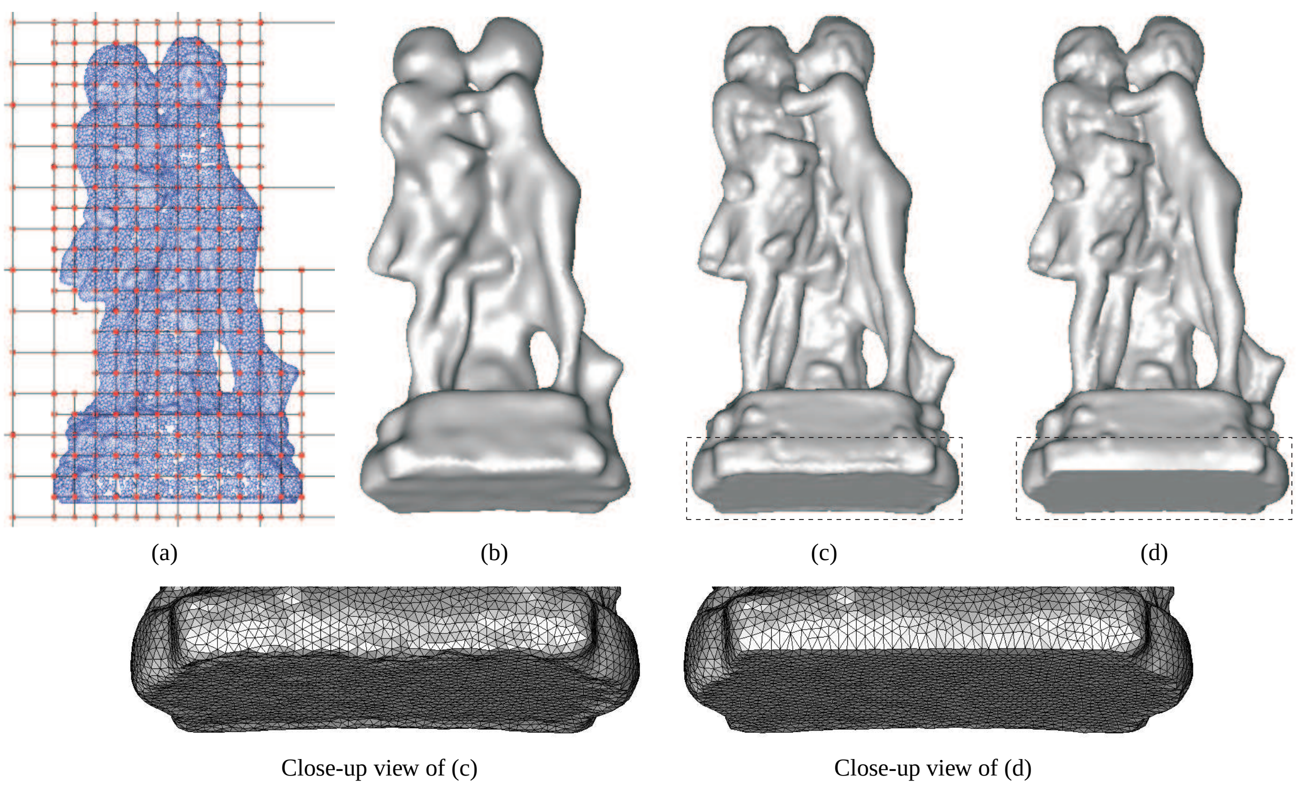

Figure 6. Mesh reconstruction of the Sculpture model. The figure shows the data points and the T-mesh in (a), the initial mesh in (b), the displaced mesh in (c), and the final mesh (after bilateral filtering) with sharp features in (d).

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 at the same time, the sharp edges are recovered, as shown in (e).

Example 2. The second example is the Bunny model with holes, as shown in Fig. 3. Since the data points are already well approximated by the displacement mapping, the last phase of our algorithm (recovering of sharp features) is discarded. As shown in (d), the holes on the bottom (cf. (a)) have been closed.

Example 3. The third example is part of the Fandisk model, which was cut by a plane, as shown in Fig. 4. 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 (e). 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 Foot model that has been shown in Fig. 5. After the displacement mapping in (d), the sharp edges can be recovered by the data-driven bilateral evolution, as shown in (e).

Example 5. The last example is a complex sculpture model, as shown in Fig. 6. Through the T-spline level set evolution, the initial mesh with a correct topology is obtained in (c). The updated mesh after displacement mapping is given in (d). Finally, the sharp edges are produced in (e) by using the bilateral evolution.

6 Conclusions

We have introduced a method 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 the implicit function, the non-uniformly sampled and incomplete data can be handled, and the displaced mesh can be efficiently computed. The sharp features of the mesh surface are produced by using a data-driven bilateral evolution. Our method is independent of the specific representation of the implicit function f_0 for the base surface, as long as f_0 is smooth.

Acknowledgments

The point sets used for our experiments were downloaded from on-line 3D scan repositories [1].

References

[1] Level of detail for 3d graphics. *http://lodbook.com/models/*.

[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] 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] 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.

[7] M. Botsch and L. Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 283–290, 2001.

[8] 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.

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

[10] K. Chan, S. Mann, and R. Bartels. World space surface pasting. In *Graphics Interface'97*, pages 146–154, 1997.

[11] 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.

[12] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proc. SIGGRAPH'98*, pages 115–122, 1998.

[13] R. Cook. Shade trees. In *Proc. SIGGRAPH'84*, pages 223–231, 1984.

[14] T. K. Dey and S. Goswami. Tight cocone: a water-tight surface reconstructor. In *Proc. SMI'03*, pages 127–134, 2003.

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

[16] 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.

[17] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):950–953, 2003.

[18] X. Gu, Y. He, and H. Qin. Manifold splines. *Graphical Models*, 68(3):23–254, 2006.

[19] 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.

[20] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proc. SIGGRAPH'00*, pages 95–102, 2000.

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

[22] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH'96*, pages 99–108, 1996.

[23] 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.

[24] 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.

[25] A. Hubeli and M. H. Gross. Multiresolution feature extraction from unstructured meshes. In *Proc. of IEEE Visualization'01*, pages 16–25, 2001.

[26] W. K. Jeong, K. Kähler, J. Haber, and H. P. Seidel. Automatic generation of subdivision surface head models from point cloud data. In *Proc. Graphics Interface*, pages 181–188, 2002.

[27] W.-K. Jeong and C.-H. Kim. Direct reconstruction of a displaced subdivision surface from unorganized points. *Graphical Models*, 64(2):78–93, 2002.

[28] T. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. (Proc. SIGGRAPH'03)*, 22(3):943–949, 2003.

[29] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proc. SIGGRAPH'02*, pages 339–346, 2002.

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

[31] 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.

[32] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proc. SIGGRAPH'96*, pages 313–324, 1996.

[33] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. SIGGRAPH'00*, pages 85–94, 2000.

[34] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. SIGGRAPH'87*, pages 163–169, 1987.

[35] S. Mann and T. Yeung. Cylindrical surface pasting. In *Geometric Modelling*, pages 233–248, 1999.

[36] M.Attene, B. Falcidino, 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.

[37] 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.

[38] 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.

[39] Y. Ohtake, A. Belyaev, and A. Pasko. Dynamic meshes for accurate polygonization of implicit surfaces with sharp features. In *Proc. SMI'01*, pages 74–81. IEEE Computer Society, 2001.

[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] C. E. Scheidegger, S. Fleishman, and C. Sliva. Triangulating point set surfaces with bounded error. In *Symposium on Geometry Processing*, pages 63–72, 2005.

[47] 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.

[48] 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.

[49] S. M. Smith and J. M. Brady. Susan ł a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, 1997.

[50] 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.

[51] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. ICCV'98*, pages 839–846. IEEE Computer Society, 1998.

[52] L. Velho, J. Gomes, and L. H. Figueiredo. *Implicit Objects in Computer Graphics*. Springer Verlag, New York, 2002.

[53] C. Wang. Bilateral recovering of sharp edges on feature-insensitive sampled meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):629–639, 2006.

[54] 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.

[55] 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. IEEE Computer Society, 2006.

[56] S.-H. Yoon. A surface displaced from a manifold. In *GMP*, 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.