Modeling and 3D Object Reconstruction by Implicitly Defined Surfaces with Sharp Features

Xinghua Song[a,b], Bert Jüttler[b]

[a] *University of Science and Technology of China*
[b] *Johannes Kepler University Linz, Austria*

Abstract

We propose a new method for describing sharp features (i.e., edges and vertices) of implicitly defined surfaces. We consider an initial implicitly defined surface, which is represented as the zero set of a C^1 smooth scalar field with non–vanishing gradients. In order to represent sharp edges and vertices, this surface is augmented by adding new types of implicit representations, called edge descriptors and vertex descriptors. They are defined with the help of the distance field of edge curves. In our implementation, we use circular splines to describe these edge curves, since they support a fast and non-iterative closest point computation.

After adding the edge and vertex descriptors to the initial scalar field, the zero set of the augmented function contains the sharp features. We apply the new representation to surface modelling by implicitly defined surfaces with sharp features and to object reconstruction. In the latter case we describe an algorithm for detecting the sharp curves and vertices of a shape which is given by an unorganized point cloud, which are then approximated by circular splines, in order to define the edge and vertex descriptors.

Key words: Sharp features, circular spline, edge descriptor, surface modeling, surface fitting

1. Introduction

We consider implicitly defined surfaces, i.e., surfaces which are described as the zero set of a C^1 smooth scalar field. Implicitly defined surfaces offer various advantages, such as the non-existence of the parameterization problem for scattered data fitting, repairing capabilities of incomplete data, and simple operations of shape editing [33].

1.1. Related work

Different possibilities for representing the scalar field have been explored in the literature. Besides polynomials [25, 31] and piecewise polynomials [3], these representations include discretized level sets [22, 41], scalar spline functions [14, 26], radial basis functions [7, 21], T-spline level sets [38, 40], Blob-Trees [29] and hybrid representations obtained by blending algebraic surfaces via radial basis functions [20]. Moreover, it is easily possible to combine implicitly defined surfaces in various ways, e.g., by blending them together or by Boolean operations (CSG), see e.g. [4, 15, 23].

Clearly, sharp features (i.e., edges and vertices) of implicitly defined surfaces can be modelled by combining several implicitly defined objects by Boolean operations. However, as a potential problem with this approach, the precise location of the sharp features is not explicitly available. This information, which is extremely helpful for guiding polygonization algorithms such as marching triangles [9], is only implicitly contained in the description of the initial objects.

This approach (via Boolean operations) also requires a decomposition of the geometric objects into patches, which are then combined using Boolean (or other) operations. If implicitly defined surfaces are used for geometry reconstruction from unorganized point cloud data, then it is not always easy to find such a decomposition of the given data.

Even for more general representations, the extraction of sharp features from 3D data and the modeling of objects with features is clearly an important and challenging problem [32]. Difficulties arise due to the feature-insensitive sampling and the noise of the given data. Various approaches have been proposed to address this problem, mostly for surfaces which are described by triangular meshes [2, 11, 36].

In particular, the so–called bilateral filter for feature-preserving mesh denoising has been proposed [8, 12, 18, 34] and it was also successfully used in our previous work [40]. In a recent paper, the fitting of Loop subdivision surfaces to data sampled from objects with sharp features has been discussed in [16].

1.2. Our contribution

We propose a new approach for feature modelling and reconstruction with implicitly defined surfaces. As the main difference to the use of Boolean operations, our approach works with explicit parametric representations of the curves which define the sharp edges.

We assume that an initial surface is already available, which describes the desired geometry very well, except for the vicinity of the sharp features. The features are then added to the object by augmenting the underling scalar field with a novel implicit representation: edge descriptors and vertex descriptors.

Consequently, we decouple the description of the features from the representation of the base geometry. We demonstrate

the potential of the new representation for sharp features by discussing shape modelling and the problem of surface reconstruction from unorganized point cloud data.

In the shape modeling part, given an initial implicitly defined surface, we define the edge descriptor functions using certain spatial curves on the offset surface. After adding the edge functions to the initial surface, an augmented surface with sharp edges at the spatial curves is obtained. By specifying suitable range and magnitude functions of the edge descriptors, we are able to edit the sharp features on the new surface.

In the shape fitting part, an initial implicitly defined surface is generated by using an existing algorithm (the method of [13, 14] in our case, but other methods can also be used). Then the sharp edges and vertices in the initial surface are detected by a triangulation and edge decimation algorithm. We add the edge descriptors, which are defined for the sharp edges and vertices, to this initial implicit function to get a new algebraic function. After an evolution process is used for fitting the edge curves, this new function represents the sharp features well.

1.3. Outline

The remainder of this paper is organized as follows. Section 2 summarizes the necessary background information concerning implicitly defined surfaces and circular splines. In particular, we show that circular splines support a particularly simple and efficient evaluation of the closest distance of a general query point to the curve.

The third section introduces implicitly defined edges and vertices. Edge descriptors are defined with the help of the distance field of a given space curve, which is referred to as the edge curve. We analyze the smoothness of the edge descriptors and show how to choose their free parameters. Further we address the problem of blending several edge descriptors whose edge curves share a common vertex.

Section 4 discusses modelling with implicit surfaces. Besides adding regular features to existing smooth surfaces, we show how to obtain vertices with only two incoming edges, and edges with "dead ends", i.e., with end points within a smooth surface.

The next section is devoted to the reconstruction of surfaces with sharp features from unorganized point cloud data. We briefly summarize an existing method for implicit surface fitting and show how to generate edge and vertex descriptors by fitting circular spline curves to the points which represent the sharp features. The algorithm is illustrated by four examples.

Finally we summarize this paper and conclude with suggestions for further research.

2. Preliminaries

We recall the notion of implicitly defined surfaces and summarize the construction and rational spline representation of circular arc spline curves.

2.1. Implicitly defined surfaces

An implicitly defined surface is the zero level set

$$\mathcal{Z}(f) = f^{-1}(0) = \{\mathbf{x} \in \Omega \mid f(\mathbf{x}) = 0\} \tag{1}$$

of a real function (a scalar field) $f : \Omega \to \mathbb{R}$ where $\Omega \subseteq \mathbb{R}^3$ is the domain of f. If f is a C^k smooth function whose gradient ∇f does not vanish in Ω, then the surface is also C^k smooth.

In many situations, the scalar field f is represented by a linear combination of certain basis functions, such as radial basis functions, T-splines or products of univariate B-splines. In this paper we adopt the latter approach. Clearly, the edge modelling techniques described below can be used for other types of scalar fields, too.

More precisely, let

$$f(\mathbf{x}) = \sum_{i,j,k} d_{rst}\, M_i(x_1)\, N_j(x_2)\, L_k(x_3), \tag{2}$$

where $\mathbf{x} = (x_1, x_2, x_3)$, be a trivariate tensor product spline function of tri-degree (m', n', l') with real coefficients d_{rst}. The domain Ω is an axis-aligned box. The basis functions $\{M_i\}_{i=1}^{m}$, $\{N_j\}_{j=1}^{n}$ and $\{L_k\}_{k=1}^{l}$ are B-splines of degree m',n' and l' with respect to certain given knot sequences. The extremal coordinates of the box are degree-fold boundary knots, and the inner knots are chosen uniformly. The zero level set $\mathcal{Z}(f)$ will be called an *algebraic B-spline surface*.

In order to simplify the notation, the coefficients and the basis functions are gathered in two column vectors

$$\mathbf{d} = (\ldots, d_{ijk}, \ldots)^\top \text{ and } \mathbf{b}(\mathbf{x}) = (\ldots, M_i(x_1)N_j(x_2)L_k(x_3), \ldots)^\top \tag{3}$$

respectively. Then we can rewrite f as

$$f(\mathbf{x}) = \mathbf{d}^\top \mathbf{b}(\mathbf{x}) \tag{4}$$

and the gradient of f is the row vector

$$(\nabla f)(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \right)\Bigg|_{\mathbf{x}} = \mathbf{d}^\top (\nabla \mathbf{b})(\mathbf{x}). \tag{5}$$

2.2. Circular splines and distance computation

A circular arc in $\mathbb{R}^3$ has the standardized rational Bézier representation

$$\mathbf{y}(u) = \frac{(1-u)^2\mathbf{b}_0 + 2u(1-u)\omega\mathbf{b}_1 + u^2\mathbf{b}_2}{(1-u)^2 + 2u(1-u)\omega + u^2}, \quad u \in [0,1], \tag{6}$$

with the control points $\mathbf{b}_i$, where $\mathbf{b}_1$ lies in the bisector plane of the line segment $\mathbf{b}_0\mathbf{b}_2$, i.e.,

$$\|\mathbf{b}_0 - \mathbf{b}_1\| = \|\mathbf{b}_1 - \mathbf{b}_2\|. \tag{7}$$

The weight ω satisfies $\omega = \cos\phi$, where

$$2\phi = \pi - \angle(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2) \tag{8}$$

is the sweep angle of the circular arc. We shall assume that the sweep angle is less than π. Longer arcs are split into shorter segments.

A tangent continuous circular spline curve $\mathbf{c} : t \mapsto \mathbf{c}(t)$ with k segments, knots $(t_j)_{j=0}^k$ and domain $I = [t_0, t_k]$ is a sequence of circular arcs $\{\mathbf{y}_j\}_{j=1}^k$ of the form (6) with local parameters

$$u_j = \frac{t - t_{j-1}}{t_j - t_{j-1}} \tag{9}$$

which are pieced together with C^1 continuity.

Compared to other representations of space curves, such as polynomial splines, the use of circular splines offers various potential benefits, such as the availability of an exact arc-length parameterization. As an important advantage, it is possible to perform closest point computation by a simple non-iterative method.

More precisely, given a query point $\mathbf{p}$, we want to find the closest point $\mathbf{f} = \mathbf{x}(t^*(\mathbf{p}))$ on the circular spline curve, where

$$t^*(\mathbf{p}) = \arg\min_{t \in I} \|\mathbf{p} - \mathbf{x}(t)\| \tag{10}$$

The shortest distance from a data point to the curve is the minimum of the shortest distances to all arcs. Consider a fixed circular arc $\mathbf{y}_j$, and let $\mathbf{m}_j$ be its center. The parameter t^* which potentially realizes the shortest distance can be computed as follows.

(1) We project $\mathbf{p}$ orthogonally into the plane which contains the arc $\mathbf{y}_j(t)$. The projected query point is denoted with $\mathbf{p}^0$.

(2) If $\mathbf{p}^0$ lies inside the sweep angle of $\mathbf{y}_j(t)$, then the candidate values t^* of the global curve parameter are found by computing the associated global parameter of the root(s) of the equation

$$0 = (\mathbf{p}^0 - \mathbf{m}_j) \cdot \mathbf{y}_j'(u_j), \quad u_j \in [0, 1], \tag{11}$$

where the prime $'$ denotes the derivative with respect to the local curve parameter u_j. Otherwise, the shortest distance is not realized by this circular arc (but see the remark about open curves).

(3) If a local candidate value of the closest point parameter has been found, the corresponding value of the global parameter value is found as $t^* = t_{j-1} + u_j(t_j - t_{j-1})$.

In the second step of the algorithm, we substitute (6) into (11) and get simply a *quadratic* equation

$$\begin{aligned} 0 &= (\mathbf{p}^0 - \mathbf{m}_j) \cdot [(1 - u_j)^2\, \omega(\mathbf{b}_1 - \mathbf{b}_0) + \\ &+ u_j(1 - u_j)\,(\mathbf{b}_2 - \mathbf{b}_0) + u_j^2\, \omega(\mathbf{b}_2 - \mathbf{b}_1)]. \end{aligned} \tag{12}$$

Since the sweep angle is assumed to be less than π, this equation possesses a unique root $u_j \in [0, 1]$

In order to keep the algorithm as simple as possible, we compute the closest point by first finding the closest points in all circular arcs, and then selecting the point with the minimum distance among them. One may improve the efficiency of the algorithm by using a suitable hierarchy of bounding volumes.

In the case of an open spline curve, the closest point of the given query point $\mathbf{p}$ can also be one of the two boundary points. Hence, the two end points have to be checked separately.

3. Implicitly defined edges and vertices

In order to model the sharp features on an implicitly defined surface, we introduce a new type of implicit representations: edge descriptor functions.

3.1. Edge descriptors

Consider a C^1 smooth space curve $\mathbf{c} : I \to \mathbb{R}^3$ with parameter t and domain $I = [t_0, t_1] \subset \mathbb{R}$, which is a collection of C^2 segments. This curve will be called the *edge curve*. For instance, $\mathbf{c}$ may be a circular spline curve, as described in Section 2.2. Let

$$\mathcal{N}(t) = \{\mathbf{x} \in \mathbb{R}^3 : \ \mathbf{c}'(t) \cdot (\mathbf{c}(t) - \mathbf{x}) = 0\} \tag{13}$$

be the normal plane of the edge curve at $\mathbf{c}(t)$.

For any point $\mathbf{x} \in \mathbb{R}^3$, let t^* be the mapping which assigns to $\mathbf{x}$ the parameter value of the closest point on $\mathbf{c}$,

$$t^*(\mathbf{x}) = \arg\min_{t \in I} \|\mathbf{x} - \mathbf{c}(t)\|. \tag{14}$$

The mapping $t^* : \mathbb{R}^3 \to I$ defines a scalar field. Except for the domain boundaries t_0, t_1 of I, the level set $t^* = t$ of this field is contained in the normal plane $\mathcal{N}(t)$.

In addition, let $a : I \to \mathbb{R}$ and $r : I \to \mathbb{R}$ be two C^1 smooth functions defined on the same interval. They will be called the *magnitude function* and the *radius function*, respectively.

Definition 1. *The **edge descriptor** g which is determined by the edge curve $\mathbf{c}$, the magnitude function a and the radius function r is the function*

$$g : \boldsymbol{x} \mapsto (a \circ t^*)(\boldsymbol{x}) \cdot \Big((r \circ t^*)(\boldsymbol{x})) - \|\boldsymbol{x} - (\boldsymbol{c} \circ t^*)(\boldsymbol{x})\|\Big)_+^2 \tag{15}$$

where $(y)_+ = \max\{y, 0\}$ for all $y \in \mathbb{R}$.

The radius function r specifies the influence region of the edge descriptor. The support of the edge descriptor contains all points $\mathbf{x}$ whose distance to the edge curve does not exceed $r(t^*(\mathbf{x}))$. The magnitude function a controls the values of the edge descriptor along the edge curve, which is equal to $a\, r^2$. The same idea has successfully been used for implicitly defined planar curves with sharp corners, see [39].

We now analyze the smoothness of the edge descriptor g.

Let ρ_0 denote the minimum curvature radius of $\mathbf{c}$. We assume that the pipe surface with a certain constant radius $\rho \le \rho_0$, which is closed by adding hemispherical caps at the two end points of the curve (which are omitted in the case of closed curves) does not have global self-intersections. Let $\mathcal{P}$ be the interior of this pipe surface. The scalar field defined by t^* is then continuous in $\mathcal{P}$.

However, the scalar field defined by t^* is not always differentiable. Discontinuities of the gradient ∇t^* occur in the normal planes $\mathcal{N}(t)$ at points where $\mathbf{c}$ is not C^2.

On the other hand, even in the presence of discontinuities of the second derivative of the edge curve $\mathbf{c}$, the scalar field

$$\mathbf{x} \mapsto \|\mathbf{x} - (\mathbf{c} \circ t^*)(\mathbf{x})\| \tag{16}$$

which is defined by the closest distance of a point $\mathbf{x}$ to the edge curve, is C^1 smooth in the interior $\mathcal{P}$ of the pipe surface, except for the points on the edge curve itself.

Finally, if $h : I \to \mathbb{R}$ is any C^1 smooth function whose first derivative vanishes at the points where $\mathbf{c}$ is not C^2, then $(h \circ t^*)$ is C^1 smooth in $\mathcal{P}$. Consequently, the edge descriptor g is C^1 smooth in the interior of the pipe surface, except for the points on the edge curve, provided that the first derivatives of a and r vanish at the points where the edge curve is not C^2.

If the value of the radius function is smaller than the radius ρ of the pipe surface $\partial\mathcal{P}$, then the support of g is a subset of $\mathcal{P}$. The edge descriptor g is then globally C^1, except

(1) for the points of the edge curve and

(2) for points in the normal planes $\mathcal{N}(t)$ of the edge curve at discontinuities of the second derivative.

Once again, if the derivatives of a and r vanish at the points where the edge curve $\mathbf{c}$ is not C^2, then the latter discontinuities of ∇g are not present.

In the remainder of the paper we shall assume that the assumption $r \leq \rho$ regarding the radius function is always satisfied.

3.2. Implicitly defined edges

By adding an edge descriptor to the function f which defines the surface $\mathcal{Z}(f)$, we obtain a new function $F = f + g$. In order to model sharp features using this new function, we need to construct the curve $\mathbf{c}(t)$ and choose suitable parameters a and r for g.

Lemma 2. *If*

$$a(t) = -\frac{(f \circ \boldsymbol{c})(t)}{r^2}, \tag{17}$$

then the curve $t \mapsto \boldsymbol{c}(t)$ lies on the surface $\mathcal{Z}(F)$.

Proof. Since every point $\mathbf{c}_0 = \mathbf{c}(t)$ of the edge curve satisfies $\mathbf{c}(t^*(\mathbf{c}_0)) = \mathbf{c}_0$, we get

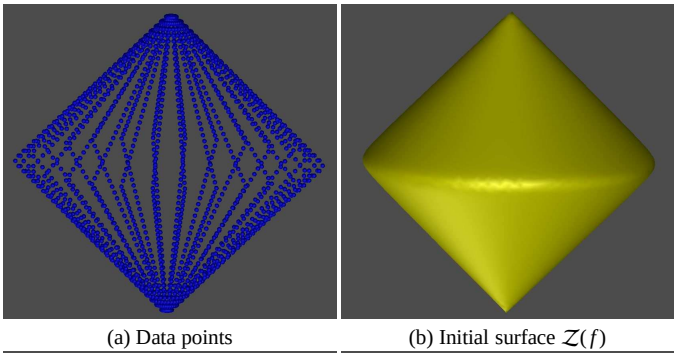
$$F(\mathbf{c}_0) = f(\mathbf{c}_0) + (a \circ t^*)(\mathbf{c}_0) \cdot r^2 = f(\mathbf{c}_0) - \frac{f(\mathbf{c}_0)}{r^2} r^2 = 0. \tag{18}$$

Consequently, the curve $\mathbf{c}(t)$ is contained in $\mathcal{Z}(F)$. □

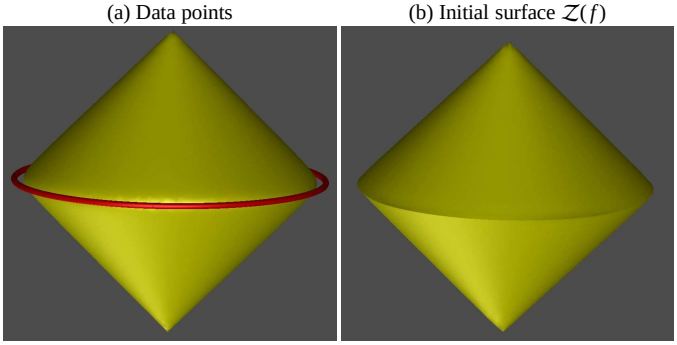
Thus, by choosing the value of a according to (17), one can modify $\mathcal{Z}(f)$ such that it contains the edge curve. We demonstrate this observation in the following example.

Example 3. Fig. 1 shows a first example. The smooth implicitly defined surface shown in (b) is an algebraic B-spline surface which was generated by approximating sample points taken from two circular cones with common base (a). Using an edge descriptor which is based on a single circle as the edge curve (c), one obtains an implicitly defined surface $\mathcal{Z}(f + g)$ with a sharp edge (d).

Next we analyze the behaviour of the surface $\mathcal{Z}(f + g)$ along the edge curve. We shall assume that the magnitude function a of the edge descriptor is chosen according to (17). For any



(a) Data points (b) Initial surface $\mathcal{Z}(f)$



(c) Edge curve (d) Final surface $\mathcal{Z}(F)$

Figure 1: By adding an edge descriptor g to f, a sharp edge on the smooth implicitly defined surface $\mathcal{Z}(f)$ is generated.

value of the curve parameter $t \in I$, let $\mathcal{N}(t)$ be the normal plane of the edge curve at the point $\mathbf{c}(t)$, and consider the gradients

$$(\nabla_{\mathcal{N}(t)} f)(\mathbf{x}) = (\nabla f)(\mathbf{x}) - \frac{(\nabla f)(\mathbf{x}) \cdot \mathbf{c}'(t)}{\mathbf{c}'(t) \cdot \mathbf{c}'(t)} \mathbf{c}'(t) \tag{19}$$

of the restriction of f to $\mathcal{N}(t)$.

Proposition 4. *If the radius r of the edge descriptor g satisfies*

$$r(t) \geq \frac{2|(f \circ \boldsymbol{c})(t))|}{\|(\nabla_{\mathcal{N}(t)} f)(\boldsymbol{c}(t))\|}, \tag{20}$$

then the surface $\mathcal{Z}(F)$ passes through $\boldsymbol{c}(t)$ and the intersection curve

$$\mathcal{L}(t) = \mathcal{Z}(f) \cap \mathcal{N}(t) \tag{21}$$

of the surface and the normal plane possesses a corner point with two tangent directions $\boldsymbol{T}_1(t)$ and $\boldsymbol{T}_2(t)$, which are different provided that the inequality is strictly satisfied. Otherwise, if (20) *is violated, then $\boldsymbol{c}(t)$ is an isolated point of $\mathcal{L}(t)$.*

Proof. We consider the restrictions $\bar{F}$, $\bar{f}$ and $\bar{g}$ of F, f and g to the normal plane $\mathcal{N}(t)$. In particular, we consider the surfaces which are defined by the graphs of $\bar{F}$, $\bar{f}$ and $\bar{g}$. The graph of $\bar{F}$ intersects the plane in the curve $\mathcal{L}(t)$. The graph of $\bar{g}$ has a singular point at $\mathbf{c}(t)$, where it touches a circular cone with slope $|2ar|$. On the other hand, the slope of $\bar{f}$ at $\mathbf{c}(t)$ is $\|(\nabla_{\mathcal{N}(t)} f)(\mathbf{c}(t))\|$. If the slope of the circular cone is smaller than the slope of $\bar{f}$, then $\mathcal{L}$ has a corner point with two directions $\mathbf{T}_1$ and $\mathbf{T}_2$. The inequality (20) is now obtained by using (17). □

We visualize this observation in the following example.

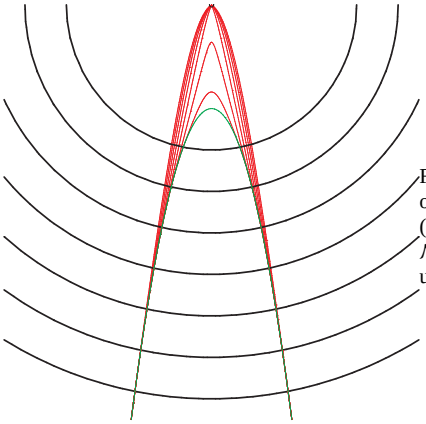


Figure 2: Intersection curves of $\mathcal{Z}(f)$ (green) and $\mathcal{Z}(f+g)$ (red) with the normal plane $\mathcal{N}(t)$ at $\mathbf{c}(t)$ for different values of r.

Example 5. Fig. 2 shows the situation in the normal plane of the edge curve. The green curve is the intersection $\mathcal{Z}(f)\cap\mathcal{N}(t)$, and the red curves are the intersections $\mathcal{Z}(f+g)\cap\mathcal{N}(t)$ for different values of the radius r. The radii are visualized by the circles. If the radius is too small, then $\mathbf{c}(t)$ is an isolated point. For sufficiently large values of r, the value of r controls the angle between the tangents at the corner $\mathbf{c}(t)$ which tends to π when r is increased.

Due to the smoothness and differentiability of g, the surface $\mathcal{Z}(f+g)$ is C^1 smooth everywhere, except along the edge curve $\mathbf{c}$ and in the normal planes of the points where $\mathbf{c}$ is not C^2. Moreover it can be shown that the two normals of the surface along the edge curve $\mathbf{c}$ then depend continuously on t, again provided that the edge curve of C^2 smooth.

3.3. Implicitly defined vertices

We extend the notion of edge descriptors to sharp vertices of an object which is represented by an implicitly defined surface. To this end, we shall consider each such vertex as a point where several edge curves meet.

Suppose that $\mathbf{v}$ is the common end point of the edge curves $\{\mathbf{c}_j\}_{j=1}^{\nu}$, enumerated according to their adjacency[1], where every edge curve has an associated edge descriptor g_j. Let $\mathbf{t}_j$ be the tangent vector of curve $\mathbf{c}_j$ at $\mathbf{v}$, oriented such that it points away from the vertex. We consider the half-planes which are spanned by the tangent directions and the gradient of f at ν,

$$\mathbf{L}_j = \{\mathbf{v} + \lambda\mathbf{t}_j + \mu(\nabla f)(\mathbf{v}) : \ \lambda,\mu\in\mathbb{R},\ \lambda > 0\}. \qquad (22)$$

They intersect in the normal of the level set $\mathcal{Z}(f - f(\mathbf{v}))$ of f through the vertex $\mathbf{v}$,

$$\{\mathbf{v} + \mu(\nabla f)(\mathbf{v}) : \ \mu\in\mathbb{R}\}. \qquad (23)$$

We denote the bisector half–plane of every two consecutive half-planes planes $\mathbf{L}_j$ and $\mathbf{L}_{j+1}$ with $\mathbf{B}_j$, where $\mathbf{L}_{\nu+1} = \mathbf{L}_1$. For each bisector half–plane we define two blending half–planes $\mathbf{B}_j^l$ and $\mathbf{B}_j^r$ which include a user-specified small angle θ with $\mathbf{B}_j$ and also contain the line (23).

Let $\mathcal{B}$ be the ball with radius $R > \rho$ and center $\mathbf{v}$. The radius R of the ball $\mathcal{B}$ is chosen such that the intersections of the ν pipe surfaces $\mathcal{P}_j$ around the ν edge curves with the sphere $\partial\mathcal{B}$ are mutually disjoint. The blending half–planes divide this ball into ν primary wedges $\mathcal{W}_j^p$ and ν blending wedges $\mathcal{W}_j^b$, where the primary wedges contain the tangent directions $\mathbf{t}_j$ and the blending wedges contain the bisector half–planes $\mathbf{B}_j\cap\mathcal{B}$.

For each blending wedge $\mathcal{W}_j^b$, we choose two blending functions β_j and γ_j which are positive in the interior of $\mathcal{W}_j^b$ and whose value and gradient vanish on one of the two blending half–planes, respectively. For instance, one may choose the squared linear equations describing the blending half–planes.

Definition 6. *The **vertex descriptor** $\bar{g}$ of the common vertex of the ν edge curves with edge descriptors g_j is obtained by blending together the ν edge descriptors,*

$$\bar{g}(\boldsymbol{x}) = \begin{cases} g_j(\boldsymbol{x}) & \text{if } \boldsymbol{x}\in\mathcal{W}_j^p \\ \dfrac{\beta_j(\boldsymbol{x})\,g_j(\boldsymbol{x}) + \gamma_j(\boldsymbol{x})\,g_{j+1}(\boldsymbol{x})}{\beta_j(\boldsymbol{x}) + \gamma_j(\boldsymbol{x})} & \text{if } \boldsymbol{x}\in\mathcal{W}_j^b \end{cases}, \qquad (24)$$

where $j = 1,\ldots,\nu$.

By adding the vertex descriptor to the scalar field f describing an initial implicit surface $\mathcal{Z}(f)$, we obtain a new scalar field whose zero set describes a surface with a vertex. More precisely, we define F as

$$F = f(\mathbf{x}) + \begin{cases} \sum_{j=1}^{\nu} g_j(\mathbf{x}) & \text{if } \mathbf{x}\in\mathbb{R}^3\setminus\mathcal{B} \\ \bar{g}(\mathbf{x}) & \text{if } \mathbf{x}\in\mathcal{B} \end{cases} \qquad (25)$$

and consider the resulting zero level set $\mathcal{Z}(F)$. Clearly, it is possible to add several vertex descriptors.

The effect of vertex descriptors is demonstrated in the following example.

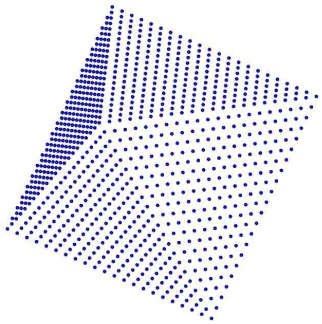
Example 7. We sampled the data points from the upper part of an octahedron and fitted them with an algebraic B-spline surface. After adding the edge and vertex descriptors, we obtain a faithful reconstruction of the object, see Fig. 3.

As an alternative, one might define vertex descriptors by replacing the distance to the edge curve on (15) with the minimum distance to all edge curves. However, the latter distance function is not C^1 smooth in the bisectors of the edge curves, and hence this definition gives vertices with additional "phantom" edges.

4. Modeling surfaces with sharp features

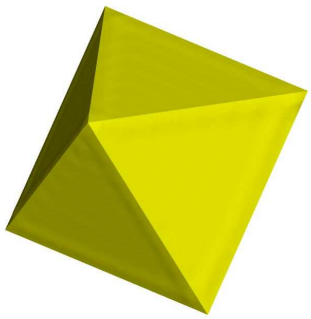
Given an implicitly defined surface $\mathcal{Z}(f)$, we model an implicitly defined surface with sharp features by adding edge and vertex descriptors to f, cf. (25). In order to obtain efficient methods for evaluating the edge descriptor functions, we represent all edge curves $\mathbf{c}(t)$ by circular splines. For any query point, we can then easily find the closest point on the edge curve using the non-iterative method described in Section 2.2.

[1] The curves are projected orthogonally in the plane with normal $(\nabla f)(\mathbf{v})$ through ν, and then ordered clockwise.



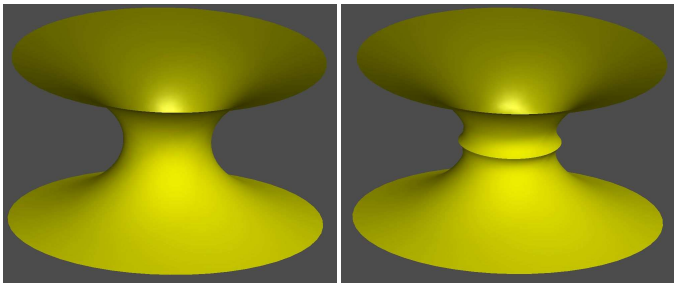
(a) data points

(b) initial surface



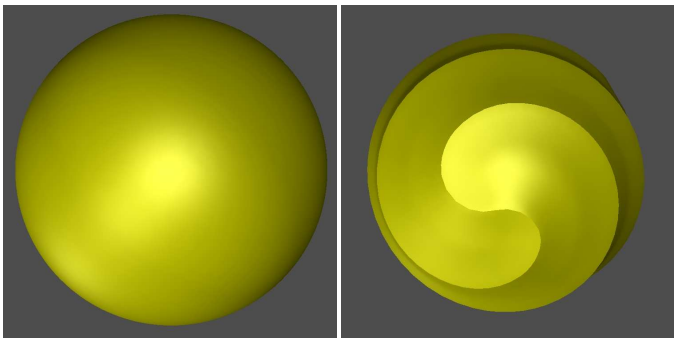
(c) surface with vertex

Figure 3: The effect of a vertex descriptor for four edge curves meeting in a vertex.



(a) initial surface (b) surface with circle feature

Figure 4: Adding a circle feature to the catenoid.



(a) initial surface (b) surface with swirl-shaped feature

Figure 5: Adding an edge descriptor to the sphere.

Clearly, the circular splines are generally only C^1 smooth, and hence the edge descriptors have a non-continuous gradient field not only along the edge curves, but also across the normal planes of the edge curves where the individual arcs are pieced together. However, according to our experiences, the effects of the latter gradient discontinuities are quite small and can be neglected. In addition, we choose the circular splines to lie approximately on offset surfaces of $\mathcal{Z}(f)$ and with a radius function r which does not change much. Thus, both a and r are almost constant along the edge curves, which again reduces the effects of the gradient discontinuities.

If required, these effects can further be decreased by choosing a circular spline with a larger number of segments, where the jumps of the second derivative are reduced. Clearly, there is a trade-off between the effort of the closest point computation, which increases with the number of segments, and the effects of the second derivative discontinuities, which decrease with the number of segments.

The method consists of three steps.

(1) We define circular splines which approximately lie on offset surfaces of $\mathcal{Z}(f)$.

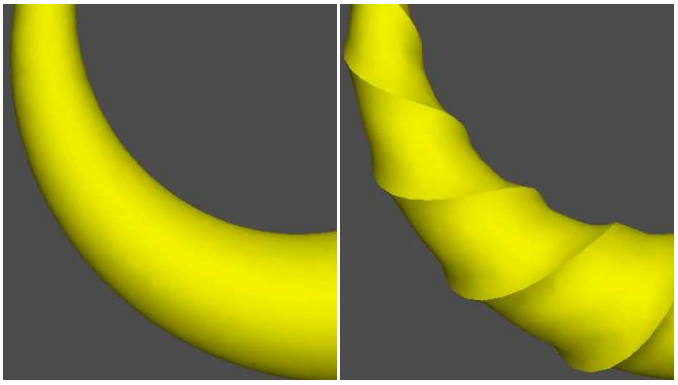
(2) We choose the magnitude functions according to Lemma 2 and the radius functions either as a certain constant multiple of the offsetting distance used in step 1 or such that the edge possesses a pre-defined angle (e.g., $\pi/2$).

(3) We add the edge and vertex descriptors to the initial scalar field f and evaluate the resulting implicitly defined surface $\mathcal{Z}(f)$.

In the second step, one can use any of the well-known techniques for circular spline generation, e.g., biarc interpolation [6, 5, 10, 17, 19, 24, 27, 35]. The performance of the method depends on the particular method for arc spline generation. In our implementation, which relies on equal chord biarc interpolation, we achieve real-time performance. Due to space limitations, we do not describe any details of these methods.

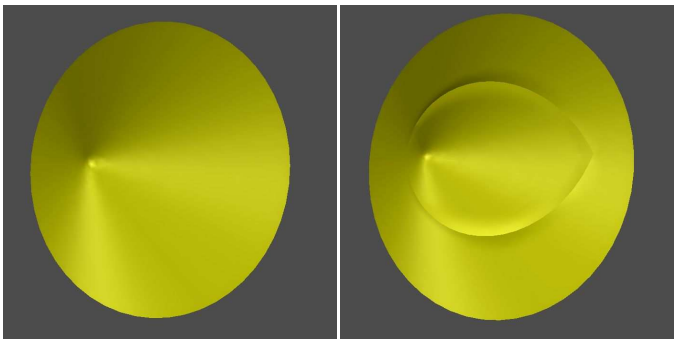
Example 8. We present four examples for surface modeling with sharp features.

1. We add a circle-shaped feature to a surface of revolution which approximately represents a catenoid, see Fig. 4.
2. Starting from a sphere, we add a swirl-shaped feature to this surface. The result is shown in Fig. 5. Since circular splines are able to model exactly spherical curves, the edge curve lies exactly on an offset surface of the initial sphere.
3. We start from a pipe surface with a circular spine curve and a linearly varying radius. The edge curve is chosen as a spiral curve on an offset surface. The initial surface and the surface with the sharp feature are shown in Fig. 6.
4. The edge descriptor technique can also be applied to edges with "dead ends", i.e., to edges which start or end somewhere within a smooth surface. In this case, the edge curve $\mathbf{c}$ starts or ends directly on the initial surface $\mathcal{Z}(f)$, and the radius function r vanishes at this point.
   Similarly, one can model vertices with only two incoming edges. This does not require any adaptation of the concept of the vertex descriptor; it can be used as described in Section 3.3. Please note that the use of a non–C^1–smooth edge



(a) initial surface V(f) (b) surface with twist feature

Figure 6: Adding a twist feature.



(a) initial surface (b) surface with features

Figure 7: Edge and vertex descriptors for "dead ends" and vertices with two incoming edges

curve would not give the desired result, since the scalar field is then not necessarily continuous, not even in a sufficiently small neighborhood of the edge curve.
Starting from an initial surface describing a circular cone, we add a feature with two dead ends and with a vertex with only two edges. The result is shown in Fig. 7.

5. Surface reconstruction

Given unorganized point data with associated normal information, we reconstruct an algebraic B-spline surface, which is augmented by adding edge and vertex descriptors. We give an outline of the method, describe the individual steps in more detail and present several examples.

5.1. Outline

We consider a given unorganized point cloud $\{\mathbf{p}_i\}_{i=1}^N$, which is assumed to represent a shape with sharp features. The following algorithm constructs an algebraic B-spline surface and augments it with edge and vertex descriptors such that the implicitly defined surface $\mathcal{Z}(F)$ represents the shape.

(1) Construct the initial implicitly defined surface $\mathcal{Z}(f)$ by using an existing method to fit the input data points $\{\mathbf{p}_i\}_{i=1}^N$. Generate a triangular mesh $\mathcal{T}_0$ representing $\mathcal{Z}(f)$ using a suitable triangulation algorithm, e.g., marching triangles [9].

(2) Select the data points $\{\mathbf{p}_i^s\}_{i=1}^{\overline{N}}$ which represent the sharp features of the original shape from the input data points by analyzing the dihedral angles of the mesh $\mathcal{T}_0$ and the distances between the input data points and the initial surface $\mathcal{Z}(f)$.

(3) Generate the initial circular splines $\mathbf{c}_i^0$ and the associated edge descriptors g_i^0. Fit the points $\{\mathbf{p}_i^s\}_{i=1}^{\overline{N}}$ representing the features with circular splines $\{\mathbf{c}_i\}_{i=1}^N$ and edge descriptors $\{g_i\}_{i=1}^N$, where N is the number of open and closed edge curves.

(4) Generate the augmented scalar field F by adding the edge and vertex descriptors to f. The surface $\mathcal{Z}(f)$ represents the object with the sharp features.

We shall now describe the individual steps in more detail.

5.2. Implicit surface fitting

Several techniques for reconstructing implicitly defined surfaces from unorganized point cloud data exist. As a major advantage compared to parametric representation, no auxiliary parameterization of the data has to be generated. In our example we use the simple method which has been described in [13, 14], which assumes that each point $\mathbf{p}_i$ is equipped with an associated unit normal $\mathbf{n}_i$.

This method uses three main ingredients to build the objective function. First, the algebraic distance

$$L(\mathbf{d}) = \sum_{i=1}^{N} [f(\mathbf{p}_i)]^2 \tag{26}$$

is used in order to measure the deviation of $\mathcal{Z}(f)$ to the given data. Second, the term

$$N(\mathbf{d}) = \sum_{i=1}^{N} \|\nabla f(\mathbf{p}_i) - \mathbf{n}_i\|^2 \tag{27}$$

expresses the deviation between the surface and the given normals $\mathbf{n}_i$. Third, we use the smoothing term

$$G(\mathbf{d}) = \iiint_{\Omega} f_{11}^2 + f_{22}^2 + f_{33}^2 + 2f_{12}^2 + 2f_{13}^2 + 2f_{23}^2 \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{28}$$

where the indices f_{ij} indicate differentiation with respect to x_i and x_j. By combining the three terms with user-specified positive weights ω_1, ω_2, where $1 > \omega_1 > \omega_2 > 0$ we obtain the objective function

$$F(\mathbf{d}) = L(\mathbf{d}) + \omega_1 N(\mathbf{d}) + \omega_2 G(\mathbf{d}) \to \text{Min} \tag{29}$$

Its minimization leads to a linear system with a symmetric positive definite and sparse (due to the local support of B-splines)

matrix which can be solved efficiently, e.g., using sparse direct solvers. The solution gives the coefficients $\mathbf{d} = (\ldots, d_{ijk}, \ldots)^\top$ of the initial algebraic B-spline surface $\mathcal{Z}(f)$.

Since the marching triangulation method [9] is able to produce a high-quality triangular mesh, we use this method to generate the mesh representation $\mathcal{T}_0$ of the initial implicit surface $\mathcal{Z}(f)$. Other polygonization methods (see e.g. [33]) can also be considered.

5.3. Edge detection and decimation

After generating the initial implicitly defined surface, we need to detect the sharp features of the original shape. More precisely, we generate triangle strips representing these features on the initial mesh $\mathcal{T}_0$, as follows.

(1) Project all the data points $\{\mathbf{p}_i\}_{i=1}^N$ to the initial mesh $\mathcal{T}_0$.

(2) Compute the dihedral angle of every edge in the mesh $\mathcal{T}_0$. If this angle exceeds a user-specified threshold, then we call the edge a sharp edge. If all three edges of a triangle are sharp edges, we call the triangle a candidate triangle.

(3) For every candidate triangle, we compute the average distance between the projected points in this triangle and the associated data points. If this exceeds a user-specified threshold, then we call the triangle a "sharp" triangle. Let $\mathcal{T}_1$ be the mesh which consists of all sharp triangles.

In the next step we generate skeletons of the mesh $\mathcal{T}_0$, i.e., polygons which correspond to the sharp features. These skeletons are generate with the help of the following decimation method.

(1) Mark the edges with at most one neighbouring triangle in $\mathcal{T}_1$ as boundary edges.

(2) If there exists a boundary edge in $\mathcal{T}_1$ which has exactly one neighbouring triangle in $\mathcal{T}_1$, then delete this edge and the triangle (but keep the remaining two edges). If all these edges have been deleted, then continue with (1). If no such boundary edge did exist, then continue with (3).

(3) If there exists a boundary edge $\mathcal{T}_1$ with the property that one of its end points is not adjacent to any other boundary edge, then delete this edge from $\mathcal{T}_1$. If all these boundary edges are deleted, then continue with (1). If no such boundary edge did exist, then continue with (4).

(4) If no boundary edges can be deleted, then the algorithm stops; the triangle strips have been decimated to polygons $\mathcal{T}_2$.

The result of this simple decimation method does not depend on the order in which the edges are considered.

Open edges on the surface (edges with one or two end points) are not yet contained in $\mathcal{T}_2$, since the third step of the algorithm will shrink them. In order to prevent this effect, one may freeze some of the points in $\mathcal{T}_2$. More precisely, steps 2 and 3 of the algorithm are modified such that these points are not allowed to be isolated during the decimation: If the deletion of an edge would isolate one of the frozen points, then it is not allowed to be deleted.

For an open edge with two end points, one should freeze an arbitrary pair of two points in the corresponding connected component of $\mathcal{T}_1$ realizing a maximum of the shortest distance between any two points in $\mathcal{T}_1$. Similarly, for an open edge with one end point, one freezes a point which realizes a maximum of the shortest distance to a skeleton $\mathcal{T}_2$ which was generated without freezing this point.

For instance, if an object has three edges with one common end point and one end point on the surface, then it will be modeled as an open edge with two end points, combined with one open edge with one end point.

5.4. Edge curve fitting and triangulation

First we define the "sharp" data points, which form the subset of the data points which corresponds to the features of the object. Two criteria are used. A point $\mathbf{p}_i$ is said to be "sharp", if

(1) the Sampson distance[2] $f(\mathbf{p}_i)/\|(\nabla f)(\mathbf{p}_i)\|$ exceeds a user–defined threshold, and

(2) the dihedral angle of the closest edge to $\mathbf{p}_i$ in $\mathcal{T}_0$ exceeds another user–defined threshold.

We use a collection of circular splines to approximate the "sharp" data points. The topology of this spline network is determined by the skeleton $\mathcal{T}_2$.

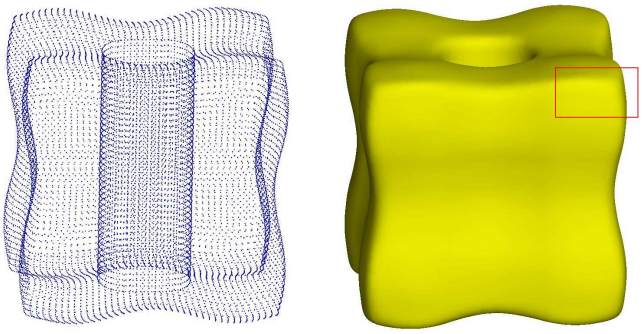
The vertices in $\mathcal{T}_2$ are all points with a valency other than two. We split the skeleton $\mathcal{T}_2$ into simple open polygons at these vertices. For each open polygon we generate a circular spline using an adaptive method for least-squares fitting of circular splines which is described in [30].

The method consists of two steps. First, an initial circular spline is obtained by biarc interpolation of decimated versions of the open polygons. Second, an evolution process – which corresponds to a Gauss-Newton type method for orthogonal distance regression – is used in order to reduce the approximation error, by driving the circular spline curve to the target point. This second step is coupled with an adaptive refinement procedure, which splits segments with large errors. The evolution process keeps all vertices in $\mathcal{T}_2$ as boundary points of the circular splines.

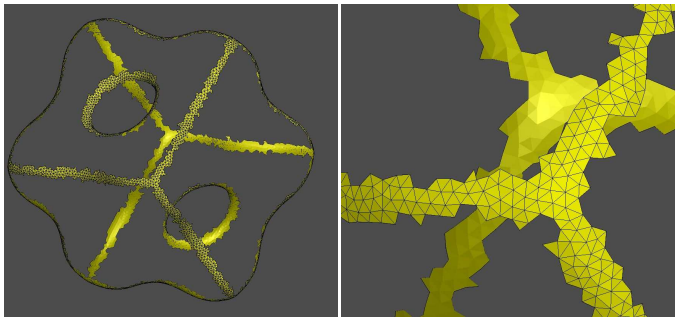
In order to define the edge descriptors, we choose the magnitude function according to Lemma 2 and the radius function depending on the distance to the initial surface, but such that the inequality (20) is satisfied.

We define the augmented scalar field F by adding the edge and vertex descriptors to f. For visualizing the surface $\mathcal{Z}(f)$, we generate a polygonization using the marching triangulation method [9]. In order to get a triangulation of the surface which preserves the features, we define the initial curve for this triangulation method as a polygon which has been sampled from the edge curves. See also [1, 37] for more advanced polygonization techniques for implicit defined objects.

[2] The value $f(\mathbf{x})/\|(\nabla f)(\mathbf{x})\|$ approximates the Euclidean distance, cf. [28].

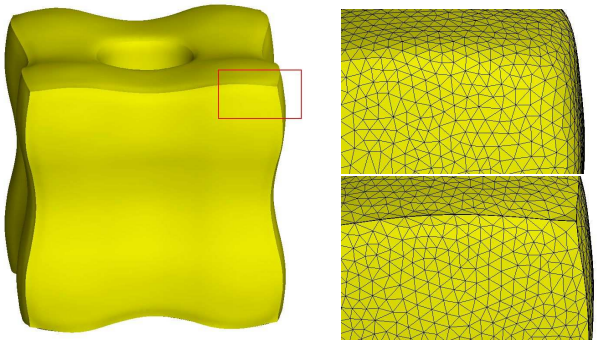


(a) Input data points and initial surface



(b) Strips of sharp triangles and close-up view

(c) Skeletons of sharp triangles and close-up view



(d) Final surface, close-up views of initial surface (top right) and final surface (bottom right)

Figure 8: Example 9: Surface reconstruction with sharp features from a data set with 13k points.

Example 9. We consider a collection of 13k data points which have been sampled from the surface of a deformed cube-shaped object with a circular hole, see Figure 8. The minimization of the objective function (28) gives the surface shown in (a), right, which does not yet contain the sharp features. A close-up view of the triangulation is provided in (d), top right. The computation time for the fitting and the triangulation was 6.4 and 6.3 seconds on a standard PC, respectively.

We apply the sharp triangle detection to the triangular mesh of the initial surface. The mesh $\mathcal{T}_1$ consisting of all "sharp" triangles is shown in Fig. 8b.

As the next step we apply the decimation algorithm to the triangle meshes of "sharp triangles". This leads to 12 open polygons, which meet in 8 vertices, and 2 additional closed polygons, see Fig. 8c. Note that the location of the vertices is detected automatically. The entire decimation (triangle strip generation and skeletonization) took less than 1 second.

Fig. 8d shows the final result, along with a close-up view of the triangulation (bottom right).

data	number of points	time (sec) T_s	T_t	T_d	T_c
Cube	13,013	6.4	6.3	0.9	2.0
Cylinder	3,270	1.8	0.39	0.08	0.86
Double torus	4,352	5.2	0.89	0.33	1.9
Fandisk	6,475	5.4	0.7	0.7	3.7

Table 1: The execution times of the given examples. T_s: initial surface reconstruction; T_t initial surface triangulation; T_d initial mesh decimation; T_c circular spline (edge curve) fitting.

Figs. 9-11 present three additional examples. The computing times for these examples, as well as for Example 9, are reported in Table 1. All computations were performed on a PC with a Pentium IV processor with 1.73GHz and 1.0GB RAM.

6. Conclusion

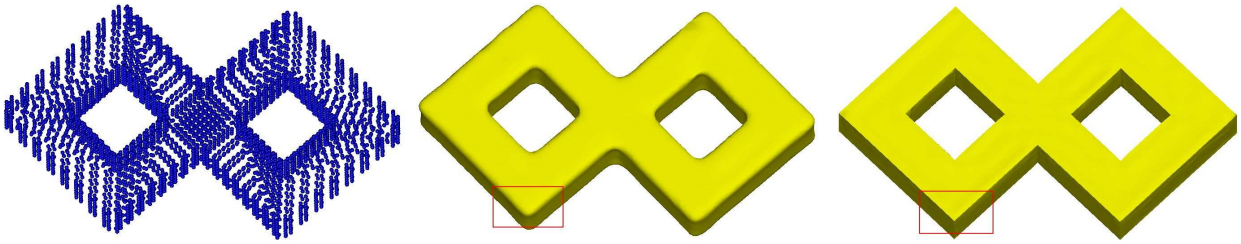
In order to model sharp features with implicitly defined surfaces, we defined the new concept of edge and vertex descriptors. These are scalar fields which can be added to any smooth implicitly defined surface.

Their definition is based on the shortest distance of a query point in space to a space curve. In order to be able to evaluate this distance efficiently, we use circular splines as edge curve. In the case of circular splines, the closest point of a query point can be computed with a non-iterative method, simply by solving a quadratic equation.

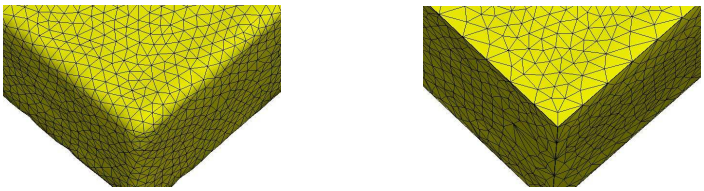
Using the concept of edge and vertex descriptors, the precise location of the edge curve is known and can be taken into account for the polygonization, e.g., by choosing this curve as the initial boundary curve in the marching triangulation algorithm. Thus, it is possible to generate a triangulation of the augmented surface which is compatible with the sharp features.

We also proposed a technique for modeling implicitly defined surfaces with sharp features, and for reconstructing them from unorganized point cloud data.

As a matter of future work, one might try to improve the reconstruction method, e.g., by developing automatic choices

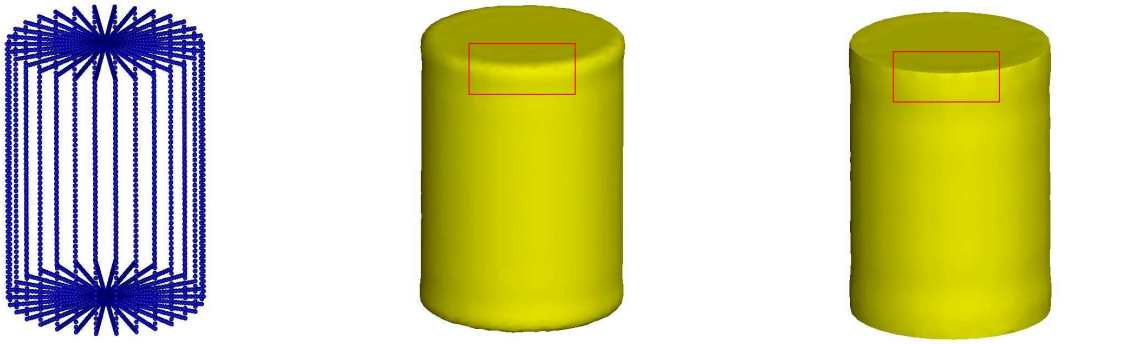


(a) Input data points (b) Initial surface (c) Final surface

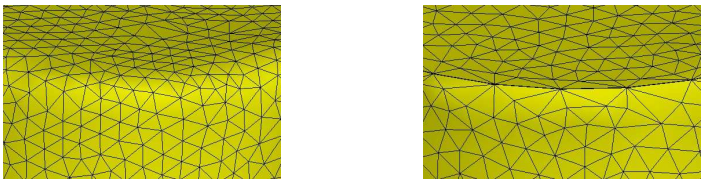


(d) Close-up view of (b) (e) Close-up view of (c)

Figure 9: Reconstruction of a double torus from 4,352 sample points (a). Initial surface (b,d) and final result (c,e).

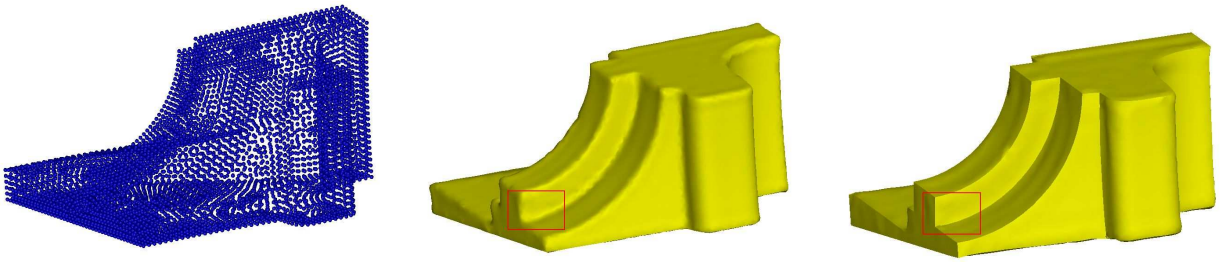


(a) Input data points (b) Initial surface (c) Final surface

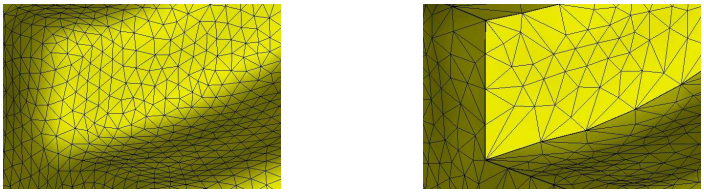


(d) Close-up view of (b) (e) Close-up view of (c)

Figure 10: Reconstruction of a cylinder from 3,270 sample points (a). Initial surface (b,d) and final result (c,e).



(a) Input data points (b) Initial surface (c) Final surface



(d) Close-up view of (b) (e) Close-up view of (c)

Figure 11: Reconstruction of the fandisk model from 6,475 sample points (a). Initial surface (b,d) and final result (c,e).

for the various user-defined parameters which take the level of noise in the data into account.

Other topics of interest include the use of other geometric primitives as edge curves, preferably with higher order of differentiability than C^1 smoothness, and the speed up of the algorithm for closest point computation, e.g., using a bounding volume hierarchy. In the case of the fitting procedure, even an dynamic hierarchy is required.

Acknowledgement. The authors were supported by the Austrian Science fund (FWF) through the national research network S92, subproject 2. The work of Xinghua Song was partially supported by the China Scholarship Council.

References

[1] S. Akkouche and E. Galin. Adaptive implicit surface polygonization using marching triangles. *Computer Graphics Forum*, 20(2): 67-80, 2001.

[2] 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.

[3] C. Bajaj, J. Chen, and G. Xu. Free-form Surface Design with A-patches. *Proc. Graphics Interface*, 174–181, 1994

[4] L. Barthe, N. Dodgson, M. Sabin, B. Wyvill and V. Gaildrat. Two-dimensional potential fields for advanced implicit modeling operators, *Computer Graphics Forum*, 22(1):23-33, 2003.

[5] U. Bauer and K. Polthier. Parametric reconstruction of bent tube surfaces. *Proc. CyberWorld: Workshop on New Advances in Shape Analysis and Geometric Modeling*, IEEE Press, 2007, 465–474.

[6] K. M. Bolton. Biarc curves. *Computer-Aided Design*, 7:89–92, 1975.

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

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

[9] E. Hartmann, A marching method for the triangulation of surfaces. *The Visual Computer*, 13(3):95-108, 1998.

[10] J. Hoschek. Circular splines. *Computer-Aided Design*, 24:611–618, 1992.

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

[12] T. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3):943–949, 2003.

[13] B. Jüttler. Least-squares fitting of algebraic spline curves via normal vector estimation. In *Proceedings of the 9th IMA Conference on the Mathematics of Surfaces*, pages 263–280, London, UK, 2000. Springer-Verlag.

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

[15] Q. Li, Smooth piecewise polynomial blending operations for implicit shapes, *Computer Graphics Forum*, 26(2):157–171, 2007.

[16] R.T. Ling, W. Wang and D.M. Yan, Fitting Sharp Features with Loop Subdivision Surfaces, *Computer Graphics Forum* (Proc. Symposium on Geometry Processing), 27(5):1383-1391, 2008

[17] D. S. Meek and D. J. Walton. Approximation of discrete data by G^1 arc splines. *Computer-Aided Design*, 24:301–306, 1992.

[18] 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.

[19] A. W. Nutbourne and R. R. Martin. *Differential geometry applied to curve and surface design, Vol. 1*. Ellis Horwood, Chichester, 1988.

[20] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H. P. Seidel. Multi–level partition of unity implicits. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):463-470, 2003.

[21] Y. Ohtake, A. Belyaev, and H. P. Seidel. 3D scattered data approximation with adaptive compactly supported radial basis functions. In *Proceedings of Shape Modeling International*, pages 31–39, 2004.

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

[23] G. Pasko, A. Pasko and T. Kunii, Bounded blending for function-basd shape modelling, *IEEE Computer Graphics and Applications*, 25(2):36-45, 2005.

[24] L. A. Piegl and W. Tiller. Data approximation using biarcs. *Engineering with Computers*, 18:59–65, 2002.

[25] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *Proc. SIGGRAPH*, pages 145–152, New York, NY, USA, 1987. ACM.

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

[27] M. A. Sabin. The use of circular arcs to form curves interpolated through empirical data points. *British Aircraft Corporation*, VTO/MS/164, 1976.

[28] P. D. Sampson. Fitting conic sections to very scattered data: An iterative refinement of the Bookstein algorithm. *Computer Graphics and Image Processing*, 18:97–108, 1982.

[29] R. Schmidt, B. Wyvill, M.C. Sousa, and J.A. Jorge. ShapeShop: sketch-based solid modeling with BlobTrees. In *2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 53–62, 2005.

[30] X.H. Song, M. Aigner, F.L. Chen and B. Jüttler. Circular spline fitting using an evolution process, available as report no. 76 at www.industrial-geometry.at/techrep.php.

[31] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(11):1115–1138, 1991.

[32] 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.

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

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

[35] W. Wang and B. Joe. Robust computation of the rotation minimizing frame for sweep surface modeling. *Computer-Aided Design* 29:379–391, 1997.

[36] 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.

[37] B. Wyvill and K.V. Overveld. Polygonization of implicit surfaces with constructive solid geometry. *Journal of Shape Modelling* , 2(4):257–274, 1996.

[38] 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 *Proceedings of Shape Modeling International*, pages 247–252. IEEE Press, 2006.

[39] H. Yang and B. Jüttler, Fitting Implicitly Defined Curves to Unorganized Points with Sharp Features, in: *Curve and Surface Design: Avignon 2006*, P. Chenin, T. Lyche and L.L. Schumaker (eds.), Nashboro Press, pages 274–283, 2007.

[40] H. Yang and B. Jüttler. Evolution of T-spline Level Sets for Meshing Non–Uniformly Sampled and Incomplete Data, *The Visual Computer*, 24:435–448, 2008.

[41] H. K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proc. IEEE Workshop on Variational and Level Set Methods*, pages 194–201, 2001.