Evolution of 3d Curves under Strict Spatial Constraints *

Klaus Hildebrandt Konrad Polthier Eike Preuss

Zuse Institute Berlin

Abstract

We present a new algorithm for fairing of space curves with respect spatial constraints based on a vector valued curvature function. Smoothing with the vector valued curvature function is superior to standard Frenet techniques since the individual scalar components can be modeled similar to curvature-based curve smoothing techniques in 2d. This paper describes a curve smoothing flow that satisfies strict spatial constraints and allows simultaneous control of both curvature functions.

1. Introduction

In most design and construction processes, 3d curves with smooth curvature distributions are needed. A feature line on a CAD surface must have smooth curvature for aesthetic reasons. In the design of free-form curves, smooth curvature is a main goal for the same reason. For the construction of patch layouts of scanned surfaces, the smoothness of the boundary curves of patches is essential for the smoothness of the patches themselves. Most curves are initially noisy though, coming from a reverse engineering process, or as the result of manual modeling. Industrial applications will normally impose that the smoothed curves must lie within a certain tolerance to the initial curve or the underlying mesh. Therefore, finding a smoothing algorithm that satisfies spatial constraints while smoothing the curvature of the curve is essential.

This paper addresses the demand for high quality curvature of 3d curves with spatial constraints. The presented algorithm is an to space curves extension of the 2d curvature smoothing flow (CS-flow) which was successfully applied to planar curves in [6]. The extension to 3d curves in space has to control two curvature distributions simultaneously, namely the components of the complex curvature. This paper provides the extended smooth formulation of the 3d CS-flow, its discretization and the smoothing algorithm that satisfies spatial constraints.

*Supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin and the Tebis AG, Germany.

1.1. Related Work

Geometric flows are a basic tool for geometry processing. Especially for smoothing and denoising of geometries, flows like the curve shortening flow for curves and the mean curvature flow for surfaces are standard approaches. In recent years the focus has shifted towards higher order flows due to their superior smoothing properties. Special attention is drawn to curve diffusion flow (resp. surface diffusion flow), the Bi-Laplace flow, and elastic energy [3, 4, 1, 13, 12].

The construction of smooth curves that respect spatial constraints is a delicate task. Though spatial constraints frequently appear in industrial applications, only few methods have been proposed. To our knowledge only the method by Hofer and Pottmann [8] has been proposed that allows to smooth curves in $\mathbb{R}^3$ with respect to spatial constraints. They propose an algorithm for energy minimizing splines in manifolds and apply it to compute energy minimizing splines in the presence of obstacles. The idea is to increment the dimension of the ambient space and then to model the obstacles as plateaus. There is a smooth transition between the obstacles and the remaining space, that gets thinner during the energy minimization. For fairing of *planar* curves with spatial constraints other methods have been suggested, e.g. the spline based methods in [11, 5, 10].

1.2. Contributions

The paper introduces a new algorithm for the smoothing of 3d curves with given strict spatial constraints. The flow is driven by carefully chosen target curvatures which evolutionary improves the curve smoothing while simultaneously ensuring strict spatial constraints. The paper contributes a discrete complex curvature which describes the bending of a polygonal curve in 3d with respect to a rotation minimizing frame along the curve, a curve fairing algorithm based on smoothed target curvatures to steer the curve evolution,

Ninth International Conference on Computer Aided Design and Computer Graphics (CAD/CG 2005)
0-7695-2473-7/05 $20.00 © 2005 IEEE

3d Curvature Smoothing Flow

Given a polygon with vertices p_i. Iterate:

1. Compute rotation min. frame $(\vec{t}, \vec{n_1}, \vec{n_2})$, curvature vector $\vec{\kappa}$ and vector valued curvature κ_c.
2. Compute smoothed vector valued curvature κ_c^s and curvature vector $\vec{\kappa}^s = \kappa_1^s \vec{n_1} + \kappa_2^s \vec{n_2}$.
3. Determine step size τ.
4. For each vertex p_i set
$$p_i \rightarrow p_i + \tau \left(\vec{\kappa}(p_i) - \vec{\kappa}^s(p_i)\right)$$

Table 1. Discretized CS-flow for space curves.

and the incorporation of spatial constraints for the curve in the target curvature.

The curve fairing algorithm is an extension of the 2d curve fairing algorithm [6].

1.3. Paper Overview

In Section 2 we present the extension of the curvature smoothing flow to curves in $\mathbb{R}^3$. The discretization of this flow, given in Section 3, uses a discretization of rotation minimizing frames and a corresponding vector valued curvature. In Section 4 we adjust the curvature smoothing flow to satisfy spatial constraints. This flow is then applied to a variety of curves from CAD applications in Section 5.

2. Smoothing Curvature

Curvature smoothing flow (CS-flow) is a class of flows which couple two processes, first a procedure that at each point in time computes a smoothed curvature κ^s from the curvature κ of the actual curve and second a flow that evolves the curve to its normal direction with a velocity depending on the difference of κ and κ^s. This construction allows to design smoothing flows of curves and to adjust the flows to specific needs by specifying a smoothing procedure for the curvature of the curve. For example, in [6] a CS-flow for planar curves is described that restricts the evolving curve to satisfy given spatial constraints.

In this section we first review the curvature smoothing flow for planar curves and then extend the flow to curves in $\mathbb{R}^3$. The construction of a CS-flow that evolves space curves within user defined spatial constraints is described in Section 4.

2.1. Review of CS-Flow of Planar Curves

The curvature smoothing flow of a planar curve is given by:

Definition 1 (2d CS-Flow) *Let $c(\cdot,t)_{t\geq 0} : I = [a,b] \rightarrow \mathbb{R}^2$ be a family of smooth planar curves, and let $\kappa(\cdot,t)$ and $N(\cdot,t)$ denote the curvature and the normal of c. The family c evolves under* curvature smoothing flow *if*

$$\begin{aligned} &\frac{d}{dt}c(x,t) = \left(\kappa(x,t) - \kappa^s(x,t)\right) N(x,t) \;\; x \in I,\; t \geq 0 \\ &c(x,0) = c_0(x) \;\; x \in I \end{aligned} \tag{1}$$

where c_0 is an initial curve and κ^s is a user defined curvature evolution chosen to control the smoothing process.

A simple example of a smoothing process for κ is to set

$$\kappa^s(x,t) = \kappa(x,t) + \Delta\kappa(x,t), \tag{2}$$

using the smoothing properties of the Laplace operator. Inserting (2) to (1) shows that the resulting flow is the curve diffusion flow that is well know for its excellent smoothing properties, see [3, 13, 4, 1].

2.2. CS-Flow of Curves in $\mathbb{R}^3$

For a planar curve the normal space at each point has dimension 1 and consequently smoothing the curvature vector in the normal bundle can be described as smoothing a function along the curve. For curves in $\mathbb{R}^3$ this is more involved since the smoothing process not only changes the length of the vector but also rotates it in the normal bundle of the curve.

Let $\{\vec{t}, \vec{n}_1, \vec{n}_2\}$ be an orthogonal unit frame $\{\vec{t}, \vec{n}_1, \vec{n}_2\}$ along a curve with tangent vector $\vec{t}$. The curvature vector can be decomposed as with respect to this frame,

$$\vec{\kappa} = \kappa_1 \vec{n_1} + \kappa_2 \vec{n_2}, \tag{3}$$

where $\kappa_i = \langle \vec{\kappa}, \vec{n}_i \rangle$. The components κ_i depend heavily on the choice of the frame, e.g. we can choose a frame such that the κ_i have arbitrary strong oscillations. The adequate choice of a frame here is a *parallel frame* (or rotation minimizing frame), since this frame has no inner rotations. With respect to the parallel frame the pair $\kappa_c = (\kappa_1, \kappa_2)^t$ is called the complex curvature. For an introduction to frames and curvature of space curves see [2] and for the rotation minimizing property of the parallel frame see [9].

Using a parallel frame, smoothing the curvature vector in the normal bundle amounts to smoothing the κ_i as functions on the curve. This leads to the following definition of the CS-flow for curves in $\mathbb{R}^3$

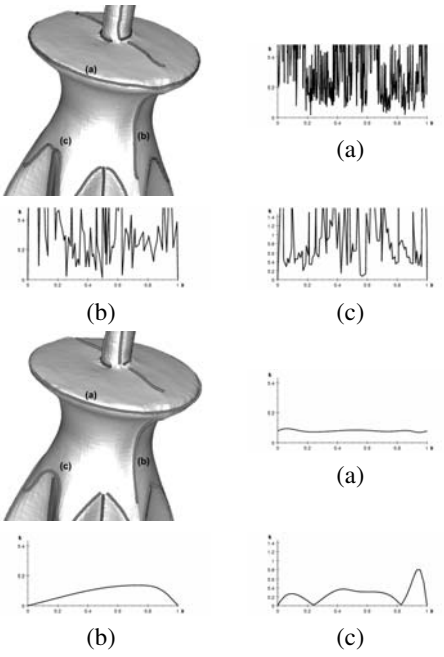

Figure 1. Feature lines on part of a screw driver, smoothed with the constraint CS-flow. Top row: The original lines created automatically by a feature detection algorithm [6] carry high frequency noise. Bottom row: CS-flow with ϵ-constraint was used for the smoothing process to prohibit large deviations of the smoothed lines from the original surface. The plots show the scalar curvature $|\kappa_c|$ of the curves.

Definition 2 (3d CS Flow) *Let $c(\cdot,t)_{t\geq 0} : I = [a,b] \to \mathbb{R}^3$ be a family of smooth space curves and let $\kappa_c(\cdot,t) = (\kappa_1(\cdot,t), \kappa_2(\cdot,t))^t$ and $\{\vec{t}(\cdot,t), \vec{n}_1(\cdot,t), \vec{n}_2(\cdot,t)\}$ denote the vector valued curvature and corresponding parallel frame of c_t. The curve c evolves according to* curvature smoothing flow *if*

$$\begin{aligned}\frac{d}{dt}c(x,t) &= (\kappa_1(x,t) - \kappa_1^s(x,t))\vec{n_1}(x,t) \\ &\quad + (\kappa_2(x,t) - \kappa_2^s(x,t))\vec{n_2}(x,t) \\ c(x,0) &= c_0(x) \end{aligned} \tag{4}$$

where $\kappa_c^s = (\kappa_1^s, \kappa_2^s)^t$ is an evolution of complex curvature of the curve.

For example, the curve diffusion flow for planar curves given in equation (2) extends to the case of curves in $\mathbb{R}^3$ by

$$\kappa_c^s(x,t) = \kappa_c(x,t) - (\Delta\kappa_1(x,t), \Delta\kappa_2(x,t))^t. \tag{5}$$

Note that for planar curves in $\mathbb{R}^3$ the 3d CS-flow reduces to the 2d case.

3. Discretization of the Flow

We discretize the flow in two steps. First, a spatial discretization based on finite elements and second a time discretization by an explicit Euler scheme. The time discretization will be carried out in Section 5.

For the spatial discretization we use polygonal curves. Such a curve is represented as an ordered list of vertices $(p_0, \ldots, p_m)$, and edges $e_i = p_{i+1} - p_i$.

Definition 3 (Discrete curvature vector) *The* discrete curvature vector $\vec{\kappa}$ *of c is defined at each vertex by*

$$\vec{\kappa}(p_i) := \frac{2}{|e_i| + |e_{i-1}|} \cdot \left(\frac{e_i}{|e_i|} - \frac{e_{i-1}}{|e_{i-1}|} \right) \tag{6}$$

and linear interpolation between the vertices.

For a smooth curve the curvature vector equals the Laplace operator of the embedding of the curve. The discrete curvature vector is based on a finite element approximation of the Laplace operator.

Discrete parallel frame. A parallel frame for a smooth curve is described by specifying a frame at one point of the curve. The frame along the curve is then uniquely defined by the rotation minimizing property of the parallel frame. The computation involves solving a system of differential equation, cp. [9].

A discrete frame of a polygonal curve is given by a set of three orthonormal vectors $\{\vec{t}, \vec{n}_1, \vec{n}_2\}$ at each vertex p_i. Here the vector $\vec{t}$ is a unit vector pointing to the tangent direction. We set

$$\vec{t}(p_i) = \left(\frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|}\right) / \left\| \frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|} \right\| \tag{7}$$

for all vertices p_i of the curve c. Then $\vec{t}(p_i)$ is orthogonal to the discrete curvature vector $\vec{\kappa}(p_i)$ and in the plane spanned by e_i and e_{i-1} for all p_i.

The *discrete parallel frame* is constructed analog to the smooth case. We specify the frame at a single vertex p_i, determines the *discrete parallel frame* on the whole curve. For all other vertices is the frame generated by iterating through the vertices, constructing the frame at a vertex p_{i+1} from the frame at the preceding vertex p_i. The frame $\{\vec{t}(p_{i+1}), \vec{n}_1(p_{i+1}), \vec{n}_2(p_{i+1})\}$ at p_{i+1} is computed by rotating the frame $\{\vec{t}(p_i), \vec{n}_1(p_i), \vec{n}_2(p_i)\}$ at p_i around the axis $\vec{t}(p_i) \times \vec{t}(p_{i+1})$, such that the vector $\vec{t}(p_i)$ is mapped onto $\vec{t}(p_{i+1})$.

Note that (similar to the smooth case) the parallel frame of a closed curve does not necessarily close up. If we start

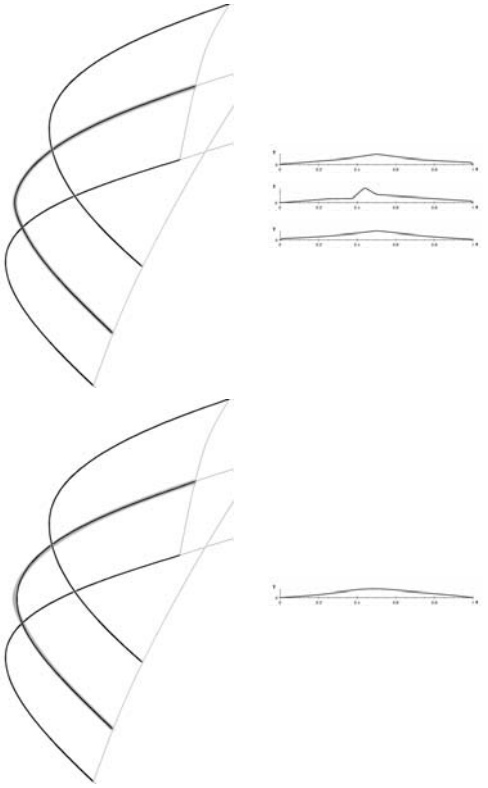

Figure 3. Smoothing of a curve of a patch layout. The curve in the middle has a bump, visible in the curvature plot. Smoothing removes the bump. The 3d ϵ-constraint is shown in gray around the curve.

the construction of the parallel frame with a frame at vertex p_0, we might arrive at p_0 with a different frame after going once around the curve. If the smoothing method used for κ_c in the CS-flow is a local one, this poses no problem, since locally the rotation minimizing frame is well-defined. Otherwise, we approximate such a frame by uniformly distributing the difference rotation angle of the two frames at p_0 over the whole curve.

Once the discrete curvature vector and the discrete parallel frame is known, the definition of the *discrete complex curvature* is straight forward.

Definition 4 (Discrete complex curvature) *Let c be a polygonal curve and $\{\vec{t}, \vec{n}_1, \vec{n}_2\}$ a discrete parallel frame on c, then for a vertex p of c the* discrete complex curvature *at p is $\kappa_c(p) = (\kappa_1(p), \kappa_2(p))^t$ with $\kappa_i(p) = \langle \vec{\kappa}(p), \vec{n}_i(p) \rangle$.*

4. Evolution of Space Curves with Constraints

In most industrial applications the smoothed curve must be in a small neighborhood of the original curve. Otherwise the resulting workpiece might no longer be suitable for the application, e.g. it might no longer fit to other pieces. We consider the following type of ϵ-constraint:

> Given a starting curve c_0 and a maximal acceptable deviation ϵ from this curve. Let $\Omega \subset \mathbb{R}^3$ be the set of points $x \in \mathbb{R}^3$ with $\mathrm{dist}(x, c_0) \leq \epsilon$. Constrain the family of curves c_t such that they lie inside Ω for all t.

In this section we will incorporate these constraints to the curvature smoothing flow described in the previous section. The extension to more complex constraints Ω is rather straight forward. The method allows to include other requirements that applications may impose, for example, interpolation of some given points are easily realized as well.

Incorporating constraints. In each step of the iterative integration method, we can compute from the 3d flow equation (4) the admitted values for κ_c^s at every vertex such that the curve stays within the ϵ-constraints. The new vertex position lies in the constraints if $\mathrm{dist}(c(p_i) + (\kappa_1(p_i) - \kappa_1^s(p_i))n_1(p_i) + (\kappa_2(p_i) - \kappa_2^s(p_i))n_2(p_i), c_0) \leq \epsilon$. A curvature distribution κ_c^s that fulfills these constraints is computed in *each step of the smoothing process*. Like in the 2d case, we reduce the complexity of this problem by prescribing $\kappa_c^s = \kappa_c$ at points that lie on the boundary of Ω, so these points do not move, and choosing a stepsize such that vertices do not leave the constraint (but may touch the boundary). At regular intervals we check, whether a vertex moves away from the boundary if we do not force the interpolation $\kappa_c^s = \kappa_c$ there.

Computing smoothed complex curvature. Since κ_c is smooth if its components are smooth, we consider the two components independently. Similar to the 2d case, we want to avoid behavior changes of the curve. A curve with curvature that linearly interpolates the prescribed values has only the necessary change in curvature to satisfy the constraints. To achieve a similar quality with smooth curvature, we demand that the smoothed curvature between two prescribed values has its only maximum and minimum at these prescribed values. The approximation scheme that was used for the 2d CS-flow with constraints has this property, so we construct a smooth curvature by filling the intervals between the prescribed curvature values with Bézier segments, separately for both components κ_j.

Let x_i denote the curve parameters of the prescribed curvature values, i.e. $\kappa_j(x_i)$ has to be interpolated. We choose the tangent direction of the Bézier segment at x_i as $(x_{i+1} - x_{i-1},\ \kappa_j(x_{i+1}) - \kappa_j(x_{i-1}))$, if this this lies in the bounding boxes spanned by the preceding and succeeding

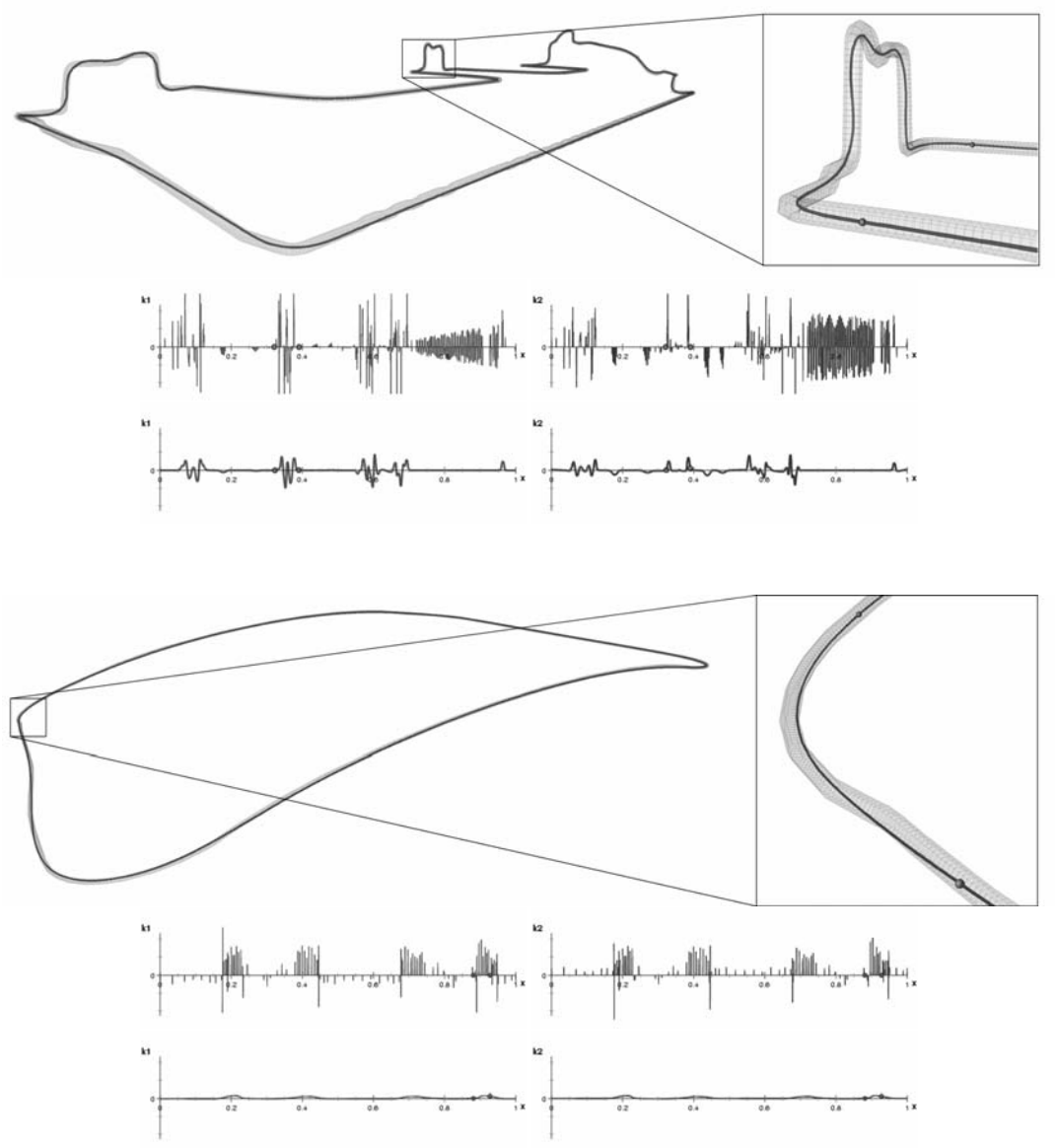

Figure 2. 3-dimensional design curves with ϵ-constraint that were smoothed with the CS-flow with constraint from this paper. These curves are outlines of different workpieces, created by a designer with a CAD-application. The images show the smoothed curve with the constraint in gray, and, in the rows below, the components κ_1 and κ_2 of the vector valued curvature of the original curve (first row) and of the smoothed curve (second row).

Ninth International Conference on Computer Aided Design and Computer Graphics (CAD/CG 2005)
0-7695-2473-7/05 $20.00 © 2005 IEEE

vertex. Otherwise we choose a horizontal tangent direction. We set the length of the tangent such that the control points lie in the bounding box and the Bézier segment is the graph of a function over x. If there are no prescribed values, i.e. the curve does not touch the boundary of the constraint anywhere, we use the mean value and set for all x $\kappa_j^s(x,t) = \frac{1}{\text{length}(c_t)} \int_{c_t} \kappa_j(s,t) ds$.

5. Experimental Results

In our experiments we use an explicit Euler scheme for the integration of the curvature smoothing flow. The usual estimate for a secure step-size for the Laplace flow is $\tau < (\sqrt{||A||_1 ||A||_\infty})^{-1}$, where A is the matrix representing the discrete Laplace operator of the current curve. To maximize the possible stepsize we discretize the curves equidistantly before integrating the flow. We use a multilevel scheme to accelerate smoothing of curves with many vertices. The only difficulty is to take care that the curve still lies in Ω when switching between different levels of detail, i.e. when switching to a coarser level, we may not remove vertices if the resulting edge does not satisfy the constraints.

Patch layout. In the first example (Figure 3), we took the boundary curves of a patch layout of a surface in 3d, that was created by CAD-professionals. Figure 3 shows three of these curves, that are parallel on the surface. They are stronger curved in the middle, and less curved in the ends. The scalar curvature $|\kappa_c|$ of the curve in the middle has an unwanted bump. We removed the bump, applying the constraint CS-flow to this curve, with an ϵ-constraint of 0.2 mm (which is about 0.3% of the length of the curve) and fixed endpoints.

Design curves. The second example in Figure 2 shows two 3d-outlines of workpieces from a car, created by a designer with a CAD-application. The original curves have noise, due to the inaccuracies of the design process. The ϵ-constraint is set by the designer, such that the behavior of the curve remains the same as the desired behavior. Therefore the constraint will normally be rather tight. The CS-flow with constraints creates a curve with nicely shaped smooth curvature within these constraints.

Feature lines. An example of smoothing feature lines of a surface is given in Figure 1. Most feature extraction methods return paths on the surface. These lines naturally still carry noise. To produce feature lines that are aesthetically looking and directly useful for a CAD-application, a smoothing process must be applied. An unconstraint smoothing algorithm would eventually move curves too far away from the original surface. Therefore, spatial constraints are essential. Figure 1 shows feature lines on a screwdriver, that were computed automatically by the extraction method in [7]. Even though this method already smooths the operators that are used for the computation of the feature lines, the feature lines are noisy. The smooth feature lines were computed automatically from the mesh by first applying the feature detection method to the surface, and then smoothing all curves with the CS-flow with a given ϵ-constraint.

6. Conclusion

We have introduced a new fairing algorithm for 3d curves which strictly satisfies given spatial constraints. The flow has demonstrated its effectiveness in industrial CAD-applications. In future work we will extend this flow to the fairing of curves on 3d surface meshes and the optimization of surface features.

References

[1] E. Bänsch, P. Morin, and R. H. Nochetto. A finite element method for surface diffusion: the parametric case. *J. Comput. Phys.*, 203(1):321–343, 2005.

[2] M. P. doCarmo. *Differential geometry of curves and surfaces*. Prentice-Hall, 1976.

[3] G. Dziuk, E. Kuwert, and R. Schätzle. Evolution of elastic curves in $\mathbb{R}^n$: Existence and computation. *SIAM J. Math. Anal.*, 33(5):1228–1245, 2002.

[4] J. Escher, U. F. Mayer, and G. Simonett. The surface diffusion flow for immersed hypersurfaces. *SIAM J. Math. Anal.*, 29(6):1419–1433, 1998.

[5] T. N. T. Goodman, B. H. Ong, and K. Unsworth. Constrained interpolation with rational cubics. In G. Farin, editor, *Nurbs for curve and surface design*, pages 59–74. SIAM, 1991.

[6] K. Hildebrandt, K. Polthier, and E. Preuss. Curvature smoothing flow for planar curves with spatial constraints. submitted, 2005.

[7] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Proceedings of the Eurographics and ACM Symposium on Geometric Processing*, 2005.

[8] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *Transactions on Graphics*, 23(3):284–293, 2004.

[9] F. Klok. Two moving coordinate frames for sweeping along a 3d trajectory. *Comput. Aided Geom. Des.*, 3(3):217–229, 1986.

[10] D. S. Meeks, B. H. Ong, and D. J. Walton. Constrained interpolation with rational cubics. *Computer Aided Geometric Design*, 20(5):253–275, 2003.

[11] G. Opfer and H. J. Oberle. The derivation of cubic splines with obstacles by methods of optimization and optimal control. *Numer. Math.*, 52:17–31, 1988.

[12] R. Schneider and L. Kobbelt. Discrete fairing of curves and surfaces based on linear curvature distribution. In P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, editors, *Curve and Surface Design*, 1999.

[13] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface smoothing via anisotropic diffusion of normals. *Proc. of IEEE Visualization*, pages 125–132, 2002.