Computers & Graphics 40 (2014) 22–35

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Adaptive and robust curve smoothing on surface meshes ☆

Kai Lawonn *, Rocco Gasteiger, Christian Rössl, Bernhard Preim

Department of Simulation and Graphics, University of Magdeburg, Germany

ARTICLE INFO

Article history:
Received 31 July 2013
Received in revised form
10 January 2014
Accepted 14 January 2014
Available online 25 January 2014

Keywords:
Surface curve
Smoothing
Cutting
Geodesic

ABSTRACT

Smoothing surface curves are an important step for surface processing applications, such as segmentation, editing and cutting. Various applications require smooth curves that follow the given initial curves closely. One example is surgical planning, in which virtual models are cut open, as in resection planning for liver surgery. There are several approaches to smoothing a surface curve that are based on energy minimization or on interpolating the control points with (piecewise) polynomial curves. These methods, however, do not ensure that the smoothed curve remains close to its initial location. This paper presents a new method for smoothing piecewise linear curves on triangular surface meshes based on a local reduction of the geodesic curvature. The method preserves the closeness to the initial curve. Moreover, the user can adjust the degree of closeness such that the smoothed curve will result in a locally straightest geodesic. We prove that the curve's geodesic curvature decreases in each iteration step, and we use it as an abort criterion. Experiments also confirm robustness to geometric and parametric noise. Finally, we evaluate our method for two surgical planning instances, the decomposition of cerebral aneurysms and resection planning for liver surgery.

1. Introduction

Curves on surfaces play an important role in many application domains, such as engineering or image-based medicine, where, for example, surface cuts or segmentations are required [1,2]. One common requirement for such curves on surfaces is smoothness. This goal is typically achieved by smoothing an initial curve that might be roughly sketched by a user and that frequently shows noise.

For the Euclidean space $\mathbb{R}^2$, there are several methods for smoothing a given curve appropriately. In recent years, several approaches have been proposed that generalize these methods on two-dimensional surface meshes and in Riemannian manifolds of arbitrary dimension. Given an initial curve, most of these approaches minimize certain energy functionals, such as the curve's length or its total curvature. Often, the energy term is designed such that a curve equations to the surface features. This arrangement leads to a curve that deviates strongly from the initial curve. For certain applications, such as treatment planning in surgery, this difference between the smoothed curve and the initial curve (where the initial curve was defined by a medical expert) should be small. Thus far, the existing solutions for this scenario suffer from a lack of convergence for noisy or irregularly tessellated surfaces. Furthermore, some of the existing approaches are limited to closed surface curves.

In this paper, we propose a novel method for performing local smoothing of initial, jagged curves on triangulated surfaces based on iterative smoothing. Our approach reduces a geodesic curvature while simultaneously controlling the deviation from the initial contour. The balance between smoothness and closeness is expressed by a single parameter. An additional parameter bounds the number of iterations, either directly or as an error threshold. Our method ensures that the curve is always located on the surface, and the final result is comparable to different surface tessellations. Both requirements are necessary for applications such as surface cutting in medical applications, for example, resection treatment planning in surgery. Our experiments show that this method is applicable to such scenarios with clinical datasets. We provide results for synthetic benchmark surfaces.

In summary, the contributions of this work consist of an adaptive and novel approach to smooth surface curves that accomplishes the following:

- *preserves closeness* to the initial curve with respect to some bounding envelope,
- uses a *single parameter* to balance the closeness versus the smoothness with respect to the geodesic curvature,
- uses an abort criterion based on a theoretical *proof* for decreasing the curve's curvature, and

☆This article was recommended for publication by A. Shamir.

* Corresponding author.
E-mail addresses: lawonn@isg.cs.uni-magdeburg.de (K. Lawonn), gasteiger@isg.cs.uni-magdeburg.de (R. Gasteiger), roessl@isg.cs.uni-magdeburg.de (C. Rössl), preim@isg.cs.uni-magdeburg.de (B. Preim).

0097-8493/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2014.01.004

- is *robust* toward geometric and parametric noise.

2. Related work

Smoothing surface curves are an important step for geometric processing, such as surface segmentation, editing, and cutting [2–4]. In most of these applications, an initial contour is defined either by direct user interaction or by (semi-)automatic feature detection. These initial contours are usually non-smooth and require a surface curve smoothing stage. This paper focuses on this smoothing stage.

A very intuitive way to smooth a polygonal line is by corner cutting, which is well known as a subdivision scheme for planar curves (see [5,6]). Morera et al. [7] presented a generalization of subdivision algorithms for curves on surfaces. The resulting subdivision curve consists of points located anywhere in the interior of the surface triangles. Hence, the polyline that connects these points is not a surface curve (according to our definition of a surface curve in Section 4), and it might "miss" essential parts of the surface. Usually, approaches to smoothing surface curves should guarantee that the resulting curve is a part of the surface. For discrete surfaces that are represented as polygonal meshes, this arrangement means that there are curve samples on the *edges* of polygons such that every line segment of the curve is a part of a surface polygon. Existing smoothing methods can be classified into methods that minimize energies or methods that approximate the given curve with (piecewise) polynomial curves.

Lee and Lee [8] and Lee et al. [9] combined automatic surface segmentation and cutting. After an initial feature contour is detected, a subsequent smoothing of the contour is performed by minimizing an energy-like functional. This functional is designed to meet different goals: to move the contour towards nearby features, to minimize the length of the contour, and to smooth the shape. Lai et al. [10] used a feature-sensitive curve smoothing for surface feature classification. They minimized a discretized tension spline energy with a subsequently projected gradient descent to obtain the smoothed boundaries. Kass et al. [11] represented curves as so-called *snakes*. A snake is a closed curve that evolves by minimizing internal forces, such as curvature, distances to features, or length, and external forces, such as distance to a feature. This approach is usually applied to image segmentation. Extensions to 2-manifold domains [3,4,12,13] are successfully used to detect features on polygonal meshes. Most of these methods ensure an adaptive sampling of the snake depending on the mesh resolution. For snakes, a rapid movement toward the features is typically expected, and their initial shape is not important. Thus, none of these methods strives for closeness to the initial curve. In a different setting, Martínez et al. [14] focused on minimizing the local length of a given curve to obtain a geodesic on the surface. This approach is implemented by iteratively reducing the local length between two subsequent curve vertices. Similar to the previous methods, the resulting geodesic can differ strongly from the shape of the initial curve.

There are several variants and extensions of the aforementioned approaches that achieve smoothing by "fitting" (piecewise) polynomial curves. Morera et al. [15] generalized Bézier curves in the Euclidean space $\mathbb{R}^2$ to geodesic Bézier curves on triangulated surfaces. The points on the initial curve are used as control points for the geodesic Bézier curves located on the surface. Hofer and Pottmann [16] determined spline curves in manifolds by minimizing quadratic energies. Although most of the presented methods converge to smooth surface curves, they do not guarantee a closeness to the initial curve shape. Hofer and Pottmann [16] overcame this issue by adding more control points on the initial curve, but this approach requires more user interaction.

Another class of methods depends mainly on the surface features. These methods are designed to move the curve close to the features [3,4,12,13]. However, depending on the field of application, both closeness and the independence of the surface features are important, i.e., the smoothing of the curve should not be related to the underlying surface features. Our approach is inspired by the work of Martínez et al. [14] in terms of minimizing the geodesic curvature between the curve segments. In addition, this method allows for adjusting the closeness to the initial curve. Our method neither depends on nor equations to the surface features. It is largely independent of the particular surface tessellation and is robust against noise.

Our approach smoothes the curves by reducing their geodesic curvature, i.e., the curve should evolve as straight as possible without unnecessary oscillations. In the limit, we obtain geodesics on the surface meshes. There are several algorithms for computing such geodesics. Our approach finds a geodesic that connect two surface points, i.e., we solve a boundary value problem. In contrast, these algorithms either integrate a geodesic curve given a starting point and a direction (see Polthier and Schmies [17]), which means that they solved an initial value problem, or they computed *all* of the geodesics that emanate from a given vertex globally. The latter usually involves evolving fronts on the surface or the solution of a linear system. Mitchell et al. [18] presented an algorithm that finds the shortest path between two given points based on a continuous variant of Dijkstra's algorithm. Surazhsky and Surazhsky [19] extended this algorithm to obtain computationally efficient and accurate approximations. Therefore, they gained an exact solution more quickly. Bommes and Kobbelt [20] generalized [19] to handle arbitrary polygons on the mesh. Kimmel and Sethian [21] used the eikonal equation to generate a propagating front. The propagating front starts from a set of points and spreads over the mesh to calculate the distances from the start set. Recently, Crane et al. [22] proposed a method for computing geodesics using heat kernels on meshes. The gradient of the heat kernel is used to find an approximation of the eikonal equation. Therefore, the gradient of the heat kernel forms a new vector field. Afterward, the new vector field is used to solve the Poisson equation and the resulting scalar field recovers the final distances.

3. Motivation and requirements

The motivation of our approach is to achieve smooth surface curves from initial jagged curves for medical surface cutting applications. In contrast to non-medical applications, the smoothing requires a closeness to the initial curve shape, which is defined by domain experts such as physicians or bioengineers. In most cases, the initial curves indicate relevant anatomical landmarks or surface regions on which data analysis or treatment planning is performed. In particular, we focus on the decomposition of vascular structures such as aneurysms for visual exploration purposes and on liver resections for preoperative treatment planning. For these applications, a patient-specific surface mesh is given, along with one or multiple user-defined cutting contours on the mesh, which represent a virtual resection or a decomposition for further analysis. In practice, the surface mesh is usually generated by the Marching Cubes algorithm and is based on a binary segmentation mask from medical image data, such as CT or MRT. The cutting contours are obtained by placing reference points on the mesh, which are connected by shortest path algorithms, such as breadth-first search, Dijkstra's algorithm [23] or similar approaches. The initial curves are continuous, and their segments are located entirely on the surface triangulation but suffer from a jagged curve shape. These noisy curves are distracting and would result in unpleasant surface cuts, which require more mental effort by the expert to conceive the cut

shape. Moreover, the outlined anatomical landmarks and surface regions exhibit smooth shapes in reality but are approximated by the jagged curves. This arrangement can lead to inaccurate data analysis, such as area or volume estimation. Thus, an appropriate smoothing of these initial curves which preserves the closeness to the initial shape is essential to support the visual perception and data analysis.

3.1. Characterization of the input data

The curve smoothing algorithm operates on an arbitrary triangular surface mesh. For the clinical application of our work, surface extraction by Marching Cubes is highly efficient. However, the generated meshes often suffer from poorly shaped triangles. The resulting surface often contains noise: vertex distortions in the normal direction (geometric noise) and in a tangent space (parametric noise). These distortions are introduced due to beam hardening artifacts and noise in the image data. Furthermore, the binary segmentation mask can lead to block or staircase artifacts. Common approaches for image noise reduction, as well as binary mask and mesh smoothing, can reduce these artifacts. The degree of smoothing, however, must be carefully adjusted to prevent the elimination of relevant surface features and to preserve the volume [24]. There can also be "topological noise," such as small handles or tunnels, which should be removed. Thus, some artifacts are still expected in the surface mesh. In practice, this circumstance also complicates or even hinders a (semi-) automatic remeshing to reduce the parametric noise, i.e., to improve the triangle quality. In summary, we identify the initial situation as follows: first, the given surface can show geometric, parametric and topological noise. Second, preprocessing of these data is not a viable option. Instead, we require algorithms that are robust enough to address these data.

This scenario is typical for medical research applications, such as investigations of simulated blood flow in cerebral aneurysms for rupture risk assessment [25]. Here, avoiding time-consuming and largely manual data preprocessing provides a significant benefit.

3.2. Goals of curve smoothing

Our overall goal is to construct a smooth curve based on the initial curve and the underlying surface mesh. By *smooth*, we refer to minimizing / reducing the initial geodesic curvature to obtain a surface curve that is "as straight as possible." However, the difference between the resulting smooth curve and the initial curve should be small because the initial curve is assumed to represent the region where the cut should occur. Given these two conflicting goals, several requirements must be fulfilled.

Adaptiveness: The resolution of the evolving curve must adapt to the local mesh tessellation. This adaption requires a refinement and simplification of the curve during the smoothing steps.

Surface domain: For the subsequent surface cutting, the smoothed curves must be located on the surface. Thus, the smoothing must be performed entirely on the surface, and every inserted or merged curve point must be located on the surface.

Robustness: The smoothing must be robust with respect to both the surface artifacts and the poor triangulation. In particular, any oscillating behavior must be avoided.

4. Overview and notation

The proposed algorithm consists of three main steps:

1. *Initialization*: The user provides the initial curve, the desired curvature of the final curve, and its maximum distance to the initial curve.

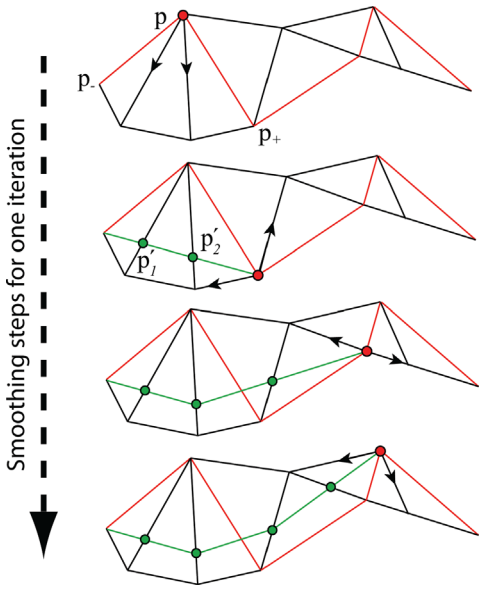

Fig. 1. Illustration of three smoothing steps to shorten the initial curve (red). The points marked with a dot were moved along the edges to shorten the length between the predecessor p_- and the successor point p_+. In the next step, the previous successor point should be moved. After reaching the end of the curve, the next iteration starts. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

2. *Smoothing*: This step computes weights for an iterative Laplacian *smoothing* of the curve. The specific choice of the weights ensures that all of the requirements are met. During the smoothing process, the curve points move along the surface edges within a user-defined region. This process could require a local adaptation: *splitting* and *merging* of the curve points.
3. *Evaluation*: Each smoothing step is followed by an evaluation of the curve to decide on the termination and to identify the critical surface vertices. Such vertices prevent a fast *movement*, and they need to be handled appropriately when the curve points are moving toward these vertices.

Steps 2 and 3 are iterated until the stopping criteria in step 3 are fulfilled. They include the degree of smoothness and the distance from the initial contour. Our general smoothing approach is illustrated in Fig. 1, where one iteration with smoothing steps for three vertices is shown.

The following notation is used in the remainder of this paper. Let $M \subset \mathbb{R}^3$ be a triangular mesh. The mesh consists of vertices V with associated positions $\mathbf{x}_i \in \mathbb{R}^3$, edges $E = \{(i,j)|i,j \in V\}$ and triangles $T = \{(i,j,k)|(i,j),(j,k),(k,i) \in E\}$. We define a surface curve C as a sequence of points $\mathbf{p}_i \in \mathbb{R}^3$, which are connected by line segments and which lie on the surface mesh M. In particular, we consider surface curves, where for any segment i, its end points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are contained in two edges of a triangle $\Delta \in T$.

5. Initialization

The input of the algorithm consists of the initial curve and the desired curvature, together with the maximum distance to the initial curve. The initial curve is typically determined by the user interaction. The user adds points on the mesh that is then connected by the shortest paths. If three adjacent points lie on one triangle, i.e., each point is associated with one vertex of that triangle, then we delete the middle point to uniquely define the curve. The initial curve could be provided by other methods, such as feature-based segmentation. In the following, we assume that

the curve consists of a sequence of vertices that are connected by edges, i.e., $\mathbf{p}_i = \mathbf{x}_j, \mathbf{p}_{i+1} = \mathbf{x}_k$ with $(j,k) \in E$.

Desired curvature: Every point on the curve is assigned its initial geodesic curvature κ_g, with

$$\kappa_g = \pi - \frac{2\pi\beta}{\theta}$$

where θ is the total point angle, i.e., the sum of the internal angles of the adjacent triangles at the point, and β is one of the two curve angles. We choose β as the minimal angle of the two curve angles (see, e.g., [17]). If the curve intersects an edge, then we have $\theta = 2\pi$, which yields $\kappa_g = \pi - \beta$.

Additionally, every curve point $\mathbf{p}_i$ is assigned a *desired geodesic curvature* $\kappa_d(\mathbf{p}_i)$. Our algorithm aims at moving curve points to positions where they have the approximate desired geodesic curvature. The desired geodesic curvature $\kappa_d(\mathbf{p}_i)$ is calculated by performing a linear interpolation between the geodesic curvature κ_g and 0 using a user-specified value t, as follows:

$$\kappa_d(\mathbf{p}_i) = t \cdot \kappa_g(\mathbf{p}_i), \quad t \in [0,1]. \tag{1}$$

Thus, for the value of $t=1$, the algorithm should return the initial curve, and for $t=0$, the smoothed curve represents the straightest geodesic. This arrangement means that the curve is as smooth as possible but can largely deviate from the initial curve. In this case, we obtain the same result as Martínez et al. [14].

Maximum distances: We restrict the movements of the curve points to an allowable region defined in terms of the Euclidean distance from the initial curve on the surface. If the initial curve points coincide with the vertex positions, then the allowable region T_{dist} is defined as the Euclidean distances of these vertices, which are less than *dist*, as illustrated in Fig. 2. Formally, the distances can be determined for instances that have the fast marching approach or the geodesics in the heat approach [21,22]. The domain experts can use a slider from 0 to the maximal distances from the initial curve on the hole surface. By interactively changing the values, the user obtains visual feedback via a contour line that depicts the corresponding distance from the curve. Therefore, the user can decide how far the smoothed curve is allowed to move.

For our experiments, we used an allowable region determined by the 2-neighborhood of the vertices that coincide with the initial curve points. If a straightest geodesic curve is desired, i.e., $t=0$, then the curve should be allowed to move freely on the entire mesh.

6. Curve smoothing and splitting step

The core part of our method is the iterative smoothing stage after initialization. In this process, the curve is relaxed such that curve points move but stay on edges. This process requires special treatment in the case where a curve point coincides with a vertex. We construct our algorithm such that a curve point never moves across a vertex. This approach leads to two cases for the *smoothing* step for a single curve point $\mathbf{p}_i$:

Case 1: $\mathbf{p}_i$ is located on an edge.
Case 2: $\mathbf{p}_i$ is located on a vertex.

In the first case, $\mathbf{p}_i$ can be moved in two directions. Its destination is on the edge or on one of the vertices that span the edge. The second case requires a *splitting*: as the point moves away from the vertex, new curve segments are required. Thus, $\mathbf{p}_i$ must be split into multiple curve points, each of which is located on edges incident to the vertex. We describe the two cases for a single point relaxation and analyze their properties. One curve smoothing step in the iteration consists of the relaxation of all of the curve points.

6.1. Case 1: $\mathbf{p}_i$ is located on an edge

Let $\mathbf{p} := \mathbf{p}_i \in C$ denote a curve point on an edge e, and let $\mathbf{p}_- := \mathbf{p}_{i-1}$ and $\mathbf{p}_+ := \mathbf{p}_{i+1}$ denote its neighbors. If $\mathbf{p}$ is placed on an edge $e \in E$, then we apply a Laplacian smoothing that determines a new position of $\mathbf{p}$ as a linear combination of $\mathbf{p}_-$ and $\mathbf{p}_+$:

$$\mathcal{L}_e(\mathbf{p}) = \mathbf{p} + (\lambda \cdot \omega_- \operatorname{proj}_e(\mathbf{p}_- - \mathbf{p}) + \lambda \cdot \omega_+ \operatorname{proj}_e(\mathbf{p}_+ - \mathbf{p})).$$

In contrast to a standard Laplacian relaxation, the linear combination *weights* $\omega_\pm$ with the *correction factor* λ are constructed such

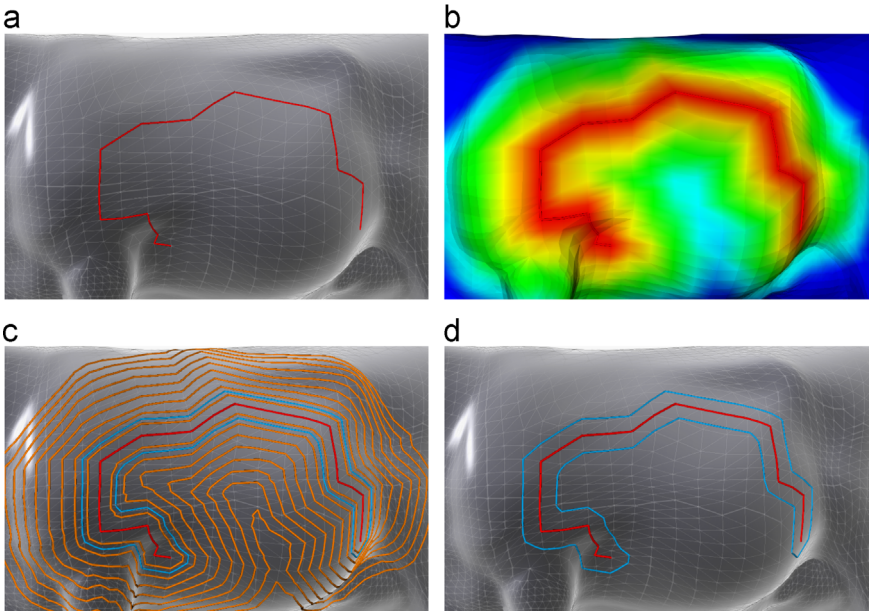

Fig. 2. Movement restriction: first, the initial curve is depicted in red (a). Afterward, the geodesic distance is computed entirely on the mesh (b). Finally, the user can interactively change the distance. The result of the allowable region is depicted in cyan, and further contours are depicted in orange (c). The final allowable region is illustrated in (d). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

that $\mathcal{L}_e(\mathbf{p})$ keeps $\mathbf{p}$ on the edge e. The operator proj_e projects point onto the line spanned by e. Furthermore, we postulate $\omega_\pm \geq 0$ and $\omega_+ + \omega_- = 1$. This arrangement leads to the following:

$$\mathcal{L}_e(\mathbf{p}) = (1-\lambda)\cdot\mathbf{p} + \lambda\cdot(\omega_-\,\text{proj}_e(\mathbf{p}_-) + \omega_+\,\text{proj}_e(\mathbf{p}_+)). \tag{2}$$

The crucial part remains the definition of the *weights* $\omega_\pm$ and the *correction factor* λ such that convergence to the desired curvature is achieved.

Computation of the weights: The curve point $\mathbf{p}$ is located on the edge e. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ denote the vertices that span e. We define

$$\omega_- = \frac{\text{dist}(\mathbf{p}_+, \text{proj}_e(\mathbf{p}_+))}{\text{dist}(\mathbf{p}_+, \text{proj}_e(\mathbf{p}_+)) + \text{dist}(\mathbf{p}_-, \text{proj}_e(\mathbf{p}_-))}$$

and

$$\omega_+ = \frac{\text{dist}(\mathbf{p}_-, \text{proj}_e(\mathbf{p}_-))}{\text{dist}(\mathbf{p}_+, \text{proj}_e(\mathbf{p}_+)) + \text{dist}(\mathbf{p}_-, \text{proj}_e(\mathbf{p}_-))},$$

where $\text{dist}(\cdot,\cdot)$ denotes the Euclidean distance.

First, we will prove that setting $\lambda = 1$ yields a shortest path that connect $\mathbf{p}_-$ and $\mathbf{p}_+$ via a point on the edge.

Theorem 6.1. *Let $\lambda = 1$. Then, the partial curve spanned by the sequence $[\mathbf{p}_-, \mathcal{L}_e(\mathbf{p}), \mathbf{p}_+]$ is a geodesic in M.*

Proof. Let $\mathbf{p}^*$ be the point on edge e such that the distance between $\mathbf{p}_-$ and $\mathbf{p}_+$ via $\mathbf{p}^*$ is minimal, i.e., $[\mathbf{p}_-, \mathbf{p}^*, \mathbf{p}_+]$ is a geodesic curve in M. We assume the general case that $\mathbf{p}_-, \mathbf{p}_+, \mathbf{x}_1$ and $\mathbf{x}_2$ are not collinear, which would degenerate to the trivial case. We simplify the problem by two rigid transformations: first, we shift $\mathbf{p}_-$ to the origin. Then, we rotate the configuration such that the linear segment $[\mathbf{x}_1, \mathbf{x}_2]$, which spans e, is located in the xy-plane and is perpendicular to the x-axis. Finally, we rotate *only* the point $\mathbf{p}_+$ around the edge e such that $\mathbf{p}_+$ lies in the xy-plane. The first translation and the rotation of the whole configuration preserve the lengths. The last rotation unfolds the two neighboring triangles at e into the xy-plane, i.e., it preserves the lengths as measured on the surface mesh M. Fig. 3(a) illustrates the situation.

We show that $\mathbf{p}^* = \mathcal{L}_e(\mathbf{p})$, and it is sufficient to do this step in $\mathbb{R}^2$ after transformation into the xy-plane with $\mathbf{p}_- = (0,0)^T$, $\mathbf{p}_+ = (x_+, y_+)^T$, $\mathbf{x}_1 = (x_e, y_1)^T$, and $\mathbf{x}_2 = (x_e, y_2)^T$. The line through $\mathbf{p}_-$ and $\mathbf{p}_+$ is given as $x_+ y = y_+ x$ (with $x_+ \neq 0$) for the nontrivial case. We obtain $\mathbf{p}^*$ as the intersection of the edge and the straight line through $\mathbf{p}_-$ and $\mathbf{p}_+$ with $\mathbf{p}^* = (x_e, (y_+/x_+)x_e)$. It remains to show the equality $\mathcal{L}_e(\mathbf{p}) = \mathbf{p}^*$. Using Eq. (2) with $\lambda = 1$, we obtain:

$$\begin{aligned}\mathcal{L}_e(\mathbf{p}) &= \frac{\text{dist}(p_+, \text{proj}_e(p_+))}{\text{dist}(p_+, \text{proj}_e(p_+)) + \text{dist}(p_-, \text{proj}_e(p_-))}\cdot \text{proj}_e(p_-) \\ &\quad + \frac{\text{dist}(p_-, \text{proj}_e(p_-))}{\text{dist}(p_+, \text{proj}_e(p_+)) + \text{dist}(p_-, \text{proj}_e(p_-))}\cdot \text{proj}_e(p_+) \\ &= \frac{x_+ - x_e}{x_+ - x_e + x_e}\text{proj}_e(p_-) + \frac{x_e}{x_+ - x_e + x_e}\text{proj}_e(p_+) \\ &= \frac{x_+ - x_e}{x_+}\cdot\begin{pmatrix} x_e \\ 0\end{pmatrix} + \frac{x_e}{x_+}\cdot\begin{pmatrix} x_e \\ y_+\end{pmatrix} = \begin{pmatrix} x_e \\ x_e\frac{y_+}{x_+}\end{pmatrix} = p^*,\end{aligned}$$

which proves Theorem 6.1. □

Computation of the correction factor: The correction factor λ ensures that a point on the curve will evolve on the surface such that the desired geodesic curvature is achieved. To determine λ, we unfold two neighboring triangles in the plane as before. However, we rotate the planar configuration such that the line ℓ through connecting points $\mathbf{p}_-$ and $\mathbf{p}_+$ coincides with the x-axis (see Fig. 3(b)). The point $\mathcal{L}_e(\mathbf{p})$ along the edge e is spanned by $\mathbf{x}_1$ and $\mathbf{x}_2$.

First, we generalize Thales' theorem. Given the points $\mathbf{p}_-, \mathbf{p}_+$, we want to determine the function $(x, y(x))$ such that the enclosed angle $\sphericalangle(\mathbf{p}_-, (x, y(x)), \mathbf{p}_+) = \vartheta$. Here, ϑ is the converted angle from κ_d, as described in [17]. With $T := \tan(\vartheta - \pi)$ and $L := \|\mathbf{p}_+ - \mathbf{p}_-\|$, we obtain

$$y(x) = \frac{L}{2T} + \sqrt{\frac{L^2}{4T^2} + x(L-x)}. \tag{3}$$

This statement can be proven by using the definition of tan and by applying trigonometric identities:

$$\vartheta = \arctan\frac{x}{y} + \arctan\frac{L-x}{y} = \pi + \arctan\frac{\frac{x}{y} + \frac{L-x}{y}}{1 - \frac{x(L-x)}{y^2}}$$

$$\tan(\vartheta - \pi) = \frac{yL}{y^2 - x(L-x)}.$$

Next, we want to determine the position $\mathbf{p}^{*\prime}$ on the edge e, where $\sphericalangle(\mathbf{p}_-, \mathbf{p}^{*\prime}, \mathbf{p}_x) = \vartheta$. Thus, we must calculate the intersection point of $(x, y(x))$ and the span of the edge e. Edge e intersects the x-axis at an angle $\gamma < \pi/2$ in such away that the span can be written as $y(x) = \tan(\gamma)(x-u)$ or

$$x = \frac{y}{\tan\gamma} + u. \tag{4}$$

Inserting (4) into (3) leads to

$$y = \frac{L}{2T} + \sqrt{\frac{L^2}{4T^2} + \left(\frac{y}{\tan\gamma} + u\right)\left(L - \frac{y}{\tan\gamma} - u\right)}.$$

Further simplification yields

$$0 = y^2\underbrace{\left(1 + \frac{1}{\tan\gamma^2}\right)}_{=:\alpha} - y\underbrace{\left(\frac{L}{T} + \frac{L - 2u}{\tan\gamma}\right)}_{=:\beta} - u(L-u).$$

A positive solution for y is obtained as

$$y^* = \frac{\beta}{2\alpha} + \sqrt{\frac{\beta^2}{4\alpha^2} + \frac{u(L-u)}{\alpha}}.$$

To determine λ, we first calculate the altitude of the triangle:

$$d = \frac{\|(\mathbf{p} - \mathbf{p}_-) \times (\mathbf{p}_+ - \mathbf{p}_-)\|}{\|\mathbf{p}_+ - \mathbf{p}_-\|}$$

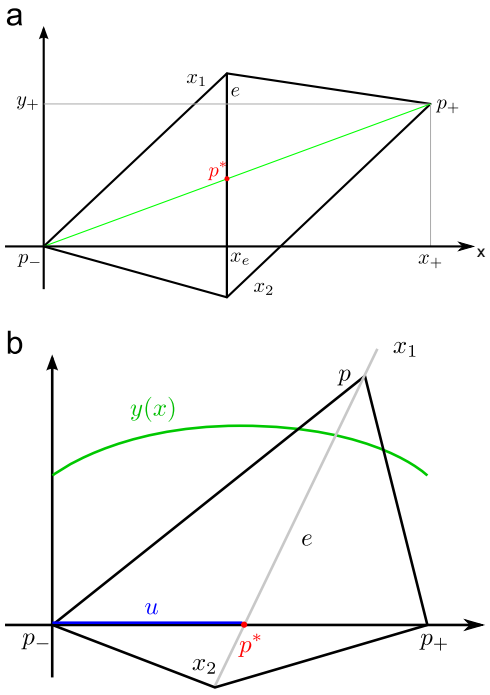

Fig. 3. We transformed the problem in the xy-plane and search for the point p*.

and set the correction factor λ to:

$$\lambda = 1 - \frac{y^*}{d}.$$

Then, for $\vartheta \to \pi$, the curve is locally a straightest geodesic with $\lambda \to 1$, and Theorem 6.1 applies. Hence, the smoothing operator $\mathcal{L}_e$ with the weights ω_- and ω_+ and the correction factor λ defined as above yields the straightest geodesic curves.

6.2. Case 2: $\mathbf{p}_i$ is located on a surface vertex

The second case applies if the curve point $\mathbf{p}$ is located on a surface vertex $i \in V$, i.e., $\mathbf{p} = \mathbf{x}_i$. In this case, plain relaxation as described in Section 6.1 is not sufficient. Instead, the curve must be locally split into multiple segments, which are spanned between the edges in star$(i) := \{(i,j) \in E | j \in V\}$. Given the curve segment $[\mathbf{p}_-, \mathbf{p}, \mathbf{p}_+]$, we partition the neighbor vertices j of i in star(i) into two sequences. The first sequence N_1 enumerates the neighbors counterclockwise, starting from the edge of $\mathbf{p}_-$ and ending on the edge of $\mathbf{p}_+$. The second sequence N_2 enumerates the remaining neighbors clockwise around vertex i. Let $\mathbf{p}_k^1$ and $\mathbf{p}_k^2$ denote the positions of the vertices in N_1 and N_2, respectively, with the additional end points $\mathbf{p}_0^1 = \mathbf{p}_0^2 = \mathbf{p}_-$ and $\mathbf{p}_m^1 = \mathbf{p}_n^2 = \mathbf{p}_+$.

Splitting: After local relaxation, $\mathbf{p}$ will be replaced by a new sequence of points located on the edges $E_1 = \{(i,j) | j \in N_1\}$ or $E_2 = \{(i,j) | j \in N_2\}$. The decision between the two options is based on a local parameterization of $\mathbf{p}_k^1$ and $\mathbf{p}_k^2$: we cut the local surface patch along $[\mathbf{p}_-, \mathbf{p}, \mathbf{p}_+]$ and unfold the two parts using an exponential map. The exponential map is a local map from the tangential space around a vertex to the mesh. Around a small neighborhood, this map is a diffeomorphism. Therefore, we use this definition to unfold the star of a vertex on the corresponding tangent space (see, e.g., [17]).

Smoothing: Based on the configuration described above, we compute new curve points between $\mathbf{p}_0^c$ and $\mathbf{p}_+$ in the split curve segment as follows. We turn counterclockwise $(\mathbf{p}_k^1)$ and clockwise $(\mathbf{p}_k^2)$ around the vertex at $\mathbf{p} = \mathbf{x}_i$. Starting with $k = 0$, we consider pairs $\mathbf{p}_k^c, \mathbf{p}_{k+1}^c$ to determine the position of a new curve point $\mathbf{p}'$ on the line segment $[\mathbf{x}_i, \mathbf{p}_{k+1}^c]$, as described in Section 6.1. Then, we set $\mathbf{p}_{k+1}^c \leftarrow \mathbf{p}'$ and proceed with $k \leftarrow k+1$ until $\mathbf{p}_k^c = \mathbf{p}_-$ is reached. Fig. 4 illustrates this procedure. There can arise invalid configurations in this process whenever a line segment is not contained in the unfolded parameter domain. In this case, we replace the line segment by parts of the domain boundary.

Finally, we compare the lengths of the two curve segments $\mathbf{p}_k^1$ and $\mathbf{p}_k^2$ after splitting and smoothing, and we choose the shorter one to replace the original segment $[\mathbf{p}_-, \mathbf{p}, \mathbf{p}_+]$. For this result, we prove the following theorem.

Theorem 6.2. *Let $\lambda = 1$. Performing splitting and smoothing as described above computes the shortest surface curve connecting $\boldsymbol{p}_-$ and $\boldsymbol{p}_+$.*

Proof. For each patch N_1 and N_2 unfolded in the plane, we find the curve points on the surface edges that minimize the distances locally for each segment of the split curve if the sum of the inner angles is less than π (Theorem 6.1). The first line from $\mathbf{p}_-$ to $\mathbf{p}_+$ over $\mathbf{p}^{c'}$ on the line segment $[\mathbf{x}_i, \mathbf{p}_1^c]$ determines the remaining points on the edges $(\mathbf{p}, \mathbf{p}_2^c), \ldots (\mathbf{p}, \mathbf{p}_{|N_c|-1}^c)$. We must calculate the intersection point of the line and the remaining edges. This approach minimizes the distance between $\mathbf{p}_-$ and $\mathbf{p}_+$ because the shortest connection on a plane is a straight line. The parameterization preserves the length of the line on the surface and the angle between the line and its edges. Because the parameterization unfolds the fan in the plane and we can find the shortest line from $\mathbf{p}_-$ to $\mathbf{p}_+$, we use the intersection points as candidates for the points that lie on the edges of the neighboring set N_c. Afterward, the resulting points are transformed back to the surface. Furthermore, if the connection line between $\mathbf{p}_-$ and $\mathbf{p}_+$ is a non-valid line, i.e., some line parts are outside the triangle fan, we then replace it with the shortest line between $\mathbf{p}_-$ and $\mathbf{p}_+$. □

6.3. Convergence of the algorithm

Our goal is to ensure that the algorithm terminates. If there is no number of iterations given, then we define constraints when a point is allowed to move. These rules ensure convergence of the curvature. First, we show that the length of the curve decreases for $\lambda = 1$. This approach guarantees that the length converges during the iteration. Then, for the case $\lambda \neq 1$, we define two constraints when a point is allowed to move. This strategy guarantees that the curvature converges during the iteration. Finally, we define the same abort criterion for $\lambda = 1$ and for $\lambda \neq 1$.

Theorem 6.3. *Let $\lambda = 1$. The length of the curve decreases for every iteration step.*

Proof. In every iteration step, a virtual line sweeps over each curve point and reduces the distance between its predecessor and successor point. Thus, instead of proving that every iteration step reduces the length of the curve from the previous iteration, we show that the sweep line shortens the length of the curve in each iteration step. For the first sweep line position, the line is placed at a curve point $\mathbf{p}$. We keep the points $\mathbf{p}_-, \mathbf{p}_+$ and move $\mathbf{p}$ to $\mathbf{p}'$ or to $\mathbf{p}_1', \mathbf{p}_2', \ldots, \mathbf{p}_n'$, depending on whether $\mathbf{p}$ lies on an edge or on a vertex in which the distance of $\mathbf{p}_-$ to $\mathbf{p}_+$ via $\mathbf{p}'$ or $\mathbf{p}_1', \mathbf{p}_2', \ldots, \mathbf{p}_n'$, respectively, is minimal. The next sweep line position is the point $\mathbf{p}_+$, which is moved to its relative optimal position in terms of its predecessor and successor (recall Fig. 1). As the sweep line reduces the length of the curve in the first sweep, it also reduces the length in the next sweep step. When the sweep line reaches the end, the length of the curve is decreased, and thus, the length is reduced after each iteration step. □

For $\lambda \neq 1$, we restrict our curve to change only if the geodesic curvature decreases in every iteration step. To ensure this aspect, we apply two different constraints to the smoothing algorithm.

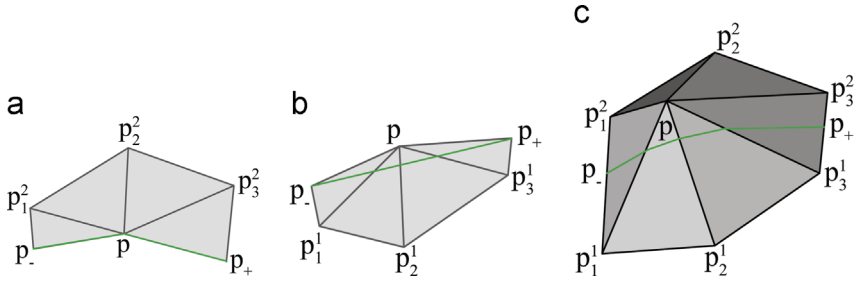

Fig. 4. Locally shortest paths after cutting and unfolding the neighbors N_1, N_2 of the domain in (a) restricted to the local domains (b) and (c). We select the shorter curve.

First, whenever the current geodesic curvature $\kappa_{cur}(\mathbf{p})$ of a certain point $\mathbf{p}$ is less than or equal to the desired geodesic curvature $\kappa_d(\mathbf{p})$ of this point

$$\kappa_{cur}(\mathbf{p}) \leq \kappa_d(\mathbf{p}), \tag{5}$$

we do not allow the point to move. Second, a point $\mathbf{p}$ is allowed to move only if the sum of the geodesic curvature of this point and its predecessor and successor point after the movement is less than or equal to the sum of the geodesic curvatures κ' before movement

$$\kappa(\mathbf{p})+\kappa(\mathbf{p}_+)+\kappa(\mathbf{p}_-) \leq \kappa'(\mathbf{p})+\kappa'(\mathbf{p}_+)+\kappa'(\mathbf{p}_-). \tag{6}$$

Therefore, the sum of the geodesic curvatures decreases for every iteration step. The proof is similar to the proof of Theorem 6.3. We use both properties as an abort criterion for our smoothing algorithm. For $\lambda \neq 1$ ($t \neq 0$), we showed that the sum of the geodesic curvatures decreases for every iteration step. Furthermore, we know that $\sum\kappa \geq 0$ is a bounded and monotonic series, which means that it converges. Thus, the abort criterion is defined in such a way that the smoothing process stops if the change in the geodesic curvature from one iteration step to the next is not significant. The case $\lambda=1$ leads to a curve shortening flow. Recent work addresses the convergence of this flow with closed initial curves. Ma and Chen [26] showed that if the shortening flow exists for a finite amount of time on a compact Riemannian manifold and the limit of the length of the curve is greater than zero, the limiting curve exists and is a geodesic. Furthermore, one characteristic is that the derivative of the curve's length is equal to the negative integral over the squared curvature:

$$\frac{d}{dt}L = -\int \kappa^2 \, \mathrm{ds}.$$

The length also converges because it is a bounded monotonic series; thus, the derivative becomes zero for infinite time steps, and the curvature goes to zero as well. Therefore, we can use our abort criterion for $\lambda=1$ as well as for $\lambda \neq 1$.

7. Curve evaluation

After each smoothing iteration, the curve is evaluated to test the stopping criteria and critical vertex configurations. Critical vertices prevent fast smoothing and could lead to sharp edges. Thus, they must be identified and handled by merging.

Abort criteria: We use two criteria to stop the smoothing process. First, the smoothed curve should not move too far from the initial curve. This constraint is ensured by restricting the movement of the curve points to the allowable envelope defined in Section 5. Whenever a point would move out of this region, the iteration is stopped. Second, the iteration stops if the curve has converged to the prescribed desired geodesic curvature κ_d. We define a new tolerance parameter τ to relax the interpolation of the desired geodesic curve. If every current geodesic curvature deviates from the desired geodesic curvature by *less* than the defined τ-percentage, then this relationship stops the iteration.

Critical vertices: Critical vertices are vertices that are "surrounded" by curve points; they converge to these vertices but do not *cross* them (see Fig. 5). This arrangement means that these vertices prevent fast convergence of the smoothing, and they must be treated specially.

A vertex $i \in V$ is *potentially* a critical vertex if there are curve segments with the associated curve points $\mathbf{p}_i, \mathbf{p}_{i+1}, \ldots, \mathbf{p}_{i+\ell}$, where a subsequence $\mathbf{p}_{i+1}, \ldots, \mathbf{p}_{i+\ell-1}$ is located on the edges $e_{i+1}, \ldots, e_{i+\ell-1}$ connected to vertex i, *and* the points $\mathbf{p}_i, \mathbf{p}_{i+\ell}$ are located on edges that do not contain the vertex i. After each iteration step, we identify these candidates and "simulate" the following merge operation: the curve points $\mathbf{p}_{i+1}, \ldots, \mathbf{p}_{i+\ell-1}$ are merged to a single point located at the vertex position $\mathbf{x}_i$. We test if the ratio of the lengths of the curve after and before merging exceeds a threshold ε. In this case, we apply the merge operation. Otherwise, we continue with the original curve. If we want to find the straightest geodesic, i.e., the parameter $t=0$ (see Section 5), then we omit ε. This circumstance means that merging is applied whenever the new length becomes shorter. For $t \neq 0$, we suggest a value of $\varepsilon=0.98$ based on empirical observations. This arrangement ensures a merging whenever the curve length from $\mathbf{p}_{i+1}$ to $\mathbf{p}_{i+1-1}$ is at least ten times shorter than the minimal distance from $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$ or from $\mathbf{p}_{i+1-1}$ to $\mathbf{p}_{i+l}$. After merging the points $\mathbf{p}_{i+1}, \ldots, \mathbf{p}_{i+\ell-1}$ to the vertex position $\mathbf{x}_i$, we use the median of the desired geodesic curvature of the points and assign it to the new point.

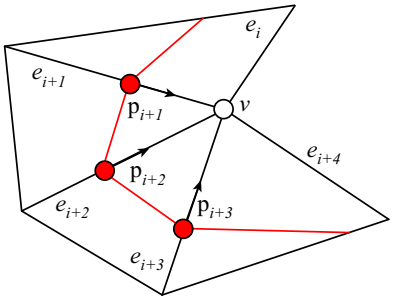

Fig. 5. Critical vertex: the red curve converges toward **v** but does not "cross" the vertex. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

8. Algorithm

Our algorithm can be summarized with the pseudo-code shown in Listing 1. The functions `initialize()`, `computeDesiredAngles()`, and `computeAllowableRegion()` create the input curve and compute the desired geodesic curvature and a maximum distance-based feasible region, as described in Section 5. Then, we start the smoothing iteration. In every iteration, we apply the following for every point $\mathbf{p}_i$: Here, the function `getNeighborSets()` computes the sets N_1 and N_2, and `selectNeighbor()` selects the appropriate set depending on the unfolded configuration. Finally, `relocatePoint()` evaluates the new position of $\mathbf{p}_i$ (see Section 6). After each iteration, the function `testAbortCriteria()` is evaluated, which eventually terminates the algorithm. Special cases are treated in the last function `handleCriticalVertices()` (Section 7).

Algorithm 1. Pseudo-code for the algorithm.

```
initialize(p)
[φ]=computeDesiredAngles(p)
T=computeAllowableRegion(p)
for smoothing iterations
  for all curve points p_i
     [N1,N2]=getNeighborSets(p_i)
     selectNeighbor(p_i,N1,N2)
     p_i=relocatePoint(p_i, φ_i,[N1,N2])
  end
  if (testAbortCriteria(p,T)) then break; end
  handleCriticalVertices()
end
```

We close the presentation with a few remarks. For simplicity, we compare only the distances of the points on the current curve to their corresponding original points. If a point is split, then we assign the k-neighborhood distance of that point to the newly inserted points for the allowable region test. Similarly, if the points are merged, then we assign the maximum k-neighborhood distance of all of the merged points to the new point. Finally, the points are relocated only if the current geodesic curvature is

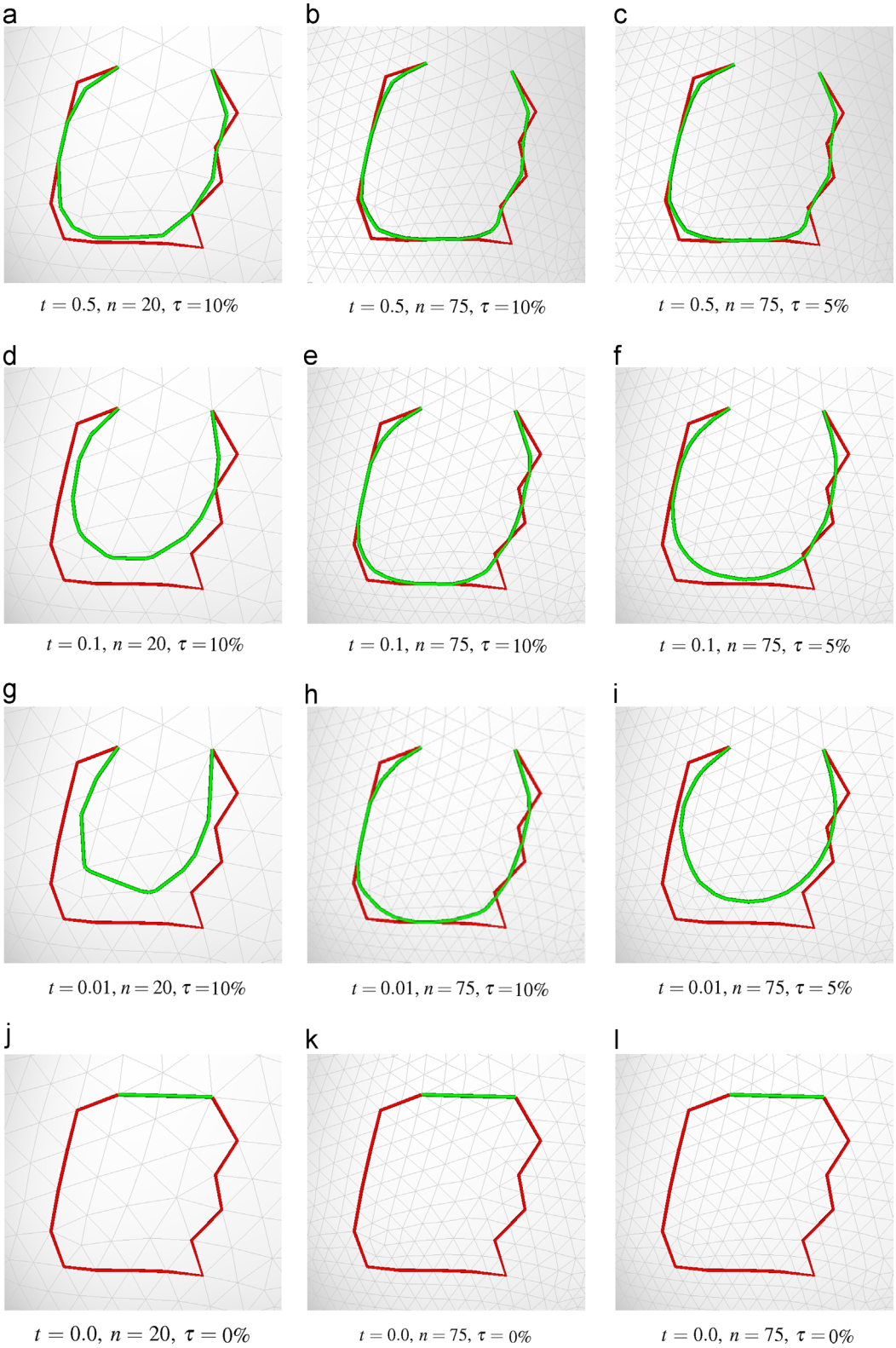

Fig. 6. Convergence effect on different tessellations when varying the parameters t, τ, and number of iterations $\in$ for a short curve.

greater than the desired geodesic curvature and if the current geodesic curvature deviates from the desired geodesic curvature by more than τ-percentage (recall Section 7). Our experiments demonstrate that $\tau = 10\%$ is a reasonable value.

9. Results and application

We evaluate our method on artificial and real-world surface datasets to verify its robustness and convergence. By convergence,

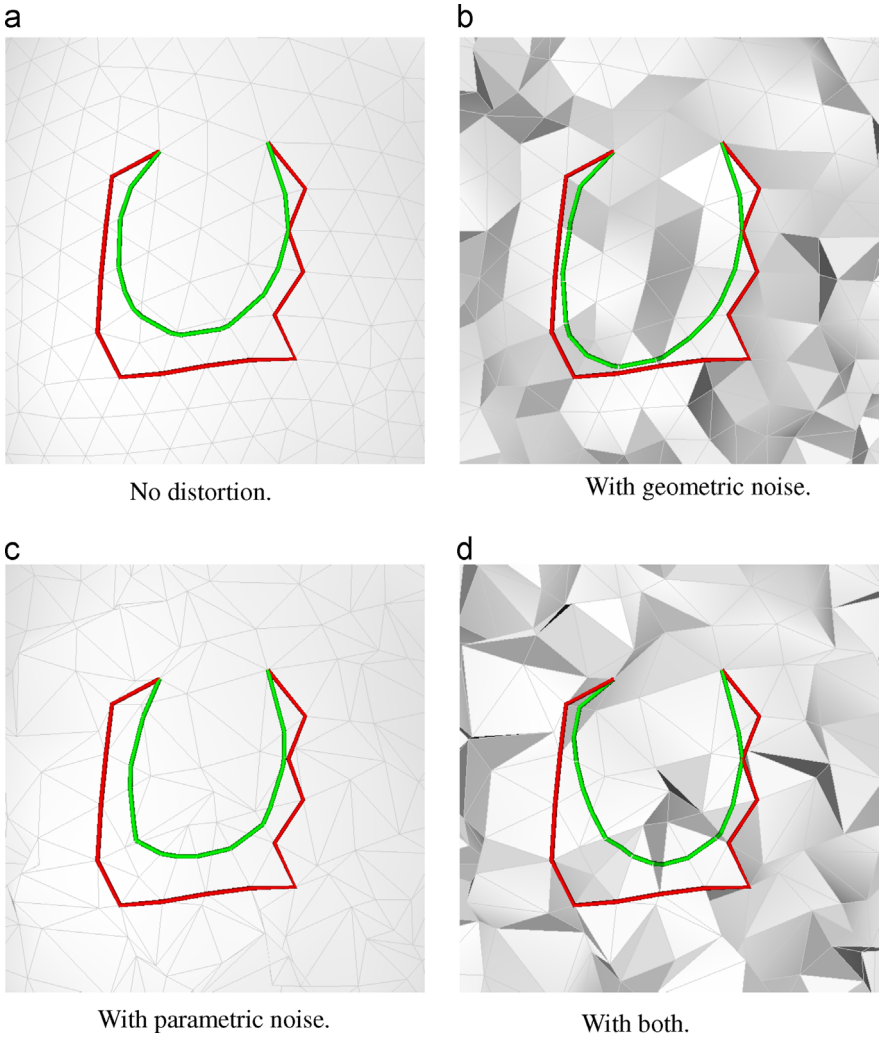

Fig. 7. Testing robustness toward geometric (distortion in the normal direction) and parametric (distortion in the tangent space) noise for $t=0.1$ and 20 iterations.

Table 1
Quantitative results of the robustness experiment based on cubic polynomial surfaces. For each parameter setting (#iteration, t, δ, and noise), several quantitative measures between the two resulting curves are compared: κ_g=geodesic curvature before smoothing, κ_g'=geodesic curvature after smoothing, ratio between κ_g and κ_g', which should correspond to t, d=Hausdorff distance, and $d\%$=percentage of deviation of the Hausdorff distance to the straightest geodesic curve.

#Iteration	t	δ	Parametric noise	Geometric noise	κ_g	κ_g'	$\frac{\kappa_g'}{\kappa_g}$	$d \cdot 10^2$	$d\%$
20	0.5	0.25	0	0	13.41	6.47	0.48	0.82	58.57
20	0.1	0.25	0	0	13.38	4.98	0.37	1.39	99.28
20	0.0	0.25	0	0	13.40	4.98	0.37	1.40	100
100	0.5	0.25	0	0	13.39	6.48	0.48	0.82	21.10
100	0.1	0.25	0	0	13.41	1.41	0.11	3.17	82.20
100	0.0	0.25	0	0	13.37	0.36	0.03	3.86	100
100	0.5	0.25	0.01	0	21.75	10.37	0.48	0.41	10.63
100	0.1	0.25	0.01	0	20.18	1.98	0.10	3.03	77.22
100	0.0	0.25	0.01	0	22.05	0.45	0.02	3.92	100
100	0.5	0.25	0.00	0.01	14.31	6.68	0.47	0.74	19.39
100	0.1	0.25	0.00	0.01	14.32	1.44	0.10	3.19	83.15
100	0.0	0.25	0.00	0.01	14.17	0.32	0.04	3.83	100
100	0.5	0.75	0.02	0.03	28.52	12.38	0.44	0.57	16.32
100	0.1	0.75	0.02	0.03	30.68	2.98	0.10	1.53	44.03
100	0.0	0.75	0.02	0.03	30.13	0.87	0.03	3.47	100

we mean that the condition $t \approx \kappa_g'/\kappa_g$ is fulfilled, with κ_g' being the geodesic curvature after smoothing. The real-world data are anatomical surfaces that are patient-specific and representative for two medical applications: vascular models of cerebral aneurysms for decomposition and liver models for resection planning. All of the tests are performed on an Intel Core 2 Duo CPU at

3.16 GHz. The memory requirements for the curve smoothing are negligible compared to the memory required by the datasets.

To specify an initial curve, the user selects a sequence of vertices connected by shortest edge paths using Dijkstra's algorithm. Furthermore, the user specifies the parameter t, which defines the globally desired geodesic curvature. Optionally, the user can relax this specification by varying the tolerance parameter τ (Section 8). For all of the experiments, we use $\tau = 10\%$ unless otherwise specified.

9.1. Convergence and robustness

We performed two experiments to assess the convergence and robustness of our approach. For the convergence, we investigated the smoothing for different parameter settings and mesh resolutions. Fig. 6 shows the results for varying the parameters $t \in \{0.05, 0.1, 0.01, 0\}$ and $\tau \in \{10\%, 5\%, 0\%\}$ after a fixed number of iterations $\in \in \{20, 75\}$. We observe that, as expected for a decrease in t, the curve changes gradually from the initial curve to the straightest geodesic. In the second column of Fig. 6, we subdivide each triangle into four triangles. In this case, we increase the number of iterations from 20 to 75. We observe that for $\tau = 10\%$, the curve converges to a smooth curve close to the original unless we set $t = \tau = 0$, for which the curve converges to a straightest geodesic. The third column of Fig. 6 represents the comparison to a setting with $\tau = 5\%$: as expected, the result is smoother at the cost of a larger distance from the original curve.

However, given a disadvantageous initial curve, the method cannot ensure that the smoothed curve fulfills the condition $t \approx \kappa_g'/\kappa_g$, with κ_g' as the geodesic curvature after the smoothing. This limitation can be easily seen for surfaces that have a hole, an initial curve that wraps around this hole and $t = 0$.

To investigate the robustness, we add geometric noise, i.e., displacements in the normal directions, and parametric noise, i.e., displacements in the tangential directions. To be comparable, we keep the vertices that coincide with the initial curve at their original position. Therefore, they are not influenced by the noise.

For this experiment, we always use 20 iterations for $t = 0.1$ and $\tau = 10\%$ (see Fig. 7(a)). For each scenario, we notice some influence of the noise on the result.

In addition to the qualitative experiment, we performed a quantitative test as well. We conducted the same experiment as Max [27] (in the context of normal fitting): we generate random cubic polynomials with coefficients in different ranges. The surfaces are of the form

$$f(x,y) = Ax^2 + Bxy + Cy^2 + Dx^3 + Ex^2y + Fxy^2 + Gy^3.$$

The coefficients A, B, C, D, E, F, and G are all uniformly distributed pseudo-random numbers in the interval $[-\delta, \delta]$ and $x, y \in [-1, 1]$. The domain was subdivided into a 40×40 grid in such a way that the distance of two neighboring vertices with the same x- and y-value is 0.05. Additionally, we add parametric and geometric noise to test for robustness. The noise value γ means that the vertices are translated randomly in a range of $[-\gamma, \gamma]$ in the domain or in the codomain for parametric and geometric noise, respectively. Table 1 shows the results of our experiments with a smooth surface and different types of added noise. We performed the test with several parameters and present the averaged total geodesic curvature κ_g before and κ_g' after the smoothing as well as the Hausdorff distance d between the two curves. For every parameter setting, we generated 50 random cubic polynomials. The relative distance d (as a percentage) expresses the deviation from the assumed maximal Hausdorff distance of the straightest geodesic curve with $t = 0$. Fig. 8 shows some results for different parameter settings. According to a quantitative comparison, several observations can be made. For each parameter setting, the geodesic curvature of the curve is decreased while remaining close to its initial curve. Thereby, the number of iterations influences how close the resulting curvature is to the prescribed curvature. A low number results in an increased deviation between κ_g'/κ_g and t compared to a higher number of iterations. Furthermore, it can be seen that the presence of noise leads to a slightly decreased geodesic curvature compared to the non-disturbed surface. The quantitative results, however, demonstrate an overall robustness with respect to the noise, which corresponds to observations from the qualitative comparisons.

As can be seen, the obtained curves are smooth and robust toward noise, and they exhibit reasonable convergence behavior. However, it is obvious that changing the vertex positions by adding geometric noise will not change the geodesic curvature (when keeping the initial curve points at their original position), but this action has, in fact, an influence on the operation space. Because the point can move only along the edges, changing the vertex position will change the intrinsic position of the smoothed curve, i.e., the relative position on the edge could have changed. Moreover, if the vertex positions are distorted in a normal

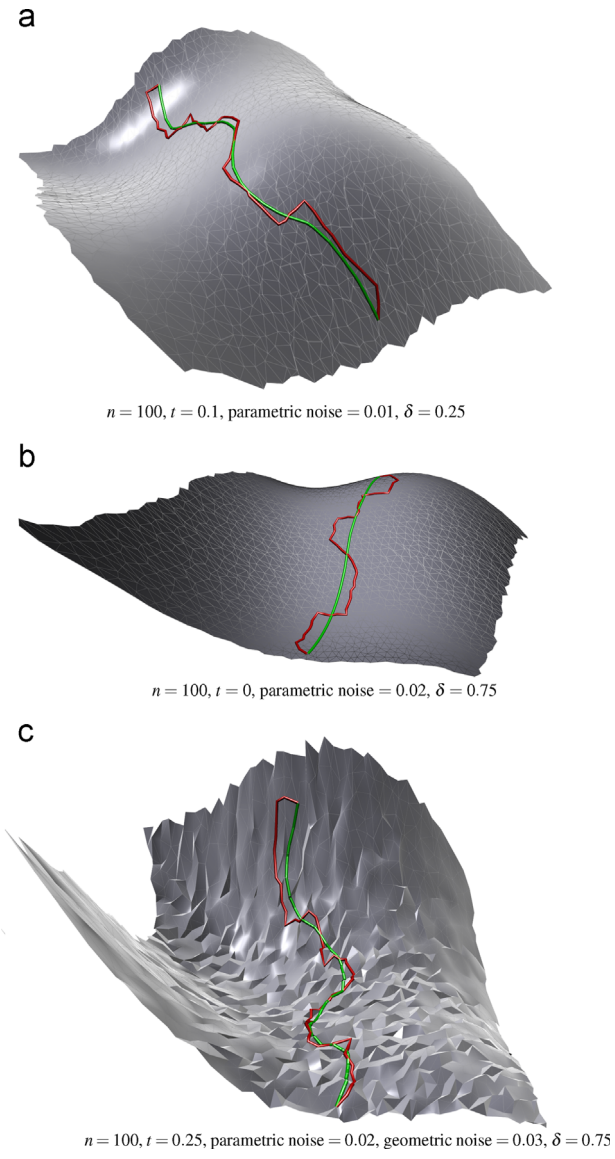

Fig. 8. Some results of qualitative experiments regarding convergence and robustness on cubic polynomials with different settings. The initial contour is indicated in red, and the smoothed contour is indicated in green. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

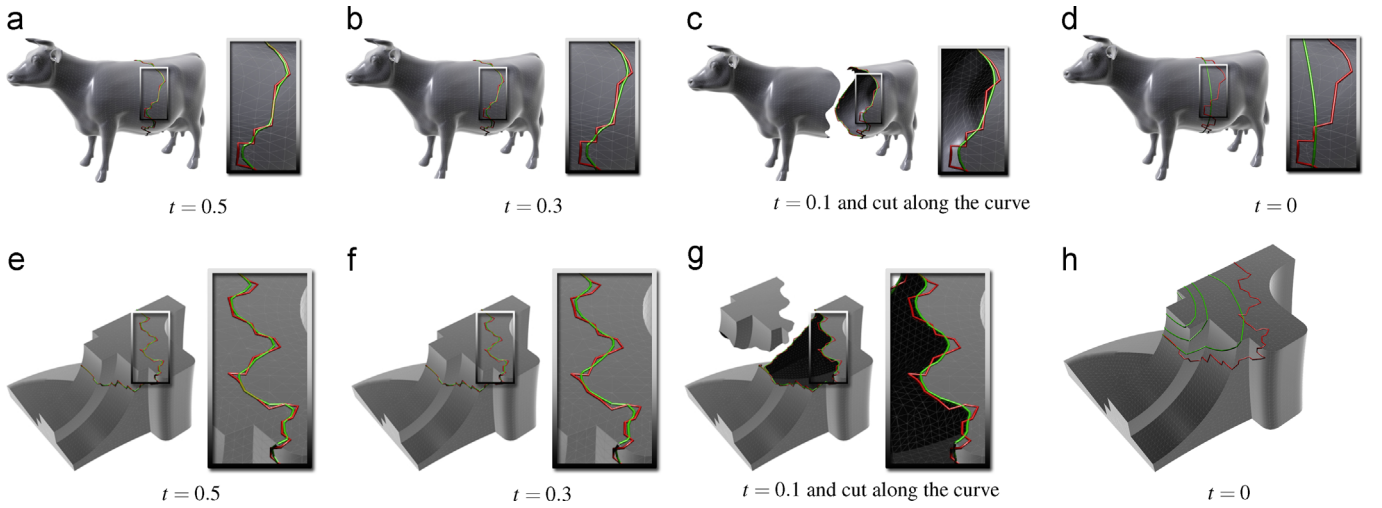

Fig. 9. The cow and the fandisk dataset with the initial curves (red) and smoothed curves (green) for different values of t. The figures (d) and (h) show the behavior of the curve for $t=0$. Iterations are performed until a sufficient desired overall curvature value is reached. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

direction, the geodesic curvature will change, and this arrangement leads to a different smoothed curve. Despite the different results, we can observe that the final curve is always smooth. Thus, the algorithm gives robust results even if the underlying surface is distorted in both the tangential and normal directions.

9.2. Application to large datasets

We applied our approach to benchmark surfaces and anatomical surfaces from medical image datasets. The anatomical surfaces exhibit low regularity and a significant amount of noise. In the experiments, we varied only the desired geodesic curvature parameter t; the number of iterations is fixed at 20, and $\tau=10\%$ is fixed.

Benchmark surfaces: Fig. 9 shows results for the *cow* and *fandisk* surface meshes. The initial surface curves are red and the resulting smoothed curves are green. The shapes of the initial curves are nontrivial; their lengths are relatively long and show additional close-ups. For the largest $t=0.5$, we obtain a smooth curve, which is located close to the initial curve. Decreasing t increases the amount of smoothing, and the curves do not move significantly away from the initial curve.

Anatomical surfaces: Fig. 10 shows results for the anatomical surfaces: bone structures (Fig. 9(a)), a cerebral aneurysm (Fig. 9 (b)), and a liver cut (Fig. 9(c) and (d)). The smooth curves remain close to the initial curves, and we do not observe any artifacts such as self-intersections. The parameter choice $t=0.1$ leads to a significant and comprehensible smoothing while closely imitating the original curve.

9.3. Comparison to the spline approximation

We compare our algorithm to a global approximation of the initial curve with B-splines. We emulate the B-spline approximation in manifolds by Hofer and Pottmann [16,28] by assuming and providing a global surface parameterization and resorting to a standard least-squares approximation. We use least-squares conformal maps [29] to construct the surface mesh parameterization. We fit cubic B-splines with a uniform knot vector. The Schoenberg–Whitney conditions are always satisfied by a regularization term, which penalizes the length and (linearized) curvature (see, e.g., [30]). This regularization not only guarantees a solution to the linear systems that arise but also accounts for minimizing exactly the same quantities as in [16]. We project the initial curve to the parameter space, apply the B-spline fitting, and map the result back onto the surface. In comparison with our new explicit curve-smoothing algorithm, we obtain similar results (see Fig. 11). Note, however, that the B-splines fitting requires either a global parameterization (whose construction is a non-trivial problem on its own) or an adapted iterative optimization scheme with projections to a tangent space in every step [16]. In contrast, our method is simpler and leads to similar results for our applications. B-spline fitting, however, is more suited for surface curves: for example, the surface curve becomes smoother if (selected) control points are removed [16]. This smoothing in the sense of generating fair curves is not our goal because we prefer curves that remain close to the initial curve. In summary, we think at the minimization of the geodesic curvature is the right choice for our applications.

9.4. User feedback

We conducted an informal interview with a domain expert to gain qualitative user feedback. The domain expert is actively involved in the reconstruction and decomposition of cerebral aneurysm surfaces as well as the exploration of their hemodynamics based on simulated or measured flow data. The surface decomposition involves several geometric operations, such as cutting the aneurysm sac from the parent vessel. The interview was designed to determine if the requirements defined in Section 3 were principally met. For several input meshes, the expert should draw an initial curve, which roughly defines the aneurysm neck. Afterward, the expert was asked to adjust the allowable region. After our smoothing approach was applied to one curve, the aneurysm surface was cut, and its result was evaluated by the domain expert. To obtain a resulting smooth curve, the participant attempted different parameter settings but was mostly satisfied with the values of $t=0.1$ and $n=20$. The expert assessed the drawing of the initial curve as being very intuitive and fast. The adjustment of the allowable region was rated as a pleasant control function to keep the smoothed curve in its eligible region. The smoothing approach was evaluated as being visually pleasant and reasonable as well as time-saving compared to the current definition of the neck contour in the geometric modeling tools. However, the expert suggested providing an overview gallery, which shows different smoothing results based on different parameter settings. This arrangement would lead to an effective selection of appropriate parameter values,

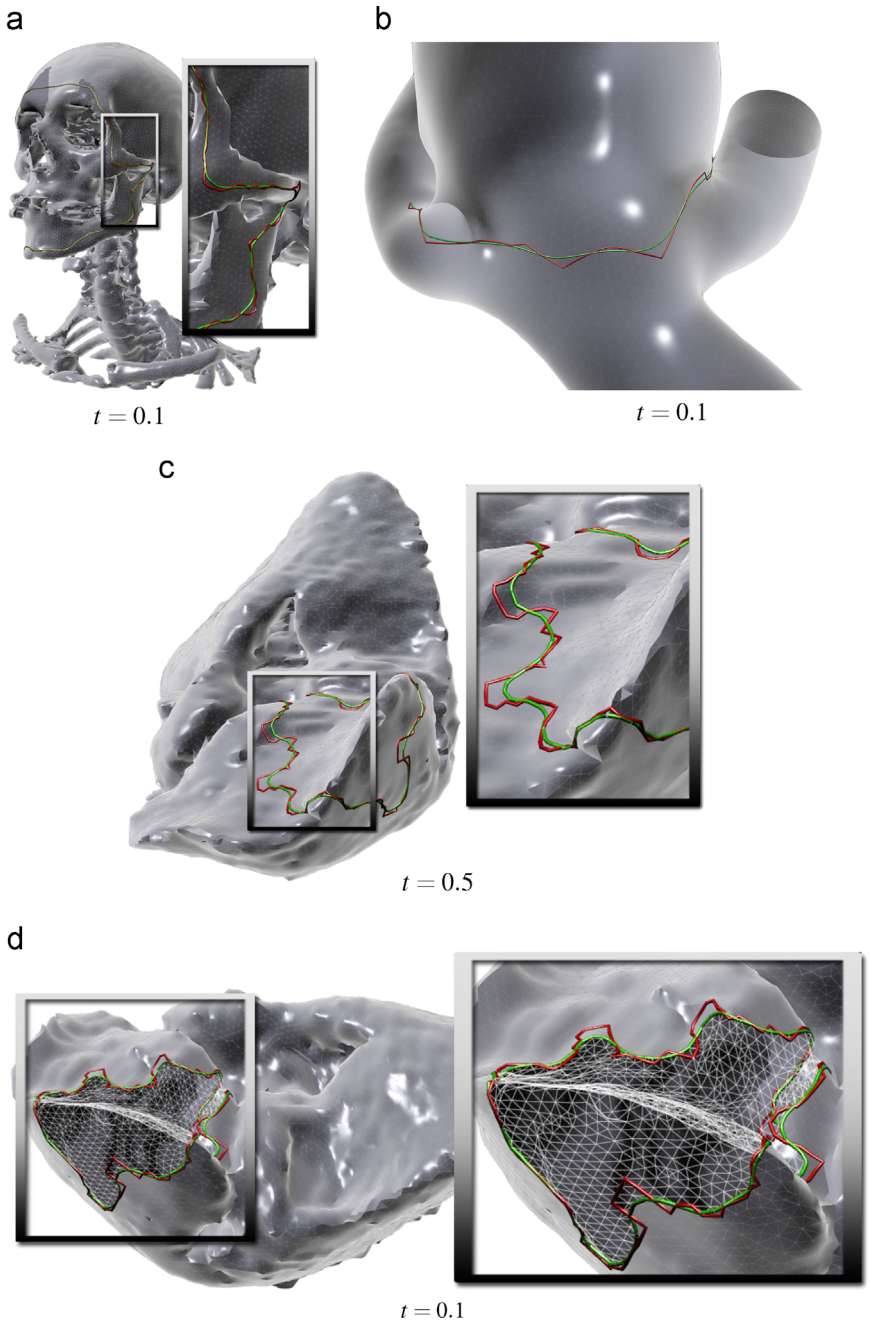

Fig. 10. Application to patient-specific medical surface datasets: initial curves (red) and smoothed curves (green) are shown on a complex bone, an aneurysm, and a liver surface dataset, respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

such as t and n, depending on the current dataset. In summary, the domain expert rated the results positively and gave feedback on improving of the interaction.

10. Conclusions

We presented a novel approach for smoothing surface curves on triangular meshes by reducing the geodesic curvature of the curves. The approach is based on an iterative Laplacian smoothing with a careful construction of the linear operator: we proved that the curve's curvature decreases during runtime, and we used this property as an abort criterion. Depending on one user-defined parameter, the result gradually changes from a smoothed curve close to the original curve to the straightest geodesic curve. The user can adjust the closeness of the smooth curve to its original shape as well as the deviation from the prescribed geodesic curvature. For these adjustments, default parameters are suggested. We tested our algorithm on both synthetic surfaces and anatomical surfaces from clinical datasets to show robustness in terms of geometric and parametric noise. In this way, our approach is not restricted to triangular meshes but is also applicable to different surface representations. Our algorithm fills a gap for the interactive computation of smooth surface curves for cases in which closeness to their initial curve shape is necessary, which was demonstrated for two medical applications. We achieved similar results in comparison with spline approximation methods, but our approach requires less user effort and does not need a

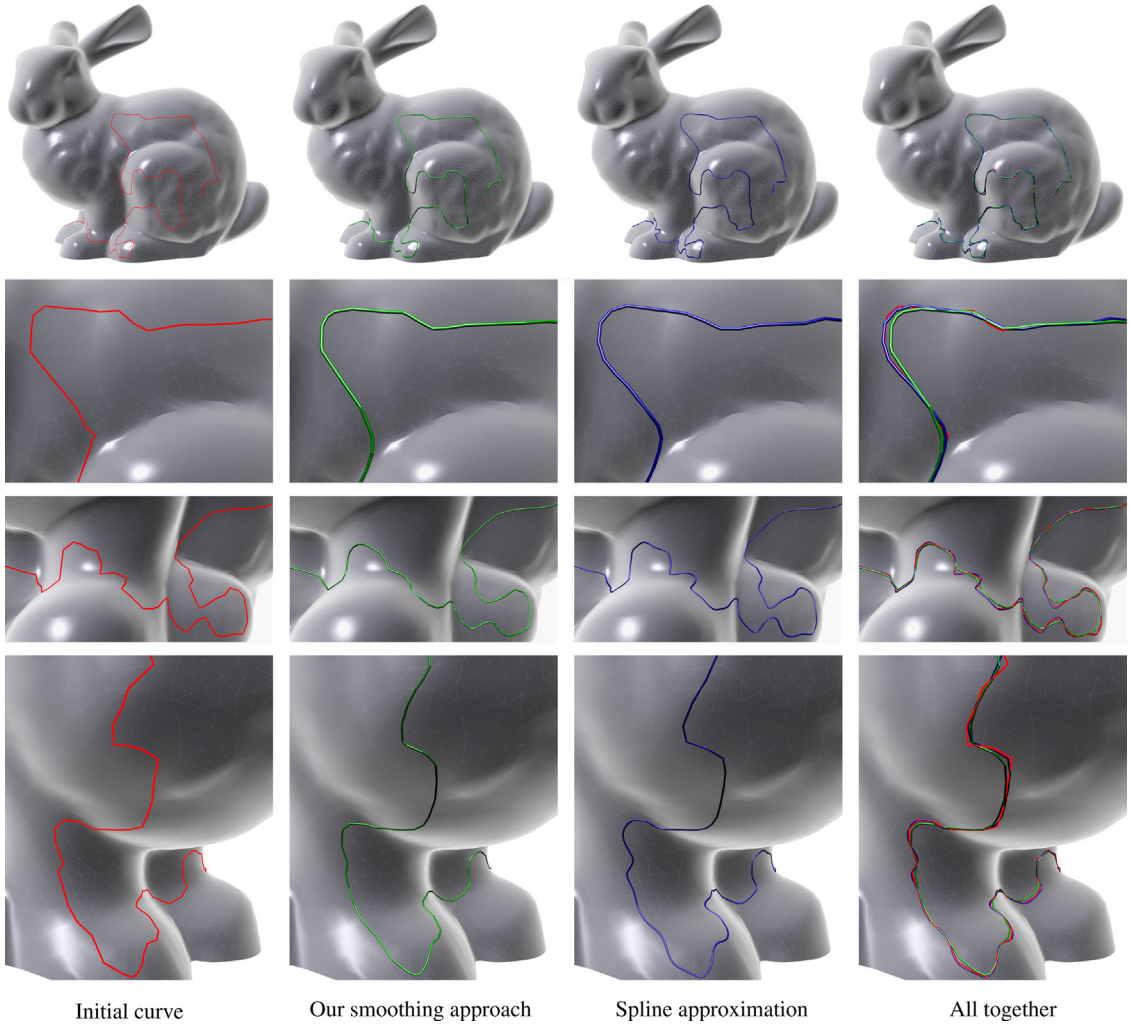

Fig. 11. Comparison of our curve smoothing approach with the spline approximation method applied on a synthetic surface, demonstrated with three enlarged views. Based on the initial curve (red), our approach (green) and the spline approximation (blue) achieve similar results. However, although the spline curve is slightly closer to the initial curve, our approach achieves more global smoothness due to the optimization between the geodesic curvature and the closeness to the initial contour. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

global parameterization. Informal user feedback with a domain expert confirmed the usefulness and robustness of our approach.

References

[1] Kaplansky L, Tal A. Mesh segmentation refinement. Comput Graph Forum 2009;28(7):1995–2003.
[2] Zachow S, Gladilin E, Sader R, Zeilhofer HF. Draw and cut: intuitive 3d osteotomy planning on polygonal bone models. In: CARS; 2003. p. 362–9.
[3] Benhabiles H, Lavoué G, Vandeborre JP, Daoudi M. Learning boundary edges for 3d-mesh segmentation. Comput Graph Forum 2011:2170–82.
[4] Ji Z, Liu L, Chen Z, Wang G. Easy mesh cutting. Comput Graph Forum 2006;25 (3):283–91.
[5] Chaikin G. An algorithm for high speed curve generation. Comput Graph Image Process 1974;3:346–9.
[6] Dyn N, Levin D, Liu D. Interpolatory convexity-preserving subdivision schemes for curves and surfaces. Comput-Aided Des 1992:211–6.
[7] Morera DM, Velho L, Carvalho PC. Subdivision curves on triangular meshes. In: Proceedings of 13th iberoamerican congress on pattern recognition (CIARP); 2008.
[8] Lee Y, Lee S. Geometric snakes for triangular meshes. Comput Graph Forum 2002:229–38.
[9] Lee Y, Lee S, Shamir A, Cohen-Or D, Seidel HP. Intelligent mesh scissoring using 3d snakes. In: Proceedings on pacific graphics; 2004. p. 279–87.
[10] Lai YK, Zhou QY, Hu SM, Wallner J, Pottmann H. Robust feature classification and editing. IEEE TVCG 2007:34–45.
[11] Kass M, Witkin A, Terzopoulos D. Snakes: active contour models. Int J Comput Vis 1988:321–31.
[12] Bischoff S, Weyand T, Kobbelt L. Snakes on triangle meshes. In: Proceedings of Bildverarbeitung für die Medizin; 2005. p. 208–12.
[13] Jung M, Kim H. Snaking across 3d meshes. In: Proceedings on pacific graphics; 2004. p. 87–93.
[14] Martínez D, Velho L, Carvalho PC. Computing geodesics on triangular meshes. Comput Graph 2005;29(5):667–75.
[15] Morera DM, Carvalho PC, Velho L. Geodesic Bezier curves: a tool for modeling on triangulations. In: Brazilian symposium on computer graphics and image processing (SIBGRAPI); 2007. p. 71–8.
[16] Hofer M, Pottmann H. Energy-minimizing splines in manifolds. In: Proceedings on SIGGRAPH; 2004. p. 284–93.
[17] Polthier K, Schmies M. Straightest geodesics on polyhedral surfaces. In: SIGGRAPH courses; 2006. p. 30–8.
[18] Mitchell JSB, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM J Comput 1987;16(4):647–68.
[19] Surazhsky V, Surazhsky T. Fast exact and approximate geodesics on meshes. ACM Trans Graph 2005;24:553–60.
[20] Bommes D, Kobbelt L. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In: Proceedings of VMV; 2007. p. 151–60.
[21] Kimmel R, Sethian JA. Fast marching methods for computing distance maps shortest paths. CPAM Report 669. Berkeley: University of California; 1996.
[22] Crane K, Weischedel C, Wardetzky M. Geodesics in heat. In: Proceedings on SIGGRAPH; 2013.
[23] Dijkstra EW. A note on two problems in connexion with graphs. Numer Math 1959;1:269–71.
[24] Bade R, Haase J, Preim B. Comparison of fundamental mesh smoothing algorithms for medical surface models. In: Simulation und Visualisierung; 2006. p. 289–304.
[25] Cebral JR, Mut F, Weir J, Putman CM. Association of hemodynamic characteristics and cerebral aneurysm rupture. Am J Neuroradiol 2011;32(2):264–70.

[26] Ma L, Chen D. Curve shortening in a Riemannian manifold. Ann Mat Pura Appl 2007;186(4):663–84.
[27] Max N. Weights for computing vertex normals from facet normals. J Graph Tools 1999;4(2):1–6.
[28] Pottmann H, Hofer M. A variational approach to spline curves on surfaces. Comput Aided Geom Des 2005;22(7):693–709.
[29] Lévy B, Petitjean S, Ray N, Maillot J. Least squares conformal maps for automatic texture atlas generation. ACM Trans Graph 2002;21(3):362–71.
[30] Hoschek J, Lasser D. Fundamentals of computer aided geometric design. Natick, MA, USA: AK Peters; 1993.