Computer-Aided Design 140 (2021) 103077

Contents lists available at ScienceDirect

Computer-Aided Design

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

A Variational Framework for Computing Geodesic Paths on Sweep Surfaces

Wenlong Meng [a], Shiqing Xin [a,*], Jinhui Zhao [a], Shuangmin Chen [b], Changhe Tu [a,*], Ying He [c]

[a] School of Computer Science and Technology, Shandong University, China
[b] School of Information and Technology, Qingdao University of Science and Technology, Qingdao, China
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 20 May 2021
Received in revised form 20 June 2021
Accepted 22 June 2021

Keywords:
Computer-aided design
Sweep surface
Variational framework
Geodesic path
Helical curve

ABSTRACT

Sweep is a natural, intuitive, and convenient 3D modeling method in computer-aided design. Sweep surface can be obtained by extruding a 2D cross-sectional profile along a guide curve $x = x(t), t \in [a, b]$. A small segment of the sweep volume can also be understood by rotating a 2D sectorial generatrix curve around the guide curve. We assume that sweep surfaces have a parametric form $\Phi = \Phi(t, \theta)$, where $\Phi([t, t + \mathrm{d}t], \theta)$ defines the sectorial generatrix curve segment at the angle of θ while $r_t(\theta) = \Phi(t, \theta), \theta \in [0, 2\pi]$, defines the circumferential closed curve. Geodesic computation on sweep surfaces is a fundamental geometric operation in many scenarios like the manufacturing process of filament winding. In order to compute a geodesic path between two points on sweep surfaces, we propose a variational framework that works on the 2D parametric domain, without the step of discretizing the surface into a polygonal mesh. The solution to the objective function is a polyline curve of n equally spaced vertices that approximates the real geodesic path, where n is a user-specified parameter for accuracy control. We prove that the polyline approaches the real geodesic in quadratic order. Furthermore, it can be easily extended to compute N-round geodesic helix curves. We also discuss various configurations of $r_t(\theta)$: (1) $r_t(\theta)$ is a constant, independent of t and θ, (2) $r_t(\theta)$ depends on only t, independent of θ, and (3) $r_t(\theta)$ depends on both t and θ. We validate the effectiveness and high performance of our method through extensive experimental results.

1. Introduction

Geodesics, as a tool to measure the distance between two points on a 2-manifold Riemannian surface, encode all the intrinsic properties of the surface, and thus have fundamental importance in digital geometry processing. Due to the nice feature that the geodesic curvature is zero everywhere, geodesics are useful in many special industrial design applications. In this paper, we consider the geodesic problem on sweep surfaces, which is a fundamental geometric operation in the manufacturing process of filament winding.

The distinguished feature of sweep surfaces $\Phi = \Phi(t, \theta)$ lies in that they are given by a pair of parameters and each parameter has clear geometric meaning. Particularly, $t \in [a, b]$ determines a position of the guide curve $x = x(t)$ while $\Phi(t, \theta)|_{\theta=\theta_0}$ gives a generatrix curve when t goes from a to b. It is computationally inefficient and inaccurate if one transforms the parametric form into a polygonal mesh and then extracts a geodesic on the discrete mesh surface. This motivates us to directly compute the geodesic on the parameter domain.

In the past research, the Runge–Kutta method [1–4] is a popular choice for computing geodesic paths on a sweep surface. In its essence, it serves as a numeric method to solve the Euler–Lagrange equations. If given "one point, one direction", it incrementally determines a sequence of points, approximating the real geodesic path. In this case, it may produce a long geodesic helical curve as long as one traces the path one point after the other. If users input two points, the Runge–Kutta method needs to solve a linear system, and thus is not easy to generate an N-round helical curve between two given points, which limits its use in the scenario of filament winding. Besides, it is hard to control the step size to achieve a guaranteed accuracy requirement. Therefore, in this paper, we propose a new variational framework to deal with the difficulties.

For a piecewise smooth curve $\gamma(t), t \in [a, b]$, its length can be measured by

$$L(\gamma(t)) = \int_a^b \sqrt{g_{\gamma(t)}(\dot{\gamma(t)}, \dot{\gamma(t)})}\mathrm{d}t \tag{1}$$

* Corresponding authors.
E-mail addresses: xinshiqing@sdu.edu.cn (S. Xin), chtu@sdu.edu.cn (C. Tu).

https://doi.org/10.1016/j.cad.2021.103077
0010-4485/

where g is the metric tensor defined on the Riemannian surface and becomes identity if considered in the Euclidean space. It is proved that the minima of the following energy functional

$$E(\gamma(t)) = \frac{1}{2}\int_a^b g_{\gamma(t)}(\dot{\gamma(t)}, \dot{\gamma(t)})\mathrm{d}t \tag{2}$$

can also be used to define geodesics [5]. Furthermore, E is independent of curve parameterization, and thus guarantees existence, uniqueness, and regularity of minimizers. Here, by 'uniqueness', we mean that among all the possible parameterization configurations, only the arc-length parameterization (or scaled by a factor) can be the minimizer of Eq. (2). Our approach for computing geodesics on sweep surfaces is based on the interesting property.

In this paper, we first discuss the parametric form of sweep surface from a different perspective. Observing that a small segment of the sweep volume can also be understood by rotating a 2D sectorial generatrix curve around the guide curve $x = x(t)$. We assume that sweep surfaces have a parametric form $\Phi = \Phi(t, \theta)$, where $\Phi([t, t + \mathrm{d}t], \theta)$ defines the sectorial generatrix curve segment at the angle of θ while $r_t(\theta) = \Phi(t, \theta), \theta \in [0, 2\pi]$, defines the circumferential closed curve. The definition enables one to define $x = x(t)$ and $r_t(\theta)$ separately. We give the definition detailedly in Section 3.1. It is worth noting that our definition only applies to a special class of sweep surfaces.

Suppose that the source point and the target point are given. Initially, users can specify an arbitrary polyline as the input. By approximating the unknown geodesic path with $n + 2$ vertices (n intermediate points are variable), we use $\sum_{i=0}^{n} \|y_i - y_{i+1}\|^2$ to define the objective function, which is the same with the above-mentioned energy functional defined in Eq. (2) but operates in the discrete setting. Driven by the objective function, we repeatedly evolve the path until the n points are equally spaced along the path and minimize the total length. Besides, we have to overcome three additional difficulties. First, we consider various configurations of $r_t(\theta)$: (1) $r_t(\theta)$ is a constant, independent of t and θ, (2) $r_t(\theta)$ depends on only t, independent of θ, and (3) $r_t(\theta)$ depends on both t and θ. Second, the objective function must be represented by θ and t so that the optimization can be conducted in the parameter domain. Finally, we need to compute the gradients w.r.t. θ and t, and use them to speed up the optimization. Our variational framework inherits a curve shortening technique. Therefore, it can be naturally extended to support the query of an N-round geodesic helical curve, which is much different from the Runge–Kutta method. It is worth noting that the base surface of our interest is a sweep surface, instead of a general parametric surface, since it is hard to precisely define an N-round spiral curve on a general surface. The by-product of this framework is that the yielding point sequence is equally spaced and able to accurately characterize how the real geodesic path winds along the surface, which is crucial to the motion planning in filament winding.

Our contributions include

1. We propose a variational framework to compute geodesics on sweep surfaces. Different from the Runge–Kutta method, our method inherits the spirit of curve shortening, and is able to report a geodesic path or an N-round geodesic helical curve when users specify the two endpoints (rather than "one point, one direction").
2. We formulate the energy functional in the parameter domain. Furthermore, we deduce the analytic gradients for three situations of the radius function. By feeding the analytic gradients into the L-BFGS solver, our algorithm has a super-linear convergence rate.
3. We establish a relationship between the number of intermediate points and the absolute error, which facilitates direct error control. Particularly, we observe that the absolute error decreases quadratically w.r.t. the number of inserted points.

2. Related works

There is a large body of literature on the geodesic problem. In this paper, we review only the most relevant works on computing geodesic paths. We divide them into two categories, depending on the type of the base surface, i.e., a smooth surface with an analytic form or a discrete polygonal surface.

2.1. Shortest paths on smooth surfaces

Available geodesic approaches on smooth surfaces can be roughly classified into analytical and numerical. Analytical approaches [6] are computationally expensive and only just a few types of surfaces have closed-form geodesic paths, which cannot be extended to a general surface. Therefore, numerical methods are more widely used than the analytical methods.

In general, the Runge–Kutta method has been widely used to find the numeric solution to the nonlinear differential equations of geodesic. Beck et al. [1] found a system of coupled, non-linear-order differential equations to solve the shortest paths of bicubic spline surfaces by using the fourth order Runge–Kutta method. Sneyd et al. [2] described a kind of specific sweep surfaces, named tubular surfaces, and computed geodesic paths on them using a second order Runge–Kutta method. Chen et al. [4] treated the surfaces as the combination of several NURBS patches. They developed an adjustable modeling scheme to compute geodesic paths through the surface combination which can be computed directly by solving the geodesic equations using the Runge–Kutta method. Kasap et al. [7] proposed a finite-difference method which is used for governing non-linear system of differential equations to iteratively solve the geodesic on surfaces. Panou et al. [8] presented the geodesic equations and their numerical method works on a triaxial ellipsoid. Numerically solving the nonlinear differential equations of two-point geodesic is time consuming when users specify a rigorous accuracy tolerance.

Geometric methods are also a typical kind of approaches to compute geodesic on smooth surfaces. They are based on the fundamental property that geodesics are a generalization of straight lines. Based on the previous point p_{i-1} and the marching direction, the next point p_i can be then predicted by simple geometric operations. Geometric methods are independent of the complex description of the surfaces. Hotz et al. [9] introduced a geometric method for the geodesic construction of arbitrary parametric surfaces. Zhang et al. [10] traced a geodesic in a simple way which is independent of the complex description of the geodesic equations. Huang et al. [11] proposed an efficient geometric approach to compute geodesics on B-spline surfaces given a start point and a directional vector. However, these methods cannot be used to infer the geodesic path between two points.

There are some other methods that utilize different techniques to compute geodesics on smooth surfaces. Maekawa [12] proposed a method for solving shortest path of free-form parametric surfaces based on a relaxation method relying on finite difference discretization. Ying et al. [13] discussed how to rapidly construct the whole geodesic flow map which allows computing any geodesic by straightforward local interpolation. Chen [14] presented a method for obtaining a geodesic-like curve that approaches to the real geodesic if the order of the curve reaches infinity. Yu et al. [15] introduced a particle system based approach to compute geodesics on implicit surfaces. Seong et al. [16] computed anisotropic geodesic distances by solving a general anisotropic Hamilton–Jacobi equation.

In this paper, we intend to achieve the following goals at the same time: (1) two-point geodesic instead of "one point one direction", (2) computationally efficient, (3) high convergence rate, and (4) can be used to compute an N-round geodesic helical curve.

2.2. Shortest paths on discrete surfaces

To our knowledge, there are many research works for computing geodesics and geodesic distance fields on discrete polygonal surfaces. Generally speaking, the task of computing a two-point geodesic path is different from computing a geodesic distance field. When users specify two points as the input, a typical technique is to iteratively shorten the path by local perturbation like shrinking a rubber band. Kanai and Suzuki [17] suggested initializing the path by Dijkstra's algorithm and then proposed a selective refinement strategy by subdividing the mesh around the path to repeatedly refine the up-to-date path. Martänez et al. [18] proposed to keep down the intersection points between the path and mesh edges and iteratively straighten a local curved segment determined by three successive points. Xin and Wang [19] gave a fast edge sequence constrained shortest path algorithm and suggested updating the edge sequence if there exists a better path. These approaches are easy to implement, however they have only linear convergence rate, and their performance is highly sensitive to initialization. Both [20] and [21] are based on energy minimizing. Their difference lies in that the former minimizes the total length, while the latter is based on differential evolution (DE).

We also mention some works on computing a distance field, which is theoretically important since it serves as a base for solving the other variants of the geodesic problem. We refer to [22] for a comprehensive survey. Some approaches are based on wavefront propagation. These methods [23–25] inherit Dijkstra's spirit to propagate discrete wavefronts from near to far. Due to the global nature, these methods are able to find the globally shortest path. Some approaches are based on PDE [26,27]. They first formulate the problem in terms of partial differential equations (PDEs) on a smooth manifold, then solve these PDEs in the discrete setting, e.g., finite element methods (FEM) or other numerical techniques. Some approaches take the input polygonal mesh as a graph. These methods [28–30] rely on the assumption when the graph is dense enough, the graph-induced distance is sufficiently close to the real geodesic distance.

In this paper, we focus on computing geodesics on the parametric sweep surfaces, instead of the polygonized counterpart. On one hand, converting a sweep surface into a polygonal mesh is very tedious and inefficient. On the other hand, the surface-to-mesh conversion suffers from accuracy issues.

3. Preliminaries

3.1. Definition of sweep surfaces

Sweep surfaces are generally defined by moving a sectional curve along a guide path, and thus can be represented by $\Theta(t,s) = x(t) + \lambda_1(s,t)f_1 + \lambda_2(s,t)f_2$, where $x = x(t)$ is the guide path and $C(\lambda_1(s,t)f_1, \lambda_2(s,t)f_2)$ gives the shape of the sectional curve at the point $x(t)$. Its geometric explanation is that the sectional curve continuously varies with the base point on the guide path.

In this paper, we understand the sweep surfaces from a different perspective. Let the parametric form be $\Phi(t,\theta)$, where $x = x(t)$ is the guide curve. As Fig. 1 shows, $\Phi(t,\theta)|_{t=t_1}$ determines a closed circumferential curve at $x(t_1)$. The vector $(\Phi(t,\theta)|_{t=t_1} - x(t_1))$ must be perpendicular to the sweep surface:

$$(\Phi(t,\theta) - x(t)) \cdot \frac{\partial \Phi(t,\theta)}{\partial t} = 0. \tag{3}$$

Specially, when the sweep surface degenerates to a conical surface, $\Phi(t,\theta)|_{t=t_1}$ becomes a circle; and when the sweep surface degenerates to a cylindrical surface, our definition is the same as the traditional definition. But in general cases, $\Phi(t,\theta)|_{t=t_1}$ varies with the parameter θ, forming a star shape with the base point $x(t_1)$, which is to guarantee every angle θ produces a unique point on the sweep surface. Note that the radius function $r_{t_1}(\theta) := \|\Phi(t,\theta)|_{t=t_1} - x(t_1)\|$ varies with θ. We have an additional requirement that the sweep surface cannot self-intersect, i.e.,

$$\frac{\partial r(t,\theta)}{\partial t} \le 1, \tag{4}$$

for every t. The purpose of this requirement is to guarantee that each point (t,θ) can contribute a point on the sweep surface; See explanation in [2]. Otherwise, the point given by (t,θ) may be located totally inside the sweep surface. To summarize, rather consider all kinds of sweep surfaces, we concentrate on a special class of sweep surfaces that can be parameterized into the above form.

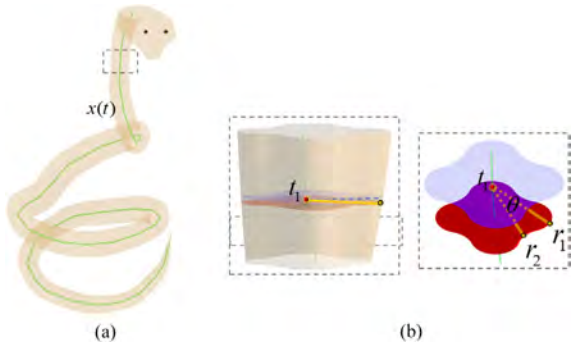


Fig. 1. Visualization of a sweep surface. (a) a sweep surface $\Phi(t,\theta)$ is defined by guide curve $x(t)$ which is shown using green curve. (b) $\Phi(t,\theta)|_{t=t_1}$ determines a closed circumferential curve at $x(t_1)$ and varies with the parameter θ, forming a star shape with the base point $x(t_1)$.

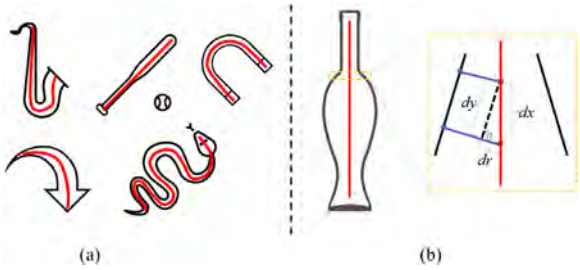


Fig. 2. (a) 2D sweep shapes can be defined by a central guide curve (colored in red) equipped with a radius function. (b) $d^2y = d^2x - d^2r$, where dx is the length of a segment of the guide curve, dr is the corresponding change of radius and dy is the length of the corresponding boundary segment of the sweep shape.

3.2. 2D sweep shapes

2D sweep shapes are much simpler than 3D sweep surfaces but have similar properties. In 2D cases, we do not need the parameter θ any more to define the sweep shapes. Instead, users need to specify the guide curve $x = x(t)$ as well as the radius function $r(t)(> 0)$ that measures the distance between $x(t)$ and the two corresponding boundary points.

We assume that there is a small segment x_1x_2 on the guide curve, where r_1 and r_2 are the radii of points x_1 and x_2, respectively. In the meanwhile, y_1, y_2 are the corresponding points on the swept boundary. As Fig. 2 shows, the differential relation between them is:

$$d^2y = d^2x - d^2r. \tag{5}$$

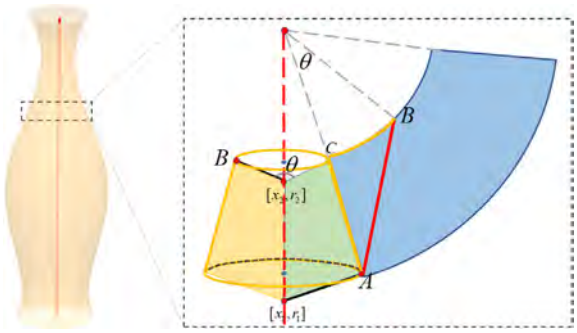


Fig. 3. For an infinitesimal patch whose parameters are in $[t, t+dt]\times[\theta, \theta+d\theta]$, we have $d^2y = (d^2x - d^2r) + (rd\theta)^2$.

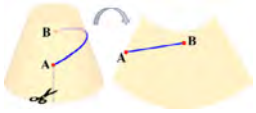
In the discrete setting, $dx(= x'(t)dt)$ becomes $\|x_1 - x_2\|$, dr becomes $|r_1 - r_2|$, and dy becomes $\|y_1 - y_2\|$. Therefore, we have

$$\|y_1 - y_2\|^2 = \|x_1 - x_2\|^2 - |r_1 - r_2|^2. \tag{6}$$

Based on this, a path lying on the sweep boundary can be approximated by a polygonal curve $y_0, y_1, y_2, \ldots, y_n, y_{n+1}$ whose total length is

$$\sum_{i=0}^{n} \sqrt{\|x_{i+1} - x_i\|^2 - |r_{i+1} - r_i|^2}.$$

3.3. 3D sweep surfaces



The 3D cases are more complicated than cases in the 2D situation since they depend on both t and θ to measure the length of a path. Geometrically speaking, an infinitesimal segment of the path can be roughly viewed as a curved segment lying on a cylindrical surface or conical surface, as shown in the inset figure. For this reason, we can study the length problem by unfolding a local part onto a 2D plane. Let $x = x(t)$ be the guide curve. In the following, we shall build the relationship about how the length dy depends on the triple $(dx, dr, d\theta)$, where r depends on t and θ generally.

As Fig. 3 shows, we unfold an infinitesimal patch whose parameters are within $[t, t + dt] \times [\theta, \theta + d\theta]$ onto a plane. The patch infinitesimal, upon unfolded, becomes a sector or simply a rectangle. Let $A = \Phi(t, \theta), B = \Phi(t + dt, \theta + d\theta)$ be the two endpoints of the infinitesimal path. Let C be the point on the surface with the parameters $(t + dt, \theta)$. On the unfolded plane, $\triangle ABC$ becomes a right-angle triangle. The Pythagorean theorem indicates

$$|AB|^2 = |AC|^2 + |BC|^2.$$

BC is approximately an arc and thus has a length of $r \cdot d\theta$, while AC can be inferred from the infinitesimal segment of the guide curve and the infinitesimal change of radius:

$$|AC|^2 = \|x_1 - x_2\|^2 - |r_1 - r_2|^2.$$

Therefore, the relation between $dx, dr, d\theta, dy$ can be characterized by

$$d^2y = (d^2x - d^2r) + (rd\theta)^2. \tag{7}$$

We further consider two problems. On one hand, the discrete form of Eq. (7) is

$$\|y_1 - y_2\|^2 = \|x_1 - x_2\|^2 - |r_1 - r_2|^2 + (\frac{r_1 + r_2}{2}|\theta_1 - \theta_2|)^2. \tag{8}$$

On the other hand, Eq. (7) has three separate situations, depending on how r varies.

Case 1. r is a constant value. In this case, the relation degenerates into

$$d^2y = (x'(t))^2d^2t + (rd\theta)^2. \tag{9}$$

Case 2. r depends on t, but is independent of θ. In this case, we have

$$d^2y = (x'(t))^2d^2t - (\frac{dr}{dt})^2d^2t + (rd\theta)^2. \tag{10}$$

Case 3. r depends on both t and θ. In this case, we have

$$d^2y = (x'(t))^2d^2t - \left(\frac{dr}{d\theta}d\theta + \frac{dr}{dt}dt\right)^2 + (rd\theta)^2. \tag{11}$$

4. Algorithm

4.1. Discrete setting

As mentioned above, we intend to minimize the following energy functional [5]:

$$E(\gamma(t)) = \frac{1}{2}\int_a^b g_\gamma(t)(\gamma(\dot{t}), \gamma(\dot{t}))\, dt.$$

In the following, we discuss the formulation in the discrete setting. Let $y_0, y_1, \ldots, y_n, y_{n+1}$ be a sequence of points on the sweep surface. The first point y_0 and the last point y_{n+1} are fixed while the other points are moveable. In fact, each point y_i has a corresponding parameter pair (t_i, θ_i). On the parameter domain, the $n + 2$ points are connected into a 2D polyline. By mapping the 2D polyline onto the sweep surface, we can get a path totally lying on the surface, which is the real path γ represented by $y_0, y_1, \ldots, y_n, y_{n+1}$. When the points are distributed very densely, the length of γ is roughly equal to that of the 3D polyline $y_0, y_1, \ldots, y_n, y_{n+1}$. Therefore, we have

$$L(\gamma) = \sum_{i=0}^{n} \|y_iy_{i+1}\| + O(h), \tag{12}$$

where h is the maximum gap between y_i. Now we assume that γ is a geodesic connecting y_0 and y_{n+1}. On one hand, $y_1, \ldots, y_n$ have many different positional configurations even if each point y_i is on γ. On the other hand, there may exist a large gap in the point sequence, making it fail to approximate the real geodesic path γ.

Now we introduce a convex function $Q = Q(s)$ satisfies $Q'(s) > 0$ and $Q''(s) > 0$ for $s > 0$. We have

$$\begin{aligned} Q(\frac{L(\gamma)}{n+1}) &= Q(\frac{\sum_{i=0}^{n}\|y_iy_{i+1}\|}{n+1}) + o(\frac{h}{n+1}) \\ &\approx Q(\frac{\sum_{i=0}^{n}\|y_iy_{i+1}\|}{n+1}) \\ &\le \frac{\sum_{i=0}^{n} Q(\|y_iy_{i+1}\|)}{n+1}. \end{aligned} \tag{13}$$

It can be seen that for a sufficiently large n (fixed), when $\sum_{i=0}^{n} Q(\|y_iy_{i+1}\|)$ is minimized, a necessary condition is

$$\|y_0y_1\| = \|y_1y_2\| = \cdots = \|y_ny_{n+1}\|.$$

In the meanwhile, minimizing $\sum_{i=0}^{n} Q(\|y_iy_{i+1}\|)$ makes $L(\gamma)$ also minimized, and thus the polyline $y_0, y_1, \ldots, y_n, y_{n+1}$ is able to approximately represent the real geodesic. Inspired by [31], we take $Q(s) = s^2$ in this paper and consider the following objective function

$$E(y_1, y_2, \ldots, y_n) = \sum_{i=0}^{n} \|y_iy_{i+1}\|^2. \tag{14}$$

When the optimal configuration of $y_1, \ldots, y_n$ is found, we keep the corresponding parameter pair (t_i, θ_i) for y_i, connect $\{(t_i, \theta_i)\}_{i=0}^{n+1}$ into a 2D polyline on the parameter domain, and map the 2D parameter polyline onto the surface.

4.2. Local rotation minimizing frame

There are two ways to define the sweep surface $\Phi(t, \theta)$. The first way is to transform the traditional representation $\Theta(t, s) = x(t) + \lambda_1(s, t)f_1 + \lambda_2(s, t)f_2$ (guide curve + circumferential curve) into $\Phi(t, \theta)$ (guide curve + radius function). The other way is to define the guide curve $x = x(t)$ and $r = r(t, \theta)$ separately. Obviously, the second way is more convenient for designers to generate a sweep surface. However, we need to define the adapted frame $(e_1(t), e_2(t), e_3(t))$ at each point $x = x(t)$.

In general, there are Frenet frame [32,33] and rotation minimizing frame (RMF) [34,35] as common frame forms. Although the Frenet frame can easily be computed, its rotation about the tangent of a general spine curve often leads to undesirable twist in motion design or sweep surface modeling. For the rotation minimizing frame (RMF), the normal plane vectors (e_2, e_3) exhibit no instantaneous rotation about e_1. Suppose that (t_i, θ_i) is the ith point on the path. In order to repeatedly update θ_i to a better configuration, one has to utilize the continuous change of the frame $(e_1(t), e_2(t), e_3(t))$ w.r.t. t. In our work, we use the double reflection method [35], a typical rotation minimizing frame, to compute the RMF sequence at the parameter points $\{(t_i, \theta_i)\}_{i=0}^{n+1}$. In fact, it is easy to infer $e_1(t_i)$ by computing the tangent direction of $x = x(t)$. Let the initial right-handed orthonormal frame be $U_0 = (o_0, v_0, w_0)$ at t_0, the next frame $U_1 = (o_1, v_1, w_1)$ at t_1 for RMF approximation is obtained in the following two steps.

Step 1. Let $\Re_1$ denote the reflection in the bisecting plane of the points t_0 and t_1 (see Fig. 4(a)). We use $\Re_1$ to map U_0 to a left-handed orthonormal frame $U_0^L = (o_0^L, v_0^L, w_0^L)$.

Step 2. Let $\Re_2$ denote the reflection in the bisecting plane of the points $t_1 + w_0^L$ and $t_1 + w_1^L$ (see Fig. 4(b)). We use $\Re_2$ to map U_0^L to a right-handed orthonormal frame $U_1^L = (o_1, v_1, w_1)$.

In this way, the RMF at each t_i can be determined. The whole variable RMF can be taken as piecewisely linear along the guide curve. Suppose that the two points y_i, y_{i+1} are sufficiently close to each other. We can map y_i, y_{i+1} simultaneously to the frame at $\frac{t_i+t_{i+1}}{2}$ and estimate their angle difference.

4.3. Objective function and gradients

By combining Eqs. (8) and (14), the objective function becomes

$$\begin{aligned} E(\{(y_i)\}_{i=1}^n) &= E(\{(t_i, \theta_i)\}_{i=1}^n) \\ &= \sum_{i=0}^{n} \|y_i y_{i+1}\|^2 \\ &= \sum_{i=0}^{n} \|x_i - x_{i+1}\|^2 - |r_{i+1} - r_i|^2 \\ &\quad + \frac{1}{4}|r_i + r_{i+1}|^2|\theta_i - \theta_{i+1}|^2. \end{aligned} \tag{15}$$

We take $\{(t_i, \theta_i)\}_{i=1}^n$ as driven variables, where $x_i = x(t_i), r_i = r(t_i, \theta_i)$. In the following, we discuss the gradients in three situations.

4.3.1. Constant sweep radius

$$\nabla_{t_i} E = 2x'(t_i)(x(t_i) - x(t_{i-1})) + 2x'(t_i)(x(t_i) - x(t_{i+1})). \qquad i = 1, 2, \ldots, n.$$

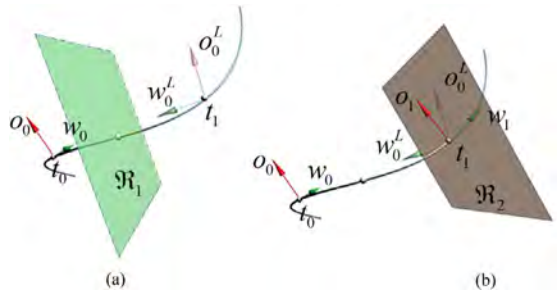


Fig. 4. (a) The first reflection $\Re_1$ of the double reflection method. (b) The second reflection $\Re_2$ of the double reflection method.

and

$$\nabla_{\theta_i} E = 2r^2(\theta_i - \theta_{i-1}) + 2r^2(\theta_i - \theta_{i+1}). \qquad i = 1, 2, \ldots, n.$$

4.3.2. Variable sweep radius $r = r(t)$

$$\nabla_{t_i} E = 2x'(t_i)(x(t_i) - x(t_{i-1})) - 2r'(t_i)(r(t_i) - r(t_{i-1})) + \frac{1}{2}r'(t_i)$$
$$(r(t_{i-1}) + r(t_i))(\theta_i - \theta_{i-1})^2 + 2x'(t_i)(x(t_i) - x(t_{i+1})) - 2r'(t_i)(r(t_i)$$
$$- r(t_{i+1})) + \frac{1}{2}r'(t_i)(r(t_{i+1}) + r(t_i))(\theta_i - \theta_{i+1})^2. \qquad i = 1, 2, \ldots, n.$$

and

$$\nabla_{\theta_i} E = \frac{1}{2}(r(t_{i-1}) + r(t_i))^2(\theta_i - \theta_{i-1}) + \frac{1}{2}(r(t_i) + r(t_{i+1}))^2(\theta_i - \theta_{i+1}). \quad i = 1, 2, \ldots, n.$$

4.3.3. Variable sweep radius $r = r(t, \theta)$

$$\nabla_{t_i} E = 2x'(t_i)(x(t_i) - x(t_{i-1})) - 2\frac{\partial r}{\partial t_i}(r(t_i, \theta_i) - r(t_{i-1}, \theta_{i-1})) + \frac{1}{2}\frac{\partial r}{\partial t_i}$$
$$(r(t_i, \theta_i) + r(t_{i-1}, \theta_{i-1}))(\theta_i - \theta_{i-1})^2 + 2x'(t_i)(x(t_i) - x(t_{i+1})) - 2\frac{\partial r}{\partial t_i}(t_i$$
$$, \theta_i)(r(t_i, \theta_i) - r(t_{i+1}, \theta_{i+1})) + \frac{1}{2}\frac{\partial r}{\partial t_i}(r(t_i, \theta_i) + r(t_{i+1}, \theta_{i+1}))(\theta_i - \theta_{i+1})^2.$$
$$i = 1, 2, \ldots, n,$$

and

$$\nabla_{\theta_i} E = -2\frac{\partial r}{\partial \theta_i}(r(t_i, \theta_i) - r(t_{i-1}, \theta_{i-1})) + \frac{1}{2}\frac{\partial r}{\partial \theta_i}(r(t_i, \theta_i) + r(t_{i-1}, \theta_{i-1}))$$
$$(\theta_i - \theta_{i-1})^2 + \frac{1}{2}(r(t_i, \theta_i) + r(t_{i-1}, \theta_{i-1}))^2(\theta_i - \theta_{i-1}) + 2\frac{\partial r}{\partial \theta_i}(r(t_{i+1}, \theta_{i+1})$$
$$- r(t_i, \theta_i)) + \frac{1}{2}\frac{\partial r}{\partial \theta_i}(r(t_{i+1}, \theta_{i+1}) + r(t_i, \theta_i))(\theta_{i+1} - \theta_i)^2 - \frac{1}{2}(r(t_i, \theta_i) + r$$
$$(t_{i+1}, \theta_{i+1}))^2(\theta_{i+1} - \theta_i). \qquad i = 1, 2, \ldots, n.$$

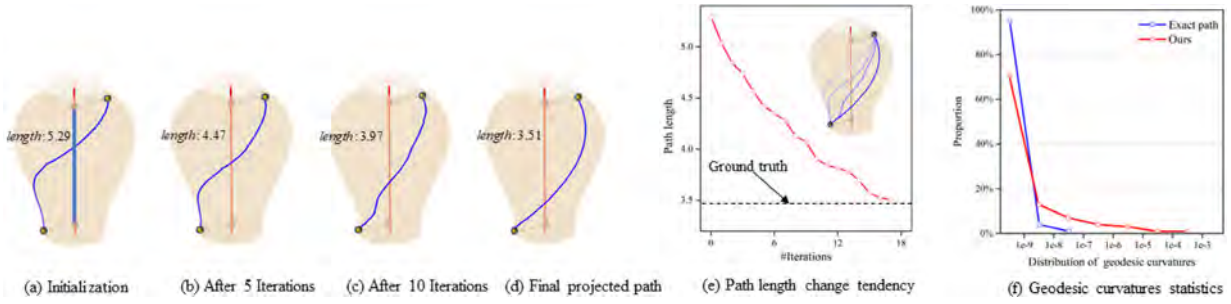


Fig. 5. A typical iterative process. Given an initial path (a), we iteratively minimize the objective function E. After 17 iterations (b–d), E is minimized and the resulting path is a shortest path. As shown in the convergence plot (e), the total length decreases when E decreases. (f) shows the distributions of discrete geodesic curvatures of our path and the exact path (discretized into a polyline.).

4.4. Implementation details

Suppose that the sweep surface $\Phi(t,\theta)$ is defined by a guide curve $x(t)$ and a radius function $r(t,\theta)$. Let y_0 and y_{n+1} be the source point and the target point respectively. With the help of explicit gradients, we find the best configuration of $y_0, y_1, \ldots, y_n$ based on the L-BFGS solver. The algorithm works on the parameter domain. The algorithmic pipeline is shown in Fig. 5.

Path initialization. First, we sample n points on guide curve $x = x(t)$ to generate an initial path γ_0 that passes through y_0 and y_{n+1}, forming a sequence:

$$y_0, y_1, \ldots, y_n, y_{n+1}.$$

We use t to serve as the arc-length parameter of the guide curve, and find the parameter pair (t_i, θ_i) for the point y_i. At the same time, we compute the rotation minimizing frame at y_i. The overall accuracy is controlled by the number of intermediate points, i.e., n. We shall analyze how the error is related to n later.

Refinement process. After that, we refine the point sequence by repeatedly reducing the squared energy defined in Eq. (15). The algorithm terminates when the length difference between two successive iterations is less than an error tolerance ϵ. During each iteration, we feed the objective function E, as well as its gradients $\{\nabla_{t_i} E\}_{i=1}^n$ and $\{\nabla_{\theta_i} E\}_{i=1}^n$ into the L-BFGS solver to produce the next generation of point sequence. It can be seen from Fig. 5 that the path becomes shorter and shorter. For the example shown in Fig. 5, the optimization terminates after 17 iterations. It is worth mentioning that the objective function E is guaranteed to monotonically decrease throughout the optimization, but the length L of the path may not monotonically decrease at the beginning of the optimization. At the end of the optimization, E is minimized and L cannot be shortened any more. We summarize the above discussion into Algorithm 1.

Accuracy validation. In the continuous setting, an exact geodesic path has zero geodesic curvature everywhere. Since the resulting paths are represented by polylines, we can measure the accuracy of the resulting paths by computing discrete geodesic curvatures. Let p_{i-1}, p_i and p_{i+1} be three successive points on the resulting path. Then the discrete geodesic curvature for point p_i is

$$\frac{\overrightarrow{p_{i-1}p_i}}{\|\overrightarrow{p_{i-1}p_i}\|} \times \frac{\overrightarrow{p_ip_{i+1}}}{\|\overrightarrow{p_ip_{i+1}}\|} \cdot \mathbf{n}(p_i),$$

where $\mathbf{n}(p_i)$ is the surface normal at p_i. In the meanwhile, we use the fourth order Runge–Kutta method with an extremely small step to extract the exact geodesic path, and then discretize it into a polyline with the same number of points. Visualization of the distribution of discrete geodesic curvatures shows that our result is highly consistent with the ground-truth, demonstrating the high accuracy of our algorithm (see Fig. 5(f)).

4.5. Error and time complexity analysis

Considering that in the yielded point sequence, all the points are equally spaced, and thus the thickness of gap is reduced to half if the number of points is doubled. Imagine that the real geodesic has a largest curvature κ. We will analyze how the error reduced w.r.t. n.

Theorem 4.1. *The error decreases quadratically w.r.t. n.*

Proof. Without loss of generality, we assume that the real geodesic has a curvature κ everywhere and the total length is L. The curvature radius R is thus $1/\kappa$. Let the gap between y_i and y_{i+1} be h when the number of inserted points amounts to n. First, due to the property that the points are equally spaced, the curved segment between y_i and y_{i+1} has a length $\frac{L}{n+1}$. The angle between y_i's normal vector and y_{i+1}'s normal vector is $\frac{L}{R(n+1)}$. Therefore, we have

$$h = 2R \sin \frac{L}{2R(n+1)}.$$

Therefore, the error between the real geodesic and the polyline should be

$$\tau = |L - 2R(n+1) \sin \frac{L}{2R(n+1)}|.$$

Algorithm 1 Computing Geodesic Paths on Sweep Surfaces Using Variational Method

Input: A sweep surface $\Phi(t,\theta)$ defined by guide curve $x(t)$ and radius function $r(t,\theta)$; two points $y_0, y_{n+1} \in \Phi(t,\theta)$; An error tolerance ϵ.

Output: A geodesic path on the sweep surface to connect the user-specified endpoints y_0 and y_{n+1}.

1: Extract a set of two-tuples $\{(t_i,\theta_i)\}_{i=0}^{n+1}$ for the initial point sequence;
2: Infer the RMF based on the current $\{t_i\}_{i=0}^{n+1}$. // See more details in Section 4.2
3: $j := 0$;
4: **while** the length difference between two successive iterations is larger than ϵ **do**
5: Compute the objective function E as well as its gradients with t_i and θ_i;
6: Perform one iteration of L-BFGS, and take the resulting point sequence as the initialization for the next iteration;
7: $j := j + 1$;
8: **end while**
9: Transform the sequence $\{(t_i,\theta_i)\}_{i=0}^{n+1}$ onto the sweep surface.

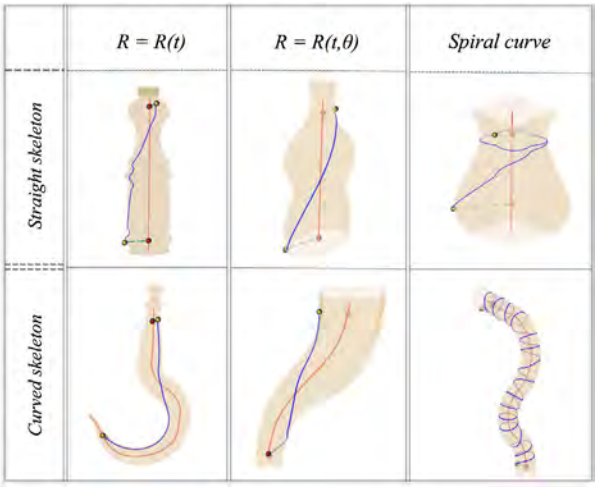


Fig. 6. Computing shortest paths on various sweep surfaces.

According to Taylor's expansion, we have

$$\tau \approx \frac{L^3}{24R^2(n+1)^2},$$

which shows that the error decreases quadratically w.r.t. n.

As far as the time complexity is concerned, our optimization based algorithm depends on the number of iterations and the timing cost spent in each iteration. Obviously, during each iteration, the required timing cost is linear to the number of intermediate points, n. Therefore, the overall timing cost can be estimated by $O(nI_{max})$, where I_{max} is the maximum number of iterations.

4.6. Path uniqueness

The uniqueness of geodesic path depends on the geometry. Roughly speaking, the geodesic paths connecting two points are not unique — different N's produce different N-round geodesic paths. But it seems more interesting to discuss the path uniqueness in each homotopic equivalence class. Image that we have a sweep surface that consists of only saddle points (negative Gaussian curvature). Two curves are said to be homotopically equivalent if one can be continuously deformed to the other (constrained on the surface). Given two points on such a surface, the geodesic (connecting the two points) is unique in every homotopic equivalence class. An extended observation is that if the input is a straight or curved circular cylinder, the uniqueness approximately holds in each homotopic equivalence class. (The observation is not rigorous since a curved cylinder shape may contain non-saddle points.)

5. Evaluation

We implemented our algorithm in Microsoft Visual C++ 2015, without using any additional numeric packages. All the experiments were conducted on a computer with Intel(R) Core(TM) i7-9700K CPU 3.60 GHz and 8 GB memory.

We list a set of experimental results to show that our algorithm can compute shortest paths on sweep surfaces with various guide curves; See Fig. 6. We shall evaluate the proposed algorithm in different indicators in next subsections.

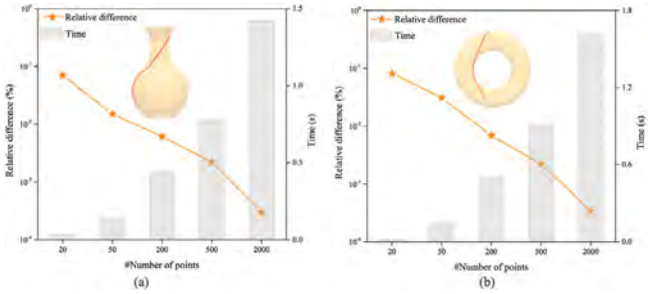


Fig. 7. Plot on how the timing cost and accuracy depend on the number of inserted points. We can see that the more inserted points, the higher accuracy we can get. In the meanwhile, the timing cost slowly grows as the number of inserted points increases.

5.1. Error control

As mentioned above, our method can control the accuracy using the number of inserted points. Users can set the number of intermediate points to obtain various levels of accuracy. We measure the relative difference $|d(s,t) - \widetilde{d}(s,t)|/\widetilde{d}(s,t)$, where $d(s,t)$ and $\widetilde{d}(s,t)$ are the approximate and exact distances respectively. Generally speaking, the exact geodesics on surfaces are not readily available, so we use the fourth order Runge–Kutta method with extremely small step size to extract the geodesic path as the ground truth. As Fig. 7 shows, the result becomes more accurate if there n is larger. For example, when n is set to be 200, the relative error of path is about 0.61% and the timing cost is only 0.39 s.

In fact, there is an alternative way to roughly predict the exact distance L using the conclusion of Theorem 4.1. Let L_1 be the approximate distance computed by setting n inserted points, while L_2 be the approximate distance computed by setting $2n$ inserted points. We have

$$\frac{L - L_1}{L - L_2} = 4,$$

from which we can quickly predict L.

5.2. Run-time performance

The length optimization (LO) algorithm [20] takes the total length as the objective function. Our function adopts a different objective function and has better convergence rate. We visualize how the path length and gradient norm decrease in Fig. 8. It can be clearly seen that our algorithm has a higher convergence rate than [20]. For example, our algorithm terminates after 15 iterations when the length difference between two successive iterations is less than $1e^{-3}$. But LO requires over 60 iterations to achieve the same level of accuracy.

In addition, higher accuracy requires more computational cost. One of the advantage of our algorithm lies in that it is able to achieve higher accuracy at a relatively small cost. We give statistics on how the timing cost depends on n in Fig. 7 and Table 1. For example, when we set n to be 50, the timing cost is about 0.13s. When n increases to 500, the timing cost is about 0.62s.

5.3. Quantitative comparison

Our algorithm distinguishes itself from the existing five relevant methods.

Runge–Kutta method on tubular surfaces [2]**.** It is a compact representation of tubular surfaces by specifying a centerline curve $x(t)$ and a radius function $r(\theta)$. Therefore, computing a geodesic

Table 1
Detailed performance statistics. We compare four typical methods that are most relevant to our method. The number of inserted points, n, is set to be 50, 500, 1000 respectively.

Methods	Surfaces			
	Performance			
Ours (n = 50)	Time (s)	**0.129**	**0.206**	**0.219**
	Relative Diff (%)	1.420	1.513	1.483
Ours (n = 500)	Time (s)	0.615	0.682	0.704
	Relative Diff (%)	0.170	0.226	0.218
Ours (n = 1000)	Time (s)	0.971	1.168	1.294
	Relative Diff (%)	**0.042**	**0.049**	**0.043**
[14]	Time (s)	0.357	0.419	0.473
	Relative Diff (%)	1.810	2.107	2.411
[10]	Time (s)	0.660	0.715	0.794
	Relative Diff (%)	0.481	0.549	0.538
[19]	Time (s)	$2.07_{(t_p)}$ + $0.362_{(t_s)}$	$2.69_{(t_p)}$ + $0.741_{(t_s)}$	$3.67_{(t_p)}$ + $0.623_{(t_s)}$
	Relative Diff (%)	0.082	0.113	0.194
[20]	Time (s)	$2.07_{(t_p)}$ + $1.960_{(t_s)}$	$2.69_{(t_p)}$ + $2.713_{(t_s)}$	$3.67_{(t_p)}$ + $2.451_{(t_s)}$
	Relative Diff (%)	0.129	0.181	0.220

[1] t_p is computing time for surface meshing and t_s is time of computing geodesic path.

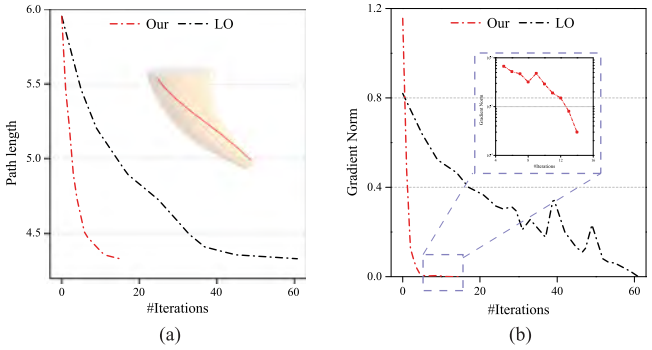


Fig. 8. The plot on how the path length and its gradient norm decrease with the number of iterations.

by utilizing the (t, θ)-domain is a proper choice. Traditionally, one can use the second-order Runge–Kutta method [2] to find a geodesic path. The differences between our approach and the Runge–Kutta method are mainly two-fold. On one hand, our algorithm runs faster and the required timing cost cannot climb sharply with the increasing of the resolution of the path (i.e., how many intermediate points are used to represent the geodesic path). On the other hand, our algorithm is more flexible and naturally supports the computation of N-round geodesic spirals.

Geodesic tracing on parametric surfaces [10]**.** As discussed in Section 4.4, one can trace a geodesic path by enforcing a zero geodesic curvature. Particularly, given a starting point as well as a starting direction, it is very easy to predict the points (in an appropriate gap) one by one. However, the tracing approach is hard to deal with the situation of two-point geodesics. In addition, it cannot report an N-round geodesic as the output.

Length minimizing methods [14,20]. In mathematics, a geodesic path can be defined as a path with zero geodesic curvature everywhere, e.g., an equatorial curve on the spherical surface. However, an equatorial curve is not stable and can be shrunk to a much different configuration upon a slight disturbance. In practical applications, those stable geodesics (with a length minimum) are more useful. Therefore, there are many approaches that optimize the total length of the (polygonal) path until the length remains unchanged. Our algorithm is much different from the existing length minimizing methods. First, our algorithm has a higher convergence rate as our experimental results show. Second, the existing length minimizing methods cannot guarantee the uniformity of the intermediate points and thus easily shortcut the "sinkhole" area.

Curve shortening methods on polygonal meshes [19]**.** There are some research works that repeatedly shorten the initial curve until the path length cannot be shortened any more. When the input is a polygonal mesh, they alternatively perform the two kinds of operations: (1) compute a shortest path restricted in a face sequence, and (2) update the face sequence to a different configuration to see if it can lead to a shorter path. Obviously, it only applies to a polygonal mesh and cannot directly deal with a parameterized sweep surface.

6. Applications

This section presents two interesting applications of the proposed variational framework, including geodesic helix for filament winding and path evolution in animation simulation.

6.1. Geodesic helix for filament winding

In material science, filament winding is an important fabrication technique mainly used for manufacturing open (cylinders) or closed end structures (pressure vessels or tanks) [36,37]. This process involves winding filaments under tension over a rotating mandrel. See inset figure for illustration. It is interesting to see if our algorithm can be extended to solve geodesic helical curve for filament winding.

Definition of geodesic helix. In fact, geodesic helical curves can be viewed as an extension of the helix curve on cylinder surface. On a cylinder surface, the circular helix H has a parametric equation:

$$x = r\cos\frac{t}{\sqrt{r^2+s^2}}, y = r\sin\frac{t}{\sqrt{r^2+s^2}}, z = \pm\frac{st}{\sqrt{r^2+s^2}}, \quad (16)$$

where r is the radius, $2s\pi$ is the pitch, and t is the arc-length parameter. If the outer normal of H is consistent to the cylinder surface for all $t > 0$, it implies that H is a geodesic that is locally shortest everywhere. In fact, accurately defining the N-round geodesic helix is not easy since one has to set up the frames first.

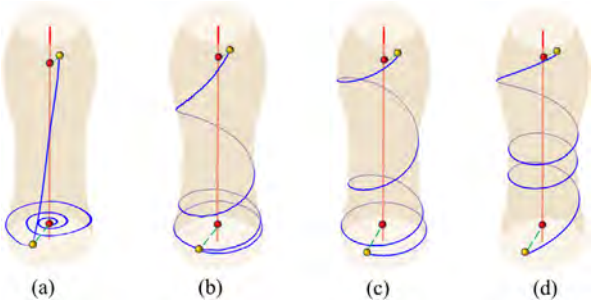


Fig. 9. Geodesic helical curves. (a) Users input an N-round initial curve. (b–c) The 4-th iteration and the 8-th iteration. After 19 iterations, we get the final geodesic spiral curve shown in (d).

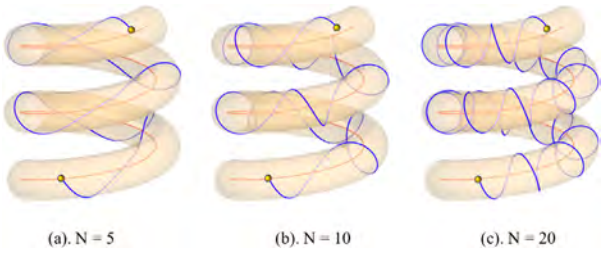


Fig. 10. We compute geodesic helical curves initialized by N circles. The circle numbers are 5, 10, 20 respectively.

In this paper, we solve the problem based on rotation minimizing frames (RMF) so that the angle parameter θ is comparable for different points. As Fig. 9 shows, users can input an N-round initial curve by referring to the RMF at the source point. Then after 19 iterations, our algorithm can report the final geodesic spiral curve.

Comparison with existing helix algorithms. To our knowledge, Xin et al. [38] extended circular helical curves to general tubular shapes and proposed a method for measuring their length and girth. Liu et al. [20] developed an optimization driven approach on discrete surfaces to compute helical curves. Both of them work on a polygonal mesh. For the sweep surface shown in Fig. 10, they need 3.67 s to discretize the parametric sweep surface to a triangle mesh with 273K faces. The former runs in 6.25 s for a 5-round geodesic helical curve shown in Fig. 10, while the latter requires 3.74 s. By contrast, our algorithm takes only 0.70 s to compute geodesic helical curve, without any pre-computing, which validates the advantages of our algorithm in terms of performance.

Filament winding. The filament winding process lays down fibers in alternating positive and negative orientations of the mandrel. In order to guarantee the design mechanical properties of products, it is necessary to keep the fibers distributing evenly on the mandrel. We further simulate the filament winding process using our algorithm. Given the winding parameter N, our method can easily generate various winding results distributing evenly on the specified mandrel. See filament results from sparse to dense in Fig. 11.

6.2. Path evolution

In the field of animation simulation, there are some scenarios where a 3D shape is progressively deformed by some non-rigid transformation techniques [39,40]. In this process, the change of geodesic has spatial coherence. Based on this observation, we take the deformation process as a sequence of geometrically similar shapes. We use the geodesic path in the previous frame to initialize the geodesic in the next frame. Due to the spatial coherence, our algorithm needs an inconspicuous timing cost to accomplish the geodesic update.

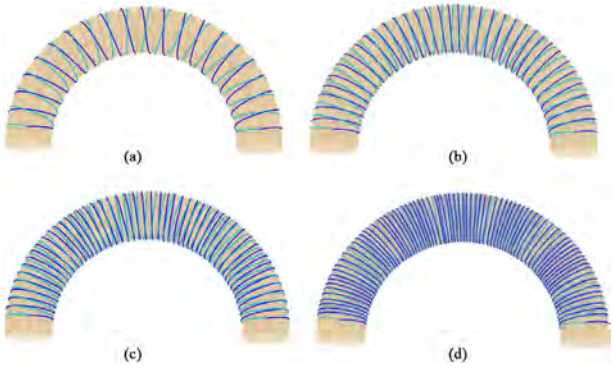


Fig. 11. Filament winding results using our algorithm. From (a) to (d), our algorithm produces four results with various winding density configurations. The timing costs are respectively 0.82 s, 1.37 s, 2.10 s and 2.85 s.

Take Fig. 12 as an example where there is a sequence of variable tubular shapes. Given two endpoints on a tubular surface, our method requires about 100 ms to compute the geodesic path (a polyline with 50 vertices) for the first frame. But after that, each frame costs only about 10 ms to finish the update of the geodesic path, which is due to the spatial coherence.

7. Conclusion

We develop a method for computing geodesic paths on sweep surfaces. Unlike the existing discrete geodesic algorithms that apply to polygonal meshes, we formulate a variational problem by taking advantage of the parametric representation of sweep surfaces. To solve the optimization efficiently, we represent the geodesic path by a polyline with n vertices, where n is a user-specified parameter for accuracy control. Instead of using total length, our energy functional is the sum of squared lengths of the polyline. Our algorithm is guaranteed to converge, and has a super-linear convergence rate. We show that the resulting polyline has least total length and the n vertices are equally spaced along the path. As a result, the limit of the polyline as n approaches infinity equals the real geodesic curve. We demonstrate the effectiveness and high performance of our method through extensive evaluations. We also show that our method can be adapted easily to compute geodesic spiral curves.

Our current implementation does not support forky guide curve due to the topology challenges. We will tackle the challenge in the near future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by National Natural Science Foundation of China (61772016, 62002190, 61772318, 62072284), NSF of Shandong Province, China (ZR2020MF036), and Singapore MOE grants (RG20/20 and T2EP20220-0014).

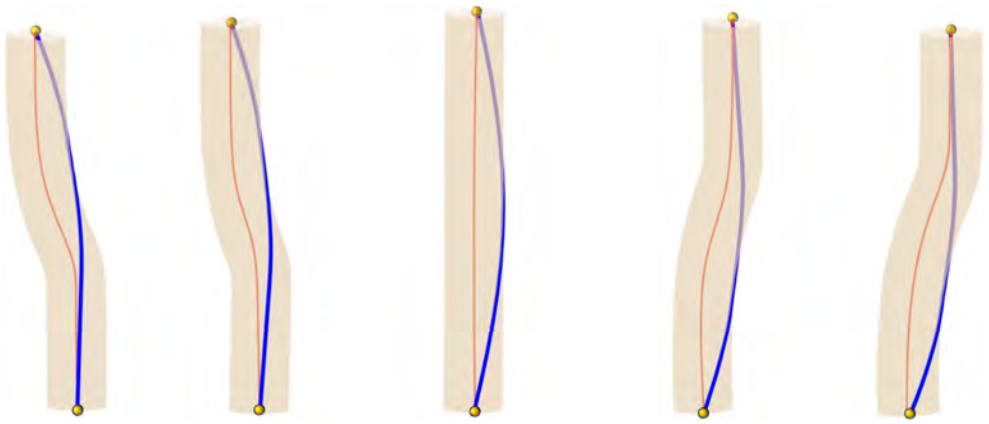


Fig. 12. Real-time path update on a dynamic sweep surface. We use 50 intermediate points to encode a geodesic path. Thanks to the coherence, it costs only about 10 ms to update the geodesic for each frame.

References

[1] Beck JM, Farouki RT, Hinds JK. Surface analysis methods. IEEE Comput Graph Appl 1986;6(12):18–36.
[2] Sneyd J, Peskin CS. Computation of geodesic trajectories on tubular surfaces. SIAM J Sci Stat Comput 1990;11(2):230–41.
[3] Del Buono N, Lopez L. Runge-kutta type methods based on geodesics for systems of ODEs on the Stiefel manifold. BIT Numer Math 2001;41(5):912–23.
[4] Chen X, He S-Y, Yu D-F, Yin H-C, Hu W-D, Zhu G-Q. Geodesic computation on NURBS surfaces for UTD analysis. IEEE Antennas Wirel Propag Lett 2013;12:194–7.
[5] Busemann H. The geometry of geodesics. Courier Corporation; 2012.
[6] Do C. MP: Differential geometry of curves and surfaces prentice hall. 1976, Englewood Cliffs, NJ.
[7] Kasap E, Yapici M, Akyildiz FT. A numerical study for computation of geodesic curves. Appl Math Comput 2005;171(2):1206–13.
[8] Panou G, Korakitis R. Geodesic equations and their numerical solution in cartesian coordinates on a triaxial ellipsoid. J Geodetic Sci 2019;9(1):1–12.
[9] Hotz I, Hagen H. Visualizing geodesics. In: Proceedings visualization 2000. vis 2000 (cat. no. 00CH37145). IEEE; 2000, p. 311–8.
[10] Zhang P, Sun R, Huang T. A geometric method for computation of geodesic on parametric surfaces. Comput Aided Geom Design 2015;38:24–37.
[11] Zhang P, Xiong J, Huang L. An efficient method for computation of geodesic on B-spline surfaces. In: 2019 International conference on advances in construction machinery and vehicle engineering (ICACMVE). IEEE; 2019, p. 311–6.
[12] Maekawa T. Computation of shortest paths on free-form parametric surfaces. J Mech Des 1996;118(4):499–508.
[13] Ying L, Candes EJ. Fast geodesics computation with the phase flow method. J Comput Phys 2006;220(1):6–18.
[14] Chen S-G. Geodesic-like curves on parametric surfaces. Comput Aided Geom Design 2010;27(1):106–17.
[15] Yu H, Zhang JJ. Geodesic computation on implicit surfaces. Int J Inform Sci Comput Math 2010;2(1):33–49.
[16] Seong J-K, Jeong W-K, Cohen E. Curvature-based anisotropic geodesic distance computation for parametric and implicit surfaces. Vis Comput 2009;25(8):743–55.
[17] Kanai T, Suzuki H. Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications. In: Proceedings geometric modeling and processing 2000. Theory and applications. IEEE; 2000, p. 241–50.
[18] Martínez D, Velho L, Carvalho PC. Computing geodesics on triangular meshes. Comput Grap 2005;29(5):667–75.
[19] Xin S-Q, Wang G-J. Efficiently determining a locally exact shortest path on polyhedral surfaces. Comput Aided Des 2007;39(12):1081–90.
[20] Liu B, Chen S, Xin S-Q, He Y, Liu Z, Zhao J. An optimization-driven approach for computing geodesic paths on triangle meshes. Comput Aided Des 2017;90:105–12.
[21] Ye Z, Liu Y-J, Zheng J, Hormann K, He Y. DE-Path: A differential-evolution-based method for computing energy-minimizing paths on surfaces. Comput Aided Des 2019;114:73–81.
[22] Bose P, Maheshwari A, Shu C, Wuhrer S. A survey of geodesic paths on 3D surfaces. Comput Geometry 2011;44(9):486–98.
[23] Mitchell JS, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM J Comput 1987;16(4):647–68.
[24] Chen J, Han Y. Shortest paths on a polyhedron. In: Proceedings of the sixth annual symposium on computational geometry. ACM; 1990, p. 360–9.
[25] Xin S-Q, Wang G-J. Improving chen and han's algorithm on the discrete geodesic problem. ACM Trans Graph 2009;28(4):104.
[26] Crane K, Weischedel C, Wardetzky M. Geodesics in heat: A new approach to computing distance based on heat flow. ACM Trans Graph 2013;32(5):152.
[27] Solomon J, Rustamov R, Guibas L, Butscher A. Earth mover's distances on discrete surfaces. ACM Trans Graph 2014;33(4):1–12.
[28] Aleksandrov L, Lanthier M, Maheshwari A, Sack J-R. An ε—Approximation algorithm for weighted shortest paths on polyhedral surfaces. In: Scandinavian workshop on algorithm theory. Springer; 1998, p. 11–22.
[29] Xin S-Q, Ying X, He Y. Constant-time all-pairs geodesic distance query on triangle meshes. In: Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games; 2012. p. 31–8.
[30] Ying X, Wang X, He Y. Saddle vertex graph (SVG) a novel solution to the discrete geodesic problem. ACM Trans Graph 2013;32(6):1–12.
[31] Godinho L, Natário J. An introduction to riemannian geometry. In: With Applications. Springer; 2012.
[32] Guggenheimer H. Computing frames along a trajectory. Comput Aided Geom Design 1989;6(1):77–8.
[33] Siltanen P, Woodward C. Normal orientation methods for 3D offset curves, sweep surfaces and skinning. In: Computer Graphics Forum. 11, (3):Wiley Online Library; 1992, p. 449–57.
[34] Klok F. Two moving coordinate frames for sweeping along a 3D trajectory. Comput Aided Geom Design 1986;3(3):217–29.
[35] Wang W, Jüttler B, Zheng D, Liu Y. Computation of rotation minimizing frames. ACM Trans Graph 2008;27(1):1–18.
[36] Haghani R, Yang J. Application of FRP materials for construction of culvert road bridges: manufacturing and life-cycle cost analysis. Rapport 2016. 2016: 3.
[37] Aldoumani N, Giannetti C, Abdallah Z, Belblidia F, Khodaparast HH, Friswell MI, Sienz J. Optimisation of the filament winding approach using a newly developed in-house uncertainty model. Eng 2020;1(2):122–36.
[38] Xin S-Q, Chen S, Zhao J, Pan Z. Measuring length and girth of a tubular shape by quasi-helixes. Comput Grap 2014;38:392–8.
[39] Wang Y, Jacobson A, Barbič J, Kavan L. Linear subspace design for real-time shape deformation. ACM Trans Graph 2015;34(4):1–11.
[40] Von-Tycowicz C, Schulz C, Seidel H-P, Hildebrandt K. Real-time nonlinear shape interpolation. ACM Trans Graph 2015;34(3):1–10.