A Variational Framework for Curve Shortening in Various Geometric Domains

Na Yuan, Peihui Wang, Wenlong Meng, Shuangmin Chen, Jian Xu, Shiqing Xin, Ying He, and Wenping Wang, *Fellow, IEEE*

Abstract—Geodesics measure the shortest distance (either locally or globally) between two points on a curved surface and serve as a fundamental tool in digital geometry processing. Suppose that we have a parameterized path $\gamma(t) = \mathbf{x}(u(t), v(t))$ on a surface $\mathbf{x} = \mathbf{x}(u, v)$ with $\gamma(0) = p$ and $\gamma(1) = q$. We formulate the two-point geodesic problem into a minimization problem $\int_0^1 H(\|\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)\|)\mathrm{d}t$, where $H(s)$ satisfies $H(0) = 0, H'(s) > 0$ and $H''(s) \geq 0$ for $s > 0$. In our implementation, we choose $H(s) = e^{s^2} - 1$ and show that it has several unique advantages over other choices such as $H(s) = s^2$ and $H(s) = s$. It is also a minimizer of the traditional geodesic length variational and able to guarantee the uniqueness and regularity in terms of curve parameterization. In the discrete setting, we construct the initial path by a sequence of moveable points $\{x_i\}_{i=1}^n$ and minimize $\sum_{i=1}^n H(\|x_i - x_{i+1}\|)$. The resulting points are evenly spaced along the path. It's obvious that our algorithm can deal with parametric surfaces. Considering that meshes, point clouds and implicit surfaces can be transformed into a signed distance function (SDF), we also discuss its implementation on a general SDF. Finally, we show that our method can be extended to solve a general least-cost path problem. We validate the proposed algorithm in terms of accuracy, performance and scalability, and demonstrate the advantages by extensive comparisons.

Index Terms—Variational method, shortest paths, geodesics, least-cost paths

1 Introduction

Computing discrete geodesics [12] is a fundamental task in computational geometry and has diverse applications ranging from computer graphics [48], [49] to computer vision [36]. Given a curved surface in 3D, a shortest path (locally or globally) between two given points can be computed by minimizing the total length using calculus of variations. When the input is a polygonal mesh, the shortest path problem is also known as the *discrete geodesic problem* [35].

In this paper, we focus on the "single-source-single-destination" shortest path problem. Let p and q be two fixed points on a 3D surface $\mathcal{S}$ and γ be an initial path connecting them. In the usual way, one solves the two-point shortest path problem by repeatedly shortening γ until its total length remains unchanged. Most of the existing geodesic algorithms [27], [32], [34], [55], [59] support only mesh surfaces as the geometric domain and operate on a certain edge/face sequence, causing a very restricted use, which motivates us to decouple computation of geodesics from geometric domains so that users don't need to care about the surface representation type.

In fact, the two-point geodesic problem can be understood from a variational perspective. Given a parametric surface $\mathbf{x} = \mathbf{x}(u, v)$ as well as a family of curves $\mathbf{x}(u(t), v(t))$ with $p = \mathbf{x}(u(0), v(0))$ and $q = \mathbf{x}(u(1), v(1))$, the geodesic between p and q can be found by minimizing $\int_0^1 |\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)|\mathrm{d}t$. There is an interesting observation [21] that $\int_0^1 |\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)|^2\mathrm{d}t$, or even $\int_0^1 H(|\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)|)\mathrm{d}t$, serves as an alternative variational form for length minimization, where the kernel function $H = H(s)$ satisfies $H(0) = 0, H'(s) > 0$ and $H''(s) \geq 0$ for $s > 0$. The different variational framework naturally leads to existence and regularity (in terms of curve parameterization), and owns nicer properties than the original variational formulation.

In this paper, our approach is based on the above observation. In implementation, we approximate the real geodesic by a sequence of points $\{x_i\}_{i=1}^n$ that are constrained to move on the surface and minimize $\sum H(\|x_i - x_{i+1}\|)$ instead of the total length $\sum \|x_i - x_{i+1}\|$. The point sequence yielded by our approach is evenly spaced along the path

• Na Yuan, Wenlong Meng, and Shiqing Xin are with the School of Computer Science and Technology, Shandong University, Ji Nan Shi, Shan Dong Sheng 250100, China. E-mail: {yuanna_sdu, longwuya}@163.com, xinshiqing@sdu.edu.cn.
• Peihui Wang is with ArcSoft, Hangzhou, Zhejiang 310012, China. E-mail: peihui_wang@163.com.
• Shuangmin Chen is with the School of Information and Technology, Qingdao University of Science and Technology, Qingdao, Shandong 266101, China. E-mail: csmqq@163.com.
• Jian Xu is with the Ningbo Institute of Materials Technology and Engineering, Chinese Academy of Sciences, Beijing 100045, China. E-mail: xujian@nimte.ac.cn.
• Ying He is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. E-mail: yhe@ntu.edu.sg.
• Wenping Wang is with the Department of Computer Science, The University of Hong Kong, Hong Kong. E-mail: wenping@cs.hku.hk.

Manuscript received 4 February 2021; revised 30 November 2021; accepted 8 December 2021. Date of publication 14 December 2021; date of current version 28 February 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2021YFB1715900, in part by the National Natural Science Foundation of China under Grants 62002190, 62272277, 52075526, and 91860204, in part by the NSF of Shandong Province under Grant ZR2020MF036, in part by the Singapore Ministry of Education under Grants RG20/20 and T2EP20220-0014, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA21010205.
(Corresponding author: Shiqing Xin.)
Recommended for acceptance by P. Sander.
Digital Object Identifier no. 10.1109/TVCG.2021.3135021

while approximating the real geodesic path. Our tests show that $H = e^{s^2} - 1$ is a better choice than $H = s^2$ and $H = s$. The biggest advantage of this framework lies in that it can operate on a signed distance function (SDF) representation that encodes the geometry of a 3D shape regardless of the geometric domain. Furthermore, we give an easy-to-use strategy to generate an initial path that fits the surface closely. Extensive experimental results show that the proposed algorithm outperforms the existing two-point geodesic algorithms in terms of accuracy, performance and scalability, which is validated on meshes, point clouds, implicit surfaces and parametric surfaces. Finally, we use it as a hammer to solve many variant geodesic problems including the least-cost path problem. Our contributions are three-fold:

1) We propose a new variational framework to find geodesics and discuss various choices of the convex kernel function. It can not only naturally lead to a geodesic path but also guarantee that the optimal points are evenly spaced when the algorithm converges.
2) Based on the observation that SDF decouples the geometry representation from geometric domains, we devise a SDF based geodesic algorithm that can work across various geometric domains. At the same time, we give different initialization strategies for different geometric domains.
3) Based on the powerful geodesic solving framework, we discuss some variant geodesic problems and extend our algorithm to the least-cost path problem.

The remaining of the paper is organized as follows. Section 2 reviews related works on discrete geodesics and least-cost paths. Sections 3 and 4 respectively give the formulation and our algorithm, followed by the evaluation on performance/accuracy in Section 5. In Section 6, we talk about some extensions of our algorithm as well as some potential applications. Finally, we conclude this paper in Section 7.

2 Related Work

In this section, we review the existing geodesic(either locally or globally) algorithms on various geometric domains. Besides, we shall introduce some related research works on variant geodesic problems and the least-cost path problem as well.

2.1 Shortest Paths on a Polygonal Mesh

2.1.1 Discrete Shortest Geodesic Distance Field

Given a polygonal mesh, computing a discrete shortest geodesic distance field rooted at some given source point is theoretically important since it serves as a base for solving the other variants of the geodesic problem. There is a large body of literature on this problem; See [7], [17] for a survey.

Wavefront Propagation. Like that Dijkstra's algorithm optimizes the distance to reach the goal, the spirit of wavefront propagation is widely used to compute the shortest geodesic geodesic distance field on a polygonal mesh [8], [17], [33], [35], [38], [56], [58], [61]. To our best knowledge, most of the existing exact shortest geodesic algorithms are based on wavefront propagation. Based on the observation that the edge sequences helpful for determining a shortest path are finitely many, one can arrange the possible shortest paths by *windows*. The key to design an exact shortest geodesic algorithm lies in enumerating all possible windows while effectively filtering out those useless windows. For exact shortest geodesic algorithms, the required timing cost climbs sharply with regard to the mesh complexity.

PDE Based Approaches. PDE is also a useful tool for reporting an approximate discrete shortest geodesic distance field. Crane *et al.* [12] proposed a heat equation based method that initializes the source point with a unit of heat, diffuses it in a very short time and finally rebuilds the shortest geodesic distance field by normalizing the gradients of the heat field.

Solomon *et al.* [47] showed that the optimal mass transport between a Dirac delta distribution and a uniform distribution reflects how a particle of unit mass undergoes geodesic motion. In addition, the fast marching method [42] is a numerical solution to the Eikonal equation but the algorithmic paradigm is closer to a wavefront propagation algorithm.

Graph Based Approximation. Another kind of approximation approaches is graph based. The rationale behind is that a sufficiently dense graph is able to well approximate the real intrinsic distance.

Aleksandrov *et al.* [3] suggested adding a logarithmic number of Steiner points along each edge of the mesh, which are placed in a geometric progression along an edge. Then it takes $O(mn\log(mn) + nm^2)$ time to compute a $(1+\epsilon)$-approximation geodesic path, where n is the mesh complexity and m is the total number of Steiner points. Xin *et al.* [57] adopted a different strategy by adding m (specified by the user) uniformly distributed auxiliary points on the mesh. Then they constructed a graph with $O(m^2+n)$ edges in $O(mn^2\log(n))$ time. Ying *et al.* [60] proposed a pre-computed data structure, named by *saddle vertex graph (SVG)*, which supports a divide-and-conquer distance query. Wang *et al.* [50] developed a discrete geodesic graph (DGG), which enables direct error control. Recently, Adikusuma *et al.* [1] developed a new method for constructing the DGG in a direct and efficient manner based on an accuracy aware window propagation scheme.

Smooth Distance Fields. Although speed and accuracy are two commonly used measurements of evaluating a discrete shortest geodesic algorithm, one may desire a smooth distance metric on some occasions. There are several research works belonging to this category, such as commute distance [20], diffusion distance [11], biharmonic distance [31] and heat diffusion induced distance [12]. For example, the above mentioned heat method [12] formulates the geodesic problem using only standard differential operators and solve a pair of standard linear elliptic problems to report a smooth distance field. Cao *et al.* [7] used quadratic programming to seek for a balance between the accuracy and the smoothness.

2.1.2 Two-Point Shortest Path Problem

There are some occasions where one needs to query the geodesic distance, and possibly the corresponding path as well, between a pair of fixed points p and q. Rather than compute the p-rooted distance field or the q-rooted distance field, the commonly used way is to shrink an initial path until it cannot be shortened any more. Kanai and Suzuki [27] suggested initializing the path by Dijkstra's algorithm and then repeatedly refining the initial path based on a selective refinement strategy. Martínez *et al.* [34] proposed to keep down the intersection points between the path and the edge sequence, followed by iteratively straightening a local curved segment given by three successive points. Xin and Wang [55] developed a fast edge sequence constrained shortest path algorithm and suggested updating the edge sequence if the nearby edge sequence is able to induce a shorter path. Liu *et al.* [32] considered the geodesic path problem from the perspective of optimization and proposed to minimize the total length $\sum \|x_i - x_{i+1}\|$ using the L-BFGS solver. Later, Ye *et al.* [59] developed a differential evolution (DE) based method by energy minimizing. Very recently, Sharp *et al.* [44] introduced an edge-flip technique. The algorithm repeatedly flips the connections between vertices until all the destinations are connected to the source point using a geodesic path. To summarize, most of the existing two-point geodesic algorithms [27], [32], [34], [55], [59] are devised for a certain kind of geometric domain, causing a restricted use. The goal of this paper is to decouple geodesic computation from geometric domains such that users don't need to care about the specific surface representation (mesh, point cloud, implicit surface and parametric surface). Even when the input is a mesh, we hope that the timing cost is almost independent of the mesh complexity.

2.2 Shortest Paths on Point Cloud or Implicit Surface

Mémoli and Sapiro [18] suggested enclosing the input point cloud with an offset band and approximating the real shortest geodesic distance in that band. The above-mentioned heat method [12] can also deal with a point cloud as the input as long as the graph based differential operators can be defined. Yu *et al.* [62], [63] proposed to generate a Cartesian grid such that the geodesic computation can be directly performed on the grid based implicit representation. Seong *et al.* [41] developed a bounded 3D Hamilton-Jacobi (H-J) equation solver to compute curvature tensor-based anisotropic geodesic distance on parametric and implicit surfaces.

In fact, it's more flexible to approximate a geodesic as a sequence of nearly evenly spaced 3D points and keep refining the path until it cannot be shortened any more. Based on the observation, Hofer and Pottmann [22] extended splines from euclidean spaces to a curved manifold surface, where they used $\sum \|x_{i+1} - x_i\|^2$ to drive the optimization. Ruggeri *et al.* [39] introduced an additional term to constrain the moveable points on the surface. In fact, $\int_0^1 H(|\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)|)\mathrm{d}t$ is a more general variational form for finding the geodesic, where $H = H(s)$ satisfies $H'(s) > 0$ and $H''(s) \geq 0$ for $s > 0$. The research work of this paper is based on this observation.

2.3 Geodesic Paths on Parametric Surface

Due to the correspondence between the original surface and the parameterization plane, the task of finding a shortest geodesic on the original surface can be reduced to the parameterization plane instead. For example, Beck *et al.* [5] suggested computing geodesic paths on a bicubic spline surface by using the fourth order Runge–Kutta method, while Sneyd and Peskin [46] proposed a numerical implementation based on a second order Runge–Kutta method. Both of the approaches operate on the parameterization plane. Jin *et al.* [26] presented the geodesic spectra in the sense of conformal structures based on an observation that under the uniformization metric, each homotopy class of closed curves has a unique geodesic.

2.4 Variant Versions of the Geodesic Problem

Although the shortest geodesic path is of most interest in the majority of cases, one may need to find other types of geodesics sometimes.

Geodesic Loops. Closed geodesic paths, also named geodesic loops [10], [54], can be used to measure the shape thickness from the intrinsic perspective. On one hand, they are helpful for interactive shape segmentation. On the other hand, Dey and his co-authors [13], [14], [15] proposed a series of research works on computing geodesic handle loops and tunnel loops that are able to characterize the topological features.

Non-Shortest Geodesics. From the perspective of differential geometry, a path is said to be a geodesic if the geodesic curvature vanishes at every point. Take the initial-value geodesic problem [9], [23], [24], [37] for example. The task of geodesic marching is to repeatedly extend the initial curve into a long zero-geodesic-curvature curve. Another example about non-shortest geodesics is to find a geodesic spiral [53] like that achieved on a cylindrical surface. In this paper, we propose to shorten an initial curve into a geodesic - generally the resulting curve and the initial curve are still homotopy equivalent.

Surfaces With Isotropic/Anisotropic Metrics. A Riemannian 2-manifold surface embedded in 3D is equipped with a tensor $T(x) \in \mathbb{R}^{2\times 2}$ at each point x, where T is a positive symmetric matrix. The length of a piecewise smooth curve $\gamma(s), s \in [0,1]$, can be measured by

$$L(\gamma(s)) = \int_0^1 \sqrt{\gamma'(s)^{\mathrm{T}} T(\gamma(s))\gamma'(s)}\mathrm{d}s. \tag{1}$$

When $T(x)$ has a form of $\lambda(x)I$, the Riemannian metric is reduced to be isotropic. Computing geodesics on surfaces with isotropic/anisotropic metrics [6] is very necessary in many applications.

2.5 Least-Cost Path

The least-cost path problem [2], [4], [28] can be viewed as an extension of the geodesic problem. It aims to find the optimal path in a cost equipped domain. Even if the least-cost problem has no analytic solution, our algorithm still works well.

The cost function can be distance, time cost, potential energy, etc. Some typical least-cost path problems include the Brachistochrone curve problem [25], the Catenary curve problem [45], the motion planning problem [43] and the trajectory planning problem [40]. In this paper, we directly compute a sequence of intermediate states to approximate the least-cost path, where the intermediate states are equally spaced in terms of the cost difference. Take the Brachistochrone curve problem [25] for example. The goal is to find the optimal curve that can minimize the total travel time. In implementation, we minimize $\sum H(\mathrm{Cost}(s_i \sim s_{i+1}))$, where $s_i \sim s_{i+1}$ denotes the transition from the ith configuration to the $(i+1)$th configuration, and $\mathrm{Cost}(s_i \sim s_{i+1})$ is the corresponding cost.

3 Formulation

3.1 Geodesic Functional

Given a differential Riemannian manifold $\mathscr{S}$, the length of a continuously differentiable curve $\gamma : [0,1] \ \rightarrow \mathcal{S}$ is defined by

$$\mathbf{L}(\gamma) = \int_0^1 \|\gamma^{'}(t)\| \mathrm{d}t, \tag{2}$$

where $p \overset{\mathrm{def}}{=} \gamma(0)$ and $q \overset{\mathrm{def}}{=} \gamma(1)$ are the source/target points. Given a convex function $H = H(s)$ satisfies $H'(s) > 0$ and $H''(s) \geq 0$ for $s > 0$, we shall show the following variational

$$\mathbf{E}(\gamma) = \int_0^1 H(\|\gamma^{'}(t)\|)\mathrm{d}t, \tag{3}$$

has the same critical points with $\mathbf{L}$, up to reparametrization. According to Jensen's inequality, we have

$$H(\mathbf{L}(\gamma)) = H\left(\int_0^1 \|\gamma^{'}(t)\|\mathrm{d}t\right) \leq \int_0^1 H(\|\gamma^{'}(t)\|)\mathrm{d}t = \mathbf{E}(\gamma). \tag{4}$$

The equality holds if and only if the path $\gamma(t)$ travels at constant speed. The first variation of $\mathbf{E}(\gamma)$ can be defined by

$$\delta\mathbf{E}(\gamma)(\phi) = \frac{\partial}{\partial h}\Big|_{h=0} \mathbf{E}(\gamma + h\phi), \tag{5}$$

whose critical points are precisely the geodesics. Based on this, Gorodski [21] proved the equivalence between minimizing $\mathbf{E}(\gamma)$ and minimizing $\mathbf{L}(\gamma)$ for $H(s) = s^2$. In fact, the proof can be naturally extended for a convex function $H = H(s)$ that satisfies $H'(s) > 0$ and $H''(s) \geq 0$ for $s > 0$. As pointed out in [21], the minimizer of $\mathbf{L}(\gamma)$ doesn't own the uniqueness and regularity - generally arbitrary reparameterizations are allowed. In contrast, the minimizer of $\mathbf{E}(\gamma)$ is able to unify those reasonable functions within each isotopy class into a unique function.

Choice of H. In the past literature [22], [39], $H(s) = s^2$ is used to approximate a geodesic. In this paper, we mainly consider three choices of H, i.e., (a) $H(s) = s$, (b) $H(s) = s^2$ and (c) $H(s) = e^{s^2} - 1$. We shall explain why $H(s) = e^{s^2} - 1$ is a good choice in Section 4.3.

Curved Surface. Imagine that we have a parametric surface $\mathbf{x} = \mathbf{x}(u,v)$, then for a curve $\gamma = \gamma(u(t), v(t)), t \in [0,1]$, we have $\gamma'(t) = \mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)$. In this case, the objective function becomes $\int_0^1 H(|\mathbf{x}_u u'(t) + \mathbf{x}_v v'(t)|)\mathrm{d}t$.

3.2 Discrete Setting

In the discrete setting, we use a strictly *convex* and *monotonically increasing* function H. Suppose that $x_0(=p), x_1, x_2, \cdots, x_n, x_{n+1}(=q)$, are points sampled from the path $\gamma(p,q)$. Then the length $\|\gamma(p,q)\|$ can be approximated by $\sum \|x_i x_{i+1}\|$ as long as $x_1, x_2, \cdots, x_n$ are sufficiently dense, i.e., $\xi \triangleq \max_i \|x_i x_{i+1}\|$ is sufficiently close to 0.

$$\begin{aligned} H\left(\frac{\|\gamma_{(p,q)}\|}{n+1}\right) &= H\left(\frac{1}{n+1}\sum \|x_i x_{i+1}\|\right) + o\left(\frac{\xi}{n+1}\right) \\ &\leq \frac{1}{n+1}\sum H(\|x_i x_{i+1}\|) + o\left(\frac{\xi}{n+1}\right), \end{aligned} \tag{6}$$

or

$$\lim_{\xi \to 0} \sum H(\|x_i x_{i+1}\|) \geq \lim_{\xi \to 0} (n+1) H\left(\frac{\|\gamma_{(p,q)}\|}{n+1}\right). \tag{7}$$

When the equality holds, it implies: (1) the point sequence nearly defines a geodesic path, and (2) the intermediate points are equally spaced, i.e., $\|x_0 x_1\| = \|x_1 x_2\| = \cdots = \|x_n x_{n+1}\|$. This observation motivates us to minimize $\sum_i H(\|x_i x_{i+1}\|)$ instead of the total length $\sum \|x_i x_{i+1}\|$

$$\begin{aligned} &\mathrm{Minimize}_{\{x_i\}_{i=1}^n} && \mathbf{E}(x_1, x_2, \cdots, x_n) \overset{\mathrm{def}}{=} \sum H(\|x_i x_{i+1}\|), \\ &\text{subject to} && x_1, x_2, \cdots, x_n \in \mathscr{S}. \end{aligned}$$

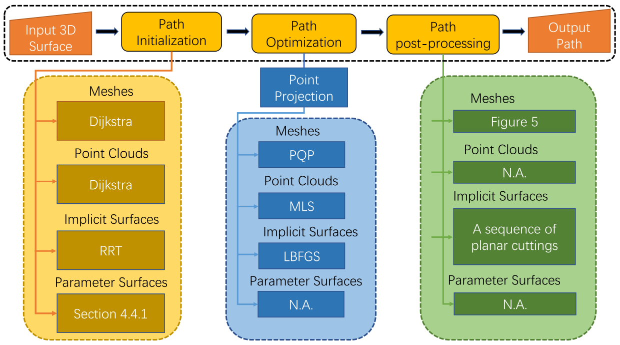


Fig. 1. Overview. Our method is a universal computational framework that works for various geometric domains, such as polygonal meshes, point clouds, implicit and parametric surfaces. Different from conventional length shortening, we minimize $\sum_i H(\|x_i x_{i+1}\|)$, where $H(s) = e^{s^2} - 1$. "N.A." means the step may not exist.

4 Algorithm

4.1 Algorithm Overview

Based on the formulation in Section 3, we propose an optimization driven approach for computing the shortest path between two given points. We first give an overview of our algorithm in Fig. 1. It mainly includes two steps, (a) path initialization and (b) path optimization; See Algorithm 1 for the pseudo-code. In the following, we shall detail the two constituent steps, followed by discussing the implementation for each separate geometric domain.

4.2 Path Initialization

Without doubt, the easiest way to take the straight-line segment between p and q as the initial path, as Fig. 2a shows. However, even if we sample a large number of points from the straight-line segment, we observe that the resulting path may be still trapped by a "sinkhole" (see Fig. 2b) although the objective function reaches a local minimum at this moment (we will explain the failure later). For this case, the resulting point sequence is not equally spaced, and there is a long segment l as shown in Fig. 2b.

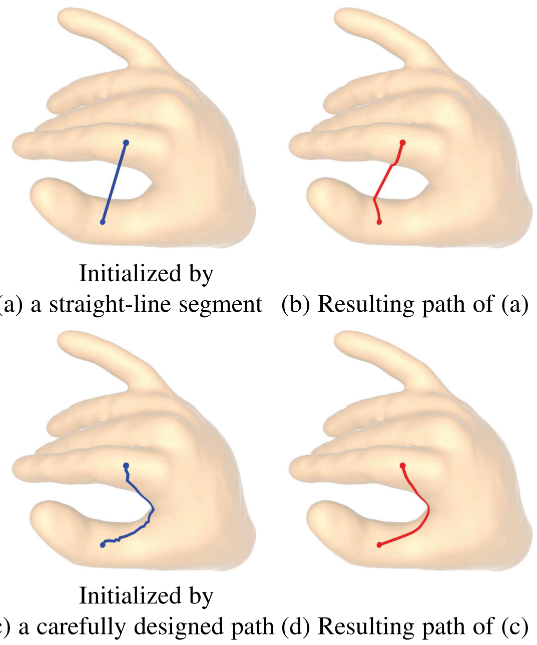

Initialized by
(a) a straight-line segment
(b) Resulting path of (a)
Initialized by
(c) a carefully designed path
(d) Resulting path of (c)

Fig. 2. If we use the straight line segment between the two endpoints as the initialization (a), the optimization process may be trapped by a curved shape variation (b). Therefore, in this paper, we use a carefully designed path as the initialization instead (c), it is able to lead to a meaningful shortest path (d).

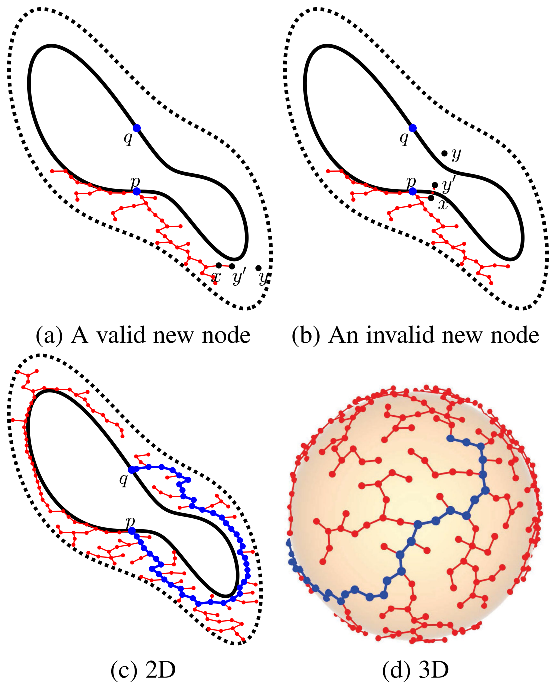


Fig. 3. Path initialization. (a) Add a valid new node into the tree. (b) Refuse an invalid node. (c) Run RRT for the 2D implicit function $f(x,y) = x^4 + y^2 + 3xy + 3\sin x \cos y - 1 = 0$. (d) Run RRT for a 3D sphere.

Algorithm 1. Computing a Shortest Path on Various Domains

```
Input:
A 3D model $\mathcal{M}$ that facilitates the projection operation;
A source $p$ and a destination $q$;
An initial path $\gamma^{(0)}(p,q)$.
Output:
An approximate shortest path $\gamma^*(p,q)$ lying on the surface $\mathcal{M}$;
1: Extract a set of samples $\{x_i^{(0)}\}_{i=1}^n$ from the initial path $\gamma^{(0)}(p,q)$;
2: Compute the objective function $\mathbf{E}(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})$, as well as its projected gradient $\triangledown\mathbf{E}(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})|_{\mathscr{M}}$ (onto the tangent plane);
3: while $\|\triangledown\mathbf{E}(x_1^{(j)}, x_2^{(j)}, \cdots, x_n^{(j)})|_{\mathscr{M}}\| \geq \epsilon$ do
4:     Project each point of $\{x_i^{(j)}\}_{i=1}^n$ onto the surface of $\mathcal{M}$ to get $\{x_i^{(j)}|_{\mathscr{M}}\}_{i=1}^n$;
5:     Compute $\mathbf{E}(x_1^{(j)}|_{\mathscr{M}}, x_2^{(j)}|_{\mathscr{M}}, \cdots, x_n^{(j)}|_{\mathscr{M}})$;
6:     Compute the gradient of $\mathbf{E}$ as well as the projected gradients $\triangledown\mathbf{E}(x_1^{(j)}|_{\mathscr{M}}, x_2^{(j)}|_{\mathscr{M}}, \cdots, x_n^{(j)}|_{\mathscr{M}})|_{\mathscr{M}}$;
7:     Perform one iteration of L-BFGS according to $\mathbf{E}$ and $\triangledown\mathbf{E}|_{\mathscr{M}}$, yielding $\{x_i^{(j+1)}\}_{i=1}^n$;
8: end while
9: Transform the resulting point sequence such that it is compatible with the original geometric domain.
```

4.2.1 Find an Initial Path

If the input is a parametric surface, the initial path can be found in the 2D parameterization plane. If the input is a mesh surface, the initial path can be given by Dijkstra's algorithm. In the following, we mainly talk about how to find an initial path assuming that the input is a general implicit function or signed distance function.

One may hope that the algorithm has better not depend on the initial path. However, there may exist multiple geodesic paths between two points (possibly with different lengths). In practice, we need to find an initial path that is as close as possible to the target geodesic path. For an implicit representation, however, it is hard to directly predict a rough path due to lack of the knowledge about global shape variation if without any precomputation. In the following, we talk about a rapidly-exploring random tree (RRT) [30] based initialization technique. We take $f(x,y,z)=0$ and $f(x,y,z)=\epsilon$ as two obstacles, followed by finding a path in the sandwiched space.

Fig. 3 gives a 2D example and a 3D example for initializing the path. In implementation, we first initialize the tree with the starting point p. We then randomly generate a point y in the sandwiched space given by $f=0$ and $f=\epsilon$. Let x be the nearest node on the up-to-date tree. We move y to y' such that $\|xy\|'$ has a length of δ (a small constant). As shown in Fig. 3a, we accept a new node y' if $f(y') < 0$. Otherwise, if y' is outside of the sandwiched space, we refuse y'; See Fig. 3b. The overall search process terminates until y' is in a distance of less than δ to the destination point q. Finally, we project the path onto the surface (see Fig. 3d), finishing path initialization.

4.2.2 Check the Validity of an Initial Path

Suppose that we obtain a point sequence $p, x_1, x_2, \cdots, x_n, q$ that cannot be shortened any more (when the optimization terminates). We keep down the value of the objective function, i.e., $\mathbf{E}_1$. After that, we find the middle point of each segment and insert the $n+1$ points back into the sequence and the sequence becomes

$$p, y_1, x_1, y_2, x_2, \cdots, x_n, y_{n+1}, x_{n+1}, q.$$

Taking it as the initial path, we can get another shortened point sequence whose corresponding value of the objective function is $\mathbf{E}_2$. It is observed that when $p, x_1, x_2, \cdots, x_n, q$ are equally spaced on the surface (no artifact exists),

$$\frac{\mathbf{E}_1}{\mathbf{E}_2} \approx \lim_{n\to\infty} \frac{n \times H(L/n)}{2n \times H(L/(2n))} = \lim_{s\to 0} \frac{H(s)}{2 \times H(s/2)}, \tag{8}$$

where L is the total length and $s \overset{\text{def}}{=} L/n$. Based on L'Hôpital's rule, we discuss $\frac{\mathbf{E}_1}{\mathbf{E}_2}$ in the following three situations.

(1) $H(0) \neq 0$: $\mathbf{E}_1/\mathbf{E}_2 \approx 1/2$ when the number of points is doubled, and $\mathbf{E} \to +\infty$ as n approaches infinity.

(2) $H(0) = 0$ but $H'(0) \neq 0$: $\mathbf{E}_1/\mathbf{E}_2 \approx 1$ when the number of points is doubled, and $\mathbf{E}$ approaches the real geodesic distance when n is sufficiently large.

(3) $H(0) = 0, H'(0) = 0$ but $H''(0) \neq 0$: $\mathbf{E}_1/\mathbf{E}_2 \approx 2$ when the number of points is doubled, and $\mathbf{E} \to 0$ as n approaches infinity.

Therefore, we can utilize the above observation to check if the resulting point sequence is able to well approximate a real geodesic path. When the resulting path may be trapped by a "sinkhole", there always exists a long segment that cannot be shortened any more. It's observed that if we select a function H belonging to the third situation, i.e., meeting $H(0) = 0, H'(0) = 0$ but $H''(0) \neq 0$, it is very easy to detect the bad path initialization. To be more detailed, $\mathbf{E}_1/\mathbf{E}_2 \approx 2$ in normal situation while $\mathbf{E}_1/\mathbf{E}_2$ is far less than 2 when the initial path doesn't fit the surface well.

We must point out that some convex functions like $H(s) = e^s, H(s) = e^{s^2}, H(s) = e^s - 1$, belonging to the first or second situation, can also be used to shorten the initial path, but cannot be used to check the validity. That's why we recommend $H(s) = s^2$ or $H(s) = e^{s^2} - 1$ in this paper. As Fig. 4 shows, $H(s) = e^{s^2} - 1$ has a better convergence rate than the other convex functions.

4.3 Path Optimization

4.3.1 Gradients

Let $x_0 \overset{\text{def}}{=} p$ and $x_{n+1} \overset{\text{def}}{=} q$ be respectively the fixed source and target points. Recall that the objective function is

$$\mathbf{E}(x_1, x_2, \cdots, x_n) = \sum_{i=0}^{n} H(\|x_{i+1} - x_i\|), \tag{9}$$

whose gradients w.r.t. $x_i, 1 \leq i \leq n$ are

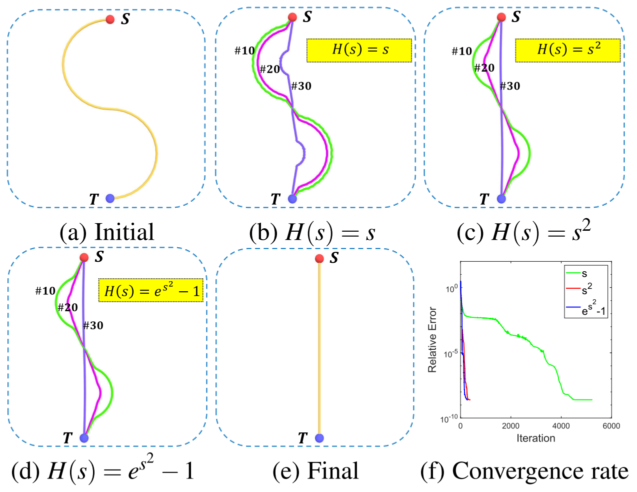
$$\triangledown_{x_i}\mathbf{E} = H'(\|x_{i+1} - x_i\|)\frac{x_i - x_{i+1}}{\|x_{i+1} - x_i\|} + H'(\|x_i - x_{i-1}\|)\frac{x_i - x_{i-1}}{\|x_i - x_{i-1}\|}. \tag{10}$$

Specially, for different choices of H: (a) $H(s) = s$, (b) $H(s) = s^2$, and (c) $H(s) = e^{s^2} - 1$, the gradients w.r.t. x_i are respectively

(a) $\nabla_{x_i}\mathbf{E} = \frac{x_i - x_{i+1}}{\|x_{i+1} - x_i\|} + \frac{x_i - x_{i-1}}{\|x_{i-1} - x_i\|}$,

(b) $\nabla_{x_i}\mathbf{E} = 2(x_i - x_{i+1}) + 2(x_i - x_{i-1})$,

(c) $\nabla_{x_i}\mathbf{E} = 2e^{\|x_i - x_{i+1}\|^2}(x_i - x_{i+1}) + 2e^{\|x_i - x_{i-1}\|^2}(x_i - x_{i-1})$
.



(a) Initial (b) $H(s) = s$ (c) $H(s) = s^2$ (d) $H(s) = e^{s^2} - 1$ (e) Final (f) Convergence rate

Fig. 4. Testing the kernel convex functions on a 2D plane: (1) $H(s) = s$, (2) $H(s) = s^2$, and (3) $H(s) = e^{s^2} - 1$, where the number of intermediate points is 200. In order to make the comparison fair, the vertical axis of (f) shows how the relative error of the total length decreases during the optimization. It can be seen that $H(s) = s^2$ and $H(s) = e^{s^2} - 1$ can achieve better convergence rate than $H(s) = s$; Furthermore, $H(s) = e^{s^2} - 1$ is slightly better than $H(s) = s^2$.

When the input is a parametric surface such that x_i can be written as $x_i = (u_i, v_i)$, then we have

$$\nabla_{u_i}\mathbf{E} = \nabla_{x_i}\mathbf{E} \cdot \nabla_u x|_{u_i, v_i},$$

and

$$\triangledown_{v_i}\mathbf{E} = \nabla_{x_i}\mathbf{E} \cdot \nabla_v x|_{u_i, v_i}.$$

4.3.2 Projection Operation

In our algorithm, the intermediate points $\{x_1, x_2, \cdots, x_n\}$ are moveable but constrained on the underlying surface. Therefore, we need a projection operation to facilitate the running of our algorithm. To be more specific, there are two occasions where one needs to project a point/vector onto the surface. First, each time when an arbitrary point x_i is updated to a new position, it needs to be projected onto the surface. Second, we need to guarantee that the gradients w.r.t. x_i are located on the tangent plane at x_i such that $\triangledown_{x_i}\mathbf{E}$ is transformed into $\triangledown_{x_i}\mathbf{E}\,|_{\mathcal{M}}$:

$$\triangledown_{x_i}\mathbf{E}\,|_{\mathcal{M}} \overset{\text{def}}{=} \triangledown_{x_i}\mathbf{E} - \text{Dot}(\triangledown_{x_i}\mathbf{E}, \vec{n})\ \vec{n}, \tag{11}$$

where $\vec{n}$ is the unit normal vector of the tangent plane at x_i. We shall explain how to implement the projection operation for various geometric domains in later subsections.

4.3.3 Optimization

With the support of gradients, we use the L-BFGS solver to minimize $\mathbf{E}$. The termination condition is $\|\triangledown_{x_i}\mathbf{E}\,|_{\mathcal{M}}\| \leq 10^{-8}$. In Fig. 4, we set up a two-point geodesic algorithm on the 2D plane to test the kernel convex functions: (a) $H(s) = s$, (b) $H(s) = s^2$, and (c) $H(s) = e^{s^2} - 1$. It's observed that $H(s) = e^{s^2} - 1$ outperforms the other two functions and achieve an accuracy of 10^{-8} after 300 iterations. As shown in Table 1, we test the kernel convex functions on different 3D models. It can be seen that $H(s) = e^{s^2} - 1$ requires less iterations to achieve an accuracy level of 10^{-3}.

TABLE 1
Testing the Kernel Convex Functions on 3D model: (1) $H(s) = s$, (2) $H(s) = s^2$, and (3) $H(s) = e^{s^2} - 1$

	$H(s) = s$	$H(s) = s^2$	$H(s) = e^{s^2} - 1$
Duck(179 vertices) Fig. 8	212	101	80
Dragon(201 vertices) Fig. 9a	249	92	85
Neptune(504 vertices) Fig. 9a	431	223	215
Squid(62 vertices) Fig. 9a	274	58	51

From the statistics about the number of iterations, it can be seen that $H(s) = e^{s^2} - 1$ requires less iterations to achieve an accuracy level of 10^{-3}. (The number of vertices past through by the initial path is also included in the table.)

TABLE 2
Three Plugin Operations on Various Geometric Domains

	Initialization	Projection	Post-processing
Meshes	Dijkstra's algorithm	PQP [29]	$\star$
Point clouds	Dijkstra's algorithm	MLS [19]	–
Implicit surfaces	RRT algorithm	$\star$	$\star$
Parametric surfaces	$\star$	–	–

4.4 Running on Various Geometric Domains

Based on the above discussion, there are three kinds of plugin operations, including initialization, projection and path postprocessing, to facilitate the running of the algorithm framework when one uses it to deal with a specific geometric domain. Table 2 summarizes the required operations for different geometric domains, where "–" means unnecessary operations.

4.4.1 Initialization

As mentioned in the previous section, one can simply connect the source point p and the target point q using a straight-line segment, and then take a set of sample points from the segment. However, even if the number of sample points is very large, the resulting path may be trapped by the highly curved shape variations.

Initialization on mesh or point cloud. For meshes or point clouds, Dijkstra's algorithm can provide an appropriate initial path.

Initialization on implicit surface. One needs to use the RRT based technique proposed in Section 4.2.1 to refine the initial path.

Initialization on parametric surface. Imagine that we have a parametric surface $\mathbf{x} = \mathbf{x}(u, v)$. For this type of geometric domain, we conduct initialization/optimization directly on the parameter domain. The objective function becomes

$$\begin{aligned}&\mathbf{E}(u_1, u_2, \cdots, u_n; v_1, v_2, \cdots, v_n)\\ &\quad= \sum_{i=0}^{n} H(\|\mathbf{x}(u_{i+1}, v_{i+1}) - \mathbf{x}(u_i, v_i)\|).\end{aligned} \tag{12}$$

Given a source point positioned at (u_0, v_0) and a target point positioned at (u_{n+1}, v_{n+1}), we connect them into a straight-line segment in the parameter plane and extract n equally spaced points from the segment. When the objective function $\mathbf{E}$ cannot be reduced any more, we check whether the points are equally spaced.

4.4.2 Projection

Recall that our algorithm framework requires that each moveable point $x \in \mathcal{S}$ moves in the tangent plane at x. In order to constrain the path on the surface throughout the optimization, we need to repeatedly project x onto $\mathcal{S}$ each time the position of x is updated.

Projection on mesh. When the geometric domain is a mesh, PQP [29] is commonly used choice for this purpose.

Projection on point cloud. When the input is a point cloud, MLS [19] becomes a desirable choice.

Projection on implicit surface. If the input is an implicit surface $f(x) = 0$, one needs a projection operation to guarantee that the points move on the surface. We can project x onto the surface along the reverse direction of $\nabla_x f$. Let x^* be the projection point. By taking $x - x^* = \mu \nabla_x f$ into

$$0 = f(x^*) \approx f(x) + (\nabla_x f)^T (x^* - x), \tag{13}$$

we get

$$\mu = f(x)/\|\nabla_x f\|^2, \tag{14}$$

and

$$x^* = x - \Big(f(x)/\|\nabla_x f\|^2\Big)\nabla_x f. \tag{15}$$

The accuracy can be enhanced by repeatedly performing the projection operation.

Projection on parametric surface. When the input is a parametric surface, the optimization is conducted directly on the parameter domain, and thus there is no need to perform projection operations.

4.4.3 Path Post-Processing

Path post-processing on mesh. In fact, for most application occasions, it suffices to report a polygonal curve as the output. However, sometimes one may have a special requirement - the geometric representation of the final path must be compatible with the input geometric domain. Take meshes for example. If we simply connect the point sequence yielded by our algorithm into a polygonal curve, then the path is not totally lying on the mesh surface - it's better to find the corresponding edge sequence and report the intersection points with the edges.

It's worth noted that the resulting path yielded by the optimization can only guarantee that each point in the point sequence is located on the surface. But the segments $x_i x_{i+1}, i = 0, 1, 2, \cdots, n$ are not completely lying on the surface generally. Sometimes there is a need to find the edge sequence past by the path. In the following, we shall use a fast projection technique to achieve this purpose.

For the trivial case that the two endpoints x_i, x_{i+1} are inside the same triangle, we directly connect them; See Fig. 5 (Case #1). Otherwise, we first project the point x_{i+1} onto the plane of the triangle $\triangle ABC$ containing x_i, yielding a projection x'_{i+1}; See Fig. 5 (Case #2). Generally x'_{i+1} is located outside $\triangle ABC$. By connecting x_i and x'_{i+1}, it is very easy to compute the exit point y_1 where the path leaves

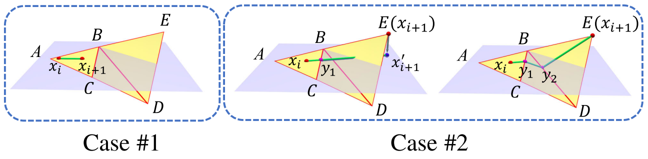


Fig. 5. Path projection. If the two endpoints x_i, x_{i+1} are inside the same triangle, we directly connect them. Otherwise, we project the direction x_ix_{i+1} onto the plane of the triangle $\triangle ABC$ containing x_i, producing a ray rooted at x_i. Suppose that the ray intersects the boundary of $\triangle ABC$ at y_1. We further consider how to project the segment y_1x_{i+1} onto the neighboring triangle $\triangle BCD$. Finally, we get the complete projected path.

$\triangle ABC$ and enters the adjacent triangle. All the edge points past through by the path can be found in turn by repeating the process; See Fig. 5 (Case #2).

The algorithm works well except the following case: the projection point x'_{i+1} is located inside $\triangle ABC$, causing failure in inferring the exit point for $\triangle ABC$. Based on our experience, the rare case happens when (a) the number of points, n, is too small, and (b) there is a huge variation of normal vectors along the path. We can find a plane Π containing x_i and x_{i+1} and trace the sub-path between x_i and x_{i+1} by intersecting the surface with Π. The plane equation is

$$(x - x_i) \cdot \left(\frac{n_i + n_{i+1}}{2} \times (x_{i+1} - x_i)\right) = 0,$$

where n_i, n_{i+1} are respectively the normal vectors at x_i and x_{i+1}, and "$\times$" represents the cross-product operation. Finally, we skip the face interior points and output the edge points as well as the corresponding edge sequence.

Path post-processing on point cloud. When the input is a point cloud, there is no need to conduct further path post-processing.

Path post-processing on implicit surface. On most occasions, it suffices to return a sequence of points as the approximate shortest path for an implicit surface as the geometric domain. If users require the output path must be completely lying on the implicit surface, each straight-line segment x_ix_{i+1}, upon being projected on the surface, becomes a curved segment. Similarly with the mesh case, we can represent the curved segment by the intersection between the base surface and the following plane

$$(x - x_i) \cdot \left(\frac{n_i + n_{i+1}}{2} \times (x_{i+1} - x_i)\right) = 0. \tag{16}$$

Recall that the point sequence is dense enough to encode the real geodesic path (see Section 4.2.2). When our algorithm terminates, the segment x_ix_{i+1} is short enough such that the difference between the straight-line segment x_ix_{i+1} and the corresponding geodesic path between them is inconspicuous, which implies that x_i and x_{i+1} cannot be on different components.

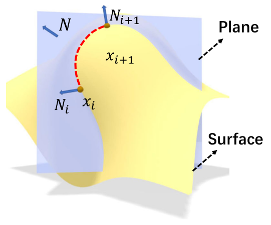


Fig. 6. Path post-processing for implicit surface: each segment can be obtained by intersecting the implicit surface with a plane.

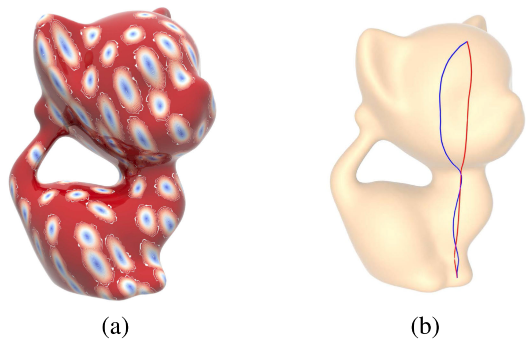


Fig. 7. Anisotropic geodesic paths. The red path in (b) is a geodesic under the normal metric, while the blue path is computed under the anisotropic metric shown in (a).

Path postprocessing on parametric surface. When the input is a parametric surface, it is enough to conduct the optimization directly on the parameter domain. There is no need to project the resulting path back to the surface.

4.5 Anisotropic Geodesic Paths

For many computer graphics occasions, the input surface may be equipped with anisotropic metrics. In the discrete setting, one can assign each face with a 2×2 matrix to denote the anisotropic metric operator T. Here we show that the proposed geodesic algorithm can be easily extended to this situation. The only change is to evaluate the length of $\|x_ix_{i+1}\|$ using $\|\frac{T(x_i)+T(x_{i+1})}{2}(x'_i - x'_{i+1})\|$, where x'_i, x'_{i+1} are respectively the corresponding 2D coordinates w.r.t. the two orthogonal directions at $\frac{x_i+x_{i+1}}{2}$, and $\frac{T(x_i)+T(x_{i+1})}{2}$ denotes the average metric operator. Fig. 6a shows the configuration of anisotropic metrics, while Fig. 6b shows two paths, where the red path is a geodesic under the normal metric and the blue path is computed under the anisotropic metric.

5 Evaluation

In this section, we evaluate our approach in terms of initialization, run-time performance, accuracy, as well as the most significant feature of our algorithm, i.e., high scalability w.r.t. various geometric domains. We implemented our algorithm in Microsoft Visual C++ 2017, and our implementation does not require any additional numeric package. All the experiments were conducted on a computer with Intel (R) Core(TM) i5-8400K CPU 2.80GHz and 16GB memory.

5.1 Existing Methods

To demonstrate the effectiveness of our algorithm, we compare it with three classes of existing algorithms.

Exact algorithms that compute the globally shortest paths on manifold triangle meshes, including the Chen-Han (CH) algorithm [8], the improved Chen-Han (ICH) algorithm [56], and the vertex-oriented triangle propagation (VTP) algorithm [38].

Approximate global algorithms that compute the single-source-all-destinations geodesics, including Dijkstra's algorithm (Dijkstra) [16], the fast marching method (FMM) [42], and the heat method (Heat) [12].

Approximate local algorithms that compute the single-source-single-destination geodesics, including the selective refinement of the discrete graph method (SR) [27], the local

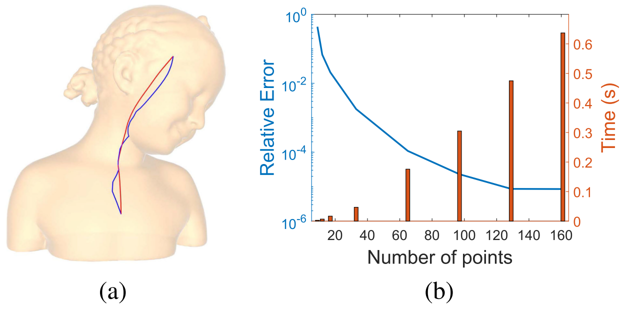


Fig. 8. Trade off between accuracy and computational cost. More sample points lead to higher accuracy but cost more computational resources. In (a), the blue path is obtained by Dijkstra's algorithm and the path passes through 33 vertices. It's observed in (b) that if we equalize the number of sample points and the number of vertices past through, the relative error can be controlled at a level of about 0.1% generally; in the meanwhile, the overall optimization takes fewer iterations and less computational time.

straightening method (LS) [34], the Differential-Evolution-Based Method (DE) [59], the square length optimization (SLO) [22], [39], and the LBFGS-based length minimization (LLM) method [32].

Experimental setting of our algorithm. If the input surface is a parametric surface or implicit surface, we directly use a straight-line segment to initialize the path. But when the input surface is a point cloud or mesh, there exists a different initialization strategy. Without doubt, if we use more sample points to approximate the real geodesic, it can lead to higher accuracy but require more computational cost at the same time. On one hand, it is optional for users to set the number of intermediate points for obtaining various levels of accuracy. On the other hand, it's observed in Fig. 7 that if we equalize the number of sample points and the number of vertices past through by Dijkstra's algorithm, the relative error can be controlled at the level of 0.1% generally; in the meanwhile, the overall optimization requires less iterations and less computational time. Therefore, for meshes and point clouds, Dijkstra's algorithm becomes a default choice.

5.2 Quantitative Comparison

In this subsection, our discussion focuses on meshes since it is the most popular geometric domain. (We shall discuss other types of geometric domains in the Section 5.3) Here we divide the existing algorithms into three kinds and discuss them kind by kind. For purpose of comparison, we subdivide the Duck model into different resolutions; See Fig. 8. It's worth noting that the mid-point subdivision is used to keep the different versions with the same geometry.

5.2.1 Comparison With Exact Algorithms

It's no wonder that our algorithm runs many times faster than the available exact algorithms, among which the VTP algorithm [38] is the state-of-the-art exact algorithm, to our best knowledge. As shown in Fig. 8b, when the number of vertices of the Duck model amounts to about 100k, the VTP algorithm requires about 3.2 seconds to compute the red path, while our algorithm costs only 0.42 seconds, which exhibits a big advantage of performance. Note that the termination condition of our algorithm is that the (projected) gradient norm is less than 1e-8. When the algorithm finishes, the relative error of our algorithm for this case is 0.01%.

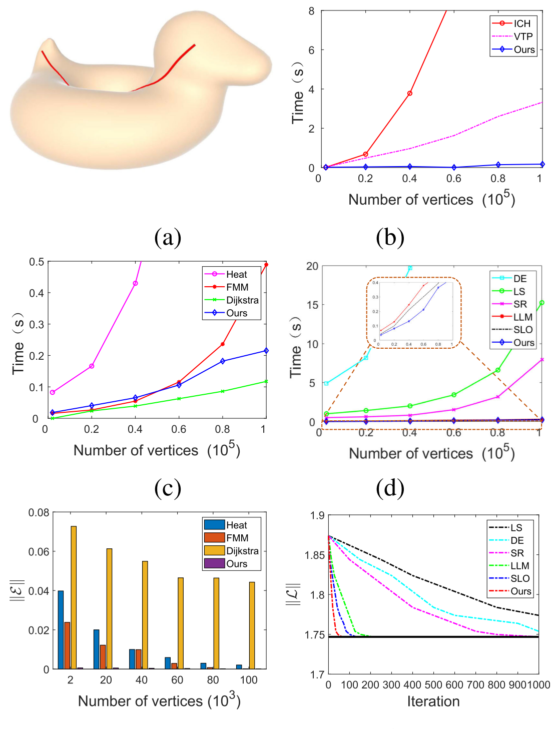


Fig. 9. Results on the Duck model under various resolutions. We subdivide the Duck model into different resolutions. We use the Duck model to compare our algorithm with various existing approaches in performance (b-d) and the accuracy (e-f).

5.2.2 Comparison With Approximate Global Algorithms

Dijkstra's algorithm, the Heat algorithm and the FMM algorithm, due to the high performance, are very popular on geodesic approximation in the research community. Although they are generally used to compute a geodesic distance field, they can also be used to query a shortest path between two points. We give a plot in Fig. 8c to show the comparison between our algorithm and the three approximate approaches. It can be clearly seen that our algorithm runs faster than FMM and Heat, and is comparable to Dijkstra's algorithm in performance (our algorithm has higher accuracy than FMM and Dijkstra's algorithm, which shall be shown later). A more interesting observation is that the timing costs of FMM, Heat and Dijkstra's algorithm climb w.r.t. the number of vertices of the input mesh while our algorithm remains almost unchanged. This is due to the fact that our optimization operates in the 3D space (except the final projection step), rather than on the edge sequence. Therefore, our approach is more suitable for computing geodesics on high-resolution models.

It can be seen from Fig. 8e that for FMM, Heat and Dijkstra's algorithm, the errors decrease with regard to the increasing of mesh resolutions. When the number of vertices increases from 2K to 100K, the absolute error of length given by FMM is reduced from 2.38% to 0.013%, that given by the Heat algorithm is from 3.98% to 0.21%, while that given by Dijkstra's algorithm is reduced from 7.27% to 4.26%. We must point that since we use Dijkstra's algorithm to do initialization, the number of sample points will also increase with regard to the increasing of mesh resolutions and thus the results reported by our algorithm are more accurate - an error of 0.0624% on the 2K-vertex model (the number of intermediate points is 39) and 0.0035% on the 100K-vertex model (179 points).

5.2.3 Comparison With Approximate Local Algorithms

There are several existing algorithms on computing a shortest path between two points. They are roughly based on keeping shrinking the original path until the path cannot be shortened any more. For example, Martínez *et al.* [34] suggested repeatedly straightening the path $\gamma(p,q)$ until the difference of length between two successive iterations is less than a prescribed tolerance, while Liu *et al.* [32] used a L-BFGS solver to minimize the total length of $\gamma(p,q)$ by taking the intersections between the path and the base edge sequence as driving variables. Both of them operate on the mesh edges and thus depend on the number of the edge sequences passed through by $\gamma(p,q)$. From Fig. 8d, we can see that our algorithm outperforms the existing approximate local algorithms when the mesh resolutions increase. As Fig. 8f shows, we compare selective refinement (SR) [27], local straightening (LS) [34], DE-Path (DE) [59] and length optimization (LLM) [32]. By contrast, our algorithm has a super higher convergence rate. For example, the initial path in Fig. 8a passes through 179 vertices. Our algorithm requires only 80 iterations to achieve an accuracy level of 10^{-3}.

5.3 Qualitative Discussion

In this paper, we mentioned some geometric domains such as meshes, point clouds, implicit surfaces and parameter surfaces. Therefore, whether a geodesic algorithm can work across various geometric domains is an important indicator. Recall that the heat based geodesic method bridges meshes and point clouds using the Laplace operator. Our algorithm, by contrast, can deal with a broader class of geometric domains (see Fig. 9), which is due to the point-to-surface projection operation. If the input is a parametric surface, our algorithm can work on the parameterization plane.

In the rightmost column of Fig. 9, the Klein bottle model has a parametric form

$$\begin{cases} x(u,v) = (a + b(\cos\frac{u}{2}\sin v - \sin\frac{u}{2}\sin 2v))\cos u \\ y(u,v) = (a + b(\cos\frac{u}{2}\sin v - \sin\frac{u}{2}\sin 2v))\cos u \\ z(u,v) = b(\sin\frac{u}{2}\sin v + \cos\frac{u}{2}\sin 2v) \end{cases}, \tag{17}$$

where $0 \le u \le \pi$ and $0 \le v \le 2\pi$.

The Mobius surface model has a parametric form

$$\begin{cases} x(u,v) = \cos 2u + v\cos u\cos 2u \\ y(u,v) = \sin 2u + v\cos u\sin 2u \\ z(u,v) = v\sin u \end{cases}, \tag{18}$$

where $0 \le u \le \pi$ and $-\frac{1}{2} \le v \le \frac{1}{2}$.

The Spiral surface model has a parametric form

$$\begin{cases} x(u,v) = u\cos v \\ y(u,v) = u\sin v \\ z(u,v) = av \end{cases}, \tag{19}$$

where $u \ge -\infty$ and $v \le \infty$.

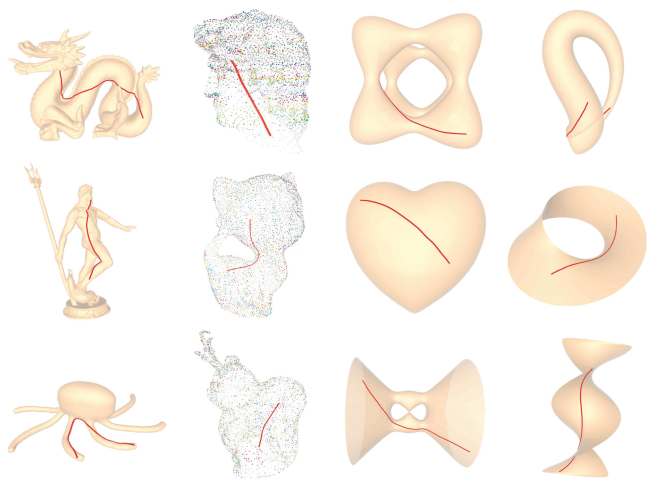

(a) Mesh (b) Point cloud (c) Implicit (d) Parameter

Fig. 10. Results on various surface domains.

6 Applications

In this section, we demonstrate the uses of our algorithm in various applications.

6.1 Geodesic Variants

There are many versions of the geodesic algorithm, depending on specific application occasions. Geodesic loops [51], [52], or closed geodesic curves, are useful in many applications such as shape segmentation, girth estimation, and topology analysis. Our algorithm can compute both geodesic paths and loops. Figs. 10a and 10b shows a 35K-vertex Cat model and three initial cutting loops. Wu and Tai [51] suggested finding the level set where the geodesic curvature is zero. Xin *et al.* [52] proposed a shortening technique to repeatedly unfold a long face sequence. The former runs in 0.23 seconds in average for this example and the latter requires 0.14 seconds in average.By contrast, our algorithm takes only 0.12 seconds in average to compute a geodesic loop when the number of sampling points is set to 84, which shows that our algorithm can be used in the scenario of interactive shape segmentation.

In addition, although globally shortest paths are mostly of researchers' interest, there are also many scenarios where one needs a long geodesic. Here we use the term "long geodesic" to denote a path that meets the geodesic property at each point but is not globally shortest. In fact, as long as users specify an meaningful initial path, our algorithm is able to find a long geodesic that resembles the initial path. In Figs. 10c, 10d, 10e, and 10f, we show four long-geodesic examples on different models.

6.2 Extension to the Least-Cost Path Problem

It's natural to extend our algorithm to infer the solution to the least-cost path problem.

Here we take the brachistochrone curve problem as an example to show its uses. A brachistochrone curve, also known as curve of fastest descent, is a curve between two fixed points in a vertical plane, on which a bead slides frictionlessly under the influence of a uniform gravitational field to a given end point in the shortest time. Imagine that

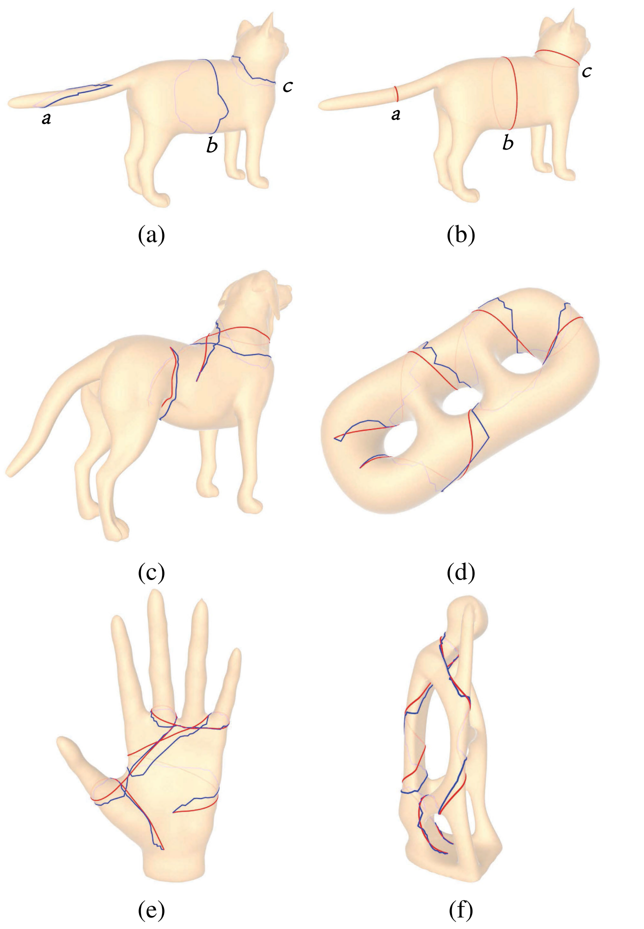


Fig. 11. After users input three initial cutting loops (a), our algorithm can shrink them quickly into geodesic loops (b). For this example, our algorithm takes only 0.12 seconds in average for finding each geodesic loop. Our method can compute long geodesics that are homologous to the initial path.

the curve can be represented by $y = y(x)$. The total travel time from (x_0, y_0) to (x_{n+1}, y_{n+1}) is

$$T(y) = \int_{x_0}^{x_{n+1}} \sqrt{\frac{1 + y'^2}{2gy}} \mathrm{d}x. \tag{20}$$

With the support of our algorithm, it is easy to report a sequence of intermediate states to approximate the real the brachistochrone curve. Let

$$\{p(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n), q(x_{n+1}, y_{n+1})\}$$

be a discrete representation of the sliding path. When sliding from (x_i, y_i) to (x_{i+1}, y_{i+1}), the average speed in this period is

$$v_i = \sqrt{2g\Big(y_0 - \frac{y_i + y_{i+1}}{2}\Big)}. \tag{21}$$

Therefore, the time duration for sliding from (x_i, y_i) to (x_{i+1}, y_{i+1}) is

$$t_i = \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{v_i}. \tag{22}$$

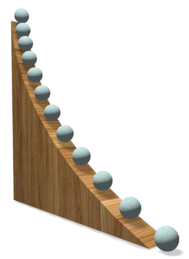


Fig. 12. A brachistochrone curve optimized by our algorithm with $n = 10$.

Similar to the geodesic path formulation, we minimize the sum of squared time $\sum H(t_i)$. Fig. 11 shows a brachistochrone curve with $n = 10$. When the optimization algorithm terminates, the sequence of points (x_i, y_i) are evenly spaced in terms of the sliding time. Our optimization method takes less than 0.1 seconds.

7 Conclusion and Discussions

We propose a novel method for computing a two-point geodesic path on 3D surfaces. Our method has two fundamental differences than the conventional curve shortening approaches. First, we discretize a geodesic path by a point sequence $\{x_i\}_{i=1}^n$ in the ambient space so that the path is independent of the surface representation. Second, we minimize $\sum H(\|x_i - x_{i+1}\|)$ instead of the path length $\sum \|x_i - x_{i+1}\|$. Furthermore, we give a set of strategies on initialization, optimization and path post-processing.

The proposed algorithm can operate on a signed distance function (SDF) representation that encodes the geometry of a 3D shape regardless of the geometric domain, and outperform the existing two-point geodesic algorithms in terms of accuracy, performance and scalability, which is validated on meshes, point clouds, implicit surfaces and parametric surfaces. Finally, we demonstrate its uses in many variant geodesic problems and the least-cost path problem (e.g., the brachistochrone curve problem).

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Y. Y. Adikusuma, Z. Fang, and Y. He, "Fast construction of discrete geodesic graphs," *ACM Trans. Graph.*, vol. 39, no. 2, pp. 1–14, Mar. 2020.
[2] L. Aleksandrov, H. N. Djidjev, H. Guo, A. Maheshwari, D. Nussbaum, and J.-R. Sack, "Algorithms for approximate shortest path queries on weighted polyhedral surfaces," *Discrete Comput. Geometry*, vol. 44, no. 4, pp. 762–801, 2010.
[3] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J. R. Sack, "An ϵ — Approximation algorithm for weighted shortest paths on polyhedral surfaces," in S. Arnborg and L. Ivansson, Eds., Berlin, Germany, 1998, pp. 11–22.
[4] L. Aleksandrov, A. Maheshwari, and J.-R. Sack, "Determining approximate shortest paths on weighted polyhedral surfaces," *J. ACM*, vol. 52, no. 1, pp. 25–53, Jan. 2005.
[5] J. M. Beck, R. T. Farouki, and J. K. Hinds, "Surface analysis methods," *IEEE Comput. Graph. Appl.*, vol. 6, no. 12, pp. 18–36, Dec. 1986.
[6] S. Bougleux, G. Peyré, and L. D. Cohen, "Anisotropic geodesics for perceptual grouping and domain meshing," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp 129–142.

[7] L. Cao *et al.*, "Computing smooth quasi-geodesic distance field (QGDF) with quadratic programming," *Comput.-Aided Des.*, vol. 127, 2020, Art. no. 102879.

[8] J. Chen and Y. Han, "Shortest paths on a polyhedron," in *Proc. 6th Annu. Symp. Comput. Geometry*, May 1990, pp. 360–369.

[9] P. Cheng, C. Miao, Y.-J. Liu, C. Tu, and Y. He, "Solving the initial value problem of discrete geodesics," *Comput.-Aided Des.*, vol. 70, pp. 144–152, 2016.

[10] W. Chunlin and T. Xuecheng, "A level set formulation of geodesic curvature flow on simplicial surfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 4, pp. 647–662, Jul./Aug. 2010.

[11] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 5–30, 2006.

[12] K. Crane, C. Weischedel, and M. Wardetzky, "Geodesics in heat: A new approach to computing distance based on heat flow," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 1–11, 2013.

[13] T. K. Dey, F. Fan, and Y. Wang, "An efficient computation of handle and tunnel loops via Reeb graphs," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–10, 2013.

[14] T. K. Dey, K. Li, and J. Sun, "On computing handle and tunnel loops," in *Proc. Int. Conf. Cyberworlds*, 2007, pp. 357–366.

[15] T. K. Dey, K. Li, J. Sun, and D. Cohen-Steiner , "Computing geometry-aware handle and tunnel loops in 3D models," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[16] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[17] J. Du, Y. He, Z. Fang, W. Meng, and S. Q. Xin, "On vertex-oriented triangle propagation algorithm: Parallelization and approximation," *Comput.-Aided Des.*, vol. 130, 2021, Art. no. 102943.

[18] G. S. Facundo Mémoli, "Distance functions and geodesics on points clouds," *SIAM J. Appl. Math.*, vol. 65, no. 4, pp. 1227–1260, 2005.

[19] S. Fleishman, D. Cohen-Or , and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3 pp. 544–552, 2005.

[20] F. Fouss, A. Pirotte, J. Renders, and M. Saerens, "Random-Walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3 pp. 355–369, Mar. 2007.

[21] C. Gorodski, *An Introduction to Riemannian Geometry*. Berlin, Germany: Springer, 2014.

[22] M. Hofer and H. Pottmann, "Energy-minimizing splines in manifolds," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 284–293, 2004.

[23] I. Hotz and H. Hagen, "Visualizing geodesics," in *Proc. Vis.*, 2000, pp. 311–318.

[24] Huang, Tao, Sun, Ronglei, Zhang, and Peng, "A geometric method for computation of geodesic on parametric surfaces," *Comput. Aided Geometric Des.*, vol. 38, pp. 24–37, 2015.

[25] S. K. Jha, H. Parthsarathy, J. R. P. Gupta, and P. Gaur, "Verification of the veracity of brachistochrone curve and evolution of optimal control, "in *Proc. India Int. Conf. Power Electron.*, 2011, pp. 1–4.

[26] M. Jin, F. Luo, S.-T. Yau, and X. Gu, "Computing geodesic spectra of surfaces," *Proc. ACM Symp. Solid Phys. Model.*, 2007, pp. 387–393.

[27] T. Kanai and H. Suzuki, "Approximate shortest path on a polyhedral surface based on selective refinement of the discrete graph and its applications," in *Proc. Geometric Model. Process. Theory Appl.*, 2000, pp. 241–250.

[28] M. Lanthier, A. Maheshwari, and J.-R. Sack, "Approximating weighted shortest paths on polyhedral surfaces," in *Proc. 13th Annu. Symp. Comput. Geometry*, New York, NY, USA, 1997, pp. 274–283.

[29] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp. Proc.*, vol. 4, pp. 3719–3726, 2000.

[30] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Ames, IA, USA, Tech. Rep. 98–11, 1999.

[31] Y. Lipman, R. M. Rustamov, and T. A. Funkhouser, "Biharmonic distance," *ACM Trans. Graph.*, vol. 29, no. 3, pp. 1–11, 2010.

[32] B. Liu, S. Chen, S.-Q. Xin, Y. He, Z. Liu, and J. Zhao, "An optimization-driven approach for computing geodesic paths on triangle meshes," *Comput.-Aided Des.*, vol. 90, pp. 105–112, Sep. 2017.

[33] Y.-J. Liu, "Exact geodesic metric in 2-manifold triangle meshes using edge-based data structures," *Comput.-Aided Des.*, vol. 45, no. 3, pp. 695–704, Mar. 2013.

[34] D. Martínez, L. Velho, and P. C. Carvalho, "Computing geodesics on triangular meshes," *Comput. Graph.*, vol. 29, no. 5, pp. 667–675, Oct. 2005.

[35] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM J. Comput.*, vol. 16, no. 4, pp. 647–668, 1987.

[36] G. Peyré *et al.*, "Geodesic methods in computer vision and graphics," *Found. TrendsÂ® Comput. Graph. Vis.*, vol. 5, no. 3–4, pp. 197–397, Dec. 2010.

[37] K. Polthier and M. Schmies, "Straightest geodesics on polyhedral surfaces," in *Proc. ACM SIGGRAPH Courses*, New York, NY, USA, 2006, pp. 30–38.

[38] Y. Qin, X. Han, H. Yu, Y. Yu, and J. Zhang, "Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–13, 2016.

[39] M. R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati, "Approximating Geodesics on Point Set Surfaces," in *Proc. 3rd Eurograph. / IEEE VGTC Conf. Point-Based Graph.*, in M. Botsch, B. Chen, M. Pauly, and M. Zwicker, Eds., 2006, pp. 85–94.

[40] G. Sahar and J. Hollerbach, "Planning of minimum-time trajectories for robot arms," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1985, pp. 751–758.

[41] J.-K. Seong, W.-K. Jeong, and E. Cohen, "Curvature-based anisotropic geodesic distance computation for parametric and implicit surfaces," *Vis. Comput.*, vol. 25, no. 8, pp. 743–755, 2009.

[42] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, pp. 1591–1595, 1996.

[43] M. Sharir, "Robot motion planning," *Commun. Pure Appl. Maths.*, vol. 48, no. 9, pp. 1173–1186, 1995.

[44] N. Sharp and K. Crane, "You can find geodesic paths in triangle meshes by just flipping edges," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–15, 2020.

[45] G. SHILOV, "Chapter III - The calculus of variations," *Mathematical Analysis*. Oxford, U.K.: Pergamon Press, 1965, pp. 78–141.

[46] J. Sneyd and C. S. Peskin. *Computation of geodesic trajectories on tubular surfaces*. Society for Industrial and Applied Mathematics, 1990.

[47] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher, "Earth mover's distances on discrete surfaces," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–12, 2014.

[48] J. Tao, J. Zhang, B. Deng, Z. Fang, Y. Peng, and Y. He, "Parallel and scalable heat methods for geodesic distance computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 579–594, Feb. 2021.

[49] J. Vekhter, J. Zhuo, L. F. G. Fandino, Q. Huang, and E. Vouga, "Weaving geodesic foliations," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[50] X. Wang, Z. Fang, J. Wu, S. Xin, and Y. He, "Discrete geodesic graph (DGG) for computing geodesic distances on polyhedral surfaces," *Comput. Aided Geometric Des.*, vol. 52, pp. 262–284, 2017.

[51] C. Wu and X. Tai, "A level set formulation of geodesic curvature flow on simplicial surfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 4, pp. 647–662, Jul./Aug. 2010.

[52] S. Xin, C. Fu, and Y. He, "Efficiently computing exact geodesic loops within finite steps," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 06, pp. 879–889, Jun 2012.

[53] S. Q. Xin, S. Chen, J. Zhao, and Z. Pan, "Measuring length and girth of a tubular shape by quasi-helixes," *Comput. Graph.*, vol. 38, no. 1, pp. 392–398, 2014.

[54] S.-Q. Xin, Y. He, and C.-W. Fu, "Efficiently computing exact geodesic loops within finite steps," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 6, pp. 879–889, Jun. 2012.

[55] S.-Q. Xin and G.-J. Wang, "Efficiently determining a locally exact shortest path on polyhedral surfaces," *Comput.-Aided Des.*, vol. 39, no. 12, pp. 1081–1090, Dec. 2007.

[56] S.-Q. Xin and G.-J. Wang, "Improving Chen and Han's alogorithm on the discrete geodesic problem," *ACM Trans. Graph.*, vol. 28, no. 4, pp. 1–8, 2009.

[57] S.-Q. Xin, X. Ying, and Y. He, "Constant-time all-pairs geodesic distance query on triangle meshes," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, New York, NY, USA, 2012, pp. 31–38.

[58] C. Xu, T. Y. Wang, Y.-J. Liu, L. Liu, and Y. He, "Fast wavefront propagation (FWP) for computing exact geodesic distances on meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 7, pp. 822–834, Jul. 2015.

[59] Z. Ye, Y. Liu, J. Zheng, K. Hormann, and Y. He, "De-path: A differential-evolution-based method for computing energy-minimizing paths on surfaces," *Comput. Aided Des.*, vol. 114, pp. 73–81, 2017.

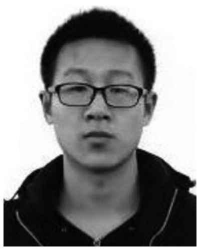
[60] X. Ying, X. Wang, and Y. He, "Saddle vertex graph (SVG): A novel solution to the discrete geodesic problem," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 170:1–170:12, 2013.

[61] X. Ying, S.-Q. Xin, and Y. He, "Parallel Chen-Han (pch) algorithm for discrete geodesics," *ACM Trans. Graph.*, vol. 33, no. 1, pp. 1–11, Feb. 2014.

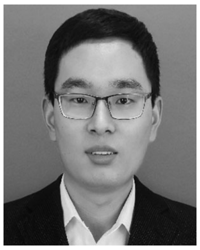
[62] H. Yu and J. Zhang, "Geodesic computation on implicit surfaces," *Int. J. Inf. Sci. Comput. Math.*, vol. 2, pp. 33–49, Jan. 2010.

[63] H. Yu, J. J. Zhang, and J. Zheng, "Geodesics on point clouds," *Math. Problems Eng.*, 2014, no. pt. 9, pp. 1–12, 2014.

Na Yuan is currently working toward the MASc degree in computer science and technology with Shandong University. Her research interests include computational geometry, computer graphics, and computer-aided design.



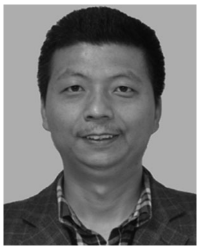
Peihui Wang received the MAEng degree from Shandong University in 2021. He is currently with ArcSoft. His research interests include computational geometry, computer graphics, and physics animation.



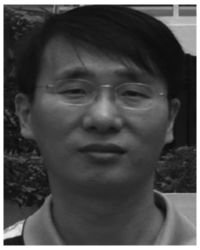
Wenlong Meng is currently working toward the PhD degree in computer science with Shandong University. His research interests include computational geometry, computer graphics, and computer-aided design.

Shuangmin Chen received the PhD degree from Ningbo University in 2018. She is currently an assistant professor with the School of Information and Technology, Qingdao University of Science and Technology, China. During the past ten years, she has authored or coauthored more than 40 research papers on famous journals or conferences. Her research interests include computer graphics and computational geometry.

Jian Xu received the PhD degree in composite material from the Catholic University of Leuven, Belgium. He is currently a professor with Dalian University of Technology. In the past 10 years, he has authored or coauthored more than 30 papers. His research interests include composite material equipment development, composite material process simulation, composite material full-scale simulation, and composite material process development. He is the principal investigator of several national research projects of China.



Shiqing Xin received the PhD degree with Zhejiang University, China, in 2009. He is currently an associate professor with the school of computer science, Shandong University. He was a research fellow with Nangyang Technological University, Singapore for three years. During the past ten years,he he has authored or coauthored more than 60 papers on top journals and conferences, including the *IEEE Transactions on Visualization and Computer Graphics* and *ACM TOG*. His research interests include various geometry processing algorithms, especially geodesic computation approaches, and Voronoi or power tessellation methods. He was the recipient of the three best paper awards and many other academic awards.



Ying He received the BS and MS degrees in electrical engineering from Tsinghua University, and the PhD degree in computer science from Stony Brook University. He is currently an associate professor with the School of Computer Engineering, Nanyang Technological University, Singapore. He is interested in the problems that require geometric computing and analysis.

Wenping Wang (Fellow, IEEE) received the PhD degree in computer science from the University of Alberta in 1992. He is currently the chair professor of computer science with the University of Hong Kong. His research interests include computer graphics, computer visualization, computer vision, robotics, medical image processing, and geometric computing. He is associate editor for several premium journals, including the *Computer Aided Geometric Design*, *Computer Graphics Forum*, *IEEE Transactions on Computers*, and *the IEEE Computer Graphics and Applications*, and has chaired more than 20 international conferences, including Pacific Graphics 2012, ACM Symposium on Physical and Solid Modeling (SPM) 2013, SIGGRAPH Asia 2013, and Geometry Submit 2019. He was the recipient of the John Gregory Memorial Award for his contributions in geometric modeling.