Computer-Aided Design 102 (2018) 128–138

Contents lists available at ScienceDirect

Computer-Aided Design

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

Lightweight preprocessing and fast query of geodesic distance via proximity graph

Shiqing Xin [a], Wenping Wang [b], Ying He [c], Yuanfeng Zhou [a], Shuangmin Chen [d,*], Changhe Tu [a], Zhenyu Shu [e]

[a] School of Computer Science and Technology, Shandong University, China
[b] Department of Computer Science, The University of Hong Kong, Hong Kong
[c] School of Computer Engineering, Nanyang Technological University, Singapore
[d] School of Information Science and Technology, Qingdao University of Science and Technology, China
[e] School of Information Science and Engineering, Ningbo Institute of Technology, Zhejiang University, China

ARTICLE INFO

Keywords:
Proximity graph
Geodesic distance
Real-time interaction
Exponential map
Spline curves

ABSTRACT

Computing geodesic distance on a mesh surface $\mathcal{S}$ efficiently and accurately is a central task in numerous computer graphics applications. In order to deal with high-resolution mesh surfaces, a lightweight preprocessing is a proper choice to make a balance between query accuracy and speed. In the preprocessing stage, we build a proximity graph $\mathcal{G}$ with regard to a set of sample points and keep the exact geodesic distance between any pair of nearby sample points. In the query stage, given two query points s and t, we augment the proximity graph $\mathcal{G}$ by adding s and t on-the-fly, and then use the shortest path between s and t on the augmented proximity graph to approximate the exact geodesic path between s and t. We establish an empirical relationship between the number of samples and expected accuracy (measured in relative error), which facilitates fast and accurate query of geodesic distance with a lightweight processing cost. We exhibit the uses of the new approach in two applications—real-time computation of discrete exponential map for texture mapping and interactive design of spline curves on surfaces.

1. Introduction

Fast and accurate query of distance map [1] is central to many computer graphics applications, such as surface segmentation and edit [2,3], mesh skinning [4] and watermarking [5], Poisson sampling [6], meshing [7], isometry-invariant shape retrieval [8], intrinsic Voronoi tessellation [9–12], minimum-distortion parametrization [13–15] and non-rigid correspondence [16]. Also, the research topic has a close relationship with many other research fields [17], such as medical imaging [18], and robot motion planning [19], and architectural geometry [20].

In light of the fact that exact geodesic algorithms [21–27] are time consuming and hard to be applied to large mesh surfaces, one has to make a balance between accuracy and speed in many real computer graphics applications. In the last decades, there is a large body of literature on approximate geodesic computation [28–39]. But the status quo is that it is still hard to find a suitable geodesic algorithm in practice. A natural question arises: Can we achieve an accurate and fast distance query with a lightweight processing (roughly linear to the model complexity)? Furthermore, in many computer graphics applications, e.g., discrete exponential map generation, we hope the reported geodesic distance is as accurate as possible if measured with relative error. Obviously, the existing approximation approaches cannot meet the couple of requirements.

In this paper we shall present a fast and accurate method for geodesic computation on a 2D manifold surface. The key idea of the method is to use a sequence of precomputed short and exact geodesic paths to approximate the geodesic path between any two given points on a given base surface. Specifically, in a processing step, we generate a set of evenly distributed sample points on the base surface and compute the exact geodesic distance from each sample point to those in its neighborhood of some specified range. This produces a proximity graph of all the sample points. In the query step, when two query points s and t are given, we add s and t to the set of sample points to generate an augmented proximity graph and then compute the shortest path from s to t on this augmented graph to approximate the exact geodesic between the two points, as illustrated in Fig. 1.

Suppose that r is the radius of the maximum empty circle, which can be used to characterize the density of a given sample set. The proximity of a sample point p_i is defined by the set of the sample points inside a geodesic disk of radius $R > 0$ centered at

* Corresponding author.
E-mail address: csmqq@163.com (S. Chen).

https://doi.org/10.1016/j.cad.2018.04.021
0010-4485/

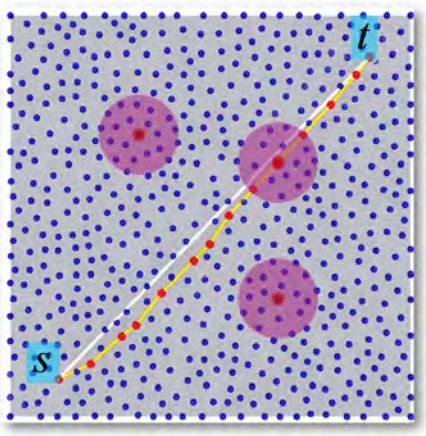


Fig. 1. Here a proximity graph is defined on 600 sample points with each sample being connected to 19 nearest neighboring samples. Then the geodesic path from s to t is approximated by the shortest path on the proximity graph. The relative error of the approximate path is 0.28%.

the p_i. When $R \geq 2r$, the proximity graph is connected and the union of the radius-R geodesic disks centered at all the samples form a coverage of the base surface. (Otherwise, let q be a point that is not covered by the set of disks. Considering the empty circle centered at q will lead to a contradiction to $R > 2r$.) Alternatively, for a given model, the density of the samples can be controlled by the total number of the samples, assuming an even distribution. Similarly, in lieu of R, for each sample one may set the number of the neighboring samples to include in its proximity to define the proximity graph.

Our method outperforms the fast marching method (FMM) in both time and accuracy. For example, on a 100k-face Bunny model, we use 5000 sample points and define the proximity of each sample to be the set of its 18 nearest neighbors. The query time of our method is 5 times faster than FMM and our approximation is more accurate—the average relative error and the worst relative error of our results are 0.9% and 3.2%, respectively, while the average relative error and the worst relative error of FMM are 1.3% and 56.7%, respectively. Admittedly, this significant improvement in efficiency and accuracy does not come free. While FMM does not need preprocessing, our method needs to generate the sample points and their proximity graph in a preprocessing step that costs 28 s and takes about 3 MB to encode the proximity graph. This is a reasonably small space requirement for most geometric processing tasks for a mesh model of this size.

The remainder of the paper is structured as follows. In Section 3 we will discuss the preprocessing step of building a geodesic proximity graph on a given surface. In Section 4 we will present the algorithms for processing geodesic queries with the assist of the proximity graph. In Section 5 the properties and parameters of our algorithms will be analyzed. In Section 6 we will present experimental results for evaluation and comparison. Finally, the paper is concluded in Section 7.

2. Building geodesic proximity graph

Let $\mathcal{S}$ be a connected Riemannian surface equipped with a metric g. For two points $s, t \in \mathcal{S}$, there exists a shortest geodesic path connecting s and t, denoted by $\Pi(s, t)$. Let $d_g(s, t)$ denote the geodesic distance between s and t, i.e., the length of $\Pi(s, t)$. We will also call $d_g(s, t)$ the *geodesic map* when s and t are regarded as two arbitrary points on $\mathcal{S}$. Suppose that there is a set of evenly distributed sample points $\mathcal{P} = \{p_i\}$ on $\mathcal{S}$ with the radius of the maximum empty circle being r. For each sample point p_i, we define a geodesic disk $\{\mathcal{D}(p_i, R) \mid p_i \in \mathcal{P}\}$ centered at p_i with geodesic radius R. Assume $R \geq 2r$. Then the union of geodesic disks, $\bigcup_{p_i \in \mathcal{P}} \mathcal{D}(p_i, R)$, form a coverage of $\mathcal{S}$, that is, every point of $\mathcal{S}$ belongs to some geodesic disk $\mathcal{D}(p_i, R)$. Two distinct sample points p_i and p_j are said to *proximate* to each other if $p_j \in \mathcal{D}(p_i, R)$, or equivalently, $p_i \in \mathcal{D}(p_j, R)$. Let $\mathcal{E}$ denote all the pairs of mutually proximate samples. Then the geodesic map $d_g(s, t)$ is approximately encoded by the *proximity graph* $\mathcal{G} = \{\mathcal{P}, \mathcal{E}\}$.

To compute the geodesic distance between any two points $s, t \in \mathcal{S}$, we propose to approximate the geodesic path $\Pi(s, t)$ on $\mathcal{S}$ by the shortest path on the proximity graph $\mathcal{G}$, i.e.

$$s \to p_1 \to p_2 \cdots \to t \tag{1}$$

where the $p_i \in \mathcal{P}$. In other words, the approximate $\Pi(s, t)$ we seek is the shortest path on the augmented proximity graph $\mathcal{G}^{s,t}$ which is obtained by adding s and t, as well as their proximate samples, to $\mathcal{G}$.

There are two stages in the implementation of our approach: (1) building the proximity graph; and (2) processing geodesic queries using the proximity graph, as shown in Fig. 2. We will present in this section the first stage of building a geodesic proximity graph on a given surface, which is a preprocessing task for our method. Building a geodesic proximity graph involves two steps. During the first step, we need to generate a set of evenly distributed sample points. Then we need to define a neighborhood for each sample and use it to compute the proximity graph on the sample points. We will elaborate on these two steps in the following, assuming that the base surface is given as a triangle mesh surface.

2.1. Sample generation

We use the farthest distance sampling method [40,6] to generate the samples on a given base surface. The algorithm requires two alternative operations: (1) updating the distance field according to the newly added sample point, and (2) retrieving a new sample point where the distance field reaches the global maximum. Furthermore, we use the fast marching method (FMM) [32] to compute the farthest distance required in the sampling process. However, when the mesh quality is so bad that it is challenging for FMM to ensure accuracy required, we switch to the slower but more reliable exact distance method [24] to ensure the uniformity of the sample points. Note that FMM is used for sampling for all the examples shown in this paper except for the cylinder model in Fig. 3 where the mesh quality is too poor for FMM to give meaningful distance estimation. The number of samples needed is a key parameter to the sampling process—it is correlated to the radius of the maximum empty disk, assuming the uniformity of the sample distribution, and thus influences significantly the accuracy of geodesic computation of the overall method. For example, to attain the average relative error of 1%, the number of sample points required ranges from 1300 to 15,000, depending on the number of faces of the mesh surface model under consideration. We will give a more detailed analysis and recommendation on parameter selection in a later section.

Care is needed for locating a farthest point insides some triangle when using the exact method [24] in the case of poor mesh quality. Note that the exact geodesic algorithm cuts mesh edges into segments, also called *windows*, such that each segment shares the same face sequence. In Fig. 3(a), we show 3 windows that arrive at the triangle $\triangle ABC$, where (x_i, y_i) are respectively their root positions, i.e., the unfolded 2D coordinates of the last vertex on the shortest path and r_i is the distance from the source to the ith root. Then locating the farthest point amounts to solving the following

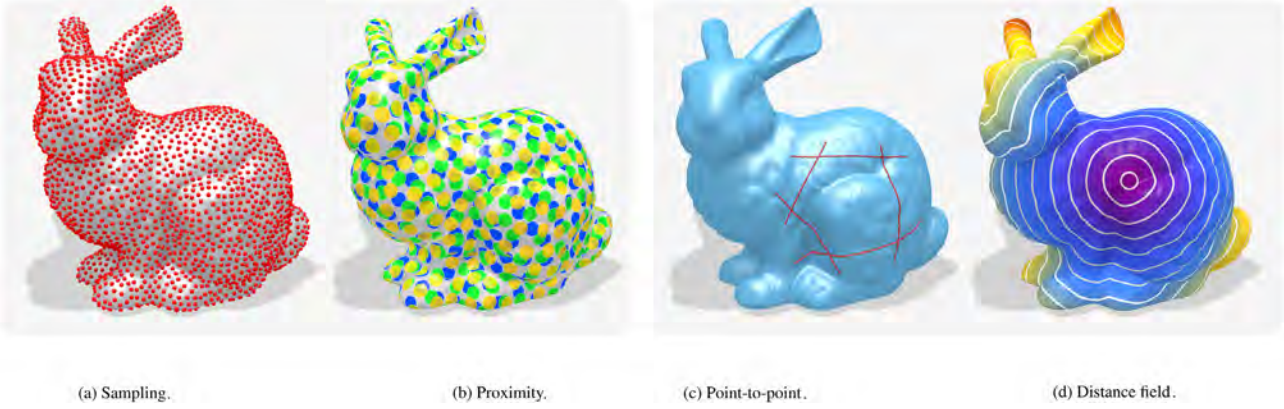
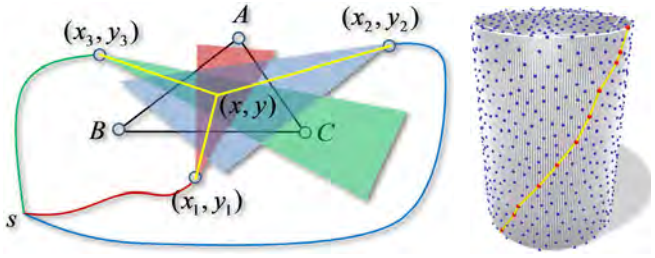


Fig. 2. Our algorithm has two stages: building the proximity graph (a–b) and processing geodesic queries using the proximity graph (c–d).



(a) Locating the farthest point inside a triangle.

(b) Sample points on a cylinder with thin and long triangle faces.

Fig. 3. Computing the farthest point using the exact geodesic algorithm on a surface with poor mesh quality.

systems of equations

$$\begin{aligned} & r_1 + \sqrt{(x-x_1)^2 + (y-y_1)^2} \\ = & r_2 + \sqrt{(x-x_2)^2 + (y-y_2)^2} \\ = & r_3 + \sqrt{(x-x_3)^2 + (y-y_3)^2} \end{aligned} \tag{2}$$

such that the straight line segment between (x, y) and (x_i, y_i) passes through the interior of the ith window's interval (the subsegment of the last edge it arrives at). This allows us to get a correct farthest point even in the presence of poor tessellation. See Fig. 3(b) for an example of using the exact geodesic methods to generate well distributed samples on a cylinder with poorly shaped (i.e. thin and long) triangle faces.

2.2. Representation of proximity graph

To define the proximity of each sample p_i, we need to specify which samples should be included in the neighborhood of p_i and compute the exact geodesic distances from p_i to these neighboring samples. Here, the exact distance from each sample to the samples in its neighborhood is computed using the exact method in [24]. The created geodesic proximity graph is saved in a file that keeps the coordinates of all the samples as well as the ID pairs of neighboring samples. In addition, we keep the geodesic distance between each pair of neighboring samples. To facilitate retrieving a geodesic path as part of a query, we also store the end direction vector of the geodesic path $\Pi(p_i, p_j)$ from p_i to p_j for each pair of neighboring samples p_i and p_j. Note that starting from p_j and backtracing along the reverse direction of $\Pi(p_i, p_j)$, we can retrieve $\Pi(p_i, p_j)$ as a polygonal line on the base surface. Thus, the information for encoding the proximity graph is stored in the following format.

```
# The number of sample points
  3800
# The coordinates of sample points
  1.2 1.1 0.8
  1.7 0.1 0.6
  ⋮
  1.4 2.3 0.7
# ID1    ID2    Distance    End_direction
  1      2      0.5534      0.15  0.83  0.5372
  1      3      0.6625      0.37  0.51  0.7765
  ⋮
  3800   3792   0.7293      0.64  0.77  0.3761
```

3. Distance query

The proximity graph computed in the preprocessing step supports two kinds of distance queries: (1) point-to-point distance query, which is to compute the geodesic distance and path between two given points on the surface; and (2) distance field computation, which is to compute the distance value from any destination point of the surface to a given source point.

3.1. Point-to-point distance query

Let $\mathcal{S}$ denote a given base surface, represented as a triangle mesh. Suppose that the proximity graph $\mathcal{G}$ of the surface has been computed. Given two points $s, t \in \mathcal{S}$, there are the following three cases when computing the geodesic distance and path between s and t:
Case 1: both of s and t are sample points;
Case 2: one of s and t is a sample point;
Case 3: neither of s and t is a sample point.

Next we will just consider Case 3 since the other two cases are handled in a similar and simpler manner. We first use the exact distance method [24] to compute the geodesic distances from s to its neighboring samples in a geodesic disk of radius R centered at s. Then we do the same for the point t. After that, we merge s and t into the proximity graph $\mathcal{G}$ to get an augmented proximity graph, denoted $\mathcal{G}^{s,t}$. Finally, we find the shortest path from s to t on $\mathcal{G}^{s,t}$ using a meet-in-middle scheme, rather than the traditional one-way sweeping process.

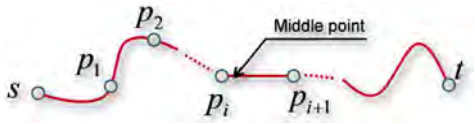
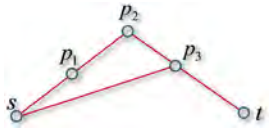


Fig. 4. Proof to Theorem 3.1.

It is known that Dijkstra's algorithm or its variant [41] can be used to find a shortest path easily. But here we propose a more efficient approach to make this step faster. The idea is simple: let two wavefronts propagate from s and t simultaneously and terminate when the two wavefronts meet each other. This procedure is about twice as fast as Dijkstra's algorithm because the number of samples that need to be visited by the two wavefronts is about half of that by the single wavefront used by Dijkstra's algorithm, which has to run from s to t in a one-way style.



The termination condition of our two-wavefront procedure needs to be established carefully to ensure correctness. In fact, the first sample point covered by both wavefronts is not necessarily located on the shortest path $\widetilde{\Pi}_g(s,t)$-such a counterexample is illustrated in the right figure. Suppose that $\|sp_2\| = \|tp_2\|$ and two worms move from s and t respectively at the same speed V. After a time period of $\|sp_2\|/V$, two worms reach p_2 simultaneously, and thus p_2 is the first sample point that can be commonly swept by both of the two wavefronts (p_3 is visited by only one worm at this moment). However, $\|sp_2\| + \|tp_2\|$ is not the shortest distance between s and t. Instead, p_3 is the sample point that defines a shorter path $\|sp_3\| + \|tp_3\|$, which can be achieved at the time point of $\|sp_3\|/V$.

It can be shown that the correct path $\widetilde{d}_g(s,t)$ is found when either wavefront exceeds $(\widetilde{d}'_g(s,t) + R)/2$, where $\widetilde{d}'_g(s,t)$ is the up-to-date optimal path that is not shorter than the real estimate $\widetilde{d}_g(s,t)$. The correctness of this condition follows from the following theorem (see Fig. 4).

Theorem 3.1. *Let $\widetilde{\Pi}_g(s,t) : s \to p_1 \to \cdots \to p_i \cdots \to t$ be the shortest path on $\mathcal{G}^{s,t}$. Denote its length by $\widetilde{d}_g(s,t)$. Then there exists a link point p_i on the path such that*

$$\max_{p_i \in \widetilde{\Pi}_g(s,t)} (\widetilde{d}_g(s,p_i), \widetilde{d}_g(p_i,t)) \le \frac{\widetilde{d}_g(s,t) + R}{2}. \tag{3}$$

Proof. Clearly, there exist two successive sample points, say, p_j and p_{j+1}, on $\widetilde{\Pi}_g(s,t)$ such that $\widetilde{d}_g(s,p_j) \le \widetilde{d}_g(s,t)/2$ while $\widetilde{d}_g(s,p_{j+1}) \ge \widetilde{d}_g(s,t)/2$. Considering each geodesic segment is at most R, we have either $\widetilde{d}_g(s,t)/2 - \widetilde{d}_g(s,p_j) \le R/2$ or $\widetilde{d}_g(s,p_{j+1}) - \widetilde{d}_g(s,t)/2 \le R/2$ holds. Taking $i = j$ for the former case and $i = j+1$ for the latter yields the proof. □

The pseudo-code for this meet-in-middle procedure is shown in Algorithm 1.

Computing geodesic path. Often one needs to compute the geodesic path $\widetilde{\Pi}(s,t)$ between the two query points s and t in addition to their geodesic distance. After computing the geodesic distance from s to t by running the above shortest path algorithm, we may retrieve the geodesic path by unfolding the face sequence

Algorithm 1 Querying the distance between two points s and t.

Input: A surface $\mathcal{S}$ and $s, t \in \mathcal{S}$;
A sweep radius R;
A proximity graph $\mathcal{G}$.
Output: The distance estimate $\widetilde{d}_g(s,t)$.

```
if s is NOT a sample point then
    Link s to the graph G by finding its proximate sample points;
end if
if t is NOT a sample point then
    Link t to the graph G similarly;
end if
Create an empty priority queue Q and push two triples (s, 0, s), (t, 0, t) into Q,
where the first component denotes which point is touched by the wavefront, the
second component denotes the up-to-date shortest distance of the wavefront
arriving at the point of interest, and the last one denotes which endpoint (s or
t) provides the distance;
Set a distance pair (d̃(s,p), d̃(p,t)) at each sample point p, where d̃(s,p) (resp.
d̃(p,t)) denotes the up-to-date shortest distance from s (resp. t) to p. Its initial
value is set to be(∞, ∞);
Set d_s^t, the up-to-date optimal distance between s and t, to be ∞;
while Q is not empty do
    Take out the top-priority element (p, d, q);
    if d ≥ (d_s^t + R)/2 then
        Break;
    end if
    if q = s then
        if d < d̃(s,p) then
            d̃(s,p) := d
            if d(s,p) + d̃(p,t) < d_s^t then
                update d_s^t by d̃(s,p) + d̃(p,t);
            end if
            for all p' is proximate to p do
                if d + d_g(p,p') is less than p''s up-to-date distance d̃_s(p') then
                    Push the triple (p', d + d_g(p,p'), s) into Q;
                end if
            end for
        end if
    else
        //q = t otherwise
        Handle the triple (p, d, q) in the same way;
    end if
end while
Let d̃_g(s,t) := d_s^t.
```

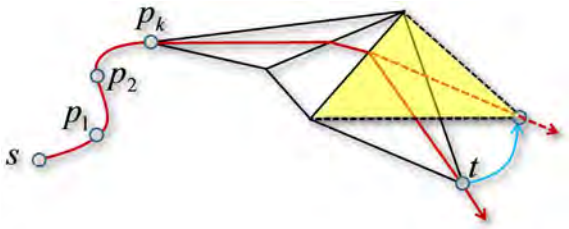


Fig. 5. Backtracing a shortest path $s \to p_1 \to p_2 \cdots \to t$.

reversely from the ending point t. Suppose that we have found the sequence of the sample points on the approximate geodesic path, denoted $s \to p_1 \to p_2 \cdots \to p_k \to t$. Our next task is to compute the polygonal line connecting consecutive samples in this sequence. Recall that the ending direction vector of the path from p_k to t is kept, so we may use this direction to trace the polygonal line from t back to p_k, as shown in Fig. 5. Similarly, we can trace similarly from p_k back to p_{k-1}, and so on. Joining all these polygonal lines together yields the final approximate geodesic path from s to t, as well as the approximate geodesic distance.

3.2. Distance field computation

Next we come to discuss how to compute a distance field with respect to a source point s on a mesh surface $\mathcal{S}$ with the help of the proximity graph $\mathcal{G}$. For a base surface given as a triangle mesh,

the distance field computation problem is specifically formulated as computing the distances from the source point, which is not necessarily a mesh vertex or a sample point, to all the mesh vertices of the base surface [33,36]. The computation of a geodesic distance field takes three steps:

1. Compute the distances from s to its k-nearest samples within a geodesic disk of radius R and use this information to augment the proximity graph $\mathcal{G}$ into a proximity graph $\mathcal{G}^s$ that also contains the source point s;
2. Compute the distance $\widetilde{d}_g(s, p)$ from s to each sample point p on the proximity graph $\mathcal{G}^s$;
3. Finally, for each triangle vertex v, we search for all its neighboring sample points within the radius-R geodesic disk centered at p_i and choose the sample t that gives the shortest total length of the path $\Pi(s, p_i) \cup \Pi(p_i, v)$ among these neighboring samples of v.

Our tests show that the last step is the most time-consuming among the three steps. A better technique for the last step is to update the distance values at vertices using those at the sample points since it can be shown that $\cup_{p_i \in \mathcal{P}} \mathcal{D}(p_i, R)$ and $\cup_{v \in V \cup \mathcal{P}} \mathcal{D}(v, R)$ accommodate the same set of shortest paths if we require the link point to be a sample point. Thus the last step can be further converted into a multi-source geodesic distance field problem ($s \cup \mathcal{P}$ serves as the ancestor sources). Furthermore, the implementation can be further improved by requiring that the increment of geodesic distance between each vertex and its ancestor be not more than R. The final approximate geodesic paths encoded by the distance field are equivalent to those computed directly by Algorithm 1 . See more details in Section 5.2.

4. Analysis

In this section we shall present the properties of our algorithm and study how R and r affect time/memory costs and accuracy.

4.1. Properties

First, we shall point out that the approximate geodesic distances computed by our method induces a metric.

Theorem 4.1. *Approximate geodesic distances computed using the proximity graph induce a metric.*

Proof. *Positivity*: $\widetilde{d}_g(s, s) = 0$ and $\widetilde{d}_g(s, t) > 0, \forall s \neq t$.

Reflexivity: Suppose that $s \to p_1 \to p_2 \cdots \to p_k \to t$ is the shortest path connecting s and t on the proximity graph $\mathcal{G}$. Then the path $t \to p_k \to p_{k-1} \cdots \to p_1 \to s$, which has the same length, can be also found in $\mathcal{G}^{s,t}$. Hence, $\widetilde{d}_g(s, t) = \widetilde{d}(t, s)$ holds for each pair of points s and t.

Triangle inequality: For three arbitrary points s, x, t, the composite path $\widetilde{\Pi}(s, x) \cup \widetilde{\Pi}(x, t)$ is an approximate geodesic path that is $\widetilde{d}_g(s, x) + \widetilde{d}_g(x, t)$ in length. This implies that the computed approximate path $\widetilde{\Pi}(s, t)$ cannot be longer, that is,

$$\widetilde{d}_g(s, t) \leq \widetilde{d}_g(s, x) + \widetilde{d}_g(x, t). \tag{4}$$

Since the positivity, reflexivity and the triangle inequality are all satisfied, we finish the proof. □

We further point out that the worst-case relative error of an approximate geodesic path is bounded, and the bound is given by r and R.

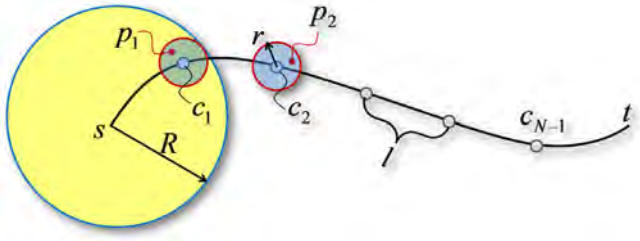


Fig. 6. Proof to Theorem 4.2. In order to construct an approximate path, we divide the geodesic path $\Pi(s, t)$ evenly into $N \triangleq \lceil \frac{d_g(s,t)}{R-2r} \rceil$ segments, and each segment is $l \triangleq \frac{d_g(s,t)}{N}$ in length. Here c_i is a dividing point and p_i is a sample point inside $\mathcal{D}(c_i, r)$, $1 \leq i \leq N-1$.

Theorem 4.2. *Let $L \triangleq d_g(s, t)$ be the length of the exact geodesic path between two points s, t, and $\widetilde{L} = \widetilde{d}_g(s, t)$ be our estimate. We have*

$$0 \leq \frac{\widetilde{L} - L}{L} \leq \frac{2r}{R - 2r}. \tag{5}$$

Proof. We divide the geodesic path $\Pi(s, t)$ evenly into $N \triangleq \lceil \frac{L}{R-2r} \rceil$ segments such that each segment is $l = L/N \leq R - 2r$ in length as Fig. 6 shows. We use $c_1, c_2, \ldots, c_{N-1}$ to denote the link sample points between successive segments. Since r is the radius of the maximum empty circle, there is at least one sample point, say, $p_i \in \mathcal{P}$, inside each geodesic disk $\mathcal{D}(c_i, r)$. From $l \leq R-2r$ it follows that

$$\begin{aligned} &d_g(s, p_1) \leq l + r \leq R - r, \\ &d_g(p_i, p_{i+1}) \leq l + 2r \leq R, \\ &d_g(p_{N-1}, t) \leq l + r \leq R - r. \end{aligned} \tag{6}$$

Therefore can construct an approximate geodesic path $s \to p_1 \to p_2 \cdots \to t$ that is not shorter than the real estimate $\widetilde{L}$, where $i = 1, 2, \ldots, N-2$. Summing up the above N inequalities yields

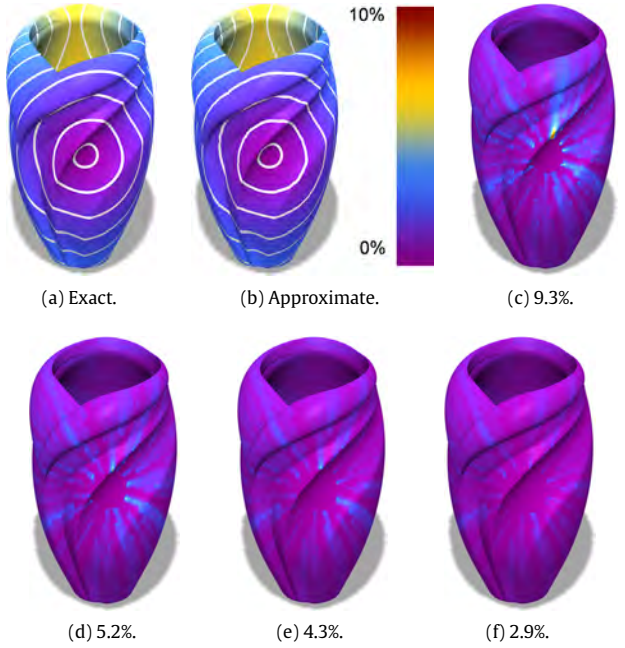
$$\begin{aligned} \widetilde{L} &\leq NR - 2r \\ &\leq \underbrace{N(R - 2r)}_{L} + 2(N-1)r \\ &\leq L + \frac{L}{R - 2r} 2r = \frac{R}{R - 2r} L. \end{aligned}$$

Together with $L \leq \widetilde{L}$, the proof is completed. □

Numerical tests show that the worst relative error produced by our approximation scheme is always much smaller than this bound. Moreover, it is observed that the worst relative error always occurs when the distance between the source s and the destination t is slightly larger than the radius R of the geodesic disk for defining proximity. In addition, enlarging the proximity of both source and destination points should help to decrease the worst relative error. See details in Section 4.3.

4.2. Parameters $n_{\mathcal{P}}$ and $m_{\mathcal{P}}$ vs. R and r

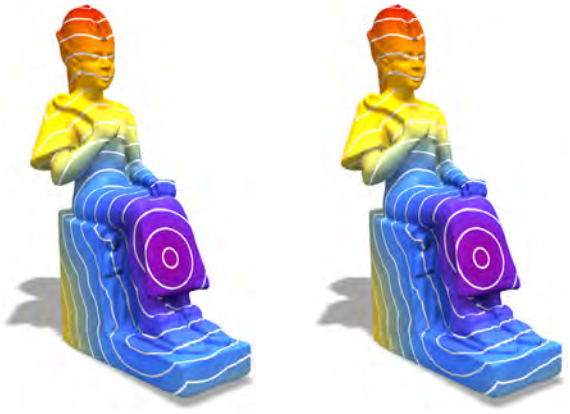
Because of the uniformity of sample distribution, instead of using the radius r of the maximum empty circle, we may use the total number of samples on a surface, denoted $n_{\mathcal{P}}$, to reflect the density of the sample points. The real relationship between $n_{\mathcal{P}}$ and r is complicated—$n_{\mathcal{P}}$ also depends on shape variations. Similarly, in lieu of using the radius R of a geodesic disk, we may define the neighborhood (i.e. proximity) of each sample p_i by including its k-nearest samples of p_i, where $k \triangleq m_{\mathcal{P}}$. Hence, $m_{\mathcal{P}}$ can be an alternative parameter for defining proximity. We will next discuss the selection of the optimal values of $n_{\mathcal{P}}$ and $m_{\mathcal{P}}$.



(a) Exact. (b) Approximate. (c) 9.3%.

(d) 5.2%. (e) 4.3%. (f) 2.9%.

Fig. 7. Enlarging the proximity of the query points to reduce the worst-case relative error. Given a 100k-face Vase model, we compute an approximate solution, shown in (b), that has a 0.82% average relative error compared to the exact distance field, shown in (a). Here $n_{\mathcal{P}} = 5000$ and $m_{\mathcal{P}} = 18$. The worst-case relative error is 9.3%. The error map is visualized in (c). By enlarging proximity of the source and destination points to 24, 30, 36 respectively, the worst relative error is reduced to 5.2% (d), 4.3% (e), and 2.9% (f), respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Exact. (b) Approximate.

Fig. 8. Comparison of an exact geodesic field in (a) and our approximate geodesic field in (b) with the same source point. Here $n_{\mathcal{P}} = 5000$, $m_{\mathcal{P}} = 18$, $m_{\mathcal{P}}^{s,t} = 36$. For our method, the average relative error is 0.9% and the worst relative error is 3.2%.

4.3. Proximity setting of the source/destination points

Experimental results suggest that the worst-case relative error usually occurs when two points are in a distance slightly larger than the proximity radius R. Consider the geodesic field shown in Fig. 7 that is computed by our method. Here $n_{\mathcal{P}} = 5000$ and $m_{\mathcal{P}} = 18$, the average relative error of all the geodesic distances is 0.82%, but the worst-case relative error amounts to 9.3%, which happens when the destination point has a distance of about R to the source point s. (Note that the theoretical error bound given by Theorem 4.2 is $r/R = 26\%$ for this example.) If we increase the proximity of the source and destination points to include 24, 30, 36 samples, respectively. Then the worst relative error will be decreased to 5.2%, 4.3% and 2.9%, respectively (see the color-coded difference field in Fig. 7(c)–(f)).

However, increasing the proximity of the endpoints s, t requires a higher time cost for finding the samples proximate to s and t. Hence, there is a tradeoff between this extra time required for enlarging the proximity of the query points s and t and benefit of approximation accuracy improvement. Based on numerous experiments, we use the empirical rule of setting the proximity of the query points s and t to be twice as large as that for a general sample point.

4.4. Parameter selection of $n_{\mathcal{P}}$ and $m_{\mathcal{P}}$

For the 100k-face Ramses model in Fig. 8, we ran a large number of tests with $n_{\mathcal{P}}$ varying from 3000 to 6000, $m_{\mathcal{P}}$ from 14 to 28 to identify the parameter values that lead to the fastest query speed subject to the user's accuracy requirement. Here, the average relative error is required to be less than 1%, and the worst-case relative error is required to be less than 3.4%. With these assumptions, we found the best parameter values to be $n_{\mathcal{P}} = 5000$, $m_{\mathcal{P}} = 18$ for sample vertices, and $m_{\mathcal{P}}^{s,t} = 36$ for the two query points s and t. With this setting of parameters, the actual average/maximum relative errors observed are 0.9% and 3.2%, respectively, for the model in Fig. 8.

For models with different numbers of faces, the best parameter settings are also different. Based on extensive experimental results, we recommend the values of the parameters in Table 1 for models of different sizes (in the number of faces), and show the corresponding actual average/maximum relative errors and the query time costs. Note that the parameter setting is largely independent of the shape of a model, according to our tests. Hence, the parameter settings in Table 1 will be used in the next section when comparing our algorithm with the fast marching method [32], Saddle Vertex Graph (SVG) [35] and the heat-based method [34] in terms of query performance and accuracy for geodesic distance computation.

5. Experiments

In this section we shall discuss the time and space performance of our method, compare it with several existing methods, and talk about its robustness with surfaces of different meshing qualities. Furthermore, we will present the extension of the method to surfaces equipped with a general Riemannian metric.

The experiments were carried out on a computer with a 64-bit version of Win7 system, a 3.07 GHz Intel(R) Core(TM) i7 CPU and 6 Gb memory. The coding language is C++ supported by OpenMP. Our implementation does not require any numeric package.

5.1. Preprocessing cost

The preprocessing task includes two parts, sampling and proximity computation. We made a test on the Bunny model ranged from 50k faces to 300k faces. As is shown in Fig. 9, where $n_{\mathcal{P}} = 6000$ and $m_{\mathcal{P}} = 20$, the preprocessing time is basically linear to the number of faces, whereas the precomputed file size linearly depends on the product of $n_{\mathcal{P}}$ and $m_{\mathcal{P}}$. Fig. 9 also shows that the preprocessing step is considerably fast and the precomputed file is portable and independent of the model complexity, generally between 2 and 6 megabytes. Since the maximum circle radius decreases monotonically with the progress of the farthest point sampling, it takes much less time in distance computation to process those samples that come later. Hence, the sampling time remains nearly unchanged even if $n_{\mathcal{P}}$ doubles.

Table 1
Parameter settings that lead to a relative error of 0.9% and a worst-case relative error of 3.2%. It outperforms the fast marching method and the heat based method whether in performance or accuracy. Compared with SVG (their parameter K is set to 100), our algorithm has a performance advantage in the same accuracy level.

#Faces		10k	20k	50k	100k	200k	300k
	Configuration	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$	$n_{\mathcal{P}}$ $m_{\mathcal{P}}$ $m^{s,t}_{\mathcal{P}}$
		1300 22 44	1500 21 42	2300 18 36	5000 18 36	10 000 18 36	15 000 18 36
Ours	Preprocessing time	0.6 s	1.4 s	4.8 s	21.5 s	80.3 s	179.7 s
	Average error	0.8%	0.8%	0.9%	0.9%	0.9%	0.9%
	Worst error	3.3%	3.3%	3.2%	3.2%	3.2%	3.2%
	Query time	4.6 ms	5.9 ms	11.5 ms	21.2 ms	40.5 ms	78.2 ms
SVG	Preprocessing time	9.7 s	26.3 s	58.9 s	155.8 s	405.6 s	829.2 s
	Average error	0.8%	0.9%	0.9%	0.9%	0.9%	0.9%
	Worst error	10.1%	8.5%	6.7%	6.3%	6.1%	5.8%
	Query time	8.2 ms	20.5 ms	132.4 ms	334.6 ms	1107.7 ms	2290.3 ms
Dijkstra	Average error	6.38%	5.27%	5.16%	5.01%	4.43%	4.28%
	Worst error	34.3%	27.7%	31.8%	26.3%	25.3%	24.7%
	Query time	6.3 ms	10.9 ms	37.3 ms	69.5 ms	158.6 ms	251.6 ms
FMM	Average error	3.14%	2%	1.5%	1.3%	1.04%	0.88%
	Worst error	27.5%	28.3%	24.8%	21.7%	20.2%	21.2%
	Query time	10.0 ms	18.9 ms	64.8 ms	112.3 ms	259.6 ms	426.5 ms
Heat	Average error	48.9%	42.5%	58.7%	56.5%		
	Worst error	68.8%	52.5%	71.1%	70.6%	Out of memory during the preprocessing stage	
	Query time	11.0 ms	23.8 ms	54.1 ms	117.0 ms		

5.2. Comparison with existing methods

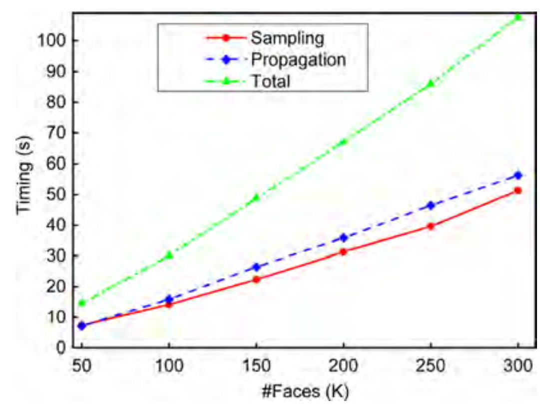
We first compare our algorithm with four state of the art methods on the Ramses model with various resolutions, i.e. Dijkstra's algorithm, the fast marching method, Saddle Vertex Graph (SVG) and geodesic in heat. Timing data and relative errors of the two-point query are available in Table 1. As shown in the table, our method significantly outperforms most of the existing methods. For instance, for a model with 10k faces, our two-point query algorithm runs twice faster and is much more accurate than the fast marching method; and, for a model with 100k faces, our algorithm is five times faster and still more accurate than the fast marching method. Compared with SVG where their parameter K is set to 100, our algorithm has an advantage of query speed and preprocessing cost in the same accuracy level.

Again, taking the Ramses model for test, we compare our method with the other four in computing a distance field and show the comparison in the two plots of timing costs and average relative errors in Fig. 11. The time for computing a distance field using our algorithm is approximately linear to the number of mesh faces. Although our method is slightly slower than the other approaches, it is more accurate. See also Table 1 for comparison. (Note that the heat-based method runs out of memory during the preprocessing stage for models of more than 160k faces.) A visual accuracy comparison of all these methods is shown in Fig. 10.

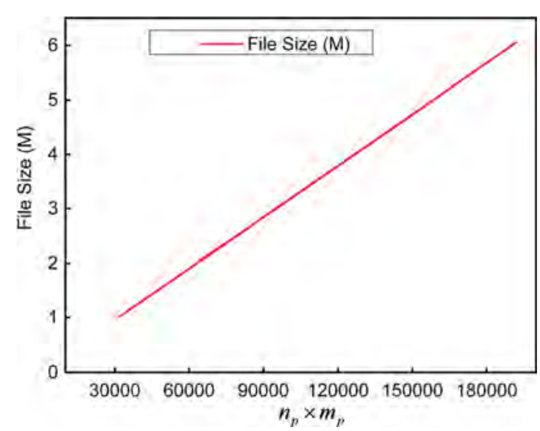
In the following, we compare our algorithm with the fast marching method, as well as some recently proposed geodesic algorithms, based on theoretical analysis or empirical statistics.

Dijkstra's algorithm. It was proposed in 1959 by Dijkstra and becomes a textbook algorithm that is widely used for computing shortest paths on graphs. It has many variant versions. For example, by replacing the priority queue into buckets, Dijkstra's algorithm can be boosted [41]. However, Dijkstra's algorithm, as well as its variants, is not intrinsic to the geometry—the approximation accuracy seriously depends on mesh triangulations and thus cannot meet the requirements for most geometry processing occasions.

Fast marching method (FMM) [32]. FMM is originally proposed as an efficient numerical method for solving the boundary value problems of the Eikonal equation on regular grids. It is extended to computing geodesic distance on triangle meshes by Kimmel and Sethian [32]. Due to its superior efficiency, it has widely been used in the computer graphics. It requires a fine and dense triangulation to maintain acceptable approximation accuracy. However, the



(a) Timing.



(b) Precomputed file size.

Fig. 9. The preprocessing time on the Bunny model with 50k to 300k faces: (a) the preprocessing time when $n_{\mathcal{P}} = 6000$ and $m_{\mathcal{P}} = 20$, and (b) the precomputed file size w.r.t. $n_{\mathcal{P}} \times m_{\mathcal{P}}$. Note that the precomputed file size is independent of the model complexity.

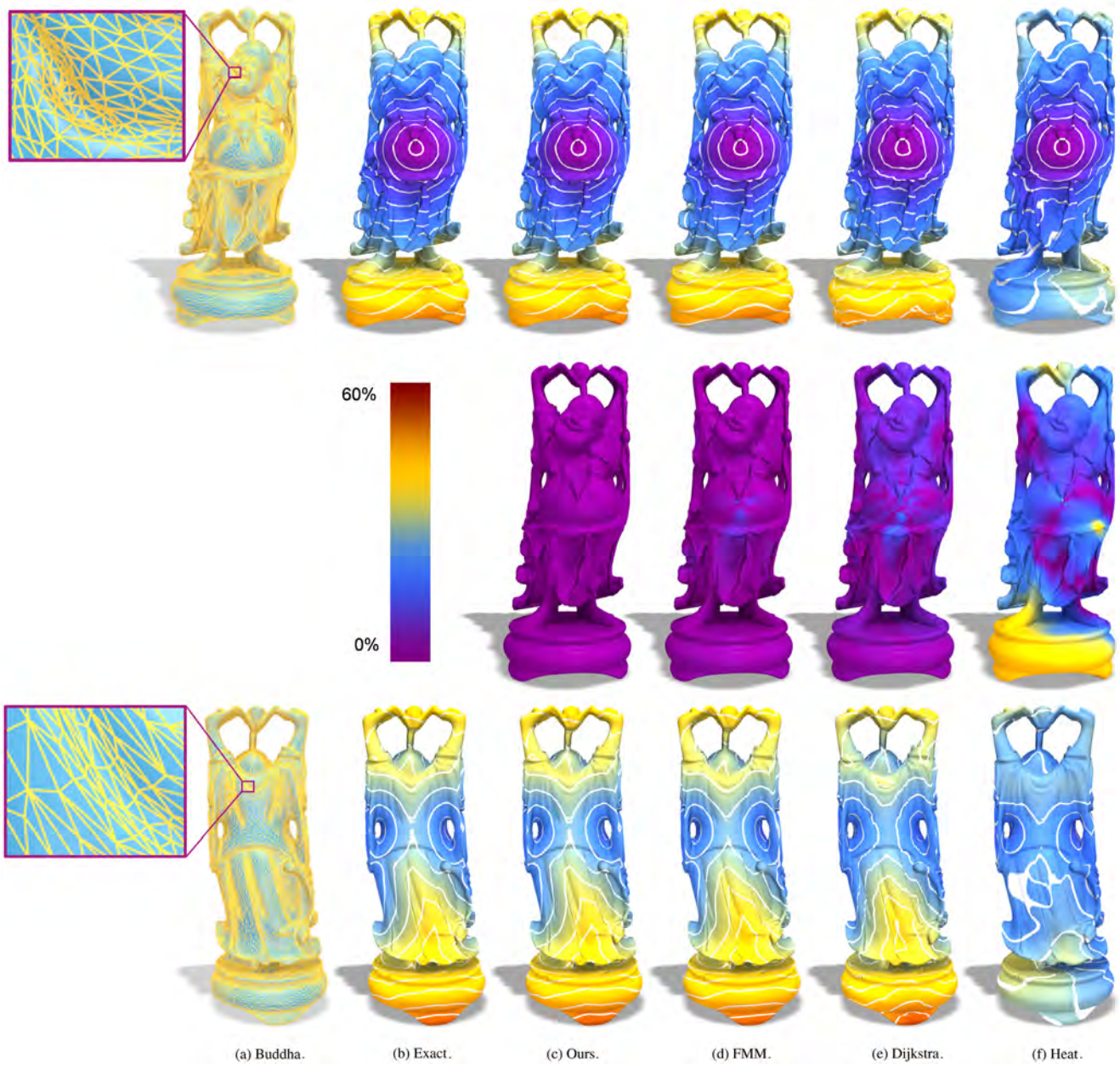


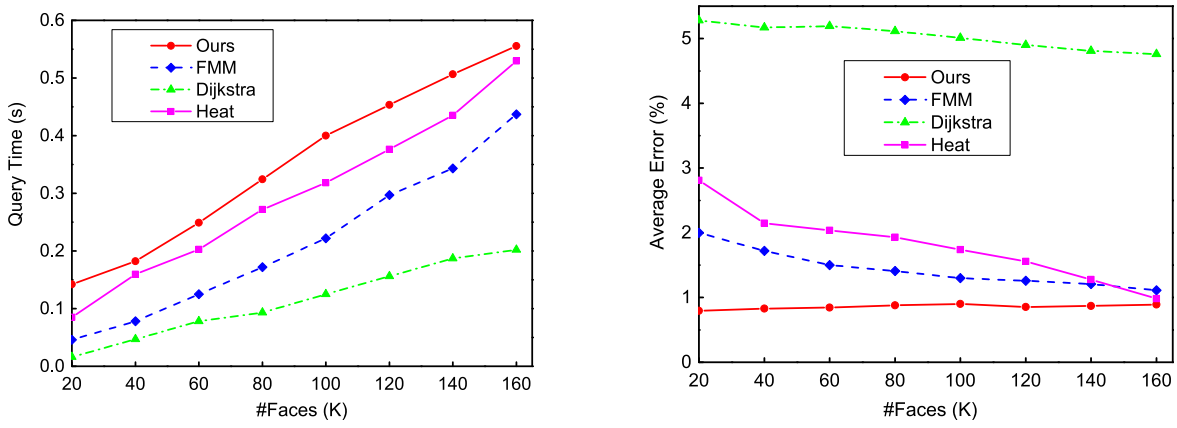
Fig. 10. Computing a geodesic distance field on the Buddha model with 100k faces, with front and back views. The color-coded difference fields are shown in the middle row. (a) Input surface; (b) Exact method; (c) Our approximate method (0.82% average, 3.4% worst-case); (d) the fast marching method (1.18% average, 21.6% worst-case); (e) Dijkstra's algorithm (5.5% average, 26.4% worst-case); and (f) the heat-based method (35.8% average, 55.9% worst-case). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

worst-case relative error of FMM can be very large in short ranges even with reasonable mesh quality.

Geodesic in heat [34]. This algorithm has a two-stage scheme, and each stage requires solving a standard linear elliptic problem. If the two coefficient matrices are pre-decomposed, then the algorithm runs very fast; See Table 1. Our tests show that it fails to produce a desirable estimation result for complicated models, such as the one shown in Fig. 10. When the input mesh is quite large, the preprocessing step will be out of memory due to the LDLT decomposition of a large linear system. Furthermore, its query processing time is longer than ours.

Earth mover's distance (EMD) [37]. This method is based on a smooth distances on a surface transiting from a purely spectral distance to the geodesic distance. It is based on the theory of optimal transportation on a discrete surface. In spite of its theoretical soundness, the algorithm is neither efficient nor accurate for practical applications, as compared with the other existing methods.

Saddle vertex graph (SVG) [35]. The SVG method is similar to our method in that it also builds a graph, in which the so called "saddle" vertices serve as transfer stops for the shortest paths. However, the saddle vertices are not intrinsic to surface geometry, and the number of saddle vertices may be zero or as many as $O(n)$, depending on mesh triangulation. Consequently, the algorithm cannot ensure that the accuracy of their approximate results meets a user specified tolerance. Moreover, the method requires a high preprocessing cost (in both time and memory), since it needs to pre-compute all the saddle–saddle and vertex–saddle relationships between all the "saddles" and their nearby vertices. Finally, one has to compute



(a) Running time. (b) Average error.

Fig. 11. Taking the Ramses model for test, Comparison of time and accuracy of our method with three other methods for computing distance fields. The three methods are: Dijkstra's algorithm, the fast marching method and geodesic in heat. The Ramses model is used. (a): Time vs. the number of faces; (b) Average relative error vs. the number of faces. See also Fig. 10 for a visualization of the accuracy comparison.

a very large geodesic disk to encompass the neighboring "saddle" vertices for a general vertex.

Geometry image based approach [33]. The algorithm utilizes the regular structure of geometry images and thus allows an efficient FMM based implementation on parallel architectures. However, the estimation accuracy depends on the parametrization quality and also inherits the disadvantages of FMM. It is very hard to obtain an accurate distance query result to meet the user-specified tolerance.

5.3. Robustness

We use the parameters recommended in Table 1 and compute the approximate distance fields on three versions of the Fertility model shown in Fig. 12: (a) the 50k-face model with a regular triangulation, (b) the 20k-face model with a regular triangulation, and (c) the 5k-face model with poor triangulation. With Lo values [42]

$$G(\triangle ABC) = \frac{S_{\triangle ABC}}{|AB|^2 + |BC|^2 + |CA|^2},$$

being the triangle quality measurement, the Lo statistics of the three models are respectively (a) $G_{\text{average}} = 0.138$, (b) $G_{\text{average}} = 0.107$, and (c) $G_{\text{average}} = 0.065$. The first column shows the exact geodesic distance fields, the middle column shows our estimated geodesic distance fields, and the right column shows the differences in a color coding. From top to bottom, the average relative errors are respectively 0.85%,0.92% and 0.93%, while the maximum errors are respectively 3.11%, 3.26% and 3.34%. Note that we used the exact geodesic algorithm for sampling for the 5k-face model, due to poor mesh quality. This test indicates that our method still produces accurate approximation for models with poor mesh quality.

5.4. Computing geodesics on Riemannian surfaces

The Riemannian metric on a manifold surface $(\mathcal{S}, g)$ is induced from an inner product $\langle \cdot, \cdot \rangle$ defined on the tangent space of the surface. It requires $g(X, X) > 0$ for all tangent vectors $X \neq 0$. In the discrete setting we enforce a simple transformation matrix for each face, according to the dual basis of the cotangent bundle, to reset the new edge lengths [7]. Note that the intrinsic geodesic algorithm only depends on edge lengths, rather than vertex coordinates. After the edge lengths are updated, our algorithm can be applied without any modification. In Fig. 13, the left figure illustrates unit disks under the Riemannian metric and the right figure shows the Riemannian geodesic distance field computed by our algorithm.

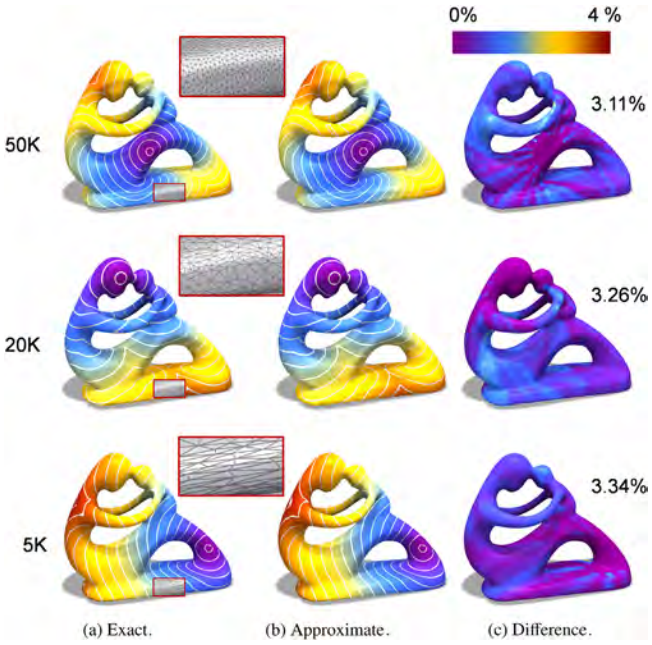


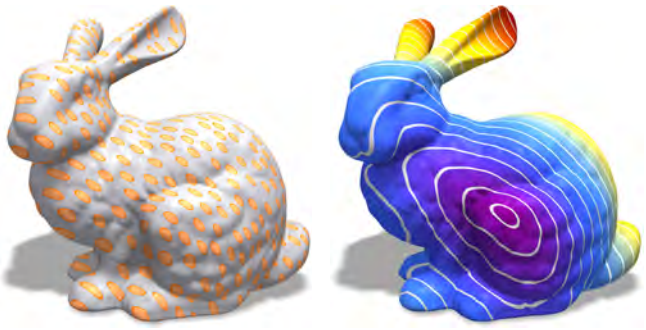
Fig. 12. Our algorithm, using the parameters in Table 1, achieves a high accuracy on models with different mesh quality (the Lo values [42] are respectively (a) $G_{\text{average}} = 0.138$, (b) $G_{\text{average}} = 0.107$, and (c) $G_{\text{average}} = 0.065$). From top to bottom, the average relative errors are respectively 0.85%,0.92% and 0.93%, while the maximum errors are respectively 3.11%, 3.26% and 3.34%.

6. Applications

In this section, we show two applications that may benefit from the superior performance of our method: (1) real-time computation of discrete exponential maps, and (2) interactive design of spline curves on surfaces.

6.1. Discrete exponential map

Given a smooth surface, the exponential map at a point of the surface establishes a diffeomorphism between a neighborhood of the point and its tangent space. The geodesic distance/direction induces an "as-rigid-as-possible" local parametrization in a neighborhood of a given point s. However, experiments show that

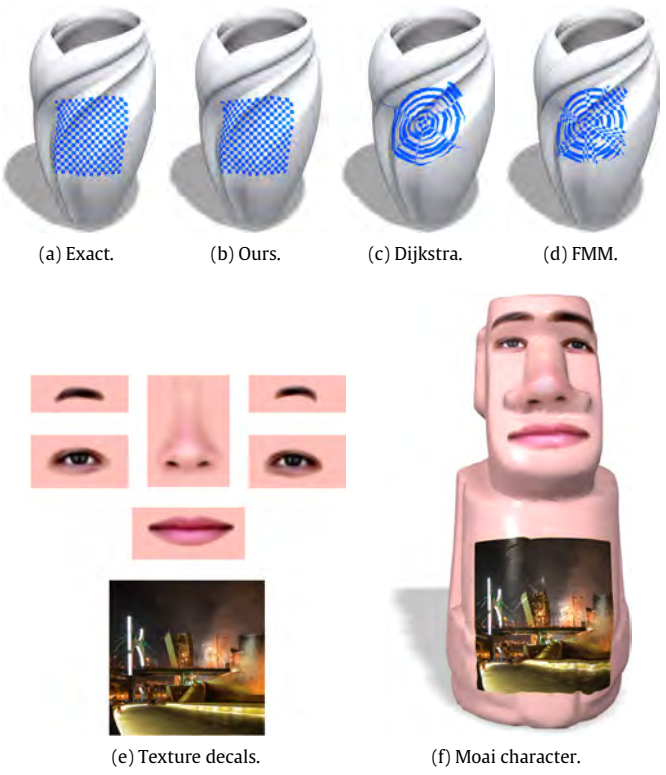


(a) Riemannian metric. (b) Geodesic distance field.

Fig. 13. Our algorithm can be applied to query the distance field in the sense of Riemannian metric.

it is non-trivial to accurately compute such parametrization in a reasonably large neighborhood of s. The reason is twofold—geometric variation that tampers the local diffeomorphism and errors of numerical computation. Schmidt et al. [43] developed an interactive method for texturing surfaces using decals using approximate discrete exponential map computed with Dijkstra's algorithm. Although it is easy to implement, its inaccuracy of path directions usually leads to conspicuous artifacts for surfaces with rich geometric details.

Consider the test example of a 100k-face Vase model in Fig. 14(a)–(d). Here we show the results computed using the exact geodesic algorithm, our approximate method ($n_{\mathcal{P}} = 5000$, $m_{\mathcal{P}} = 18$, $m_{\mathcal{P}}^{s,t} = 36$), Dijkstra's algorithm, and the fast marching method, respectively, to compute the discrete exponential map. Dijkstra's algorithm and the fast marching method result in a large distortion because the large numerical errors in path direction. In contrast, our algorithm for computing a geodesic distance field produces a more accurate exponential map. Here, the distance and direction information is converted into texture coordinates. In Fig. 14(e)–(f), we use our method to texture the Moai model with human eyes, eyebrows, mouths and the Bilbao building.



(a) Exact. (b) Ours. (c) Dijkstra. (d) FMM.

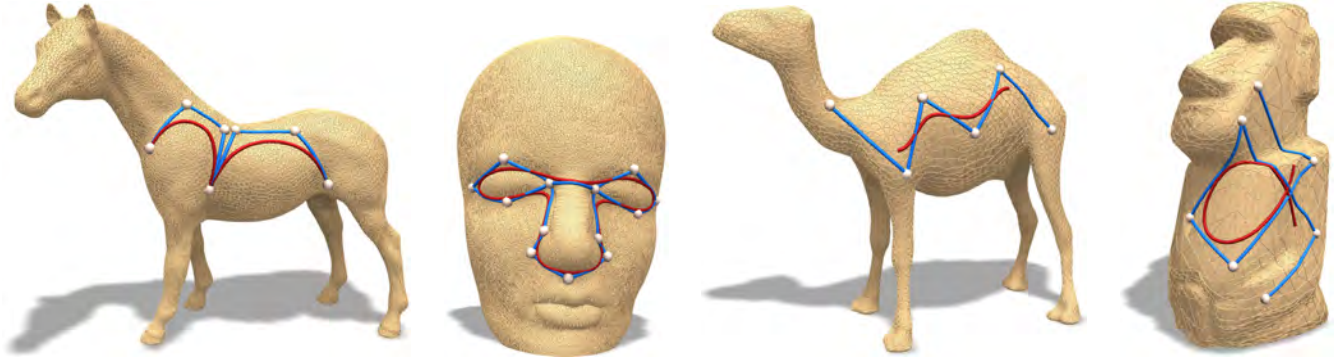
(e) Texture decals. (f) Moai character.

Fig. 14. (a)–(d) Using different geodesic algorithms to compute a discrete exponential map: Exact, ours, Dijkstra's algorithm and the fast marching method. Conventional approximation algorithms usually result in a large distortion is due to the serious deviation of path direction, while ours cannot. (e)–(f) Interactive decal composition with our algorithm.

6.2. Spline curves on surfaces

Hofer and Pottmann [44] proposed energy-minimizing splines on manifolds. Their method needs to project the intermediate spline curves onto the tangent spaces of $\mathcal{S}$, and requires a smoothing step as the postprocessing. Nava-Yazdani and Polthier [45] investigated spline curves on manifolds, Lie group or more generally symmetric space. Their major contribution is to generalize the ordinary de Casteljau algorithm to manifolds, requiring that successive control points be sufficiently close to ensure the local flow property.

In fact, the idea is helpful to interactive spline design, where computing shortest paths is the most frequent operation. Our algorithm can used for this purpose due to its superior efficiency and high accuracy. Fig. 15 shows four examples, with (a) showing some cubic Bézier curves and the others showing cubic B-spline curves. In terms of performance, taking the 40k-face Horse model as an example, we computed 100 points on the spline curve and



(a) Horse, 40k faces. (b) Face, 31k faces. (c) Camel, 18k faces. (d) Moai, 6k faces.

Fig. 15. Spline curves on surfaces: two cubic Bézier curves (a) and three cubic B-splines (b–d). The blue curves are control polygons, while the red are spline curves. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

connected consecutive points into the final result. The total computation time is about 0.6 s. Compared with the implementation in [44] that often requires several seconds, our algorithm is more suitable for interactive spline design on manifold surfaces.

7. Conclusions

We have proposed a proximity graph based approach for encoding and querying the geodesic distance map. It supports fast and accurate distance query while requiring only a lightweight preprocessing. The major difference between our approach and the existing approaches lies in that our approach is capable of controlling the relative error, rather than absolute error, which is useful in many occasions. We exhibit the uses of the new approach in two applications—real-time computation of discrete exponential map for texture mapping and interactive design of spline curves on surfaces.

Acknowledgments

We are grateful to the editors and anonymous reviewers for their insightful comments and suggestions. The models in this paper are provided courtesy of the AIM @SHAPE Shape Repository. This work is partially supported by NSF of China (61772016, 61772312, 61772318, 61572021), the key project of NSFC (61332015), NSF of ZheJiang (LY17F020018), the Open Project Program of the State Key Lab of CAD & CG (A1702), Zhejiang University and Singapore Ministry of Education Grant RG26/17.

References

[1] Bose P, Maheshwari A, Shu C, Wuhrer S. A survey of geodesic paths on 3D surfaces. Comput Geom Theory Appl 2011;44(9):486–98.
[2] Katz S, Tal A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans Graph 2003;22(3):954–61.
[3] Funkhouser T, Kazhdan M, Shilane P, Min P, Kiefer W, Tal A. et al. Modeling by Example.
[4] Sloan P-PJ, Rose III, CF, Cohen MF. Shape by example. In: Proceedings of the 2001 symposium on interactive 3D graphics. I3D '01, 2001. p. 135–43.
[5] Praun E, Hoppe H, Finkelstein A. Robust mesh watermarking. In: SIGGRAPH'99. 1999. p. 49–56.
[6] Schlömer T, Heck D, Deussen O. Farthest-point optimized point sets with maximized minimum distance. In: Proceedings of the ACM SIGGRAPH symposium on high performance graphics. HPG '11, 2011. p. 135–42.
[7] Zhong Z, Guo X, Wang W, Lévy B, Sun F, Liu Y, et al. Particle-based anisotropic surface meshing. ACM Trans Graph 2013;32(4):99:1–99:14.
[8] Rabin J, Peyré G, Cohen LD. Geodesic shape retrieval via optimal mass transport. In: Proceedings of the 11th european conference on computer vision: Part V. ECCV'10, 2010. p. 771–84.
[9] Liu Y-J, Chen Z, Tang K. Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. IEEE Trans Pattern Anal Mach Intell 2011;33(8):1502–17.
[10] Xu C, Liu Y, Sun Q, Li J, He Y. Polyline-sourced Geodesic Voronoi diagrams on triangle meshes. Comput Graph Forum 2014;33(7):161–70.
[11] Liu Y, Xu C, Yi R, Fan D, He Y. Manifold differential evolution (MDE): a global optimization method for geodesic centroidal Voronoi tessellations on meshes. ACM Trans Graph 2016;35(6):243:1–243:10.
[12] Liu Y, Xu C, Fan D, He Y. Efficient construction and simplification of Delaunay meshes. ACM Trans Graph 2015;34(6):174:1–174:13.
[13] Kimmel R, Sochen N, Weickert J, editors. Texture mapping via spherical multidimensional scaling. In: Scale space and PDE methods in computer vision. Lecture notes in computer science, vol. 3459, 2005.
[14] Zhou K, Synder J, Guo B, Shum H-Y. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on geometry processing. SGP '04, 2004. p. 45–54.
[15] Int J Comput Vis 2006;69(1):145–56.
[16] New AT, Mukhopadhyay A, Arabnia HR, Bhandarkar SM. Non-rigid shape correspondence and description using geodesic field estimate distribution. In: ACM SIGGRAPH 2012 posters. 2012. p. 96:1–96:1.
[17] Mitchell JSB. Geometric shortest paths and network optimization handbook of computational geometry. Elsevier Science Publishers B V; 2015. p. 49–119.
[18] Fletcher PT, Lu C, Pizer SM, Joshi S. Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans Med Imaging 2004;23(8):995–1005.
[19] Storer JA, Unilwslty B, Reif JH. Shortest paths in the plane with polygonal obstacles. J ACM 1994;41:982–1012.
[20] Pottmann H, Huang Q, Deng B, Schiftner A, Kilian M, Guibas L, et al. Geodesic patterns. ACM Trans Graph 2010;29(4):43:1–43:10.
[21] Mitchell JSB, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM J Comput 1987;16(4):647–68.
[22] Chen J, Han Y. Shortest paths on a polyhedron. In: Proceedings of the sixth annual symposium on computational geometry. 1990. p. 360–9.
[23] Surazhsky V, Surazhsky T, Kirsanov D, Gortler SJ, Hoppe H. Fast exact and approximate geodesics on meshes. ACM Trans Graph 2005;24(3):553–60.
[24] Xin S, Wang G-J. Improving Chen and Han's algorithm on the discrete geodesic problem. ACM Trans Graph 2009;28: 104:1–104:8.
[25] Xu C, Wang TY, Liu Y, Liu L, He Y. Fast Wavefront Propagation (FWP) for Computing Exact Geodesic Distances on Meshes. IEEE Trans Vis Comput Graph 2015;21(7):822–34.
[26] Xin S, Wang W, Chen S, Zhao J, Shu Z. Intrinsic girth function for shape processing. ACM Trans Graph 2015;35(3):1–14.
[27] Qin Y, Han X, Yu H, Yu Y, Zhang J. Fast and exact discrete geodesic computation based on triangle-oriented wavefront propagation. ACM Trans Graph 2016;35(4).
[28] Cheng S-W, Jin J. Shortest paths on polyhedral surfaces and terrains. In: Proceedings of the 46th annual ACM symposium on theory of computing. STOC '14, 2014. p. 373–82.
[29] Cheng S-W, Na H-S, Vigneron A, Wang Y. Approximate shortest paths in anisotropic regions. In: Proceedings of the 18th annual ACM-SIAM symposium on discrete algorithms. SODA '07, 2007.
[30] Chechik S, Larkin DH, Roditty L, Schoenebeck G, Tarjan RE, Williams VV. Better approximation algorithms for the graph diameter. In: Proceedings of the 25th annual ACM-SIAM symposium on discrete algorithms. SODA '14, 2014.
[31] Bernstein A. Near linear time $(1+\epsilon)$-approximation for restricted shortest paths in undirected graphs. In: Proceedings of the 23rd annual ACM-SIAM symposium on discrete algorithms. SODA '12, 2012.
[32] Kimmel R, Sethian JA. Computing geodesic paths on manifolds. Proc Natl Acad Sci 1998;95(15):8431–5.
[33] Weber O, Devir Y, Bronstein A, Bronstein M, Kimmel R. Parallel algorithms for approximation of distance maps on parametric surfaces. ACM Trans Graph (TOG) 2008;27(4):1–16.
[34] 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.
[35] Ying X, Wang X, He Y. Saddle vertex graph (SVG): a novel solution to the discrete geodesic problem. ACM Trans Graph 2013;32(6):170.
[36] Ying X, Xin S, He Y. Parallel Chen-Han (PCH) algorithm for discrete geodesics. ACM Trans Graph 2014;33(1):9:1–9:11.
[37] Solomon J, Rustamov R, Guibas L, Butscher A. Earth mover's distances on discrete surfaces. ACM Trans Graph 2014;33(4):67:1–67:12.
[38] Belyaev AG, Fayolle PA. On variational and PDE-based distance function approximations. Comput Graph Forum 2015;34(8):104–18.
[39] Campen M, Heistermann M, Kobbelt L. Practical anisotropic geodesy. Comput Graph Forum 2013;32(5):63–71.
[40] Peyré G, Péchaud M, Keriven R, Cohen LD. Geodesic methods in computer vision and graphics. Found Trends Comput Graph Vis 2010;5(3–4):197–397.
[41] Schwartz WR, Rezende PJ, Pedrini H. Faster approximations of shortest geodesic paths on polyhedra through adaptive priority queue. In: VISAPP (1). 2015. p. 371–8.
[42] Lo SH. A new mesh generation scheme for arbitrary planar domains. Internat J Numer Methods Engrg 1985;21(8):1403–26.
[43] Schmidt R, Grimm C, Wyvill B. Interactive decal compositing with discrete exponential maps. ACM Trans Graph 2006;25(3):605–13.
[44] Hofer M, Pottmann H. Energy-minimizing splines in manifolds. ACM Trans Graph 2004;23(3):284–93.
[45] Nava-Yazdani E, Polthier K. De Casteljau's algorithm on manifolds. Comput Aided Geom Design 2013;30(7):722–32.