Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 860136, 12 pages
http://dx.doi.org/10.1155/2014/860136

Research Article

Geodesics on Point Clouds

Hongchuan Yu,[1] Jian J. Zhang,[1] and Zheng Jiao[2]

[1] *National Centre for Computer Animation, Bournemouth University, Poole Bh12 5BB, UK*
[2] *School of Environmental & Chemical Engineering, Shanghai University, China*

Correspondence should be addressed to Jian J. Zhang; jzhang@bournemouth.ac.uk

Received 26 July 2013; Revised 27 February 2014; Accepted 19 March 2014; Published 27 April 2014

Academic Editor: Alexei Mailybaev

We present a novel framework to compute geodesics on implicit surfaces and point clouds. Our framework consists of three parts, particle based approximate geodesics on implicit surfaces, Cartesian grid based approximate geodesics on point clouds, and geodesic correction. The first two parts can effectively generate approximate geodesics on implicit surfaces and point clouds, respectively. By introducing the geodesic curvature flow, the third part produces smooth and accurate geodesic solutions. Differing from most of the existing methods, our algorithms can converge to a given tolerance. The presented computational framework is suitable for arbitrary implicit hypersurfaces or point clouds with high genus or high curvature.

1. Introduction

Geodesics play an important role in differential geometry. The applications range from finite element computation to computer aided geometric design, from computer animation to robotic navigation, and from brain flattening and warping in computational neuroscience to machine learning on manifolds. However, little research has been undertaken on point clouds so far, despite the fact that affordable 3D acquisition devices are becoming increasingly available. One of the main challenges is that all computation must be carried out on a set of scattered points rather than parameterized surfaces or meshes.

In this paper, we will tackle this challenge and concentrate on two scenarios: implicit surfaces and point clouds, which are the common forms used for representing point set surfaces. The presented geodesic computation framework works well for implicit surfaces and point clouds and can be easily applied to high dimensional datasets. Our framework consists of three parts, approximate geodesics on implicit surfaces, approximate geodesics on point clouds, and geodesic correction. Although point set surfaces might be given in an implicit form in many applications [1–6], it is impractical to generate an implicit surface for every point cloud.

Our work is motivated by the results presented in [7], which computes intrinsic distance functions and geodesics on implicit hypersurfaces by embedding surfaces into a Cartesian grid. The computation is performed with the well-known fast marching method [5]. This approach can handle surfaces of a high curvature, such as that shown in Figure 1. However how to generate a Cartesian grid for an implicit function remains a challenging issue, since the size of the Cartesian grids grows exponentially when they are subdivided iteratively, rendering the computing process unbearably slow. Moreover, the resulting geodesic paths lie on a grid offsetting from the surface; that is, they do not lie on the surface. To the best of our knowledge, other algorithms, such as variational curve design schemes [8–10], are unable to handle surfaces like that shown in Figure 1.

Our research is built on the work of [4], which presents a particle based approach of modelling implicit surfaces, and [11], which presents rendering point set surfaces by moving least squares. Our framework differs from the previous methods in that the geodesic computation is directly performed on the implicit surfaces or point clouds, not on the proximity mesh or grid. Our main contributions include the following.

(1) Accurate Geodesic Computation. When an approximate geodesic path is available, which may be an approximate path lying on the grid offsetting from the point set surface or a zigzag path on the implicit surface, our method can effectively reach a smooth and accurate geodesic solution on the point

Figure 1: 3D model of a roll. The geodesic curve (blue) is computed by the variational subdivision approach in [8]. It can be noted that the resulting curve does not lie on the surface.

set surface. The distinct advantage is that the precision of the geodesic solution is controllable.

(2) Arbitrary Implicit Surfaces or Point Clouds. Our algorithms can work well on arbitrary implicit surfaces (or point clouds) irrespective of genus or curvature (positive, nonpositive, high, or low).

Moreover, our presented algorithms can perform geodesic computation on both 3D and high dimensional manifolds.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related work. Our computational framework is presented in Section 3. The details of the implementation and discussions are given in Section 4. Finally, our conclusions are given in Section 5.

2. Previous Work

For a triangle mesh, the pioneering work is the MMP algorithm proposed by Mitchell et al. in [12], which provides a solution for the "single source, all destination" shortest path problem. For the worst case, the running time is $O(N^2 \log N)$. Another geodesic algorithm was presented in [13], whose time complexity is $O(N^2)$. Almost all recent work focuses on producing approximate geodesics with a guaranteed error bound. For example, additional edges are introduced into a mesh for geodesic computation [14]. Speeding up the MMP algorithm with a merging operation is also of interest [15]. A well-known algorithm is the fast marching method (FMM) [5], which computes approximate geodesics in $O(N \log N)$ time. Working on a triangular mesh, the FMM's distance functions are sometimes error-prone. The resulting geodesic path is not always accurate. Some correction methods are consequently proposed as a post procedure following the FMM, such as applying the "straightest geodesics strategy" [16] to correct the geodesic path in [17].

In contrast, there have been only a few methods relating to implicit surfaces. A parametric curve can be easily represented in an implicit form, but an implicit curve cannot always be represented parametrically in dimensions other than two or one. A practical treatment in dealing with an implicit surface is to embed it into a mesh. In [7], an implicit surface is embedded into a Cartesian grid for geodesic computation. However, the resulting geodesic paths lie on the Cartesian grid, not on the implicit surface. Witkin and Heckbert in [4] proposed an alternative method of directly sampling and controlling implicit surfaces by using a particle system. Hence, one of the objectives of this paper is to develop an approach to computing geodesics directly on an implicit surface to improve the computational accuracy.

For point clouds, several authors have presented their individual methods. Klein and Zachmann [18] approximated geodesics as shortest paths on a geometric proximity graph over point clouds, called the spheres-of-influence graphs (SIG). Ruggeri et al. in [19] further improved the accuracy of the approximate geodesics defined on the SIG. Hofer and Pottmann [8] proposed to compute geodesic curves as energy minimizing discrete curves constrained on a moving least square surface. However, a distinct defect is that these methods are usually sensitive to noise. Hence, it is hard to control the precision. Our proposed algorithms can converge to a desired solution with some specified tolerance.

3. Geodesic Computational Framework

A geodesic is calculated between two specified points on a surface. The proposed approach in this paper calculates the geodesics of all pairs of specific sample points on an implicit surface or a point cloud. Usually, this set of specific sample points is given in advance. However, for measuring an implicit surface or a point cloud, resampling the surface is a vital step. We discuss this issue on two scenarios as follows.

3.1. On Implicit Surfaces. The basic idea of our sampling method is to drive the particles to float from the sources (where the geodesics start) to the destinations (where the geodesics end, also called sinks) on an implicit surface. The resulting trajectories are used as the initial estimates of the geodesics between the sources and the destinations. In order to produce smooth geodesics on the surface, in this paper, we introduce two modifications to the W-H formulation proposed in [4]. (1) We modify the particle density model and add a curvature term. This makes particle distribution dependent on the surface curvature resulting in more optimal distribution of the particles. (2) We present a repulsion-attraction scheme. It can not only yield smooth particle trajectories but also further speed up the process. To compute geodesics, the resulting particles are to form a connection graph covering the surfaces. However, unlike a mesh, the edges of the connection graph are the trajectories of the particles, not straight lines. The nodes of the connection graph (i.e., the initial given samples) can be arbitrarily placed on the surface. To obtain a smooth and accurate solution, geodesic curvature flow will be employed as the geodesic correction procedure, which is addressed in Section 3.3.

Our algorithm consists of three steps: (1) particle based sampling on implicit surfaces, (2) Dijkstra's algorithm, and (3) geodesic correction. We first illustrate how to manipulate

particles directly on an implicit surface, followed by geodesic computation.

3.1.1. Density Model. We are concerned with only the particle position on stationary surfaces here. For simplicity, the surface deformation terms are ignored from the W-H formulation thereafter. Consider an initial collection of n floating particles randomly lying on an implicit surface S of the implicit function $F(\mathbf{p}^i(t)) = 0, 1 \le i \le n$, where $\mathbf{p}^i(t) \in R^d$ is the trajectory of the ith particle. We use boldface letters to denote vectors and matrices and italics for scalars thereafter. The particles then repel each other so as to reach equilibrium with a uniform distribution on S. The particle motion can be described as follows:

$$\mathbf{p}_t^i = \left(\mathbf{P}^i - \frac{\nabla F^i \cdot \mathbf{P}^i}{\left\|\nabla F^i\right\|^2}\nabla F^i\right) - \beta F^i \frac{\nabla F^i}{\left\|\nabla F^i\right\|^2}, \tag{1}$$

where $\mathbf{p}_t^i$ is the derivative of ith particle w.r.t. time t, representing a tangent vector at $\mathbf{p}^i$ which is orthogonal to the normal ∇F^i, $\mathbf{P}^i$ is a velocity (or estimate) of $\mathbf{p}_t^i$, and β is a constant. In our experiments, β is set to 1. The first term of the right side of (1) is viewed as the tangent component of $\mathbf{P}^i$ that pushes the particles away in the tangent plane at $\mathbf{p}^i$. This results in the particles offset to S, especially in the areas of high curvature. Hence, the second term of (1) plays a role that pulls back the offset particles towards S by using the Newton-Raphson approximation scheme. (For details, refer to [4].)

However, the original scheme of (1) has many deficiencies. The main problem is how to estimate the velocity $\mathbf{P}^i$ in (1). Usually, the original W-H method can produce a homogeneous distribution of particles on an implicit surface. But, it is difficult to give an inhomogeneous distribution based on curvature. It has been observed that the new particles are only pushed out onto flat areas, which will become too crowded, while the particles are fleeing from the high curvature areas.

3.1.2. Curvature Term. A Hessian matrix is usually used to describe the differential structure of an implicit surface S, and the eigenstructure of the Hessian matrix is used to gauge the curvature of S. For an implicit surface S in R^d, the Hessian matrix at $\mathbf{p}^i$ can be given as $\mathrm{Hess}(\mathbf{p}^i) = \nabla\nabla F^i$. It is well-known that the maximum and minimum eigenvalues of the Hessian at $\mathbf{p}^i$, respectively, correspond to the principal curvatures of S at $\mathbf{p}^i$, while the corresponding eigenvectors are, respectively, the principal directions in R^d. For an implicit surface S in R^d, the principal curvatures may be defined in a directly analogous fashion. Hence, the curvature vector of S at a point $\mathbf{p}^i$ can be described by the eigenvectors of the Hessian at $\mathbf{p}^i$ as follows:

$$\mathrm{Curv}\left(\mathbf{p}^i\right) = \sum_j^d \mu_j\left(\mathbf{p}^i\right)\mathbf{v}_j\left(\mathbf{p}^i\right), \tag{2}$$

where μ_j is the jth eigenvalue of $\mathrm{Hess}(\mathbf{p}^i)$ and $\mathbf{v}_j$ is the corresponding eigenvector. The curvature term of (2) only depends on the differential structure of S rather than others. It will be introduced into the estimate of $\mathbf{P}^i$ in (1) as an extra force.

3.1.3. Repulsion-Attraction Scheme. For point to point geodesic computation, the particles from a single point (source) are expected to be attracted into a specified destination point (sink) quickly; therefore the trajectory of the particle which reaches the sink the quickest is the closest approximation of the shortest path from the source to the sink. Our repulsion scheme is thus extended to a repulsion-attraction scheme as follows.

The energy functional of attraction for each particle $\mathbf{p}^i$ is defined as

$$E^{i*} = \lambda\sum_j^J\left(1 - \exp\left(-\left\|\mathbf{p}^{*j} - \mathbf{p}^i\right\|\right)\right), \tag{3}$$

where $\mathbf{p}^*$ denotes a sink on S, λ is a constant, and J denotes the sink number. The attraction force that each particle $\mathbf{p}^i$ receives from $\mathbf{p}^*$ is independent of each other. Minimizing E^{i*} by gradient descent with respect to the particle position $\mathbf{p}^i$ yields the desired attraction force exerted to $\mathbf{p}^i$, that is, $-E^{i*}_{\mathbf{p}^i}$. Hence, there are three forces exerted to $\mathbf{p}^i$ here, that is, the repulsion, the attraction forces, and curvature term. The resultant force is written as

$$\begin{aligned}\mathbf{P}^i &= -\left(E^i_{\mathbf{p}^i} + E^{i*}_{\mathbf{p}^i}\right) + \mathrm{Curv}\left(\mathbf{p}^i\right)\\ &= \alpha\sum_j^I\frac{\left(\mathbf{p}^i - \mathbf{p}^j\right)}{\left\|\mathbf{p}^i - \mathbf{p}^j\right\|}\exp\left(-\left\|\mathbf{p}^i - \mathbf{p}^j\right\|\right)\\ &\quad - \lambda\sum_j^J\frac{\left(\mathbf{p}^i - \mathbf{p}^{*j}\right)}{\left\|\mathbf{p}^i - \mathbf{p}^{*j}\right\|}\exp\left(-\left\|\mathbf{p}^i - \mathbf{p}^{*j}\right\|\right) + \mathrm{Curv}\left(\mathbf{p}^i\right),\end{aligned} \tag{4}$$

where I denotes the number of nonsink particles. In the repulsion-attraction scheme, the sink number J is a variable. The determination of the sinks is given later. It can be noted that there exist two balance constants (α, λ) in order to estimate $\mathbf{P}^i$. No parameters need to be tuned during iteration. This is desirable for the stability and effectiveness of the particle system. In our experiments, the particle system works well when setting both α and λ to 1.

3.1.4. Approximation of Geodesics. The scheme of (1)–(4) describes the particle behaviour on an implicit surface. For the purpose of geodesic computation, the repulsion-attraction scheme is employed here. Suppose that the sources on surface S are known. The goal is to compute the geodesics from some specified source to all destinations. Herein, we provide a procedure for particle generation. The particles are then distributed over S by the scheme of (1)–(4). The basic idea is to utilize the rule of particle repulsion-attraction; that is, particles from the same sources repel each other; otherwise, they attract each other.

Figure 2 shows the particles' trajectories form a network that covers the implicit surface. The given sources (i.e., A,

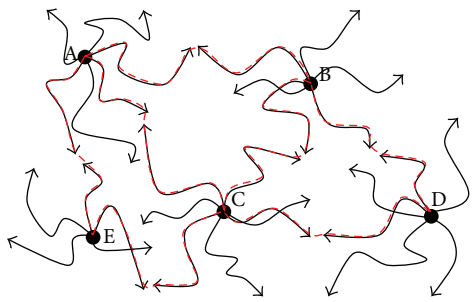

Figure 2: Illustration of the repulsion-attraction rule. Points A–E denote the sources, and the dashed lines denote the connected paths.

B, C, D, and E) generate the new particles according to the distances from each source to their individual nearest neighboring particles. Let the derived particles from the same source be the sinks of the particles coming from other sources. The particles from the same sources repel each other; otherwise, they attract each other. Note that the sources are fixed, while the sinks are floating on S here. When two particles derived from two different sources meet, these two sources are connected by the particles' trajectories (see the dashed lines in Figure 2). At that time, the particles derived from these two sources are regarded as the homologous particles and repel each other. As a result, all the sources will be connected finally by the particles' trajectories as shown in Figure 2. Our presented sampling approach is summarized as follows.

Procedure of Sampling an Implicit Surface

(i) Input a given threshold ε, implicit function $F(\mathbf{p}) = 0$, and a set of sources.

(ii) A loop is as follows:

  (a) each source generates l new particle, if the shortest distance to its neighbours is more than ε;

  (b) repulsion-attraction scheme equations (1)–(4);

  (c) if $\|\mathbf{p}^i - \mathbf{q}^j\| < \varepsilon$, where particles $\mathbf{p}^i$ and $\mathbf{q}^i$ are derived, respectively, from two different sources, then these two sources are connected by the recorded trajectories of $\mathbf{p}^i$ and $\mathbf{q}^i$;

  (d) recording the current location of each particle.

(iii) This is done, until all the sources have been connected.

Moreover, we can apply Dijkstra's algorithm [20] to the resulting connected graph for the approximate geodesic paths on S. The whole algorithm for geodesic approximation is then described as follows.

Algorithm 1 (for approximate geodesic paths on implicit surface). Consider the following.

(1) Input a set of sources.

(2) Apply the procedure of sampling an implicit surface.

(3) Apply Dijkstra's algorithm to the resulting connected graph.

Now, let us consider the time complexity of the algorithm for sampling the implicit surface. Assume that there are N given sources and l drifting particles are produced on S each time. Applying the repulsion-attraction scheme of (1)–(4), we can note that it involves an inner iteration, that is, to pull back l new particles to S by the Newton-Raphson iteration. In our experiments, this can be accomplished through a small number of iterations. For simplicity, we ignore this inner iteration here. There are $(N + kNl)$ particles on S at the kth iteration. It takes $(N + kNl)(N + kNl)$ times for the repulsion-attraction computation between all pairs of points at that time. Hence, the upper bound of computational time is estimated as $O(cN^2)$, where $c = (kl + 1)^2$. When N is large, that is, k becomes small, only a small number of iterations are required to generate the connection graph.

Applying Dijkstra's algorithm to the resulting graph, we can compute the approximate geodesic paths of all pairs of sources at the cost of $O(N^2 \log N)$. Hence, for all pairs of sources, the complexity of Algorithm 1 is of $O(N^2(c + \log N))$ in total. This means that adding the process of implicit surface sampling does not cause the overall running time to increase exponentially.

In general, the implicit surfaces are utilized to fit discrete sampling points. It is straightforward to select the sources from the original sampling points manually. Moreover, the resulting geodesic paths are only approximations on the implicit surface. We will show that the approximate paths can converge to geodesic curves on the implicit surfaces through a geodesic correction procedure described in Section 3.3.

3.2. On Point Clouds. For a point cloud, an alternative is to approximate the point set surface by the Cartesian grid as used in [7]. The basic idea is conceptually simple, that is, to determine the Cartesian grid envelope which surrounds the implicit hypersurface and then apply "continuous Dijkstra"-like strategy to the grid for generating the geodesic estimations.

Herein, we set the mean of the points within a specified cell of the grid as the source node on the surface. This makes the source nodes evenly distributed over the point cloud surface. Moreover, the unorganized points are in general much more than the source nodes. However, the drawback is to build up a whole grid with request for an additional $O(N^d)$ space, where d denotes the dimensionality of manifolds and N denotes the node number. It can be noted that this is a sparse multidimensional array. To avoid such large space complexity, the simplest approach is to store the nonempty cells into an array. Thanks to the ANN searching [21], we may then conveniently determine the neighboring cells of a query cell by it. The ANN searching usually needs $O(dN)$ spaces for the nonempty cell storage and an extra time of partitioning the point clouds into a data structure (e.g., K-d tree), that is, $O(N \log N)$.

Based on the resulting grid, we can propagate the distance information from a specified node to all other nodes on it

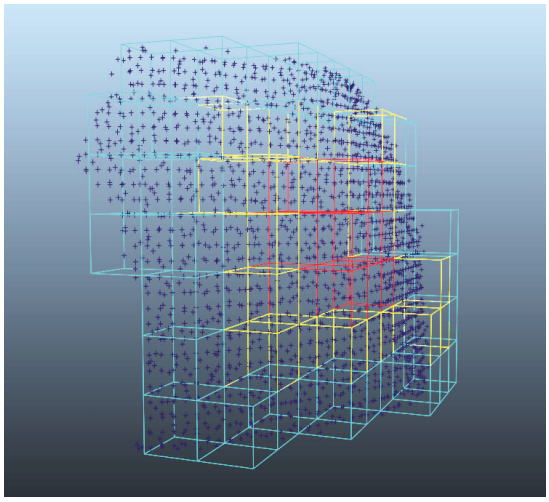

Figure 3: Illustration of a regular grid covering a point cloud. Propagation is from the red cells to the yellow ones. In practice, we store the nonempty cells in an array instead of the whole grid.

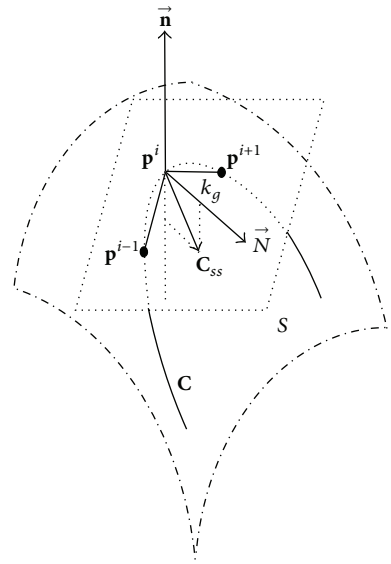

Figure 4: Approximation of the 2nd-order derivative of a curve and the geodesic curvature vector on surface S. $\vec{N}$ denotes the direction of the geodesic curvature vector, which is tangent to S at point $\mathbf{p}^i$.

in a "continuous Dijkstra"-like manner. This is illustrated in Figure 3. The algorithm for geodesic approximation on point clouds is summarized as follows. As a result, computing all pairs of geodesics is of $O(N^2 \log N)$ complexity.

Algorithm 2 (for approximate geodesics on a point cloud). Consider the following:

(i) input: a point cloud S, a source p_0 within the cell c_0, and the intervals of a grid, that is, Δx, Δy, Δz;

(ii) build up a searching tree for ANN running on the grid;

(iii) loop:

  (a) determine the neighboring cells of the visited cell set by ANN searching on the grid, and denote them as N_c;

  (b) loop (within N_c):

    (1) set the mean of points within the ith neighboring cell as the destination node, and denote it as p_i, which is not visited;

    (2) obtain the "shortest" edge linking the p_i and the visited cell set, that is, the "shortest" edge to the visited destination node set;

    (3) combine this edge with the existing geodesic path to form the initial estimation of the geodesic from p_0 to p_i;

(iv) until all the nonempty cells are visited.

3.3. Geodesic Correction. A geodesic curve has vanishing geodesic curvature everywhere. This suggests that an exact geodesic curve should have zero geodesic curvature at all of its points. For a discrete representation, we have to add another requirement that all samples must lie on the implicit surface. The presented correction approach below can guarantee that the sampling points of a curve converge to the implicit surface or point cloud surface and the curve on the surface converges to the geodesic. It is therefore more accurate than the existing mesh or grid based algorithms.

In order to make a curve converge to a geodesic curve, geodesic curvature flow is adopted here. This is because geodesic curvature flow eliminates the geodesic curvatures pointwise on a curve. For clarity, we first consider a regular surface S and two endpoints $\mathbf{p}^1$, $\mathbf{p}^2$, between which we intend to compute a geodesic curve on S. Let $\mathbf{C}(s,0)$ be an initial smooth parametric curve on S with $\mathbf{C}(s^1, 0) = \mathbf{p}^1$ and $\mathbf{C}(s^2, 0) = \mathbf{p}^2$, where s denotes the arc-length parameter of the curve. If these two endpoints are fixed, we deform $\mathbf{C}(s,0)$ by the geodesic curvature flow on S. The curve converges to a geodesic curve. The geodesic curvature flow is described as

$$\mathbf{C}_t(s,t) = k_g \vec{N}, \tag{5}$$

where $k_g \vec{N}$ denotes the geodesic curvature vector of $\mathbf{C}(s,t)$. The geodesic curvature vector can be given as

$$k_g \vec{N} = \mathbf{C}_{ss} - \langle \mathbf{C}_{ss}, \mathbf{n} \rangle \mathbf{n}, \tag{6}$$

where $\mathbf{C}_{ss}$ is the 2nd-order derivative of $\mathbf{C}(s,t)$ w.r.t arc-length s and $\mathbf{n}$ is the normal to S. This flow is also known as the Euclidean curve shortening flow [22] and has been widely applied in computational physics, image processing, and material sciences.

For a discrete case, the geodesic curve has to be approximated by a polyline containing a set of nodes. Correcting the approximate geodesic curve is thus carried out on the resulting polyline. Since we have no parametric forms, the second-order derivative of $\mathbf{C}(s,t)$ w.r.t s is approximated using a vector triangle as shown in Figure 4. This is the increment

of the tangent vector $\mathbf{p}_t^i$ at $\mathbf{p}^i$; that is, $\mathbf{C}_{ss} \approx \Delta\mathbf{p}_t^i = (\overrightarrow{\mathbf{p}^{i+1}\mathbf{p}^i} - \overrightarrow{\mathbf{p}^i\mathbf{p}^{i-1}})/\Delta h$. If $\mathbf{C}_{ss}$'s projection onto the tangent plane at $\mathbf{p}^i$ vanishes, the geodesic curvature of (6) is also bound to vanish at $\mathbf{p}^i$. When the geodesic curvature vanishes pointwise on S, the deforming curve given by (5) reaches its stable state, that is, the geodesic curve.

Moreover, we can write the discrete version of (5) in a matrix form as follows:

$$\mathbf{C}_{ss} = \begin{pmatrix} 0_{3\times3} & & 0_{3\times3} & & 0_{3\times3} & & \cdots \\ -1 & & 2 & & -1 & & \\ & -1 & & 2 & & -1 & \\ & & -1 & & 2 & & -1 \\ & & & -1 & & 2 & \\ & & & & -1 & & 2 \\ & & & & & -1 & \\ & & & & \cdots & & \\ 0_{3\times3} & & \cdots & & \cdots & & 0_{3\times3} \end{pmatrix}_{3m\times3m} \begin{pmatrix} \cdots \\ -1 \\ & -1 \\ & & -1 \\ & & & -1 \\ 2 & & & & \cdots \\ & \cdots & & & \\ & & 0_{3\times3} & \cdots & \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \cdots \\ x_N \\ y_N \\ z_N \end{pmatrix} \equiv \mathbf{K}\mathbf{p}(t), \tag{7}$$

where m denotes the number of sample points on $\mathbf{C}(t)$, $\mathbf{p}(t)$ contains the coordinates of m sample points, and $\mathbf{K}$ is a coefficient matrix. Then, the geodesic curvature of (6) can be rewritten as

$$\begin{aligned} k_g\vec{N} &= \mathbf{K}\mathbf{p}(t) \\ &\quad - \begin{pmatrix} \mathbf{n}_1 & & \\ & \ddots & \\ & & \mathbf{n}_m \end{pmatrix}_{3m\times m} \begin{pmatrix} \mathbf{n}_1^T & & \\ & \ddots & \\ & & \mathbf{n}_m^T \end{pmatrix}_{m\times3m} \mathbf{K}\mathbf{p}(t) \\ &\equiv \mathbf{K}_{3m\times3m}\mathbf{p}_{3m\times1} - \mathbf{n}^*_{3m\times3m}\mathbf{K}_{3m\times3m}\mathbf{p}_{3m\times1}, \end{aligned} \tag{8}$$

where $\mathbf{n}_i$ denotes the normal vector at the ith sample point $\mathbf{p}^i$. As a result, the geodesic curvature flow of (5) can be rewritten as

$$\mathbf{C}_t(\mathbf{p}) = \mathbf{K}\mathbf{p}(t) - \mathbf{n}^*\mathbf{K}\mathbf{p}(t). \tag{9}$$

To speed up the iteration of (9), we apply the backward Euler method [23] to it, which yields

$$\mathbf{C}(\mathbf{p}(t+1)) = \left(I - \mu\left(\mathbf{K} - \mathbf{n}^*\mathbf{K}\right)\right)^{-1}\mathbf{C}(\mathbf{p}(t)), \tag{10}$$

where μ denotes the iterative step length, which is an arbitrary constant due to the backward Euler method. To solve the scheme equation (10) efficiently, we may use the preconditioned conjugate gradient (PCG) method here. Let $\mathbf{A} = (I - \mu(\mathbf{K} - \mathbf{n}^*\mathbf{K}))$. It can be noted that $\mathbf{A}$ is sparse and symmetric. By having a good precondition, it can significantly improve the convergence rate of the conjugate gradient solver. The simpler the precondition, the faster the convergence. The $\mathbf{n}^*$ is a block diagonal matrix, while the $\mathbf{K}$ is a block tridiagonal matrix. All the eigenvalues of $\mathbf{K}$ lie in the interval of $[-1, 0]$. Matrix $\mathbf{A}$ is well conditioned for typical values of μ. In our implementation, we employ the usual diagonal preconditioner $\mathbf{A}^*$; that is, $A^*_{ii} = 1/A_{ii}$, which speeds up the iteration with almost no overhead.

In this paper, we need to deal with two cases, implicit surfaces and point clouds. For the former, it can be noted that $\mathbf{C}_t(\mathbf{p}^i) = \mathbf{p}_t^i$; that is, we can regard $\mathbf{C}_t(\mathbf{p}^i)$ as a velocity estimate $\mathbf{P}^i$ at $\mathbf{p}^i$ in (1). This is the constraint of the particles moving on the geodesic curve. Consequently, we can set $\mathbf{P}^i = \mathbf{C}_t(\mathbf{p}^i)$ and substitute it into (1) leading to the following evolution equation:

$$\mathbf{p}_t = \mathbf{C}_t(\mathbf{p}) - \beta\begin{pmatrix} \mathbf{n}_1 & & \\ & \ddots & \\ & & \mathbf{n}_m \end{pmatrix}_{3m\times m} \begin{pmatrix} \dfrac{F^1}{\|\nabla F^1\|} \\ \cdots \\ \dfrac{F^m}{\|\nabla F^m\|} \end{pmatrix}_{m\times1}. \tag{11}$$

It can be noted that the 2nd term pulls back the sample points onto the implicit surface S. All the sampling points will lie on S and converge to the geodesic accordingly. In our implementation, we split (11) into a two-step scheme. The 1st step employs (10) for updating the $\mathbf{C}_t(\mathbf{p})$ in (11), while the 2nd step is the Newton-Raphson iteration given by

$$\mathbf{p}_t^i \longleftarrow -\frac{\beta F^i \nabla F^i}{\|\nabla F^i\|^2}. \tag{12}$$

For point clouds, there is no implicit function available and we have to seek an alternative approach for estimating the normal vectors. We also have to consider how to restrict the geodesics on the point cloud surfaces similarly to the 2nd term in (11). Because of the above issues, we employ the moving least squares [11] to the point clouds and estimate the normal vectors for the curvature flow of (10).

For a sample point $\mathbf{q}$, we solve for the best tangent plane $\mathbf{A}(\mathbf{q})\mathbf{x} = 0$ at $\mathbf{q}$ that minimizes

$$J = \frac{1}{2}\sum_i w_i\left(\left\|\mathbf{p}^i - \mathbf{q}\right\|\right)\left\|\mathbf{A}(\mathbf{q})\mathbf{p}^i\right\|^2, \tag{13}$$

where $\mathbf{p}^i$ denotes all the sample points. Herein, we employ the weight that is defined as

$$w_i(d) = e^{(-d^2/h^2)}, \tag{14}$$

where h is a constant reflecting the anticipated spacing between neighboring points. The unknowns include the $\mathbf{q}$'s coordinates and the parameters of affine $\mathbf{A}$ here. For simplicity, let $\mathbf{q} = \mathbf{q}_0 + \lambda\mathbf{n}$ and $\mathbf{A}(\mathbf{q})\mathbf{x} = \langle \mathbf{n}, \mathbf{x} - \mathbf{q}_0 - \lambda\mathbf{n} \rangle$, where $\mathbf{q}_0$ denotes an initial clustering centre and $\mathbf{n}$ denotes the normal vector. Such the minimization problem of (13) can be solved in an iterative manner; that is, we start with $\lambda = 0$ and first approximate $\mathbf{n}$ as follows:

$$\mathbf{n} = \arg\min_{\mathbf{n}} \sum_i w_i \left(\left\| \mathbf{p}^i - \mathbf{q}_0 \right\| \right) \left\langle \mathbf{n}, \mathbf{p}^i - \mathbf{q}_0 \right\rangle^2 \quad (15)$$

and then, fixing the normal $\mathbf{n}$, we solve the following nonlinear minimization problem with the single unknown λ:

$$\lambda = \arg\min_{\lambda} \sum_i w_i \left(\left\| \mathbf{p}^i - \mathbf{q}_0 - \lambda\mathbf{n} \right\| \right) \left\langle \mathbf{n}, \mathbf{p}^i - \mathbf{q}_0 - \lambda\mathbf{n} \right\rangle^2. \quad (16)$$

In our implementation, we found that there existed a satisfied solution within $\lambda \in [-h, h]$. Note that we select the nodes of the approximate geodesic paths from Algorithm 2 in Section 3.2 as the initial $\mathbf{q}_0$. Once $\mathbf{n}$ and λ are available, it is natural to update the nodes accordingly, resulting in the resulting geodesic paths being restricted on the point cloud surfaces.

There are scenarios where a specified approximate geodesic path is given. In this case, we first compute the normal vectors and update the location of each node equation (13) and then apply (10) to it for improving the geodesic accuracy.

In order to sample a geodesic curve evenly according to the geometric features, the samples are distributed depending on the curvature of the geodesic curve. This can be given by the proposed repulsion-attraction scheme equation (4), which is modified as

$$\mathbf{P}^i = \alpha \sum_{j=1}^{m} \frac{\left(\mathbf{p}^i - \mathbf{p}^j \right)}{\left\| \mathbf{p}^i - \mathbf{p}^j \right\|} \exp\left(- \left\| \mathbf{p}^i - \mathbf{p}^j \right\| \right) + \mathbf{C}_{ss}\left(\mathbf{p}^i \right). \quad (17)$$

Note that all the samples on a curve $\mathbf{C}(s,t)$ repel each other and the curvature term is replaced by the 2nd derivative of $\mathbf{C}(s,t)$. Substituting it into (1), the particle motion equation is then rewritten as

$$\mathbf{p}_t^i = \mathbf{P}^i - \mathbf{n}^* \mathbf{P}^i. \quad (18)$$

It can be noted that (18) only contains the repulsion motion. Once the equilibrium is reached, the samples will be distributed depending on the curvature of the curve.

Let us summarize the geodesic correction procedure below.

Procedure of Geodesic Correction

(i) For the approximate geodesic paths from Algorithm 1,

  (a) apply Newton-Raphson iteration in (12) for pulling back the particles;
  (b) apply the scheme (10) for correcting geodesics;
  (c) apply the scheme equation (18) for a distribution depending on curvature;
  (d) check the interval of the successive samples and interpolate new samples accordingly;
  (e) repeat until the geodesics converge.

(ii) For the approximate geodesic paths from Algorithm 2,

  (a) apply the scheme of (13) to the input geodesic paths for updating the nodes and the associated normal vectors;
  (b) apply the scheme of (10) for correcting geodesics;
  (c) apply the scheme equation (18) for a distribution depending on curvature;
  (d) check the interval of the successive samples and interpolate new samples accordingly;
  (e) repeat until the geodesics converge.

The time complexity of the geodesic correction procedure depends on the number of the sample points on a geodesic, that is, m in (10). First, we need to pull back the samples to implicit surfaces one by one in both scenarios. Then, (10) is employed individually. Since matrix $\mathbf{A}$ of (10) is sparse and block diagonal, the multiplication of matrix-vector only costs a linear time in the implementation of the preconditioned conjugate gradient method [23]. Assume that pulling back the samples to the surface costs $O(m)$, and solving (10) in $O(m)$ time and solving (10) can be iterated k times. Thus, one geodesic costs around $O((k+1)m)$. For one single source geodesic computation, the upper bound of the whole time complexity can be estimated as $O(kmN)$, where N denotes the destination number. Due to the backward Euler scheme and the precondition technique, the convergence rate of (10) is drastically improved. In general, 2 iterations are sufficient to reach the solution with the relative error less than 10^{-6}. Thus, the time complexity of one single source geodesic computation is of $O(mN)$. For all pairs of geodesics, it costs $O(mN^2)$. We also need to store each geodesic individually, which costs $O(mN^2)$ space in total.

3.4. Error Analysis. In this section, we will show that regardless of whether the approximate geodesic paths are obtained from Algorithm 1 (such as the zigzag curves on implicit surfaces) or from Algorithm 2 (such as the polylines lying on the Cartesian grid), with different intervals, the geodesic correction procedure can achieve the same precision automatically.

In order to estimate the approximation error, we assume that the number of nodes is fixed; that is, the correction procedure does not generate new nodes. Let us consider the local structure of a curve with an osculating circle as shown in Figure 5(a). The local approximate error is described as

$$\text{err} = \widehat{AB} - \left\| \overrightarrow{AB} \right\|, \quad (19)$$

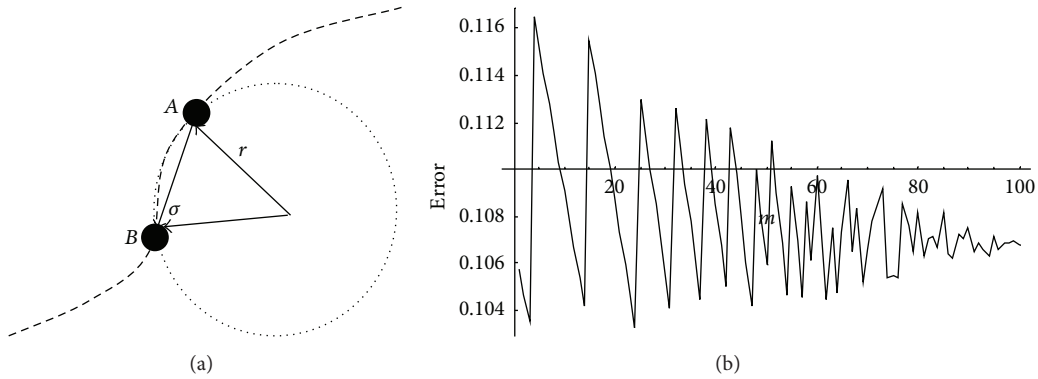

Figure 5: (a) Illustration of the approximation error. (b) The error diagram of geodesic correction. Herein, the chord length is assumed to be equidistant for each m and the curve length is fixed.

where $\widehat{AB}$ denotes the arc-length from A to B and $\|\overrightarrow{AB}\|$ is the chord length σ. Applying the cosine law yields $\widehat{AB} = r\cos^{-1}(1-(\sigma^2/2r^2))$, where r denotes the radius of curvature. Substituting $\widehat{AB}$ into err gives $\text{err} = r\cos^{-1}(1-(\sigma^2/2r^2)) - \sigma$. For a whole geodesic curve, the error is expressed as

$$\text{err} = \sum_{i}^{m} r_i \cos^{-1}\left(1 - \frac{\sigma_i^2}{2r_i^2}\right) - \sum_{i}^{m} \sigma_i, \tag{20}$$

where m denotes the number of samples on a curve and r_i and σ_i depend on the local curvature of a curve. With regard to (10), we assume that $\text{Cst} = \sigma_i/r_i$ is constant for a given geodesic curve. The error of the whole geodesic curve can thus be given by

$$\text{err} = \frac{L}{\text{Cst}} \cos^{-1}\left(1 - \frac{\text{Cst}^2}{2}\right) - L, \tag{21}$$

where $L = \sum_i^m \sigma_i$ is the estimate of the curve's length. Because of the scheme equation (18), the samples are distributed depending on the curvature of the geodesic curve. It can guarantee that the Cst is constant. This means that our algorithms can converge to a desired solution with some specified tolerance.

Moreover, if the number of samples m is variable, Figure 5(b) further shows that raising m can improve numerical stability effectively.

4. Implementation

Our experiments consist of two parts, implicit surfaces and point clouds. All the codes are run on MatLab in a Pentium IV 3.2 GHz PC with 4 GB RAM. Implicit surfaces were generated from three point clouds (i.e., Stanford bunny, hand, and sculpture) by using the FastRBF toolbox (at http://www.farfieldtechnology.com/products/toolbox/).

4.1. Implicit Surfaces. To illustrate the curvature dependent distribution, we first performed the repulsion-attraction scheme equations (1)–(4) on the 3D model of the Stanford bunny. The reason why we used this model is because the ears have high curvatures (as shown in Figure 6, there are two images for front and back views), which can test the effectiveness of our method. The particles are generated from 39 fixed sources (which are marked as *). The sources disperse the particles over the implicit surface S by continuously generating new particles. Due to the curvature term equation (2), it can be noted that many particles remain at the high curvature regions in Figure 6 allowing higher curvature regions to be more densely represented.

Table 1: Comparison of the running times of Algorithm 1 and Algorithm 2 and the procedure of geodesic correction for models of Figure 9. The first 2 rows show the running times of applying Algorithm 1 on implicit surfaces, while the last 2 rows show those of applying Algorithm 2 on point clouds.

Model	Hand (sources: 272) (cells: 290)	Sculpture (sources: 289) (cells: 311)
Time of Algorithm 1 (s)	63.23	74.37
Time of geodesic correction procedure (s)	244.39	290.42
Time of Algorithm 2 (s)	51.71	59.37
Time of geodesic correction procedure (s)	245.11	299.74

We further performed Algorithm 1 described in Section 3.1 on a synthetic 3D model of a roll that has two high curvature regions and a large surface area as shown in Figure 7. To visually demonstrate the trajectories of the particles, the particles are generated from only 5 fixed sources (which are marked as *). It can be noted that our algorithm does not produce the approximate geodesic paths for all pairs of sources on S. This is because the particle system equations (1)–(4) terminate once all sources have been linked into a connected graph. Hence, applying Dijkstra's algorithm to the resulting graph produces all pairs of geodesic approximations on S. These approximate geodesic paths were further corrected by the geodesic correction procedure described in Section 3.3 as shown in Figure 7(b). For comparison, the approximate geodesics and the corrected ones are both depicted in Figure 7(c). One

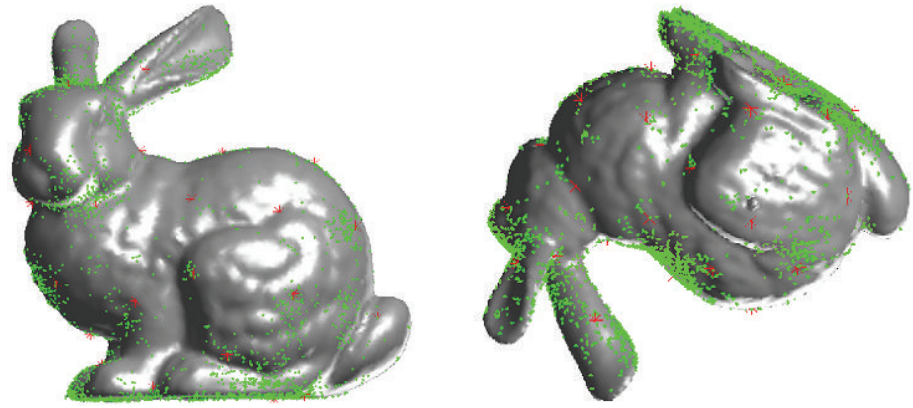

Figure 6: Illustration of particle distribution using the repulsion-attraction scheme. The 39 sources are marked as *, while 4041 floating particles are denoted by dots. Each source can generate 5 particles each time. The particle system is terminated depending on the maximal particle number rather than the particle density.

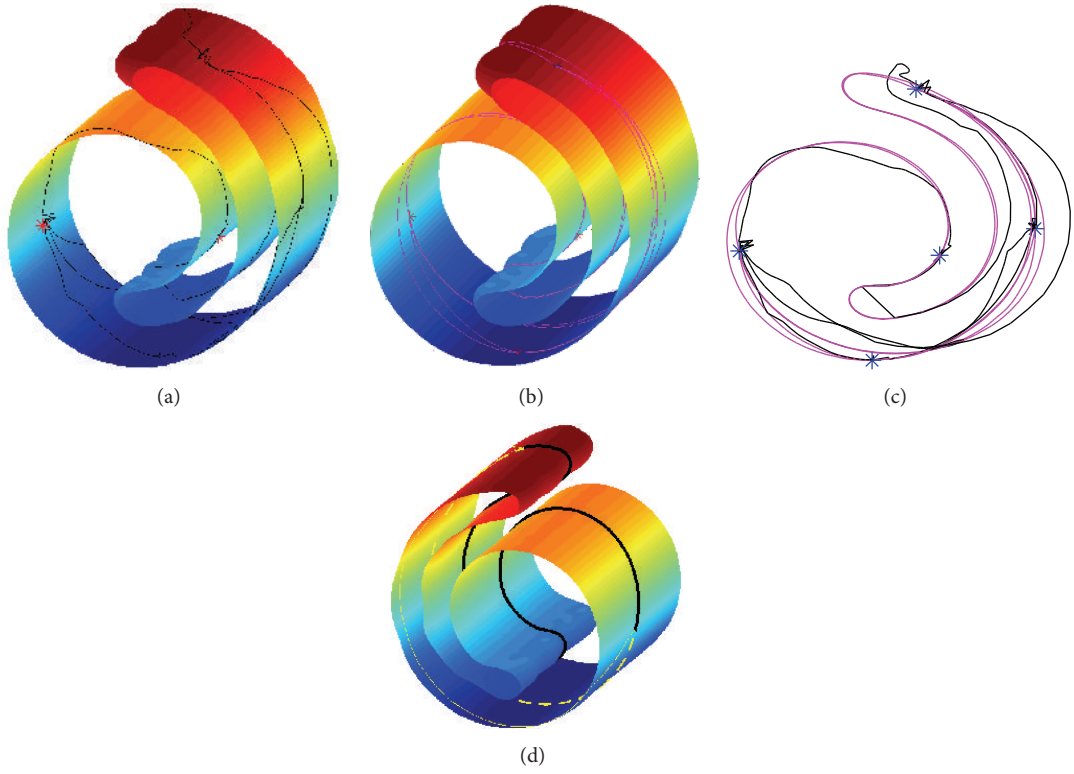

Figure 7: Comparison of the approximate geodesic paths and the corrected ones. The 5 sources are marked as *. (a) The approximated geodesic paths computed using Algorithm 1 described in Section 3.1; (b) the corrected geodesic paths by the geodesic correction procedure described in Section 3.3; (c) both approximate and corrected geodesic paths are depicted together, where the magenta curves represent the corrected paths, while the black ones represent the approximate paths; (d) the solid curve (yellow) represents the approximate geodesic path, while the dashed curve (yellow) represents the corrected path. The bold solid curve (black) is the other geodesic.

can see that some trajectories have some small zigzag lines around the sources. This indicates that the movement of the particles is very random in the neighborhood of the sources, since new particles are generated constantly around the sources. Moreover, the roll surface is a cylinder-like surface. Due to the random movement of particles, it is possible to generate two approximate geodesic paths sharing the same endpoints on the two sides of the surface. Performing the geodesic correction procedure on them results in two different geodesic curves on the surface (see yellow and black lines in Figure 7(d)). This can be easily understood by looking at the great circle on a sphere or a cylinder.

If the sources are evenly distributed over the surface, Algorithm 1 can generate the connected graph quickly. To illustrate it, we performed Algorithm 1 on a 3D human hand model, which contains 272 vertices. These vertices

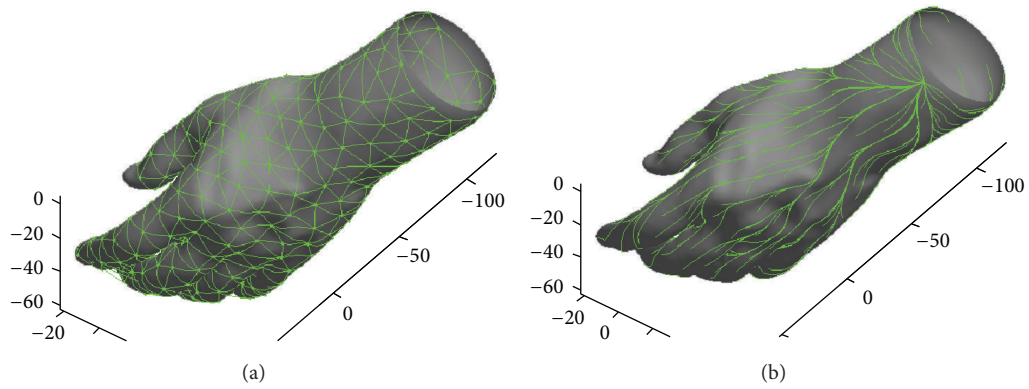

Figure 8: Comparison of the connected graph by Algorithm 1 and the geodesics by the geodesic correction procedure. (a) The connected graph with the known 272 vertices by Algorithm 1. For simplicity, the link between two endpoints is drawn by a straight line, not a particle trajectory. (b) The corrected geodesics on the surface.

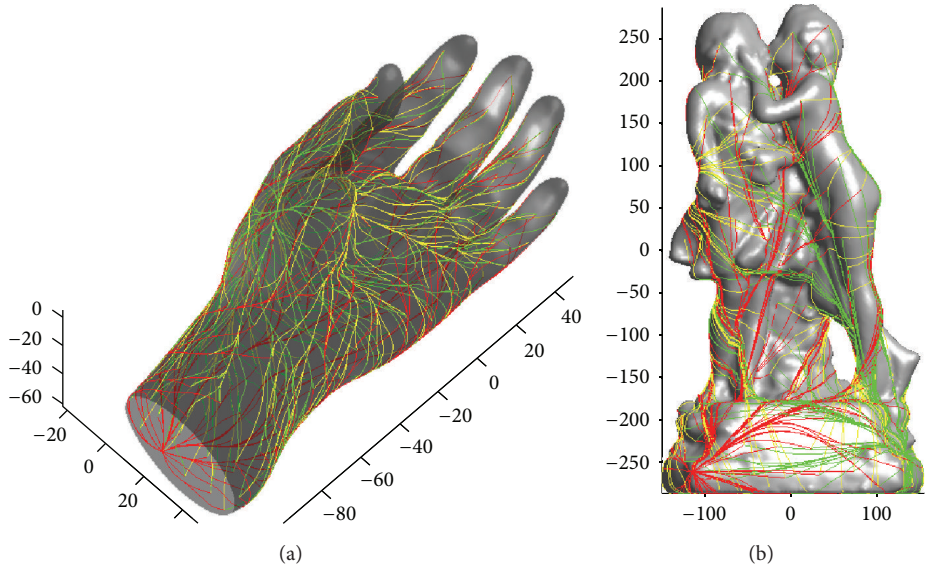

Figure 9: Illustration of pairwise geodesics on 2 models. For each model, only 3 sets of geodesics of one source to all destinations are depicted together on the surface with 3 different colours.

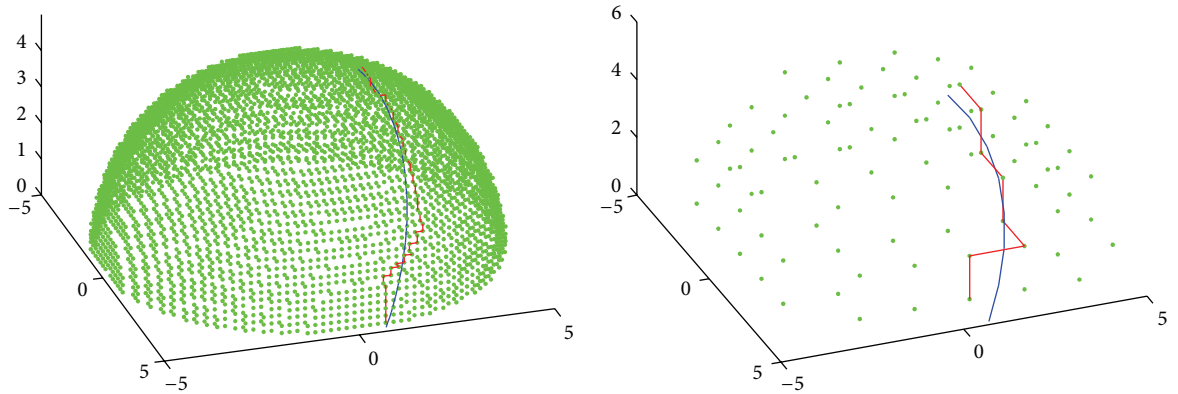

Figure 10: Illustration of geodesic computation on different grid sizes. The red curves represent the geodesic paths by Algorithm 2, while the blue ones by the geodesic correction procedure, and the green points denote the Cartesian grid cells.

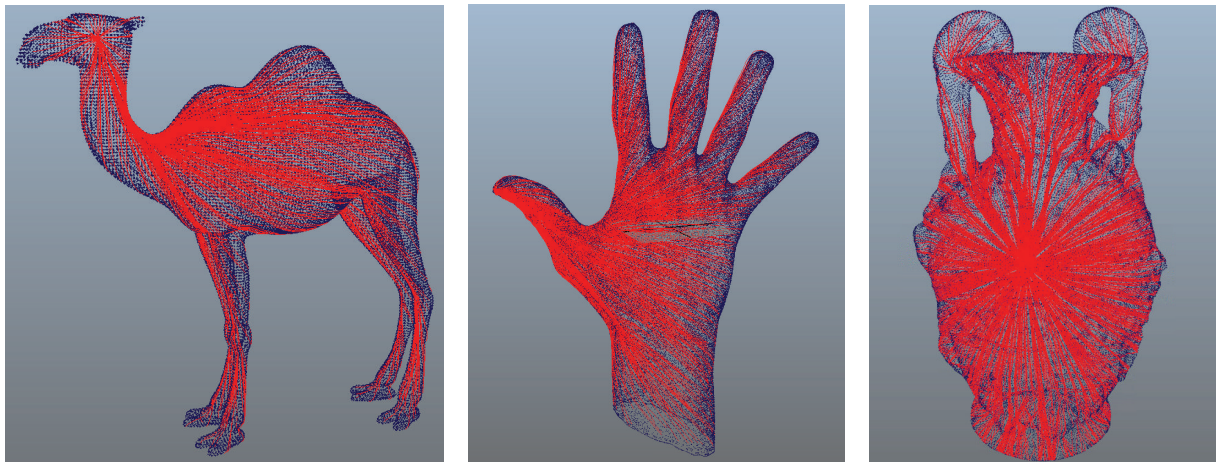

Figure 11: Performance of Algorithm 2 with the geodesic correction procedure on 3 point clouds, that is, camel (cells: 10366), hand (cells: 8008), and jar (cells: 20416). For simplicity, we only compute a set of one single source geodesics here.

are regarded as the fixed sources and continue to generate new particles. However, the resulting network shown in Figure 8(a) is not a reasonable mesh for modelling the human hand. How to remesh it is beyond the remit of this paper. Herein, our goal is to carry out Dijkstra's algorithm on a resulting connected graph for initial geodesic estimations. Each edge in Figure 8(a) corresponds to a particle trajectory, which is, namely, the initial geodesic path between two sources. Following Dijkstra's algorithm, the geodesic correction procedure was further performed on the resulting initial geodesic estimations. We show the geodesics of "one source to all destinations" in Figure 8(b).

We also performed Algorithm 1 with the geodesic correction procedure on 2 implicit surfaces, that is, human hand and sculpture, for computing all pairs of geodesics. For graphical rendering purposes, only 3 sets of the geodesics of "one source to all destinations" are depicted on the surface of each model in Figure 9. Table 1 shows the running times. It can be noted that our method can handle the genus surfaces (e.g., sculpture surface with genus 2 in Figure 9).

4.2. Point Clouds. For a point cloud, we first utilize a Cartesian grid to cover it and then apply Algorithm 2 in Section 3.2 to the resulting grid. Independent of the cell size, the geodesic correction procedure in Section 3.3 can automatically interpolate new samples for the improvement of the geodesic quality. We performed Algorithm 2 on a semisphere that is covered by two Cartesian grids with different cell sizes as shown in Figure 10. It is evident that the corrected geodesic curves by the geodesic correction procedure are becoming the same if these two curves contain the same number of samples. It can also be noted that the error depends on the sample number rather than the original grid cell sizes.

For the comparison of the running times of Algorithms 1 and 2, we performed Algorithm 2 with the geodesic correction procedure on these 2 point clouds in Figure 9. The running times are shown in Table 1. It can be noted that the running time of Algorithm 2 is less than that of Algorithm 1. However, the subsequent application of the geodesic correction procedure takes nearly the same time. This is because, for the same specified precision, the procedure of geodesic correction automatically inserts new samples accordingly. This will result in the similar sample size regardless of Algorithms 1 or 2.

To further illustrate the effectiveness of Algorithm 2, we performed it on 3 point clouds with complex topological and geometric structures. The results are shown in Figure 11. It can be noted that Algorithm 2 can deal with point clouds with genus (e.g., point cloud of jar with genus 2).

5. Conclusions

In this paper, we have presented a novel framework for computing geodesics on implicit surfaces and point clouds. The main feature of this work is its ability to produce smooth and accurate geodesics on the surface. This is due to the introduction of geodesic curvature flow in our framework. In addition, since the geodesic correction procedure can distribute the sample points according to the curve curvature and further interpolate new samples automatically, the resulting geodesic solution can achieve the specified precision. Experiments have shown that our algorithm works well on arbitrary implicit surfaces and point clouds.

For a general case, if no sources are given, the implicit surface concerned must be modelled beforehand. The surface modelling step is not included here. Moreover, it can be noted that the geodesic correction procedure costs most of the running time compared to Algorithms 1 and 2. This is because geodesic curvature flow is introduced. Optimisation of the computational efficiency and accuracy of the geodesic correction procedure is left as future work.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, "Minimal surfaces based object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 394–398, 1997.

[2] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: a general representation of shape for computer graphics," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 249–254, New Orleans, La, USA, July 2000.

[3] A. Yezzi Jr., S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, "A geometric snake model for segmentation of medical imagery," *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 199–209, 1997.

[4] A. Witkin and P. Heckbert, "Using particles to sample and control implicit surfaces," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*, pp. 269–277, Orlando, Fla, USA, 1994.

[5] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.

[6] A. Bartesaghi and G. Sapiro, "A system for the generation of curves on 3D brain images," *Human Brain Mapping*, vol. 14, no. 1, pp. 1–15, 2001.

[7] F. Mémoli and G. Sapiro, "Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces," *Journal of Computational Physics*, vol. 173, no. 2, pp. 730–764, 2001.

[8] M. Hofer and H. Pottmann, "Energy-minimizing splines in manifolds," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 284–293, 2004.

[9] M. Hofer, H. Pottmann, and B. Ravani, "From curve design algorithms to the design of rigid body motions," *The Visual Computer*, vol. 20, no. 5, pp. 279–297, 2004.

[10] L. Kobbelt and P. Schröder, "A multiresolution framework for variational subdivision," *ACM Transactions on Graphics*, vol. 17, no. 4, pp. 209–237, 1998.

[11] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003.

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

[13] J. Chen and Y. Han, "Shortest paths on a polyhedron—part I: computing shortest paths," *International Journal of Computational Geometry & Applications*, vol. 6, no. 2, pp. 127–144, 1996.

[14] M. Lanthier, A. Maheshwari, and J.-R. Sack, "Approximating weighted shortest paths on polyhedral surfaces," in *Proceedings of the 13th Annual Symposium on Computational Geometry*, pp. 274–283, Nice, France, June 1997.

[15] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," in *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, pp. 553–560, Los Angeles, Calif, USA, August 2005.

[16] K. Polthier and M. Schmies, "Straightest geodesics on polyhedral surfaces," in *Mathematical Visualization*, H. C. Hege and K. Polthier, Eds., pp. 135–150, Springer, Berlin, Germany, 1998.

[17] D. Martínez, L. Velho, and P. C. Carvalho, "Computing geodesics on triangular meshes," *Computers & Graphics*, vol. 29, no. 5, pp. 667–675, 2005.

[18] J. Klein and G. Zachmann, "Point cloud surfaces using geometric proximity graphs," *Computers & Graphics*, vol. 28, no. 6, pp. 839–850, 2004.

[19] M. R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati, "Approximating geodesics on point set surfaces," in *Proceedings of the 3rd Eurographics/IEEE VGTC Conference on Point-Based Graphics (SPBG '06)*, pp. 85–94, 2006.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, McGraw-Hill, 2nd edition, 2001.

[21] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[22] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[23] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 2nd edition, 1992.