A Triangulation-Invariant Method for Anisotropic Geodesic Map Computation on Surface Meshes

Sang Wook Yoo, Joon-Kyung Seong, Min-Hyuk Sung, Sung Yong Shin, and Elaine Cohen

Abstract—This paper addresses the problem of computing the geodesic distance map from a given set of source vertices to all other vertices on a surface mesh using an anisotropic distance metric. Formulating this problem as an equivalent control theoretic problem with Hamilton-Jacobi-Bellman partial differential equations, we present a framework for computing an anisotropic geodesic map using a curvature-based speed function. An ordered upwind method (OUM)-based solver for these equations is available for unstructured planar meshes. We adopt this OUM-based solver for surface meshes and present a triangulation-invariant method for the solver. Our basic idea is to explore proximity among the vertices on a surface while locally following the characteristic direction at each vertex. We also propose two speed functions based on classical curvature tensors and show that the resulting anisotropic geodesic maps reflect surface geometry well through several experiments, including isocontour generation, offset curve computation, medial axis extraction, and ridge/valley curve extraction. Our approach facilitates surface analysis and processing by defining speed functions in an application-dependent manner.

Index Terms—Geodesic, anisotropy, surface mesh, Hamilton-Jacobi-Bellman, curvature minimization, curvature variation minimization, shape analysis.

1 Introduction

Computing geodesics on 2-manifolds has been a recurrent theme for the last two decades in both numerical analysis and computer graphics communities. Formulated as first-order partial differential equations with boundary conditions, the problem of finding geodesic curves is important in its own right and also as an integral part of applications, including surface segmentation and editing [1], [2], distortion-minimizing parameterization [3], [4], surface remeshing [5], [6], [7], isometry-invariant shape classification [8], [9], [10], medical imaging [11], [12], and geophysics [13], to name a few.

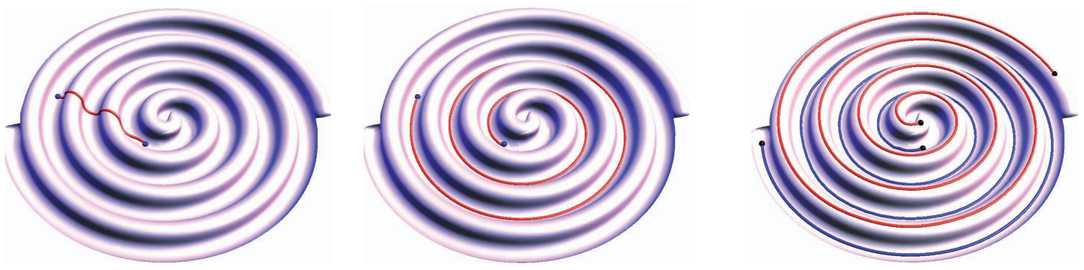
A geodesic is a distance-minimizing curve in the sense that any perturbation of this curve increases its length. The geodesic computation problem has been investigated in various settings under isotropic distance metrics. Much effort has been focused on the Eikonal equation that represents the propagation of a front in an isotropic manner. Adopting the Dijkstra-type algorithmic framework of *fast marching*, Kimmel and Sethian [14] proposed a method to compute approximate geodesic paths by numerically solving this equation.

A more general form of geodesics, called anisotropic geodesics, has been considered under anisotropic distance metrics, each of which is a function of both the position of a point and the direction at it. Sethian and Vladimirsky [15] presented a class of methods referred to as ordered upwind method (OUM)-based solvers for Hamilton-Jacobi partial differential equations (H-J PDEs), which can be regarded as a generalization of Eikonal equations under anisotropic distance metrics (or the inverse of speed functions). Thus, an OUM-based solver also adopts the algorithmic framework of fast marching. They argued that their OUM-based solver could be extended to manifolds without providing explicit algorithms.

In this paper, we discuss how to build the anisotropic geodesic map (AG map for brevity) on a surface mesh. Given an anisotropic distance metric, we formulate the problem of computing the AG map as an equivalent control theoretic problem with Hamilton-Jacobi-Bellman (H-J-B) PDEs. In order to solve the H-J-B PDEs on a surface mesh, we adopt an OUM-based solver and present a triangulation-invariant method for the solver. Since a triangulation scheme would impose connectivity information to the underlying surface, the AG map constructed by an original OUM-based solver could be dependent on the given triangulation. Our approach is based on exploring proximity among the vertices on a surface mesh rather than connectivity information given by a specific triangulation scheme, which is therefore independent of a given triangulation for the surface. We further propose two

- *S.W. Yoo and S.Y. Shin are with the Computer Graphics Laboratory, Korea Advanced Institute of Science and Technology (KAIST), 291, Daehak-ro, Yuseong-gu, Daejeon 305-701, Korea. E-mail: {ysw81, syshin}@jupiter.kaist.ac.kr.*
- *J.-K. Seong is with the School of Computer Science and Engineering, Soongsil University, 511 Sangdo-dong, Dongjak-gu, Seoul 156-743, Korea. E-mail: seong@ssu.ac.kr.*
- *M.-H. Sung is with the Image Media Research Center, Korea Institute of Science and Technology (KAIST), Hwarangno 14-gil 5, Seongbuk-gu, Seoul 136-791, Korea. E-mail: smh0816@imrc.kist.re.kr.*
- *E. Cohen is with the Geometric Design and Computation Group, School of Computing, 50 Central Campus Drive, University of Utah, Salt Lake City, UT 84112. E-mail: cohen@cs.utah.edu.*

Manuscript received 18 May 2011; revised 10 Dec. 2011; accepted 8 Jan. 2012; published online 26 Jan. 2012.
Recommended for acceptance by P. Cignoni.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-05-0109.
Digital Object Identifier no. 10.1109/TVCG.2012.29.



(a) Euclidean geodesic (b) Curvature-minimizing geodesic (c) Curvature variation-minimizing geodesic

Fig. 1. Three different geodesic paths are computed using three different distance metrics: euclidean speed function (a), curvature-minimizing speed function (b), and curvature variation-minimizing speed (c). In right figure, the ridge and valley curves are marked red and blue, respectively.

anisotropic distance metrics for surface analysis and processing. Based on the fact that curvature information encodes rich surface features, we use normal curvature-based tensors of a surface to derive curvature-based speed functions as well as their respective anisotropic distance metrics. Fig. 1 shows examples of AG paths on a surface under three different distance metrics: euclidean speed function, curvature-minimizing speed function, and curvature variation-minimizing speed function. In Fig. 1c, the ridge and valley curves are marked red and blue, respectively. The effectiveness of the proposed metrics is demonstrated through experiments including geodesic offset curve extraction, medial axis computation, and ridge/valley curve extraction.

The rest of this paper is structured as follows: Section 2 reviews related work. Section 3 describes a Hamilton-Jacobi-Bellman formulation. In Section 4, we present a triangulation-invariant solver for computing an AG map on a surface mesh. Section 5 proposes two anisotropic curvature-based distance metrics. In Section 6, we show experimental results to support the efficacy of the proposed method. Finally, Section 7 concludes this paper.

2 Related Work

Rich results have been reported on geodesic computation in various domains. However, we focus our review on those most closely related to our work.

Mitchell et al. [16] proposed the exact method for solving the shortest path problem on a triangular mesh in R^3. They computed the geodesic distances from a single source to all vertices on a meshed surface by partitioning each mesh edge into a set of intervals over which the exact distance computation can be performed. Later, an approximation algorithm with a bounded error was implemented [17]. Bommes and Kobbelt [18] further generalized the algorithm to handle an arbitrary, possibly open, source region on the mesh.

To approximate geodesic distances on a point set surface, Klein and Zachmann [19] provided a method for finding a shortest path on a geometric proximity graph called a spheres-of-influence graph. Hofer and Pottmann [20] computed a geodesic curve by formulating it as an energy-minimizing discrete curve constrained on a moving least squares (MLS) surface, and later this formulation was generalized to estimate the proximity of the geodesic path to the point cloud, utilizing surfel disks [21]. Our approach in this paper considers the proximity of the points in a more general sense by adopting anisotropic distance metrics.

Sethian [22] proposed the fast marching framework to compute an approximate geodesic. Qin et al. [23] presented a similar approach earlier based on front propagation without using a heap data structure. The fast marching framework adopts the Dijkstra-type algorithmic structure of computing the shortest paths from a single source to all destinations in order to solve Eikonal equations. The time complexity of this method is $O(N\log N)$, which is optimal in the worst case. Tsitsiklis [24] also presented a Dijkstra-type algorithm for solving eikonal equation. The fast marching method was generalized to arbitrary triangulated surfaces [14], implicit unorganized surfaces [25], point set surfaces [26], and parametric surfaces [27]. Recently, an efficient parallel algorithm for first-order approximation of geodesic distances on parametric surfaces was proposed by Weber et al. [28]. However, all these methods can solve only Eikonal equations, which are a special type of H-J PDEs.

H-J PDEs are a generalization of Eikonal equations under anisotropic metrics. There are two groups of solvers for H-J PDEs from an algorithm point of view: sweep-based solvers and OUM-based solvers. Sweep-based solvers adopt the plane sweep framework from computational geometry [29], [30]. Based on a control theoretic perspective, Qian et al. [31] proposed a sweep-based method, which uses a Gauss-Seidel updating order for fast convergence. Kao et al. [32] introduced a new interpretation of Hamiltonians based on the Legendre transformation for solving arbitrary static H-J PDEs. Based on prior approaches [33], [34], a parallel algorithm called the fast iterative method (FIM) was recently proposed to solve a class of H-J PDEs on massively parallel systems [35]. Seong et al. [36] used the FIM for segmenting and matching parametric surfaces using a curvature-based local distance function. It is challenging, however, to extend the sweep-based solvers to surface meshes with arbitrary boundaries.

The other group consists of OUM-based solvers, which can be regarded as an extension of fast marching methods. An early work of Sethian and Vladimirsky [37] and its later enhancement [15] were able to solve H-J PDEs on unstructured meshes in R^n. The underlying idea of OUM-based solvers is that the solution of a H-J PDE at a point depends only on its "upwind neighbors" along the characteristic direction. We generalize the later version to handle surface meshes, which is independent of a triangulation scheme for the underlying surface.

Mémoli and Sapiro [25] presented an algorithm for computing the geodesic distance map on an implicit surface under an isotropic metric. This solver approximates the geodesic distance map on a regular grid embedded in a thin offset band surrounding the surface. As the offset band gets thinner, the approximate solution converges to the geodesic distance map. The authors later showed that their approach can be generalized for solving Hamilton-Jacobi PDEs on point set surfaces [26]. However, they mainly dealt with Eikonal equations without providing an explicit solver for the more general class of equations, Hamilton-Jacobi PDEs. Recently, Konukoglu et al. [38] presented a recursive algorithm for solving anisotropic Eikonal equations on mesh surfaces. This method improves the fast marching algorithm by employing a recursive correction scheme under an anisotropic metric. However, the solution to the anisotropic Eikonal equations depends on a triangulation of the surface since the nearest neighbors of each vertex are determined by its incident edges of the specific triangulation.

3 Hamilton-Jacobi-Bellman (H-J-B) Formulation

In this section, we briefly summarize the control theoretic view of AG computation based on the Hamilton-Jacobi-Bellman (H-J-B) formulation in [15]. We use this formulation to construct an AG map for a surface mesh under an anisotropic distance metric $\psi(\mathbf{x}, \mathbf{a}(\mathbf{x}))$ on a 2-manifold Ω, where $\mathbf{a}(\mathbf{x})$ is a unit vector in the tangent plane at $\mathbf{x} \in \Omega$, that is, $\mathbf{a}(\mathbf{x}) \in S^1 = \{\mathbf{v} \in R^2 | \ \|\mathbf{v}\| = 1\}$. Let $\mathcal{C}_\mathbf{x} = \{c \mid c(s) \in \Omega,\ 0 \leq s \leq L;\ c(0) = \mathbf{x},\ c(L) \in \partial\Omega\}$, where c is a curve on the surface connecting $\mathbf{x}$ and a point on the boundary $\partial\Omega$. Then, the length of the AG curve from $\mathbf{x}$ to the point on the boundary $\partial\Omega$, denoted by $u(\mathbf{x})$, is given as follows:

$$u(\mathbf{x}) = \inf_{c \in \mathcal{C}_\mathbf{x}} \int_0^L \psi(c(s), \frac{c'(s)}{\|c'(s)\|}) ds. \tag{1}$$

Interpreting ψ as the inverse of speed (the time per unit distance along the curve), the length (or cost) of the curve is the time needed to travel along it. Thus, $u(\mathbf{x})$ is the time to travel along the AG curve from $\mathbf{x}$ to the boundary, which is the infimum over all possible curves between $\mathbf{x}$ and the boundary.

The infimum in the above definition is due to the fact that there are infinitely many possible paths and the optimal path in general does not necessarily exist. By using a suboptimal path instead, $u(\mathbf{x})$ is the unique viscosity solution of the following H-J-B equation [15]:

$$\begin{cases} \min_{\mathbf{a} \in S^1}\{(\nabla u(\mathbf{x}) \cdot \mathbf{a}) f(\mathbf{x}, \mathbf{a}) + 1\} = 0, & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = q(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{cases}$$

where $q(\mathbf{x})$ is the time penalty for exiting the domain at the point $\mathbf{x} \in \partial\Omega$ and $f = \frac{1}{\psi}$ is a speed function [11]. In our problem setting, we set $q(\mathbf{x}) = 0$. Let a Hamiltonian be chosen as follows:

$$\begin{aligned} H(\nabla u, \mathbf{x}) &= -\min_{\mathbf{a} \in S^1}\{(\nabla u(\mathbf{x}) \cdot \mathbf{a}) f(\mathbf{x}, \mathbf{a})\} \\ &= \max_{\mathbf{a} \in S^1}\{(\nabla u(\mathbf{x}) \cdot (-\mathbf{a})) f(\mathbf{x}, \mathbf{a})\} = 1. \end{aligned} \tag{2}$$

This Hamiltonian turns out to be convex and homogeneous of degree one in the first argument, and thus the OUM-based solver converges to solutions of H-J-B PDEs. We design the speed function $f(\mathbf{x}, \mathbf{a})$ in an application-dependent manner, which is Lipschitz-continuous and bounded by some constants f_1 and f_2

$$0 < f_1 \leq f(\mathbf{x}, \mathbf{a}) \leq f_2 < \infty. \tag{3}$$

In order to exploit the framework of the OUM-based solver, we propose to use normal curvature-based speed functions that satisfy these conditions (Section 5).

Given a set of seed points with boundary conditions, the AG map for a surface mesh provides the viscosity solution of (2) at every vertex on the surface under an anisotropic distance metric ψ. In other words, this map gives the arrival time $u(\mathbf{x})$ to every vertex $\mathbf{x}$ on the surface from the given seed points. Also, the maximizer $\mathbf{a}$ in the last part of (2) corresponds to the characteristic direction for vertex $\mathbf{x}$. Therefore, starting from any vertex on the surface, we can trace back an approximate geodesic curve to a seed guided by this map.

4 Triangulation-Invariant AG Map Construction

4.1 Overview

Motivation. In this section, we present a triangulation-invariant solver for AG map construction by adopting an OUM-based solver [15] for surface meshes. The OUM-based solver converges to the solution of the H-J-B PDE as a mesh is successively refined. Given a mesh of fixed resolution, the quality of the solution depends on the quality of the mesh, that is, how well the surface is triangulated.

A triangulation of a set of points on a 2-manifold encodes proximity information. Specifically, in a Delaunay triangulation, the nearest neighbors of each vertex in euclidean sense are connected by its incident edges. Guided by this information, an isotropic geodesic map can be constructed naturally while traversing the triangular mesh from vertex to vertex starting from the seeds. However, the proximity information that is embedded in a Delaunay triangulation provides a limited guide to AG map construction for a given vertex set on a 2-manifold although the convergence of the OUM-based solver is guaranteed as the mesh is successively refined.

From the algorithmic point of view, the OUM-based solver chooses the next closest vertex to the seeds under an anisotropic distance metric among the vertices in the candidate vertex set. However, these vertices have been chosen based on the connectivity of a triangular mesh, which does not reflect the proximity of the vertices in the sense of the same anisotropic metric. Unfortunately, it is difficult, if not impossible, to obtain a generalized Delaunay triangulation for an arbitrary surface under an anisotropic distance metric. Therefore, our strategy is to locally enumerate proximity of the vertices in a triangulation-invariant manner rather than to obtain it from a specific triangulation.

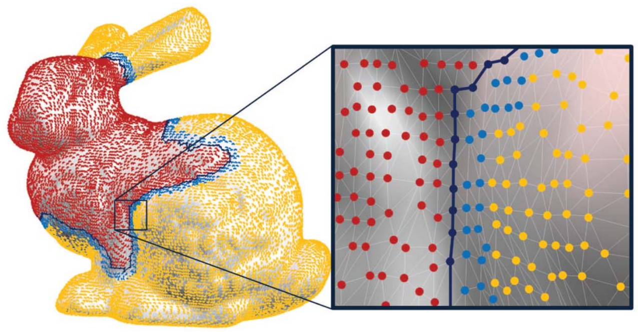


Fig. 2. Three vertex sets: *Far* (yellow), *Considered* (blue), and *Accepted* (red). The accepted front (AF) is represented in bold lines.

Framework. Being inherited from the OUM-based solver, the framework of our triangulation-invariant solver is similar to that of the Dijkstra's shortest path finding algorithm from a single source to all other vertices. A pseudocode for our triangulation-invariant solver is given in **Algorithm 1** (see Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.29). Starting from the user-provided seed points, the solver estimates the arrival times to all vertices, one by one, guided by the proximity information at each vertex on the surface mesh. While the solver is being executed, the vertices on the surface are partitioned into three sets: *Far*, *Considered*, and *Accepted*, as illustrated in Fig. 2. The set *Far* consists of vertices for which no information on the arrival times are available. On the other hand, the arrival time to every vertex in the set *Accepted* has already been computed. The set *Considered* is the boundary of *Far*, along which a tentative arrival time to each vertex has been computed. Each vertex in *Considered* is a neighbor of a vertex in *Far* and also that of a vertex in *Accepted*. The set *Considered* consists of all candidates for the next vertex that moves to *Accepted*, and the tentative arrival times to these vertices are used as the criteria to choose the next vertex.

The vertices on the boundary of *Accepted* form an accepted front (*AF*), which can be regarded as a snapshot of a discretization of the front formulated by a H-J PDE [15]. More precisely, we define the accepted front *AF* in terms of two sets, *Accepted* and *Considered* as follows:

$$AF = \left\{ \begin{array}{ll} \mathbf{x_j x_k}|, & \mathbf{x}_j \text{ and } \mathbf{x}_k \text{ are adjacent to each other in} \\ & \textit{Accepted} \text{ and also adjacent to a vertex in} \\ & \textit{Considered} \end{array} \right\}.$$

As shown in Fig. 2, the set *AF* is a piecewise linear curve (bold) which is progressively evolved from time to time by the triangulation-invariant solver.

4.2 Triangulation-Invariant Solver

In this section, we explain the main algorithm of our triangulation-invariant solver. Since the framework of our solver is similar to that of the OUM-based solver, we concentrate our discussion on the following issues that arise for the triangulation-invariant solver:

- How to determine the neighbors of a vertex (steps 4 and 9 in **Algorithm 1**).
- How to compute a tentative arrival time $V(\mathbf{x})$ (steps 4 and 9).
- How to construct *Considered* (step 8).
- How to update *AF* and *Considered* (steps 7 and 8).
- Why our solver converges to the viscosity solution.

Neighbor determination. A triangular mesh specifies the neighbors of each vertex explicitly via its incident edges, thereby providing partial information on the proximity for a vertex set on a mesh in a euclidean sense, although biased to a specific triangulation. However, our objective is to avoid such a mesh so as to obtain a triangulation-invariant solver. Specifically, for each vertex p_i of the mesh, we choose as its neighbors the vertices that are located within distance h from p_i in a geodesic manner. In order to estimate h, one may employ the recent result for optimal "bandwidth" selection for a MLS surface, where the bandwidth of a sample vertex is its influence radius [39]. For efficiency, however, we use a heuristic scheme to choose h as follows:

$$h = \max_i \left\{ \sigma \sqrt{\pi s_i^2 / n_i} \right\}, \tag{4}$$

where n_i is the number of vertices from which geodesic distance to the vertex p_i is less than s_i and σ is a user-defined constant. $\pi s_i^2 / n_i$ approximates the average area per vertex near p_i. Given p_i and sufficiently large n_i, we choose s_i using the Dijkstra's shortest path finding algorithm such that all n_i neighbors of p_i are within distance s_i from p_i in a geodesic manner on the surface mesh. In our experiments, we empirically set $n_i = 60\text{-}70$ and $\sigma = 4$.

Tentative arrival time computation. We describe how to evaluate the tentative arrival time to a vertex $\mathbf{x}$ in *Considered* (steps 4 and 9 in **Algorithm 1**). Consider the projection of a line segment $\mathbf{x}_j\mathbf{x}_k$ in the set *AF* onto the tangent plane of the surface at $\mathbf{x}$ (Fig. 3a). Since both $\mathbf{x}_j$ and $\mathbf{x}_k$ are from *Accepted*, their minimum arrival times $u(\mathbf{x}_j)$ and $u(\mathbf{x}_k)$ have already been estimated. Let $\bar{\mathbf{x}}_j$ and $\bar{\mathbf{x}}_k$ be the projected points of $\mathbf{x}_j$ and $\mathbf{x}_k$, respectively, on the tangent plane. Then, an ordered upwind approximation of the tentative arrival time $V_{\mathbf{x}_j,\mathbf{x}_k}(\mathbf{x})$ to $\mathbf{x}$ through the projected simplex $\mathbf{x}\bar{\mathbf{x}}_j\bar{\mathbf{x}}_k$ on the tangent plane can be computed as follows:

$$V_{\mathbf{x}_j,\mathbf{x}_k}(\mathbf{x}) = \min_{t\in[0,1]} \left\{ \frac{\tau(t)}{f(\mathbf{x},\mathbf{a})} + t u(\mathbf{x}_j) + (1-t) u(\mathbf{x}_k) \right\}, \tag{5}$$

where $\tau(t) = \|\tilde{\mathbf{x}} - \mathbf{x}\| = \|(t\bar{\mathbf{x}}_j + (1-t)\bar{\mathbf{x}}_k) - \mathbf{x}\|$ and $\mathbf{a} = \frac{\tilde{\mathbf{x}}-\mathbf{x}}{\tau(t)}$. Appendix B, available in the online supplemental material, describes how to compute the tentative value $V_{\mathbf{x}_j,\mathbf{x}_k}(\mathbf{x})$ using a speed function $f(\mathbf{x},\mathbf{a})$ based on the normal curvature at a vertex (Section 5). In general, there are multiple line segments in *AF*, and thus the minimum of $V_{\mathbf{x}_j,\mathbf{x}_k}$'s over all line segments in *AF* is chosen for the tentative arrival time $V(\mathbf{x})$ to $\mathbf{x}$ (see (1) and (2) in Appendix A, available in the online supplemental material).

For a vertex $\mathbf{x} \in \textit{Considered}$, the part of *AF* that affects the arrival time to $\mathbf{x}$ turns out to be a portion of *AF* near $\mathbf{x}$. Denoting this part as the near front $NF(\mathbf{x})$, we define

$$NF(\mathbf{x}) = \{\mathbf{x}_j\mathbf{x}_k \in AF | \exists\, \tilde{\mathbf{x}} \text{ on } \mathbf{x}_j\mathbf{x}_k \text{ such that } \|\tilde{\mathbf{x}} - \mathbf{x}\| < h\gamma\},$$

where h is a bandwidth computed using (4) and γ is the anisotropy ratio f_2/f_1, for the bounding constants f_1 and f_2 of the speed function f defined in (3). Fig. 3b shows the $NF(\mathbf{x})$ for a vertex $\mathbf{x}$ in *Considered*. In Fig. 3b, every line

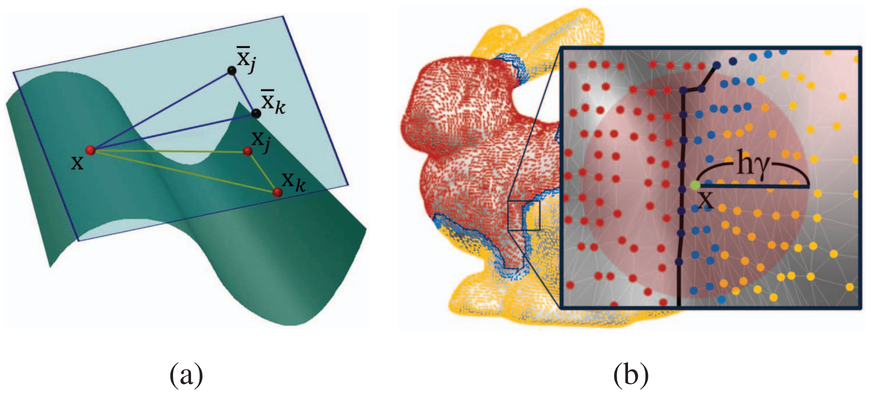


Fig. 3. Tentative arrival time computation: (a) A line segment $\mathbf{x}_j\mathbf{x}_k$ in the set *AF* is projected onto the tangent plane at $\mathbf{x}$. (b) The near front $NF(\mathbf{x})$ for a vertex $\mathbf{x}$ is represented in bold lines.

segment $\mathbf{x}_j\mathbf{x}_k \in NF(\mathbf{x})$ is represented in bold lines. Both h and γ are computed once in a preprocessing step. Since the arrival time to $\mathbf{x}$ is affected only by the line segments in $NF(\mathbf{x})$, we could discard a large portion of *AF* except $NF(\mathbf{x})$ when updating the tentative arrival time $V(\mathbf{x})$.

Proximity-based construction of *Considered*. We explain a new method for constructing the set *Considered* based on proximity among vertices on the surface. We modify the original definition of the set *Considered* to make our method for AG map construction triangulation-invariant. As described in **Algorithm 1**, the accepted front *AF* propagates gradually by moving a vertex in *Considered* to *Accepted*, one by one, where the tentative arrival times are used as the criteria to choose the next vertex. The vertices in the set *Considered* are candidates for the next closest vertex to *AF*. In order to achieve triangulation-invariance in front propagation, we newly define a *candidate set* for a line segment in AF based on its proximity with adjacent vertices in *Far*.

While the solver is being executed, the two vertex sets *Considered* and *AF* are boundaries of *Far* and *Accepted*, respectively. Since there are no vertices between *Considered* and *AF*, each vertex in *Considered* forms one or more *empty triangles* together with line segments in *AF* as illustrated in Fig. 4: we define an empty triangle for a vertex $\mathbf{x}$ as a triangle that contains no neighbor vertices projected on the tangent plane at $\mathbf{x}$. In Fig. 4, for example, a vertex $\mathbf{x}_4$ forms empty triangles together with line segments $\mathbf{x}_i\mathbf{x}_j$ and $\mathbf{x}_j\mathbf{x}_k$ in *AF*. Given the accepted front *AF*, we define a candidate set C_{jk} for a line segment $\mathbf{x}_j\mathbf{x}_k$ in *AF* to be a set of vertices on the surface that form empty triangles with $\mathbf{x}_j\mathbf{x}_k$. Specifically, the candidate set construction method consists of two steps: neighbor vertex projection and empty triangle check. First, the neighbor vertices of both $\mathbf{x}_j$ and $\mathbf{x}_k$ are projected onto the tangent plane at each neighbor vertex $\mathbf{x}$. Second, an empty triangle check is done for $\mathbf{x}$ by testing if a triangle connecting $\mathbf{x}$, $\mathbf{x}_j$, and $\mathbf{x}_k$ contains any of the projected neighbor vertices on the tangent plane at $\mathbf{x}$. In Fig. 4, for example, $C_{jk} = \{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$ since every vertex $\mathbf{x}_l, l = 1, 4, 5$, forms empty triangles $\mathbf{x}_l\mathbf{x}_j\mathbf{x}_k$ with the line segment $\mathbf{x}_j\mathbf{x}_k$. Once the candidate sets C_{jk} are constructed for all line segments $\mathbf{x}_j\mathbf{x}_k$ in *AF*, the set *Considered* is finally constructed as their union.

A candidate set C_{jk} encodes the proximity information with respect to $\mathbf{x}_j\mathbf{x}_k$. Each candidate set may have multiple vertices in *Considered*, and also two different candidate sets may have the same vertices as elements. For example, $C_{ij} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and $C_{jk} = \{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$ in Fig. 4. For the original version of the OUM-based solver, however, every candidate set is reduced to a single vertex on the surface which would restrict the choice of the next vertex that moves to *Accepted*. The triangulation-invariant solver is allowed to use more candidates for the next vertex, which leads to a better solution for a given surface mesh. On the other hand, we have to pay an extra cost for constructing the candidate sets. Specifically, an empty triangle check requires projection of the neighbors of $\mathbf{x}_j$ or $\mathbf{x}_k$ onto the tangent plane at each of the neighbors when constructing C_{jk}. However, the candidate set construction is a local operation on the neighbors of $\mathbf{x}_j$ or $\mathbf{x}_k$, which can be performed on the fly.

Updating *AF* and *Considered*. *AF* and *Considered* are updated based on the candidate sets. Note again that *Considered* is the union of the candidate sets C_{jk} for all $\mathbf{x}_j\mathbf{x}_k$ in *AF*. Suppose that a vertex $\mathbf{x}_*$ in *Considered* is newly accepted. Then, our solver identifies the line segment $\mathbf{x}_j\mathbf{x}_k$ in *AF* that is closest to $\mathbf{x}_*$ in euclidean metric, removes this line segment from *AF*, and adds $\mathbf{x}_*\mathbf{x}_j$ and $\mathbf{x}_*\mathbf{x}_k$ to *AF*. Accordingly, the vertices in C_{jk} is removed from *Considered*,

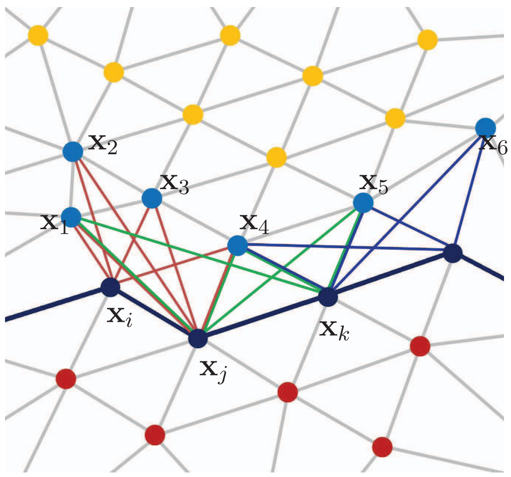


Fig. 4. Candidate sets for edges in AF: $C_{ij} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and $C_{jk} = \{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$. Color scheme: *Far* (yellow), *Considered* (blue), *Accepted* (red), and *AF* (bold lines).

and those in C_{*j} or C_{*k} are added to it. For example, suppose that $\mathbf{x}_4$ in *Considered* is newly accepted in Fig. 4. Then, the line segment $\mathbf{x}_j\mathbf{x}_k$ is removed from AF while $\mathbf{x}_j\mathbf{x}_4$ and $\mathbf{x}_4\mathbf{x}_k$ are added to AF. Accordingly, the solver removes the vertices $\{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$ in C_{jk} from *Considered*, computes candidate sets C_{j4} and C_{4k} for the newly added line segments to AF, and finally adds the vertices in C_{j4} and C_{4k} to *Considered*.

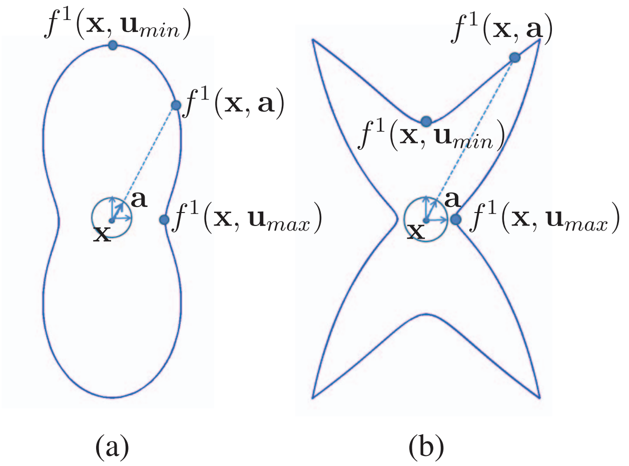
Convergence of the solver. As illustrated in Fig. 4, each vertex in a candidate set forms an empty triangle with a line segment in *AF*. Therefore, the candidate sets provide the information on all possible triangles formed by the vertices in *Considered* together with the line segments in *AF*. Therefore, the triangulation-invariant solver implicitly enumerates all possible triangulations, in a local sense, by building *AF* and *Considered* as explained in this section. In [15], the authors proved that the OUM-based solver converges to the viscosity solution in R^n, and also noted that their solver can be adapted to a triangulated 2-manifold. Since the triangulation-invariant solver locally enumerates all possible triangulations of a given surface, the convergence of our solver follows directly from that of the original OUM-based solver.

We finally analyze the time complexity of our triangulation-invariant solver. For updating the tentative arrival time $V(\mathbf{x})$ of each vertex $\mathbf{x}$ in *Considered* (steps 4 and 9 in **Algorithm 1**), we need to evaluate $V_{\mathbf{x}_j,\mathbf{x}_k}(\mathbf{x})$ for every $\mathbf{x}_j\mathbf{x}_k \in NF(\mathbf{x})$, where the size of $NF(\mathbf{x})$ is asymptotically proportional to $h^2\gamma^2$. Assuming that the surface mesh has M vertices, the main loop of our solver (step 5) has the time complexity of $O(h^2\gamma^2 M log M)$ since there are a total of M vertices to accept, and at every such iteration we need to reevaluate $V(\mathbf{x})$ at most $h^2\gamma^2$ times. A factor of $log M$ is for maintaining an ordering of *Considered* in terms of the tentative arrival time V.

5 Curvature-Based Speed Functions

Given a general H-J-B PDE solver on a surface mesh, a speed function $f(\mathbf{x}, \mathbf{a})$ (2) can be chosen in an application-dependent manner. For surface analysis and processing (Section 6), we propose two curvature-based speed functions based on classical curvature tensors on a 2-manifold.

Curvature-minimizing speed function. Consider a vehicle that moves on a surface. It will move fast on a low curvature region of the surface while moving slowly on a high curvature region. In order to capture this intuition, the speed function f should reflect the curvature characteristics of the surface. Let $\kappa_{\mathbf{a}}(\mathbf{x})$ be the normal curvature in the direction of a unit vector $\mathbf{a}$ on the tangent plane at a vertex $\mathbf{x}$ on a surface. Then,



$f^1(\mathbf{x}, \mathbf{u}_{min})$
$f^1(\mathbf{x}, \mathbf{a})$
$\mathbf{a}$
$\mathbf{x}$
$f^1(\mathbf{x}, \mathbf{u}_{max})$
$f^1(\mathbf{x}, \mathbf{a})$
$f^1(\mathbf{x}, \mathbf{u}_{min})$
$\mathbf{a}$
$\mathbf{x}$
$f^1(\mathbf{x}, \mathbf{u}_{max})$
(a)
(b)

Fig. 5. The curvature-minimizing speed function f^1 with $B = 0.4$ at an elliptic point (a) and a hyperbolic point (b).

$$\kappa_{\mathbf{a}}(\mathbf{x}) = \mathbf{a}^T \mathcal{N} \mathbf{a},$$

where $\mathcal{N}$ is the second fundamental tensor. The symmetric matrix $\mathcal{N}$ can be diagonalized using the principal curvature directions, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$, giving the minimum and maximum curvatures, κ_{min} and κ_{max}, respectively, to yield

$$\mathcal{N} = [\mathbf{u}_{min} \;\; \mathbf{u}_{max}] \begin{bmatrix} \kappa_{min} & 0 \\ 0 & \kappa_{max} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{min}^T \\ \mathbf{u}_{max}^T \end{bmatrix}. \quad (6)$$

We now define the speed function using the magnitude of $\kappa_{\mathbf{a}}(\mathbf{x})$ at vertex $\mathbf{x}$ as follows:

$$f^1(\mathbf{x}, \mathbf{a}) = \exp^{-B|\kappa_{\mathbf{a}}(\mathbf{x})|}, \quad (7)$$

for user-provided constant B. Fig. 5 shows the speed functions in terms of the unit vector $\mathbf{a}$ on the tangent plane at a vertex $\mathbf{x}$ on a surface. Here, B controls the anisotropy of the speed function. Fig. 6 illustrates how B affects AG paths. Specifically, large B results in high anisotropy ratio γ. In an extreme case ($B = 0$), the curvature-minimizing speed

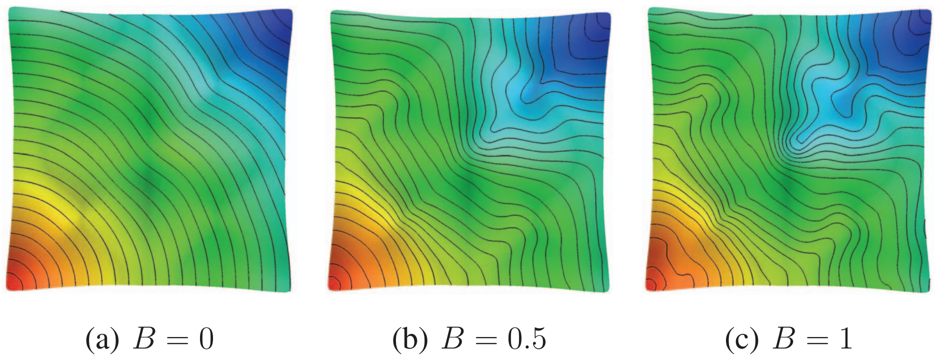


Fig. 6. Level sets of the AG map for different constants B. High B results in high anisotropy ratio γ. The distance metric becomes isotropic when $B = 0$.

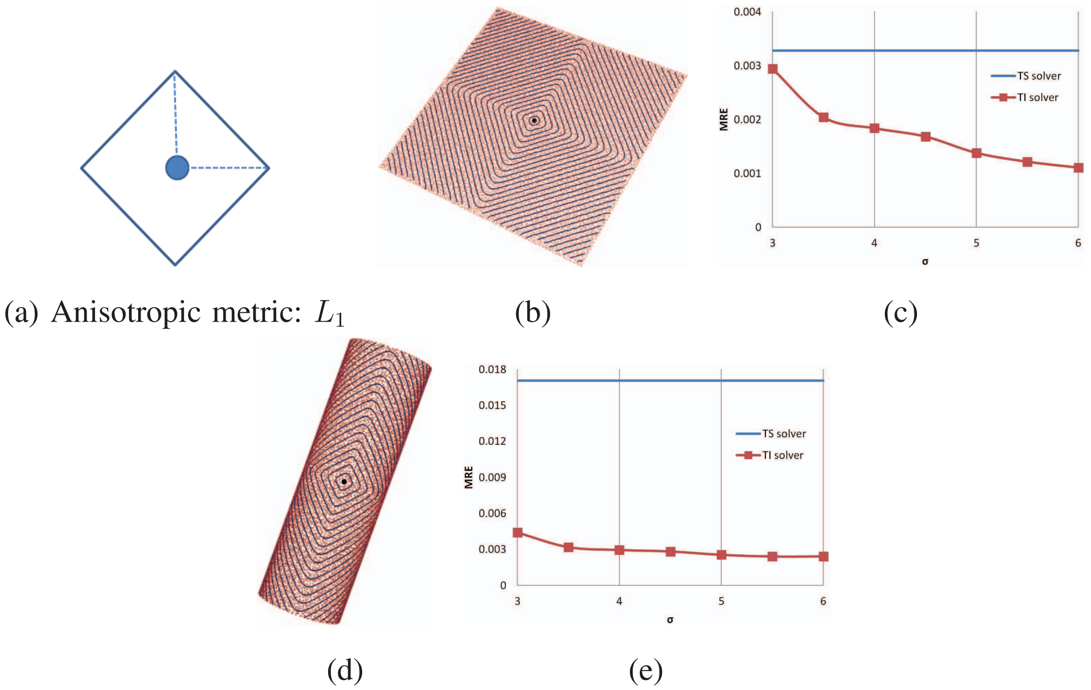


Fig. 7. Average errors by both the triangulation-invariant (TI) solver and the triangulation-specific (TS) solver are shown for various values of σ under the L_1 metric.

function f^1 is reduced to a constant, which gives rise to an isotropic metric. Recently, Seong et al. [36] presented a similar curvature-based anisotropic speed function. However, their speed function is not general since it cannot handle a hyperbolic point on the surface.

Curvature variation-minimizing speed function. Derivatives of curvature have been used to characterize interesting surface features such as creases [40] and suggestive contours [41]. The variations of curvature tend to be minimized along such features. Inspired by this observation, we derive curvature variation-minimizing curves on a surface. Let $\mathbf{C}$ be the derivative of the normal curvature tensor $\mathcal{N}$ defined in (6). Then, $\mathbf{C}$ can be represented as a $2 \times 2 \times 2$ rank-3 tensor as follows [12]:

$$\mathbf{C} = [D_{\mathbf{u}}\mathcal{N} \;\; D_{\mathbf{w}}\mathcal{N}] = \left[\begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{bmatrix} b & c \\ c & d \end{bmatrix}\right], \tag{8}$$

where $\mathbf{u}$ and $\mathbf{w}$ are a pair of orthogonal vectors defining a tangent frame. $\mathbf{C}$ is a symmetric tensor having only four unique entries, $a = D_{\mathbf{u}_{min}}\kappa_{min}$, $b = D_{\mathbf{u}_{max}}\kappa_{min}$, $c = D_{\mathbf{u}_{min}}\kappa_{max}$, and $d = D_{\mathbf{u}_{max}}\kappa_{max}$. A similar curvature variation was estimated on a parametric surface by measuring the difference of curvature tensors at neighbor points [36]. It depends, however, on a global parametrization of parametric surfaces since it requires a regular grid for neighbor information.

The directional derivative of curvature in the direction of $\mathbf{a}$, denoted by $\mathbf{C}(\mathbf{a}\cdot\mathbf{a}\cdot\mathbf{a})$, is computed by multiplying the tensor $\mathbf{C}$ with $\mathbf{a}$ three times. $\mathbf{C}(\mathbf{a}\cdot\mathbf{a}\cdot\mathbf{a})$ gives the curvature variation along the direction $\mathbf{a}$. Similarly to (7), the curvature variation-minimizing speed function is defined using the magnitude of $\mathbf{C}(\mathbf{a}\cdot\mathbf{a}\cdot\mathbf{a})$

$$f^2(\mathbf{x}, \mathbf{a}) = \exp^{-B|\mathbf{C}(\mathbf{a}\cdot\mathbf{a}\cdot\mathbf{a})|}. \tag{9}$$

With the speed functions f^1 and f^2, the AG map for a surface mesh is constructed with the speed highly adaptive to the surface geometry. By the way in which f^1 is defined, a solution $u(\mathbf{x})$ of a H-J-B PDE (2) tends to minimize the normal curvature along the AG curve from the seed points to $\mathbf{x}$, where the curvature is measured in the curve's tangent direction. With the speed function f^2, a solution $u(\mathbf{x})$ of (2) tends to minimize the curvature variation along the AG curve. As shown in Fig. 1, the AG path under the curvature-minimizing speed f^1 (Fig. 1b) differs from the conventional (isotropic) geodesic curve under euclidean metric (Fig. 1a). Fig. 1c shows an example of two AG paths, ridge, and valley curves under the curvature variation-minimizing speed f^2. The ridge and valley curves are marked red and blue, respectively.

6 Results

This section consists of three parts: we first validate the triangulation-invariant (TI) solver (Section 4) and provide timing data on AG map construction for various models with this solver. We then show geometric structures extracted from AG maps. All experiments were performed on a PC equipped with an Intel Core2 Duo 2.6 GHz CPU with 2.75 GB memory.

6.1 Validation of the Triangulation-Invariant Solver

In order to demonstrate the accuracy of the triangulation-invariant (TI) solver in comparison to the triangulation-specific (TS) solver (the OUM-based solver [15] with fixed connectivity information), we performed two experiments with different surface meshes: a rectangular surface in the plane and a cylindrical surface (Figs. 7 and 8). We first built the former surface by sampling a set of 22 K points from a rectangular region and performed a Delaunay triangulation

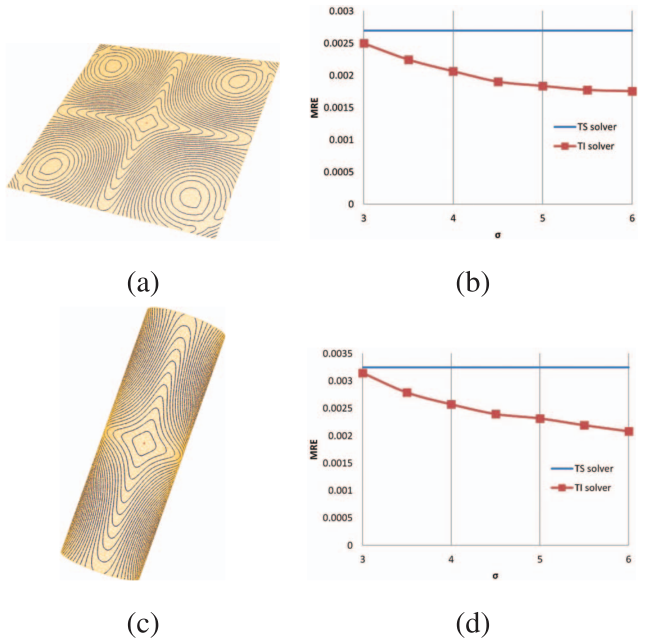


Fig. 8. Average errors by both the triangulation-invariant (TI) solver and the triangulation-specific (TS) solver are shown for various values of σ under the G metric.

algorithm to get a surface mesh [42]. We then created the latter by bending the former and stitching a pair of its boundaries. Thus, there is a one-to-one correspondence between the vertices of the two surfaces. For each experiment, we used two different anisotropic speed functions: the L_1 metric and a speed function in [15]. For convenience, the latter is referred to as the G metric. For the L_1 metric, we set the speed function $f(\mathbf{x}, \mathbf{a})$ in a direction $\mathbf{a}$ at each vertex $\mathbf{x}$ to be equal to the radius of the L_1-unit circle centered at $\mathbf{x}$. As illustrated in Fig. 7a, the radius of this circle varies with respect to the direction $\mathbf{a}$. We chose L_1 metric as the anisotropic metric for ease of error analysis since it gives the analytic solution for a H-J-B PDE on each of the two surfaces, which we used as the ground truth value. For the G metric, $f(\mathbf{x}, \mathbf{a}) = (1 + (\nabla g(\mathbf{x}) \cdot \mathbf{a})^2)^{-\frac{1}{2}}$, where $g(\mathbf{x}) = 0.75 \sin(3\pi x_1) \sin(3\pi x_2)$ for $\mathbf{x} = (x_1, x_2)$. As in [15], the AG maps constructed from finer surface meshes (1,000 K) were regarded as ground truths for the second speed function.

For the rectangular surface with 22 K vertices on it, we constructed the AG map using the TS solver with the fixed connectivity information. We also constructed a series of AG maps independently using the TI solver by varying the coefficient σ in (4) while fixing the number of vertices on the surface (22 K). Since the radius of the sphere (h in the equation) that determines the neighbors of a vertex is proportional to σ, the number of neighbors increases as σ gets large. For each of these AG maps, we estimated the mean relative error (MRE) over all vertices on the surface. The error for each vertex is measured as the relative deviation of the solution from the ground truth value. Fig. 7c plots the estimated errors by the two solvers under the L_1 metric. Blue and red curves in the figure exhibit the errors by the TS and TI solvers, respectively. The same experiment was repeated for the cylindrical surface under

TABLE 1
Timing Data for Numerical Error Estimation

metrics	models	TS solver	TI solver
L_1 metric (Figure 7)	Plane	9.15	15.95
	Cylinder	9.29	17.36
G metric (Figure 8)	Plane	21.54	33.84
	Cylinder	28.74	38.54

the L_1 metric and Fig. 7e shows a plot of estimated errors. Table 1 summarizes the timing data. We set $\sigma = 4$ for measuring the execution time of the TI solver. The experimental results under the other speed function were consistent with those under the L_1 metric for both the planar and the cylindrical surfaces (Fig. 8 and Table 1).

The results on both the rectangular and the cylindrical surfaces supported our intuition on the TI solver. As σ increases while fixing the number of vertices on a surface, the number of the neighbors of each vertex increases accordingly. Thus, the solver becomes more accurate since it enumerates more candidates for the next vertex that moves to the set *Accepted* at each time step, as mentioned in Section 4. However, we have to pay cost for accuracy. Specifically, the TI solver spent 1.2-2 times more time than the TS solver for both surfaces.

6.2 AG Map Construction

We performed an experiment to collect timing data for constructing the AG maps for eight models: Rocker Arm, Bunny, Armadillo Head, Hand, MIT Face, Leaf, Santa, and Igea. The AG map construction consists of two steps: preprocessing and solving H-J-B PDEs (Section 4).

In the preprocessing step, a speed function was computed by estimating a curvature tensor on the surface. We computed the curvature tensor to build a speed function using a library CGAL [43]. Let f_1 and f_2 be the minimum and maximum speed function values over all surface vertices, respectively. Then, the anisotropy ratio γ was estimated as f_2/f_1.

Given a set of seed points together with a speed function and an anisotropy ratio γ, the proposed TI solver was applied to the surface in order to compute an AG map. The resulting AG map contains the solution of a H-J-B PDE at every surface vertex, that is, the arrival time $u(\mathbf{x})$ to each vertex $\mathbf{x}$. Fig. 9 shows that the AG map on various surface meshes under a curvature-minimizing speed f^1. We used a single seed point for each model and set $B = 0.05$. In Fig. 9, the seed point of a model is represented as a black sphere. A color map visualizes the distance (time) from a seed. Red and blue colors represent "close to" and "far from" the seed, respectively, while yellow does "inbetween." Table 2 summarizes the models used in all examples and timing data.

6.3 Geometric Structure Extraction

We performed an experiment to extract geometric structures from AG maps on surface meshes: AG paths,

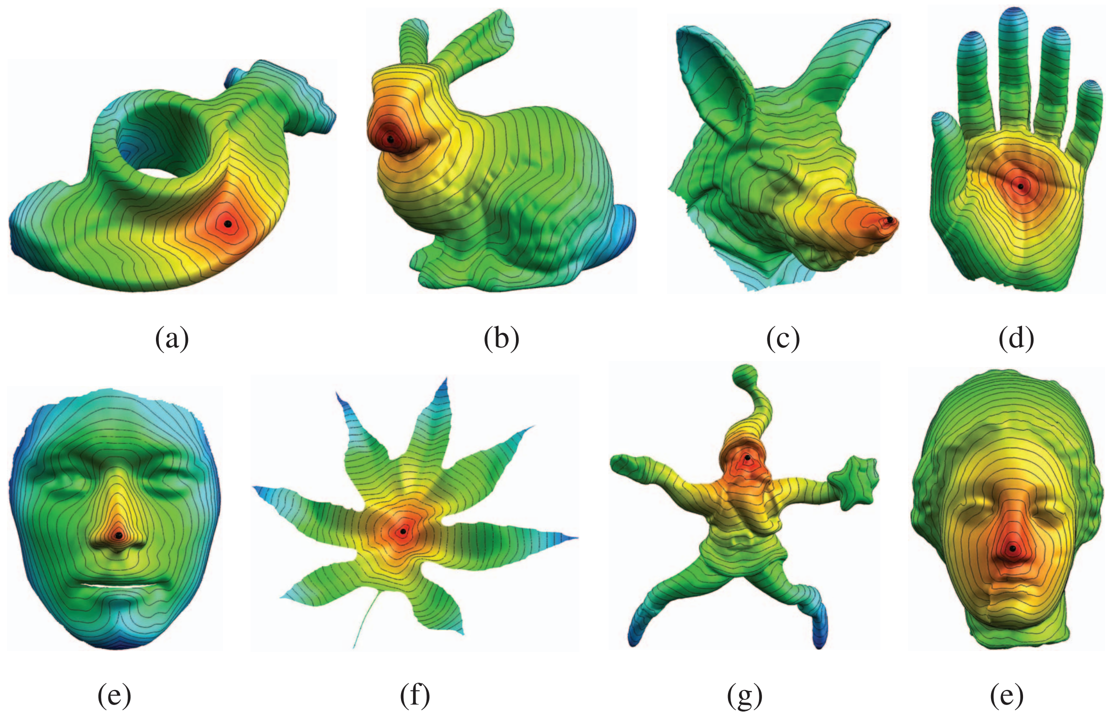


Fig. 9. AG map construction from a seed point.

isocontours, offset curves, medial axes, and ridges/valleys. To extract such structures, we developed a simple algorithm for tracing a curve on a surface mesh based on a piecewise linear approximation.

AG paths. An AG path was traced back from a query vertex $\mathbf{x}$ to a seed point, guided by the AG map. Given the query vertex $\mathbf{x}$, the geodesic path was followed in the characteristic direction $W(\mathbf{x})$ at this vertex, which was computed while constructing the AG map (Section 4). AG path tracing was done in three steps: sliding, projection, and characteristic direction interpolation. Given a query vertex $\mathbf{x}$, the sliding step moves on the tangent plane at $\mathbf{x}$ in the direction $W(\mathbf{x})$ by a small step size, $d = \min_i\{\sigma\sqrt{\pi s_i^2/n_i}\}$ (Section 4) to arrive at a point $\mathbf{x}'$. Then, $\mathbf{x}'$ is projected onto the surface. Finally, a characteristic direction at the projected point is estimated by averaging those at its neighbor vertices. This process was repeated until a seed point was reached.

Figs. 10a and 10b show curvature-minimizing geodesic curves on the Armadillo Head model using euclidean metric ($B = 0$) and the curvature-minimizing metric ($B = 2.5$), respectively. The geodesic curves from 20 random query vertices were traced using the respective AG maps on the input surface. In Fig. 10b, the AG paths (blue) reflect the anisotropic nature of the geometry better than those of Fig. 10a since the anisotropy is amplified for a large value of

TABLE 2
Models and Timing Data for AG Map Construction

models	$\lvert V\rvert$	metrics	B	pre. time	AG maps	total time
Rocker Arm	9,400	f^1	0.05	0.485	8.308	8.793
Bunny	20,002	f^1	0.05	0.953	15.959	16.912
Armadillo Head	20,246	f^1	0.05	0.938	16.211	17.149
Hand	22,303	f^1	0.05	1.301	19.151	20.182
MIT Face	28,299	f^1	0.05	1.157	24.280	25.437
Leaf	62,211	f^1	0.05	3.094	94.016	97.110
Santa	75,781	f^1	0.05	3.406	149.082	152.488
Igea	134,345	f^1	0.05	22.188	371.636	393.824

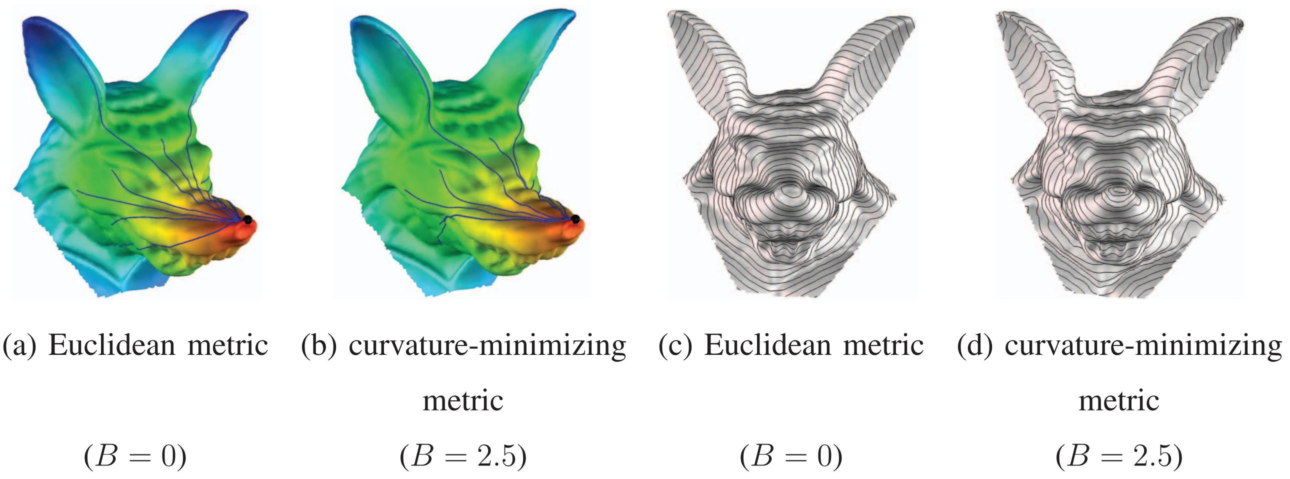


Fig. 10. AG paths computation (a) and (b). Isocontours extraction (c) and (d).

TABLE 3
Models and Timing Data for Geometric Structure Extraction

structures	models	metrics	B	# of seeds	AG map	extraction
AG paths	Armadillo Head	f^1	2.5	1	16.211	0.219
contours	Armadillo Head	f^1	2.5	1	16.211	30.391
	Igea	f^1	0.0035	1	371.636	255.36
offset curves	MIT Face	f^1	1.2	400	24.280	45.016
medial axis	Leaf	f^1	0.5	2,542	94.016	0.375
ridges/valleys	Rocker Arm	f^2	1.0	49	151.5	1.55
	Igea	f^2	0.035	319	251.9	9.3

B. The curvature-minimizing AG paths from several vertices on the surface may follow the narrow valley to avoid high curvature regions. In such a case, the approximate AG paths look like merging with each other due to the approximate nature of the AG path tracing algorithm. Model and timing data are given in Table 3. Most of computation time was spent in the AG map construction.

Isocontours. The level set of the AG map was visualized using a series of equidistance curves from the seed points. In this context, a geodesic distance map was regarded as a geodesic distance function $u(\mathbf{x})$ sampled at the vertices on the surface. Given the constructed AG map from the seed points, our approach to isocontour tracing is based on the AG paths extracted from the seed points to every vertex on the surface. Fig. 11a shows an example of extracted AG paths from a single seed point. We then generate a series of isocontours by exploiting the AG paths. Appendix C, available in the online supplemental material, describes a first order approximation method for isocontour generation. Fig. 11b presents an example of isocontours traced using this method.

Figs. 10c and 10d show two sets of equidistance curves under different distance metrics on the Armadillo Head model: isocontours under euclidean metric (Fig. 10c) and those under the curvature-minimizing metric f^1 (Fig. 10d). In the latter, feature-aligned isocontours were observed in high curvature regions, which reflect the local geometry of the model. This supports our intuition on the curvature-minimizing speed f^1 for surface analysis and processing. Timing data are available in Table 3.

Offset curves and medial axes. Equidistance curves arise naturally in computing geometric structures such as geodesic offset curves [5] and geodesic medial axes. Using

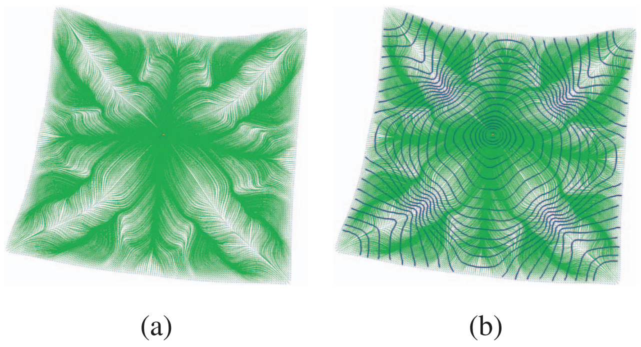


Fig. 11. Isocontour tracing: AG paths from the seed points to every vertex on the surface were extracted (a). Each isocontour of the AG map was traced from sampled points on the AG paths.

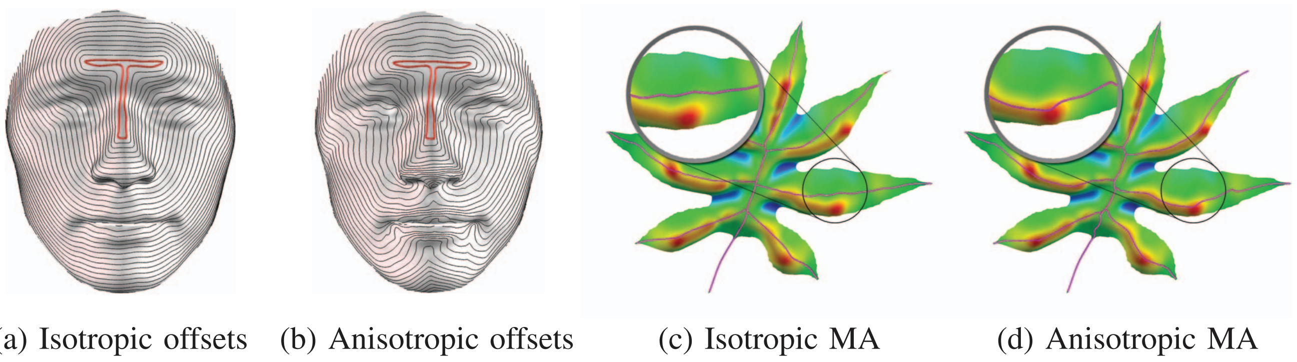


Fig. 12. Geodesic offset curves and geodesic medial axis are constructed under two different distance metrics: euclidean metric ((a) and (c)) and curvature-minimizing metric ((b) and (d)). In (c) and (d), colors represent mean curvature of the Leaf model (red to blue: inward folding to outward folding).

the curvature-minimizing speed function f^1, these structures can be generalized: Figs. 12a and 12c show the geodesic offset curves and a medial axis under euclidean metric, respectively, while Figs. 12b and 12d exhibit the same structures under the anisotropic metric. The offset curves were obtained by sampling a series of equidistance points from a connected set of seed points. The medial axis of a surface was extracted by identifying a set of vertices in a geodesic map at which the closest neighbors on the boundary change abruptly. We employed the method of Hamilton-Jacobi skeleton [44], [45] to extract the geodesic medial axis of a surface. In the first example (Fig. 12), the offset curves nicely trace along feature lines under the anisotropic distance metric (Figs. 12b). In the second example (Fig. 12), the AG medial axis reflects the geometry of the Leaf model quite well under the curvature-minimizing metric (Fig. 12d). In Figs. 12c and 12d, colors represent mean curvatures of the surface model: the red color corresponds to a valley region with high curvature. The anisotropic medial axis captures local valleys of the Leaf model better than the isotropic medial axis.

Ridges and valleys. Finally, we show that ridge/valley curves can be extracted from AG maps on surface meshes. A ridge or valley is defined to be a curve along which the derivative of a principal curvature vanishes. Thus, the derivative of the principal curvature in the tangent direction is zero at every point on the curve so as to follow a curvature variation-minimizing curve with the speed function f^2 as defined in Section 5.

The ridge/valley extraction consists of three steps: initialization, AG map construction, and curves tracing. A pseudocode for the extraction method is given in **Algorithm 2** (see Appendix A, available in the online supplemental material). Steps 2-6 are for initialization, step 7 is for AG map construction, and steps 8-11 are for curves tracing. For initialization, a set of seed vertices were acquired by sampling the points on the surface, where the derivative of a principal curvature vanishes. We employed a method presented in [46] in order to sample the ridge/valley vertices: for noise removal the sample points with smaller mean curvature were discarded using a user-defined threshold, and every connected ridge/valley vertices were grouped together. Then, a seed vertex is selected for each group so that the seed vertex has the maximum value of the mean curvature in the group. Figs. 13a, 13b, and 13c show the input model, the ridge/valley vertices, and the groups with

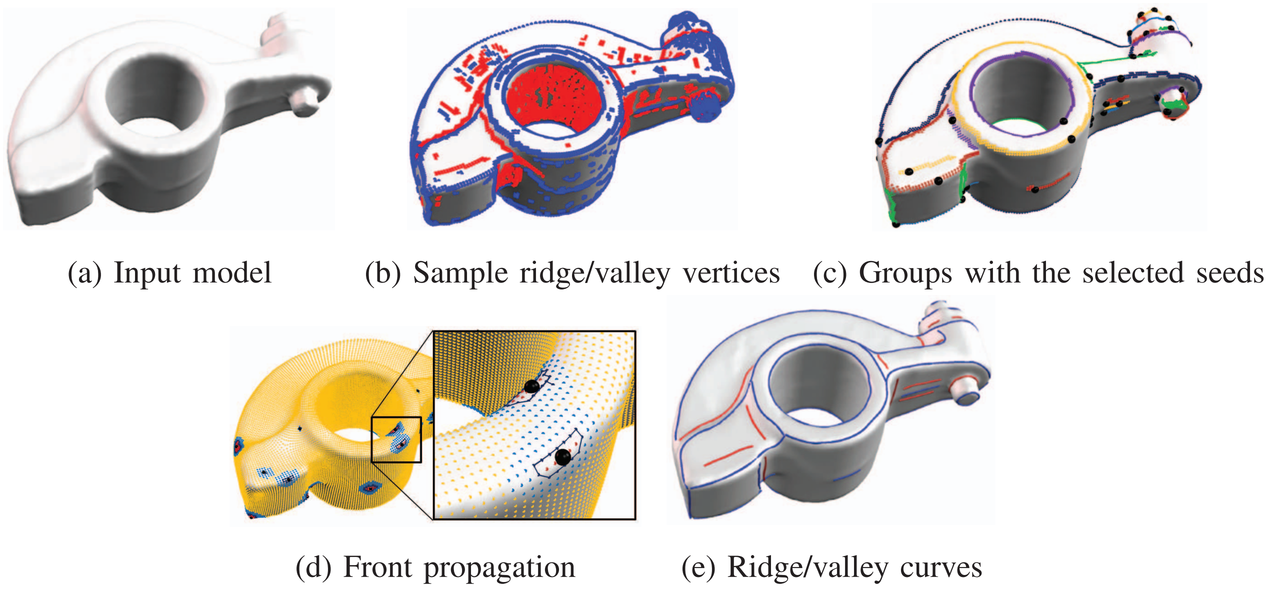

(a) Input model
(b) Sample ridge/valley vertices
(c) Groups with the selected seeds
(d) Front propagation
(e) Ridge/valley curves

Fig. 13. Ridge/valley curves extraction under a curvature variation-minimizing metric.

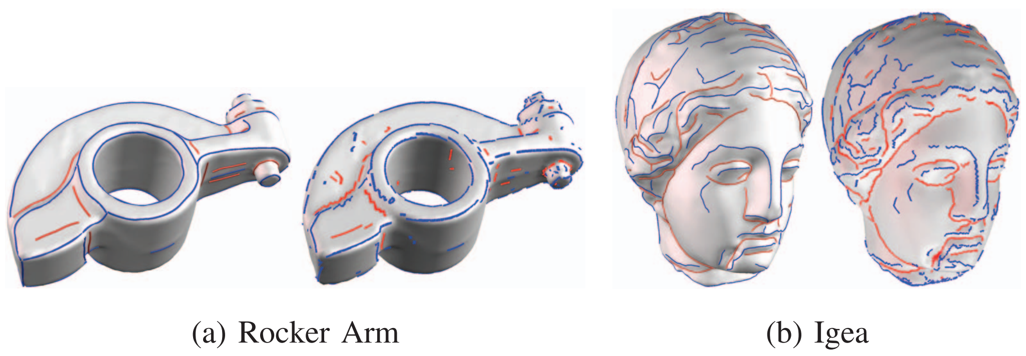


Fig. 14. Ridge (blue) and valley (red) curves.

the selected seed vertices, respectively. In the second step, an AG map was built from these seed vertices under the curvature variation-minimizing metric. Fig. 13d shows a snap shot of front propagation during the AG map construction. Finally, a ridge or valley curve was extracted from a query vertex by tracing back the curvature variation-minimizing curve from it guided by this map (Fig. 13e). In our ridge/valley extraction method, a user manually selects a query vertex for each group of the ridge/valley vertices. Fig. 14 shows examples of ridge/valley curves from complex models, Rocker Arm and Igea. Ridges and valleys are marked blue and red, respectively. The last two rows of Table 3 show timing data for the ridge/valley extraction. For visual comparison, Fig. 14 also shows the ridge/valley curves generated by a different method [46].

6.4 Discussion

The local neighbors of a vertex provide its proximity information on a surface, which plays the central role in the triangulation-invariant solver. These are contained within distance h from the vertex in a geodesic manner. We provide a heuristic method to determine h in Section 4. However, h should be computed in a more principled manner to guarantee the sufficient proximity information for every vertex while minimizing the number of the neighbors.

Another limitation is stemmed from the way of setting parameters B for curvature-based speed functions. In our experiments, we manually adjusted B case by case, which is time consuming. AG distances depend greatly on these parameters. Moreover, B determines the anisotropy ratio γ, which directly affects the efficiency of our OUM-based solver since the set NF becomes large as the anisotropy ratio γ increases. Thus, a more principled way is needed again to choose B depending on applications.

Since our method constructs the AG map in a triangulation-invariant manner, it can be extended for surfaces represented in point clouds. For point clouds, determining the neighbors of each point in the data set is not a trivial task due to the lack of information on the underlying surface geometry. One could employ some recent method for optimal selection of the neighbors using a MLS surface. However, in order to show the potential of our triangulation-invariant solver, we modified our neighbor determination method such that the neighbor points are chosen based on the euclidean distance rather than the geodesic distance. The extended method was successfully applied to most point clouds data set except for some high-curvature regions with insufficient sample points or regions with topological errors. Fig. 15 shows the AG map on the Lucy model represented by 119 K points. The AG map was constructed well for surface regions (e.g., face) with dense sample points, while it has errors for sharp regions (e.g., right hand).

7 Conclusions and Future Work

We have presented a triangulation-invariant scheme for computing AG maps on surface meshes. Our approach builds the AG distance maps by numerically solving Hamilton-Jacobi-Bellman PDEs. We generalized the OUM-based solver for surface meshes. In particular, a triangulation-invariant algorithm has been proposed to compute the AG map exploiting the underlying geometry as much as possible rather than depending on a specific triangulation. For surface analysis and processing, we further proposed two curvature-related speed functions, which are intuitive and easy to compute. The effectiveness of our approach is

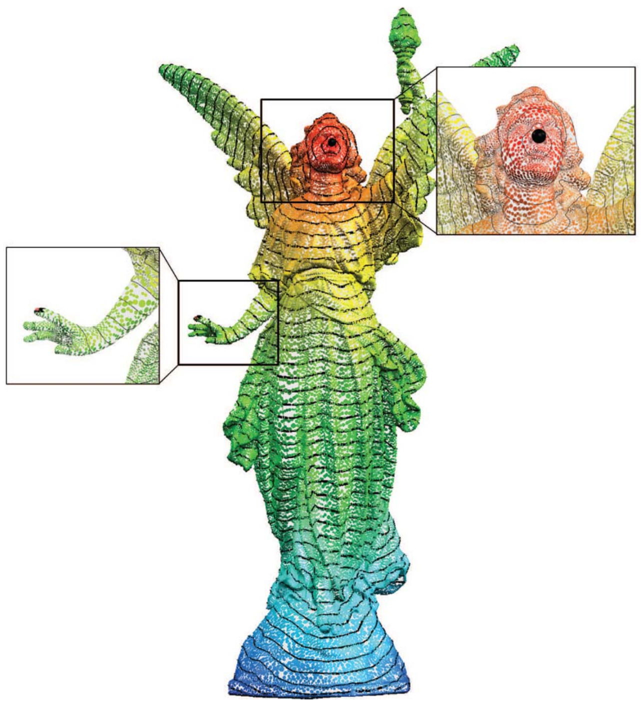


Fig. 15. We extended the triangulation-invariant solver for point clouds: a seed point is located at the tip of the nose and the color represents the AG map (red $\rightarrow$ blue: close $\rightarrow$ far).

demonstrated by extracting geometric structures, including isocontours, geodesic medial axes, and ridge/valley curves.

In the future, we wish to further explore surface analysis and processing using proposed curvature-related AG distance metrics. Reflecting the local geometric information of a surface, AG curves could be used as an effective means for distortion-minimizing surface parameterization. Also, we are currently investigating the possibility of the curvature-minimizing geodesic curves for surface segmentation.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-004157).

References

[1] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *Proc. ACM SIGGRAPH '03,* pp. 954-961, 2003.

[2] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by Example," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 652-663, 2004.

[3] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture Mapping Using Surface Flattening via Multidimensional Scaling," *IEEE Trans. Visualization and Computer Graphics,* vol. 8, no. 2, pp. 198-207, Apr.-June 2002.

[4] K. Zhou, J. Synder, B. Guo, and H.-Y. Shum, "Iso-Charts: Stretch-Driven Mesh Parameterization Using Spectral Analysis," *Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing (SGP '04),* pp. 45-54, 2004.

[5] G. Peyré and L.D. Cohen, "Geodesic Remeshing Using Front Propagation," *Int'l J. Computer Vision,* vol. 69, no. 1, pp. 145-156, 2006.

[6] O. Sifri, A. Sheffer, and C. Gotsman, "Geodesic-Based Surface Remeshing," *Proc. 12th Int'l. Meshing Roundtable,* pp. 189-199, 2003.

[7] Q. Du and D. Wang, "Anisotropic Centroidal Voronoi Tessellations and Their Applications," *SIAM J. Scientific Computing,* vol. 26, no. 3, pp. 737-761, 2005.

[8] A. Elad and R. Kimmel, "Bending Invariant Representations for Surfaces," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 168-174, 2001.

[9] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," *Proc. ACM SIGGRAPH '01,* pp. 203-212, 2001.

[10] A.M. Bronstein, M.M. Bronstein, and R. Kimmel, "Generalized Multidimensional Scaling: A Framework for Isometry-Invariant Partial Surface Matching," *Proc. Nat'l Academy of Sciences of USA,* vol. 103, no. 5, pp. 1168-1172, 2006.

[11] E. Pichon, C.-F. Westin, and A. Tannenbaum, "A Hamilton-Jacobi-Bellman Approach to High Angular Resolution Diffusion Tractography," *Proc. Eighth Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI '05),* pp. 180-187, 2005.

[12] S. Rusinkiewicz, "Estimating Curvatures and Their Derivatives on Triangle Meshes," *Proc. Second Int'l Symp. 3D Data Processing, Visualization and Transmission,* pp. 486-493, Sept. 2004.

[13] J.A. Sethian and A.M. Popovici, "3D Traveltime Computation Using the Fast Marching Method," *Geophysics,* vol. 64, no. 2, pp. 516-523, 2006.

[14] R. Kimmel and J. Sethian, "Computing Geodesic Paths on Manifolds," *Proc. Nat'l Academy of Sciences of USA,* vol. 95, no. 15, pp. 8431-8435, 1998.

[15] J.A. Sethian and A. Vladimirsky, "Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms," *SIAM J. Numerical Analysis,* vol. 41, no. 1, pp. 325-363, 2003.

[16] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, "The Discrete Geodesic Problem," *SIAM J. Computing,* vol. 16, no. 4, pp. 647-668, 1987.

[17] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, and H. Hoppe, "Fast Exact and Approximate Geodesics on Meshes," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 553-560, 2005.

[18] D. Bommes and L. Kobbelt, "Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes," *Proc. VMV '07,* pp. 151-160, 2007.

[19] J. Klein and G. Zachmann, "Point Cloud Surfaces using Geometric Proximity Graphs," *Computers and Graphics,* vol. 28, no. 6, pp. 839-850, 2004.

[20] M. Hofer and H. Pottmann, "Energy-Minimizing Splines in Manifolds," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 284-293, 2004.

[21] M.R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati, "Approximating Geodesics on Point Set Surfaces," *Proc. IEEE/Eurographics Symp. Point-Based Graphics,* pp. 85-93, 2006.

[22] J.A. Sethian, "A Fast Marching Level Set Method for Monotonically Advancing Fronts," *Proc. Nat'l Academy of Sciences of USA,* vol. 93, pp. 1591-1595, 1996.

[23] F. Qin, Y. Luo, K. Olsen, W. Cai, and G. Schuster, "Finite-Dierence Solution of the Eikonal Equation along Expanding Wavefronts," *Geophysics,* vol. 57, no. 3, pp. 478-487, 1992.

[24] J. Tsitsiklis, "Efficient Algorithms for Globally Optimal Trajectories," *IEEE Trans. Automatic Control,* vol. 40, no. 9, pp. 1528 -1538, Sept. 1995.

[25] F. Mémoli and G. Sapiro, "Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces: 730," *J. Computational Physics,* vol. 173, no. 2, p. 764, 2001.

[26] F. Memoli and G. Sapiro, "Distance Functions and Geodesics on Points Cloud," *Proc. IEEE Workshop Variational Geometric and Level Set Methods in Computer Vision,* 2003.

[27] A. Spira and R. Kimmel, "An Efficient Solution to the Eikonal Equation on Parametric Manifolds," *Interfaces and Free Boundaries,* vol. 6, pp. 315-327, 2003.

[28] O. Weber, Y.S. Devir, A.M. Bronstein, M.M. Bronstein, and R. Kimmel, "Parallel Algorithms for Approximation of Distance Maps on Parametric Surfaces," *ACM Trans. Graphics,* vol. 27, no. 4, pp. 1-16, 2008.

[29] P.-E. Danielsson, "Euclidean Distance Mapping," *Computer Graphics and Image Processing,* vol. 14, no. 3, pp. 227-248, 1980.

[30] M. Boue and P. Dupuis, "Markov Chain Approximations for Deterministic Control Problems with Affine Dynamics and Quadratic Cost in the Control," *Proc. IEEE 36th Conf. Decision and Control,* vol. 4, pp. 3424 -3429, Dec. 1997.

[31] J. Qian, Y.-T. Zhang, and H.-K. Zhao, "A Fast Sweeping Method for Static Convex Hamilton-Jacobi Equations," *J. Scientific Computing,* vol. 31, nos. 1/2, pp. 237-271, 2007.

[32] C.-Y. Kao, S. Osher, and J. Qian, "Legendre-Transform-Based Fast Sweeping Methods for Static Hamilton-Jacobi Equations on Triangulated Meshes," *J. Computational Physics,* vol. 227, no. 24, pp. 10 209-10 225, 2008.

[33] L. Polymenakos, D. Bertsekas, and J. Tsitsiklis, "Implementation of Efficient Algorithms for Globally Optimal Trajectories," *IEEE Trans. Automatic Control,* vol. 43, no. 2, pp. 278-283, Feb. 1998.

[34] F. Bornemann and C. Rasch, "Finite-Element Discretization of Static Hamilton-Jacobi Equations Based on a Local Variational Principle," *Computing and Visualization in Science,* vol. 9, pp. 57-69, 2006.

[35] W.-K. Jeong, P.T. Fletcher, R. Tao, and R. Whitaker, "Interactive Visualization of Volumetric White Matter Connectivity in DT-MRI Using a Parallel-Hardware Hamilton-Jacobi Solver," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1480-1487, Nov. 2007.

[36] J.-K. Seong, W.-K. Jeong, and E. Cohen, "Anisotropic Geodesic Distance Computation for Parametric Surfaces," *Proc. IEEE Int'l Conf. Shape Modeling and Applications,* pp. 179-186, June 2008.

[37] J.A. Sethian and A. Vladimirsky, "Fast Methods for the Eikonal and Related Hamilton?? Jacobi Equations on Unstructured Meshes," *Proc. Nat'l Academy of Sciences of the USA,* vol. 97, no. 11, pp. 5699-5703, 2000.

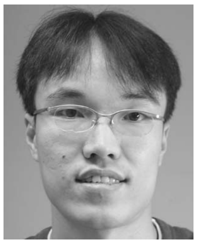
[38] E. Konukoglu, M. Sermesant, O. Clatz, J.-M. Peyrat, H. Delingette, and N. Ayache, "A Recursive Anisotropic Fast Marching Approach to Reaction Diffusion Equation: Application to Tumor Growth Modeling," *Proc. 20th Int'l Conf. Information Processing in Medical Imaging (IPMI '07),* pp. 687-699, 2007.

[39] H. Wang, C. Scheidegger, and C. Silva, "Optimal Bandwidth Selection for Mls Surfaces," *Proc. IEEE Int'l Conf. Shape Modeling and Applications,* pp. 111-120, June 2008.

[40] K. Watanabe and A. Belyaev, "Detection of Salient Curvature Features on Polygonal Surfaces," *Computer Graphics Forum,* vol. 20, pp. 385-392, 2001.

[41] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive Contours for Conveying Shape," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 848-855, 2003.
[42] N. Amenta, S. Choi, T.K. Dey, and N. Leekha, "A Simple Algorithm for Homeomorphic Surface Reconstruction," *Proc. 16th Ann. Symp. Computational Geometry (SCG '00),* pp. 213-222, 2000.
[43] CGAL, "Computational Geometry Algorithms Library," http://www.cgal.org, 2009.
[44] K. Siddiqi, S. Bouix, A. Tannenbaum, and S.W. Zucker, "Hamilton-Jacobi Skeletons," *Int'l J. Computer Vision,* vol. 48, no. 3, pp. 215-231, 2002.
[45] Y. Shi, P. Thompson, I. Dinov, and A. Toga, "Hamilton? Jacobi Skeleton on Cortical Surfaces," *IEEE Trans. Medical Imaging,* vol. 27, no. 5, pp. 664-673, May 2008.
[46] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Ridge-Valley Lines on Meshes via Implicit Surface Fitting," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 609-612, 2004.

Sang Wook Yoo received the BS degree in computer science from Sogang University, Korea, in 2005 and the MS degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 2008. He is currently working toward the PhD degree at the KAIST. His research interests include computer graphics, computational brain imaging, and machine learning.



Joon-Kyung Seong received the BS and the PhD degrees from Seoul National University in 2000 and 2005, respectively. He was a postdoctoral fellow in the School of Computing at the University of Utah from 2005 to 2008. After that he was a research professor in the department of computer science at Korea Advanced Institute of Science and Technology (KAIST). He is an assistant professor in the school of computer science and engineering at Soongsil University, Korea. His research interests include computer graphics, geometric modeling, and computational brain imaging.

Min-Hyuk Sung received the BS and the MS degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2008 and 2010, respectively. He is currently affiliated with Imaging Media Research Center, Korea Institute of Science and Technology (KIST), Korea, as a researcher. His research interests include computer graphics, geometric processing, and object recognition.

Sung Yong Shin received the BS degree in industrial engineering from Hanyang University, Seoul, in 1970 and the MS and PhD degrees in industrial engineering from the University of Michigan in 1983 and 1986, respectively. Since 1987, he has been with the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, where he is currently a professor, teaching computer graphics and computational geometry. He also leads a computer graphics research group that has been nominated as a national research laboratory by the Government of Korea. His recent research interests include data-driven computer animation and geometric algorithms.

Elaine Cohen received the MS degree in 1970 and the PhD degree in mathematics in 1974 from Syracuse University after receiving the BS (cum laude) degree in mathematics in 1968 from Vassar College. She is a professor in the School of Computing at the University of Utah since 1974 and has coheaded the Geometric Design and Computation Research Group since 1980. Her research interests include theory and algorithms for computer graphics, geometric modeling, geometric computation, haptics, and manufacturing, with emphasis on complex sculptured models represented using NURBS (Non-Uniform Rational B-splines) and NURBS-features. In addition to her research papers, she is first author of a reference textbook entitled Geometric Modeling with Splines.