International Journal of Information Science and Computer Mathematics
Volume …, Number …, 2010, Pages …
This paper is available online at http://www.pphmj.com

GEODESIC COMPUTATION ON IMPLICIT SURFACES

HONGCHUAN YU and JIAN J. ZHANG

National Centre for Computer Animation
Bournemouth University
Poole, U. K.

Abstract

Geodesics have a wide range of applications in CAD, shape design and machine learning. Current research on geodesic computation focuses primarily on parametric surfaces and mesh representations. There is little work on implicit surfaces. In this paper, we present a novel algorithm able to compute the exact geodesics on implicit surfaces. Although the existing Fast Marching Method can generate a geodesic path on a Cartesian grid that envelopes the implicit surface in question, this method, as well as other existing methods, is unable to compute a geodesic on the original surface. The computed geodesic path is actually a polyline offsetting from the surface. Our approach provides a solution to two existing fundamental problems, which are (1) to produce a Cartesian grid that can tightly embed the implicit surface concerned, which remains challenging; and (2) to formulate exact geodesics on the original implicit surface itself. Our algorithm consists of two steps, Cartesian grid based geodesic computation and geodesic correction. The later corrects an approximate geodesic path so that it can be on the implicit surface. In addition, in comparison with other existing work, our methods can handle both low dimensional and high dimensional surfaces (hyper-surfaces).

2010 Mathematics Subject Classification: Kindly Provide.

Keywords and phrases: geodesics, implicit surface, geodesic curvature flow.

Communicated by Kewen Zhao

Received April 8, 2010

1. Introduction

Geodesic computation has a wide range of applications in medical imaging, robotics, machine learning and computer graphics, such as surface remeshing (Peyre and Cohen [1]), all-geodesic algorithm for filament winding (Seereeram and Wen [2]), inertia matrix constructed by geodesics (Kim and Luo [3]), bending invariant (Elad and Kimmel [4]), motion planning (Hu et al. [26]), Isomap (Tenenbaum et al. [27]). Because a geodesic represents the shortest path on a surface between two points, it is also important for length measurement on a curved surface. These measurements are fundamental to many applications relating to distances. Many recent efforts have been made for geodesic computation. The main objectives are to achieve accurate representations with a fast computation speed. The recent progress was documented in (Surazhsky et al. [14]), which discussed many effective algorithms computing a geodesic on a mesh. The resulting geodesics however, are not the true representation, as they are not on the surface concerned.

In this paper, we present novel algorithms for computing geodesics, which will be on an implicit surface. Although there is an extensive literature for parametric surfaces (Ravi Kumar et al. [5] and Wang et al. [6]), little research was reported for implicit surfaces. This is due to the fact that implicit forms have no natural material coordinates and they are not as manipulateable as parametric surfaces. However, in many applications, surfaces are given in implicit forms, e.g. (Caselles et al. [7]; Frisken et al. [8] and Yezzi et al. [9]), where we need to extract geodesic paths for a surface distance metric. We could of course triangulate an implicit surface in advance, and use the available methods to compute geodesics on the mesh. The triangulation process usually introduces errors depending on the resolution and adds extra computational costs. Also, we can observe that working directly with implicit representations is more robust and accurate than on a triangulated surface when dealing with the differential characteristics of a surface (Memoli and Sapiro [10]).

Our work was motivated by the works given in (Witkin and Heckbert [11] and Memoli and Sapiro [10]). The former gives a particle based approach to model implicit surfaces, while the latter addresses geodesics computing on implicit hyper-surfaces through embedding surfaces into Cartesian grids. The resulting geodesic paths lie on a Cartesian grid. Focusing on the discretization of implicit surfaces, our algorithm can result in geodesics lying exactly on implicit surfaces instead of Cartesian grids or meshes. Our contributions can be summarized as follows:

(1) Implicit surface discretization. An important step identified in Memoli and Spairo [10] is to embed an implicit surface into a Cartesian grid so that the Fast Marching Method (FMM) (Sethian [12]) can be applied. However how to generate a Cartesian grid for an implicit function remains an unsolved problem, since the size of Cartesian grids grows exponentially when they are subdivided iteratively, rendering the computing process unbearably slow. We present two methods to deal with this issue for different scenarios.

(2) Exact geodesics. Existing methods only compute geodesic paths on a grid (or a triangle mesh). Although the revised MMP algorithm (Surazhsky et al. [14]) can improve the computational accuracy, the resulting geodesic paths still lie on a mesh offsetting from the surface. We present a geodesic correction procedure to ensure an exact geodesic curve on the implicit surface is produced.

Additionally, the presented methods are valid for high dimensional cases. The remainder of this paper is organized as follows: In Section 2, we briefly overview the related work. Our methods, including the Cartesian grid based algorithms and geodesic correction procedure, are presented respectively in Sections 3 and 4. The details of the implementation and discussions are given in Section 5. Finally, our conclusions are given in Section 6.

2. Related Work

For a triangle mesh, the pioneering work is the MMP algorithm (Mitchell et al. [13]), which provides an exact solution for the "one source to all destinations" shortest path problem. In the worst case, the running time is up to $O(N^2 \log N)$. (Surazhsky et al. [14]) discussed its implementation. Another exact geodesic algorithm was presented in Chen and Han [15], whose time complexity is up to $O(N^2)$. The majority of the research focuses on the computation of approximate geodesics with guaranteed error bounds, such as adding extra edges into the mesh for accurate geodesics (Lanthier et al. [16]), and iterative optimization (Kanai et al. [17]). A well-known work is the Fast Marching Method, which computes approximate geodesics in $O(N \log N)$ time complexity. However working on a triangle mesh, FMM sometimes makes the distance functions error-prone. The resulting geodesic path is not always ideal. Some correction post-processing procedures were subsequently proposed following FMM, such as applying the

"straightest geodesics strategy" (Polthier and Schmies [18]) to correct the geodesic path (Martinez et al. [19]).

For parametric surfaces, an exact numerical algorithm on parametric surfaces was presented in Do Carmo [20] though it is computationally expensive. Many approaches for discrete geodesic computation on tessellated surfaces were subsequently presented (Ravi Kumar et al. [5], and Polthier and Schmies [18]). In contrast, little research was undertaken for implicit surfaces. It is the aim of this paper to address this issue. Mathematically speaking, a parametric curve (or surface) can always be represented in an implicit form, but not vice verse other than degree two or degree one cases. Although embedding an implicit surface into a mesh is a viable alternative, as seen in Memoli and Sapiro [10], where an implicit surface is embedded into a Cartesian grid, there exist some numerical challenges in practice. An immediate issue is how to efficiently embed an implicit surface into a Cartesian grid. Without it is being done, the strategy presented in (Memoli and Sapiro [10]) can not be used. One of the objectives of this paper is to resolve this issue.

Modeling implicit surfaces can also be implemented with the particle sampling approach. The pioneering work is the W-H method, which samples and controls implicit surfaces by distributing particles on the surface (Witkin and Heckbert [11]). Our correction procedure presented later makes use of the particle system. Theoretically speaking, our approach can infinitely approximate the geodesic leading to an exact solution. Thus, we say that our approach produces the exact geodesics on implicit surfaces.

To the best of our knowledge, existing methods can compute geodesics on grids or meshes but not on implicit surfaces. Our objective is to calculate a smooth geodesic curve on an implicit surface. The basic idea is to first discretize an implicit surface by embedding it into a Cartesian grid, and then apply existing techniques, such as FMM and Dijkstra's algorithms, to the resulting grid for the approximation of geodesics. Finally, a geodesic correction procedure is presented to refine on the geodesics.

3. Cartesian Grid Based Geodesic Computation

3.1. Cartesian grid generation

The basic idea is conceptually simple, that is, to determine a Cartesian grid

envelope which surrounds the implicit hyper-surface for carrying out the subsequent FMM. We describe two algorithms for different scenarios.

The first case is concerned with implicit functions without given initial sample points. Consider an implicit surface S in R^d, which can be represented as the zero level set of a distance function

$$S = \{p : F(p) = 0,\ p \in R^d\}. \tag{1}$$

It is straightforward to build a Cartesian grid on the variable domain of the implicit function F at a given grid cell size, and then do a cell-wise scan. This however is a time-consuming task, particularly for dimensionality $d \geq 3$. Our approach is to employ a multi-resolution strategy, discussed as follows:

Algorithm I. (1) Input an implicit function F with variable domain Ω having an initial grid cell size of the form $\Delta x_1 = \cdots = \Delta x_d = \sigma_0$ and the expected grid cell size σ.

(2) Looping until $\sigma_i < \sigma$ $(i = 0, 1, 2, \ldots)$;

(3) For each grid cell with size σ_i, check whether S goes through it;

(4) Retaining these cells that S goes through, but deleting others;

(5) Halving the preserved cells in every dimension so that $\sigma_i \leftarrow \sigma_i/2$, then go to Step 2.

The computational complexity is $O(2^d N)$, where N denotes the grid cell number. This algorithm is suitable for low-dimensional implicit surfaces. If d becomes large, then it will become very expensive. (For details, please refer to Appendix I.)

Now let us look at the second case. In many applications, implicit surfaces are often employed to fit a set of discrete sampling points. For the 3D case, these sampling point sets are called *point clouds* or *range data*, which are likely from 3D shape acquisition devices, such as laser or structured light range scanners. They have a wide range of applications in geoscience, artworks, medical imaging, and reverse engineering. In machine learning, for example, we need to construct a hyper-surface

based on a given training set in a high-dimensional space for various learning purposes. The resulting implicit hyper-surfaces are usually defined by a weighted sum of local basis functions over the given sampling points, such as radial basis functions.

Once an implicit surface is constructed to represent the original geometry from a point cloud, we are given two types of information, which are the control points and the implicit surface itself. Compared with the first case, now we have additional data, i.e., the control points. This allows different strategies to be considered. Our basic idea is to employ the region growing strategy, i.e., to consider the given sample points as the seeds which iteratively grow over the implicit surface. Our approach is presented as follows:

Algorithm II. (1) Input an implicit function *F*, having initial sample set Ω, with the expected grid cell size σ.

(2) Computing the grid coordinates $p' \in G$ of samples $p \in \Omega$ at the grid cell size σ.

(3) Looping until the set of new interpolated cells *U* is empty.

(4) Making samples $p' \in G$ grow over their neighborhood (as shown in Figure 1) on a Cartesian grid, the new neighbors are stored in *U*.

(5) Updating *U* by identifying if the surface *S* goes through the current cells.

(6) $G \leftarrow G + U$, then go to Step 3.

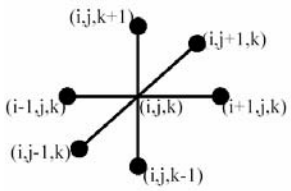

Figure 1. Illustration of the point (i, j, k)'s neighborhood in a 3D Cartesian grid.

Due to non-uniform sample distributions, many points within the initial Ω usually share one grid cell. But then, each new added point will occupy one cell exclusively. Different from Algorithm I, Algorithm II has polynomial complexity of $O(dN^2)$, where *N* denotes the grid cell number. (For details, please refer to Appendix I.)

3.2. Geodesic computation on Cartesian grids

Once the Cartesian grid envelope around the implicit surface is obtained, we can carry out the standard FMM on it for geodesic computation. The error bound is given as, $|d_g(p, q) - d_E(p, q)| \le \lambda\sqrt{\sigma}, \quad p, q \in R^d,$ where d_g denotes the geodesic distance on S, d_E denotes the Euclidean distance and λ is a constant (Memoli and Sapiro [10]). However, it can be noted that the resulting geodesic path is a path on a Cartesian grid, not on the surface itself, similar to other existing methods.

Usually for pair-wise geodesic computation, FMM has an $O(N^2 \log N)$ complexity. Taking this into account, the combined complexity of the presented algorithm II and FMM is around $O(N^2(d + \log N))$.

4. Geodesic Correction

So far, the resulting geodesic paths lie on a grid or mesh offsetting from the surface. In this section, we present an evolution procedure of geodesic curvature flow, which makes the approximate paths converge to the geodesic curves on the surface instead. This is based on the principle that a geodesic curve has a vanishing geodesic curvature everywhere.

We first consider a regular surface S and two endpoints $p^1, p^2 \in R^d$, between which we intend to compute a geodesic curve. Let $C(s, 0)$ be an initial smooth parametric curve on S with $C(s^1, 0) = p^1$ and $C(s^2, 0) = p^2$, where s denotes the arc-length parameter of the curve. If both endpoints are fixed, then we deform $C(s, 0)$ using the geodesic curvature flow, which eliminates the geodesic curvature point-wise on a curve. The curve will then converge to a geodesic curve. The geodesic curvature flow is described as

$$C_t(s, t) = k_g\vec{N}, \tag{2}$$

where $k_g\vec{N}$ denotes the geodesic curvature vector of $C(s, t)$, and the geodesic curvature vector can be expressed as

$$k_g\vec{N} = C_{ss} - \langle C_{ss}, \mathbf{n}\rangle\mathbf{n}, \tag{3}$$

where C_{ss} is the second order derivative of $C(s, t)$ w.r.t. arc-length *s*, and **n** is the normal to *S*. This flow is also known as the Euclidean curve shortening flow and has been proved that it can shrink a curve on *S* to a geodesic (Grayson [24]).

For the discrete case, the geodesic curve has to be approximated by a polyline containing a set of nodes. Correcting the geodesic approximation is carried out on the resulting polyline. Since we have no parametric representation of the initial approximation, the second order derivative of $C(s, t)$ w.r.t. *s* has to be approximated using a vector triangle shown in Figure 2. This gives $C_{ss} \approx (\overline{p^{i+1}p^i} - \overline{p^i p^{i-1}})/\Delta h$, where Δh is the mean of the intervals between points. If C_{ss}'s projection onto the tangent plane at p^i vanishes, then the geodesic curvature of equation (3) is bound to vanish at p^i. When the geodesic curvature vanishes pointwise on *S*, the deforming curve given by equation (2) reaches its stable state, i.e., the geodesic curve.

Thus, projecting the nodes *p* of the resulting geodesic paths on a grid or mesh onto the surface *S* yields a polyline as the initial estimation $C(s, 0)$ of the geodesic on *S*. This can be achieved by the Newton-Raphson iteration as follows:

$$\frac{d}{dt} p = -F(p)\nabla F / \| \nabla F \|^2.$$

Applying equation (2.3) to $C(s, 0)$ can drive it to converge to the geodesic on *S*. The precision can be improved with more nodes adding into the approximation $C(s, t)$. To this end, we introduce the particle system (Witkin and Heckbert [11]) into equation (2) as follows:

$$p_t = C_t(p) - (\nabla F \cdot C_t(p) + \beta F(p))\nabla F / \| \nabla F \|^2, \tag{4}$$

where β is a balance parameter. The nodes are not only attracted to surface *S*, but also are repelled away to each other along the curve $C(s, t)$ to make even particle distribution on $C(s, t)$. In general, during the evolution of equation (4), it is required to add a new sampling node between two successive nodes on *C*, if the interval is greater than a given threshold ε_E. However due to lack of information of the whole surface *S*, there is no guarantee that the curve $C(s, t)$ will pass arbitrary concave convex areas we desire. Figure 3 shows that the new adding node between

$\overline{AB}$ is always projected on the two points *A* and *B* in *S*, and the gap between $\overline{AB}$ cannot be reduced.

To overcome this challenge, we insert a plane to break the concave or convex areas as shown in Figure 4, i.e., plane π is inserted into the gap between two points *A* and *B*. It is perpendicular to $\overline{AB}$ and goes through the midpoint of $\overline{AB}$. Moreover, we can project the midpoint p' of $\overline{AB}$ onto the surface *S* along the plane π by solving the following constrained optimization problem

$$\begin{cases} \min\limits_{p} F(p), \\ \text{subject to } (p - p') \cdot \overrightarrow{AB} = 0. \end{cases} \tag{5}$$

The resulting projection *p* is regarded as a new node and is added into $C(s, t)$. Plane π leads the curve $C(s, t)$ to some desired concave or convex areas on *S*. There are a lot of standard techniques to solve the optimization problem of equation (5), such as the method of Lagrange multipliers.

It is not required to compute the scheme of equation (5) for every node. Herein, the surface *S* has been embedded into a grid as described in Section 3. The evaluated curve $C(s, t)$ is thus partitioned into a set of grid cells. The constraint optimization of equation (5) will have to be computed when the gap shown in Figure 4 falls in one grid cell. Due to the grid cell's size, it is usually impossible that such a complicated structure as shown in Figure 3 would be covered in a grid cell. Hence, the scheme of equation (5) does not result in a significant computing overhead in practice. The approximation error of the evolution curve *C* to the geodesic is unrelated to the original grid cell size, since new nodes could be generated by the particle system. For the analysis of the approximation error and complexity of equation (4), please refer to Appendix II.

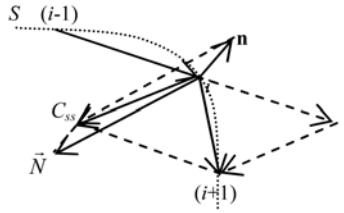

Figure 2. Approximation of the 2nd order derivative of a curve and the geodesic curvature vector on surface *S*. Note that $\vec{N}$ denotes the direction of the geodesic curvature vector, which is tangent to *S* at point p^i.

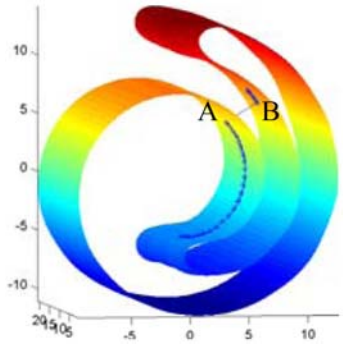

Figure 3. Illustration of a geodesic curve (blue) on a 3D roll's model.

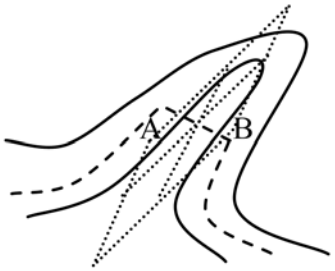

Figure 4. Illustration of gap rising in a grid cell. The dashed line denotes the estimated geodesic curve. The plane (dotted line) is inserted into the gap between A and B.

Remark. Equations (4) and (5) can be performed on arbitrary surfaces. But due to the computational complexity (see Appendix II), it is wise to start from a coarse estimation. The Cartesian grid based geodesic computation can provide a reasonable initial estimation.

5. Implementation and Analysis

The implementation consists of the Cartesian grid based geodesic computation and geodesic correction. Our fist example is to perform the Cartesian grid based geodesic algorithm in Section 3 on a semi-sphere. This is to compare geodesic curves with different Cartesian grid cell sizes. Since the surface is represented by an implicit function without any initial sampling points, the Cartesian grids are generated by Algorithm I described in Subsection 3.1. We constructed two Cartesian grids with different grid cell sizes, respectively, shown in Figures (5a) and (5b). To each Cartesian grid, we utilized FMM and FMM + geodesic correction procedure to

produce two geodesic curves, i.e., the red and blue curves in Figure 5. It can be noted that the corrected geodesic curves by the geodesic correction procedure are independent of the original Cartesian grids. The sampling points have moved from their original cells (red curves) to the geodesics (blue curves). When the geodesic correction procedure continuously interpolates new nodes, this improves the smoothness of the curves while the approximation error becomes unrelated to the original grid cell sizes.

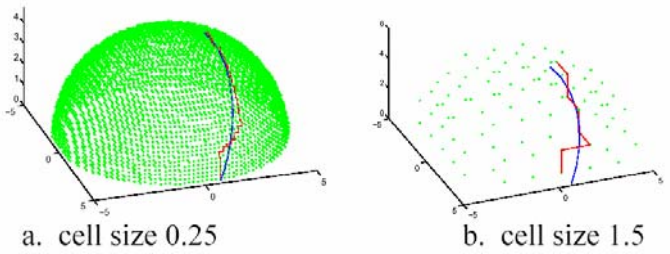

Figure 5. Illustration of geodesic computation on different grid sizes. The red curves represent the geodesic paths by FMM while the blue ones by FMM + geodesic correction procedure, and the green points denote the Cartesian grid cells.

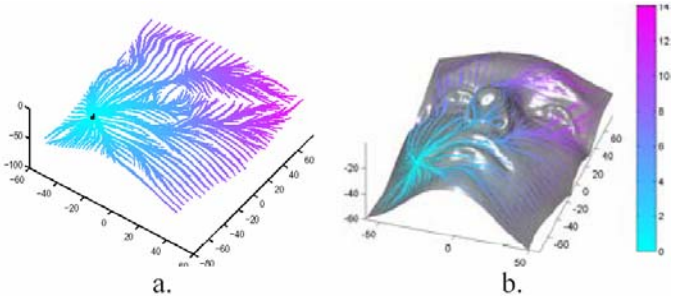

Figure 6. Illustration of the resulting geodesic paths. (a) Geodesic paths of single source to all destinations on Cartesian grids. (b) Geodesic paths on the human face surface. (The face range data is from FRGC (Phillips et al. [25]).)

We performed the Cartesian grid based algorithm on a 3D human face model, which was acquired as a point cloud, and modeled as an implicit surface with the radial basis function approach. Algorithm II as described in Subsection 3.1 is employed to the Cartesian grid generation. Figure 6 shows the result of applying our approach to the scenario of “one source to all destinations” on this face model. The color bar indicates the change of the geodesic distance between two points.

We further performed our algorithms on a set of models with more complex shapes, including human hand, watering pot and sculpture, for the scenario of “one source to all destinations” in Figure 6. Herein, both the water pot and sculpture are closed surfaces with genus two. We also list the running times in the Table below. All the codes were run on MatLab in a Pentium IV 3.2GHz PC with 1GB RAM. Applying Algorithm II to these three models results in their individual Cartesian grids. Specifying some points on the surfaces as fixed points, geodesic computation is then performed to the scenario of “one source to all destinations”. Figure 6 shows three sets of the geodesics (red, blue and green) of one source to all destinations depicted on the surface of each model.

Table. Running time of Figure 7 (times are in seconds)

Model	Cartesian Grid Size	Time (FMM + geodesic correction)
Human Hand	290 cells	51.71 s
Watering Pot	302 cells	56.42 s
Sculpture	311 cells	59.37 s

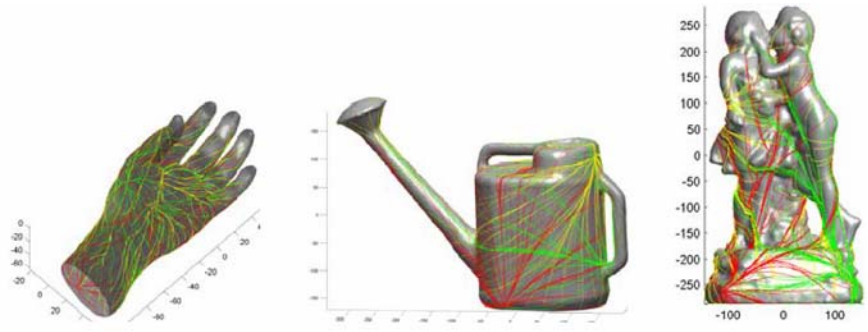

Figure 7. Illustration of geodesic computation on three complicated models. For each model, three sets of geodesics of one source to all destinations are depicted together on the surface with three different colors. (These models are acquired from http://www.farfieldtechnology.com/download/.)

6. Conclusions

We have presented an algorithm for geodesic computation on implicit surfaces, including Cartesian grid based geodesic computation (i.e., algorithm I/II+FMM) and geodesic correction. To suit different applications, we have presented two sub-algorithms for the generation of Cartesian grids on implicit surfaces. Their time

complexities are analyzed. In order to produce exact geodesics on implicit surfaces, we have introduced the particle system to the geodesic correction procedure. The distinct advantage over other approaches is the ability to produce geodesics on implicit surfaces rather than on Cartesian grids or triangle meshes. This improves the accuracy of geodesic computation and helps the numerical stability. The algorithms proposed in this paper can be applied to other, non-implicit and surface representations.

Although all the examples are concerned with 3D models, the algorithms presented are valid for higher dimensional spaces. They therefore have applications in many areas other than surface manipulation, such as manifold interpolation in machine learning and motion planning in robotics.

The geodesic correction procedure is the most time-consuming step, more expensive than other steps presented in this paper. This is because it involves the resolution of partial differential equations equation (4) by adding new nodes gradually. Optimizing for time complexity and accuracy of the geodesic correction procedure will be investigated as future work.

References

[1] G. Peyre and L. Cohen, Geodesic remeshing using front propagation, Internat. J. Comput. Vision 69(1) (2006), 145-156.

[2] S. Seereeram and J. T.-Y. Wen, An all-geodesic algorithm for filament winding of a T-shaped form, IEEE Trans. Indust. Electron. 38(6) (1991), 484-490.

[3] Y. Kim and R. C. Luo, Validation of 3D curved objects: cad model and fabricated work piece, IEEE Trans. Indust. Electron. 41(1) (1994), 125-131.

[4] A. Elad and R. Kimmel, On bending invariant signatures for surfaces, IEEE Trans. Pattern Anal. Mach. Intell. 25(10) (2003), 1285-1295.

[5] G. Ravi Kumar et al., Geodesic curve computations on surfaces, Comput. Aided Geom. Design 20(2) (2003), 119-133.

[6] G. Wang, K. Tang and C. Tai, Parametric representation of a surface pencil with a common spatial geodesic, Comput.-Aided Design 36(5) (2004), 447-459.

[7] V. Caselles et al., Minimal surfaces based object segmentation, IEEE Trans. PAMI 19(4) (1997), 394-398.

[8] S. Frisken et al., Adaptively sampled distance fields: a general representation of shape for computer graphics, Proc. Siggraph'00 (2000), 249-254.

[9] A. Yezzi et al., A geometric snake model for segmentation of medical imagery, IEEE Trans. Medical Imaging 16(2) (1997), 199-209.

[10] F. Memoli and G. Sapiro, Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces, J. Comput. Phys. 173(2) (2001), pp.

[11] A. Witkin and P. Heckbert, Using particles to sample and control implicit surfaces, Proc. of Siggraph' 94 (Year), pp. 269-277.

[12] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999.

[13] J. Mitchell, D. Mount and C. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16(4) (1987), 647-668.

[14] V. Surazhsky et al., Fast exact and approximate geodesics on meshes, Proc. ACM Siggraph' 05 (Year), 553-560.

[15] J. Chen and Y. Han, Shortest paths on a polyhedron; part I: computing shortest paths, Int. J. Comp. Geom. Appl. 6(2) (1996), 127-144.

[16] M. Lanthier et al., Approximating weighted shortest paths on polyhedral surfaces, Proc. of the 13th Annual Symposium on Computational Geometry, Nice, France, 1997, pp. 274-283.

[17] T. Kanai and H. Suzuki, Approximate shortest path on a polyhedral surface and its applications, Comp.-Aided Design 33(11) (2001), 801-811.

[18] K. Polthier and M. Schmies, Straightest geodesics on polyhedral surfaces, Mathematical Visualization, H. C. Hege and K. Polthier, eds., Springer-Verlag, 1998, 391pp.

[19] D. Martinez et al., Computing geodesics on triangular meshes, Comput. Graphics 29 (2005), 667-675.

[20] M. Do Carmo, Differential Geometry of Curves and Surfaces, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[21] P. Heckbert, Fast surface particle repulsion, SIGGRAPH'97, New Frontiers in Modeling and Texturing Course, 1997, pp. 95-114.

[22] M. D. Meyer, P. Georgel and R. T. Whitaker, Robust particle systems for curvature dependent sampling of implicit surfaces, Proc. of the Int'l Conf. on Shape Modeling and Applications (2005), 124-133.

[23] A. Rosch, M. Ruhl and D. Saupe, Interactive visualization of implicit surfaces with singularities, Comput. Graphics Forum 16(5) (1997), 295-306.

[24] M. A. Grayson, Shortening embedded curves, Ann. Math. 129(1) (1989), 71-111.

[25] P. J. Phillips et al., Overview of the face recognition grand challenge, IEEE Conf. Computer Vision Pattern Recognition, 2005.

[26] J. Hu et al., Hybrid geodesics as optimal solutions to the collision-free motion planning problem, Hybrid Systems: Computation and Control, G. Goos, J. Hartmanis and J. van Leeuwen, eds., Vol. LNCS-2034, 2001.

[27] J. B. Tenenbaum, V. Silva and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000), 2319-2323.

Appendix I

Complexity analysis of Algorithm I

For the bisection subdivision operation on a cell, there are 2^d sub-cells, where d denotes the dimensionality. Assume that there are N_i sub-cells surviving after a bisection subdivision operation on a cell, $N_i \le 2^d$, $i = 1, ..., n$, where n denotes the number of iteration. It can be noted that we have to check the result around $(1 + N_1 + \cdots + N_n)2^d$ times after n iterations, where n depends on the cell sizes σ_0 and σ. Without loss of generality, we assume that only a half of the cells is retained after a bisection subdivision, i.e., $N_1 = 2^{d-1}$, $N_2 = 2^{2(d-1)}$, ..., $N_n = 2^{n(d-1)}$. The whole computational time is consequently around $2^d(2^{(n+1)(d-1)} - 1)/2^{(d-1)} - 1$. This yields the complexity of $O(2^d N)$, where N denotes the grid cell number. Thus the complexity of Algorithm I increases exponentially along with the increase of the dimensionality.

Complexity analysis of Algorithm II

For one cell in R^d, there are $2d$ neighbors. Assume that when a cell is connected to a patch, only one neighbor of this cell belongs to this patch. It can be observed that there are $2d + 2(d-1)(n-1)$ neighbors around the patch containing n connected grid cells. These neighbors need to be further identified in Algorithm II. This is the extreme case. In general, there is more than one neighbor belonging to the patch. Hence in practice, the neighbor's number is usually less than this estimate. However, if there is only one seed in the initial sample set Ω, then we have to take the check operation on the neighbors approximately $n^2(d-1) + n(d+1)$ times for the extreme case. Since the size of the current patch n is also the current iterative number, we therefore obtain the complexity as $O(dN^2)$, where N denotes the grid cell number.

This is the worst case. When there are many samples in the initial Ω as seeds with uniform distribution over an implicit surface, the practical running time is less than this estimate.

Appendix II

In order to estimate the error of geodesic correction procedure as described in Section 4, let us consider the local structure of a curve with an osculating circle as shown in Figure A(1). The local error can be described as $\text{err} = \overset{\frown}{AB} - \|\overrightarrow{AB}\|$, where $\overset{\frown}{AB}$ denotes the arc-length from A to B, and $\|\overrightarrow{AB}\|$ is the Euclidean distance of two points ε. Applying the Cosine law yields, $\overset{\frown}{AB} = r\cos^{-1}\left(1-\frac{\varepsilon^2}{2r^2}\right)$, where r denotes the curvature radius. Substituting $\overset{\frown}{AB}$ into err gives $\text{err} = r\cos^{-1}\left(1-\frac{\varepsilon^2}{2r^2}\right) - \varepsilon$. For a whole geodesic curve C, the error is expressed as

$$\text{err} = \sum_i^m r_i \cos^{-1}\left(1-\frac{\varepsilon_i^2}{2r_i^2}\right) - \sum_i^m \varepsilon_i,$$

where m denotes the nodes' number on the curve, and ε_i and r_i depend on the local curvature of the curve. During the evolution of the geodesic curvature flow (4), all nodes are floating on surface along the curve $C(t)$ except the two fixed endpoints. Moreover, the final distribution of the nodes on a geodesic curve depends on the local curvature. For simplicity, we therefore assume $\varepsilon_i / r_i = Cst$ and $\sum_i^m \varepsilon_i = L$. This gives

$$\text{err} = L\left(\frac{1}{Cst}\cos^{-1}\left(1-\frac{Cst^2}{2}\right)-1\right), \tag{A1}$$

for the whole geodesic curve. L depends on the size of surface while Cst depends on the local structure of surface. For a given discrete geodesic curve with the fixed nodes' number m, L and Cst are constant, while for different discrete geodesic curves (e.g., m is variable), they are variable. The error distribution is shown in

Figure A(2). Furthermore, we assume that L and Cst are constant while ε_i is variable. Varying ε_i will change m. Figure A(3) shows that raising the node's number m can improve the numerical stability effectively but cannot make err converge to ZERO. The other part of err is from Cst, i.e., the local curvature of the curve. More nodes on the geodesic curve can effectively reduce Cst. The given threshold ε_E described in Section 4 is the maximum interval of two successive nodes. It can guarantee that there are dens samples at the curved areas of a curve.

In addition, let us consider the time complexity of equation (4). Assume that it takes the same time K to move a node to a geodesic. The number of nodes can be estimated as $m = L/\bar{\varepsilon}$, where $\bar{\varepsilon}$ is the mean of intervals ε_i, $\bar{\varepsilon} < \varepsilon_E \cdot \bar{\varepsilon}$ depends on the maximum interval of successive nodes ε_E. In terms of equation (A1), the running time can be estimated as

$$\text{Time} = \text{err} \cdot K/\bar{\varepsilon}\,(Cst^{-1}\cos^{-1}(1 - Cst^2/2) - 1). \tag{A2}$$

The time complexity is also illustrated in Figure A(4). It can be noted that reducing the nodes' number by increasing $\bar{\varepsilon}$ and increasing the ratio Cst of (ε_i, r_i) can reduce running time. But this will increase approximate error. We have to tradeoff between the accuracy and the running time.

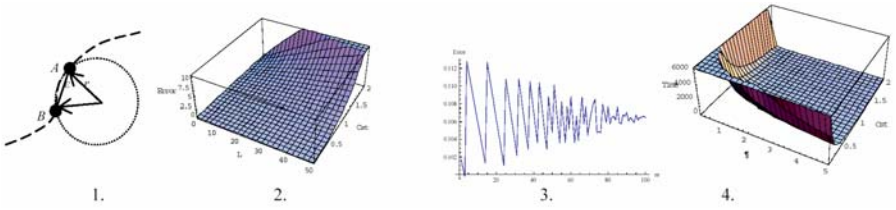

Figure A. (1) Illustration of approximate error. (2) (3) Error distribution surface of equation (A1). (4) Illustration of running time change at $\text{err} = 10$ and $K = 2$sec.

Paper # PPH-1004044-IS Kindly return the proof after correction to: *The Publication Manager* *Pushpa Publishing House* *Vijaya Niwas* 198, *Mumfordganj* *Allahabad*-211002 (*India*) along with the print charges* by the <u>fastest mail</u> ***Invoice attached**	Proof read by: …… Copyright transferred to the Pushpa Publishing House Signature: ……..…… Date: ……..…… Tel: ……..…… Fax: ……..…… e-mail: ……..……..…… Number of additional reprints required …… Cost of a set of 25 copies of additional reprints @ Euro 12.00 per page. (25 copies of reprints are provided to the corresponding author ex-gratis)