International Journal of Automation and Computing 04(1), January 2007, 8-13
DOI: 10.1007/s11633-007-0008-5

Fast Approximate Geodesic Paths on Triangle Mesh

Jie Tang[1*] Gang-Shan Wu [1] Fu-Yan Zhang[1] Ming-Min Zhang[2]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, PRC
[2]School of Computer Science, Zhejiang University, Hangzhou 310027, PRC

Abstract: We present a new algorithm to compute a geodesic path over a triangle mesh. Based on Novotni′s propagating wavefront method which is similar to the well known Dijkstra algorithm, we made some improvements which Novotni had missed and we also gave the method to find out the geodesic path which Novotni had not. It can handle both convex and non-convex surfaces or even with boundaries. Experiment results show that our method works very well both in efficiency and precision.

Keywords: Triangle mesh, geodesic path, virtual reality.

1 Introduction

Geodesic curves are useful in many areas of mesh processing, such as segmentation[1], terrain navigation, parameterization[2], remeshing[3] and skeleton generation[4]. The increasing development of discrete surface models demanded the definition of geodesic curves for polyhedral surfaces[5], and hence the study of efficient algorithms to compute them. Such curves are called Discrete Geodesics and there exist some different definitions for them, mostly depending on the application in which they are used. Usually, we consider a geodesic as a shortest path between two points on the surface. In most applications, the efficiency of computations is preferred over accuracy. Instead of optimal solutions efficiently computable high quality approximations are called for.

There are two kinds of geodesic computing problems. One is the single source shortest path problem, in which one wishes to find a shortest path between a source point and any other point on the surface. The other, more complex, version of the problem asks for a subdivision on the surface such that a shortest path between any pair of points in the surface can be found quickly; this is known as the all pairs shortest path problem.

In this paper we present a new algorithm to compute a geodesic path over a triangle mesh. Based on Novotnis propagating wavefront method which is similar to the well known Dijkstra algorithm, we made some improvements which Novotni had missed and we also proposed a method to find out the geodesic path while Novotni had not. It can handle both convex and non-convex surfaces or even with boundaries. Experiment results show that our method works very well both in efficiency and precision.

In Section 2, we do a quick review of the related work. Section 3 presents the algorithm, which is the main contribution of this paper. We begin with an approximate geodesic distance field and then give the method to find out the geodesic path. In Section 4 we show some experimental results and finally in Section 5, we make a conclusion.

Manuscript received July 28, 2006; revised December 11, 2006.
This work was supported by National Natural Science Foundation of PRC (No. 60503058, No. 60533080) and the Science and Technology Fund of Huawei Technologies Co., Ltd.
*Corresponding author. E-mail address: tangjie@nju.edu.cn

2 Related work

Most of the algorithms use front propagation or some other kind of Dijkstras-like algorithm. In 1987 Mitchell *et al.*[6] provided an exact solution for the "single source, all destination" shortest path problem on a triangle mesh. Their algorithm partitions each mesh edge into a set of intervals (windows) over which the exact distance computation can be performed atomically. These windows are propagated in a "continuous Dijkstra"-like manner. They proved that the worst case running time is of $O(n^2 \log n)$ and requires $O(n^2)$ space, where n is the number of mesh edges. Surazhsky[7] implemented the algorithm and prove that the algorithm runs in sub-quadratic time. Chen and Han[8] proposed an exact geodesic algorithm with worst case time complexity of $O(n^2)$. Kaneva and ORourke[9] partially implemented this algorithm. Another exact geodesic algorithm with worst case time complexity of $O(n\log^2 n)$ was described by Kapoor[10]. This is a very complicated algorithm which calls as subroutines many other complicated computational geometry algorithms. It is not clear if this algorithm will ever be implemented.

Exact geodesic computing costs too much resources and sometimes approximate geodesic is enough. Approximate geodesic again can be categorized generally into two groups. The first kind obtains the geodesics by adding extra edges into the mesh and running Dijsktra on the one-skeleton of this augmented mesh[11,12]. These algorithms require the addition of numerous extra edges to obtain accurate geodesics. The algorithm described in [11] relies on the selective refinement, and therefore significantly depends on the first approximation path found. If this approximation is far from the actual solution, the subdivision method might converge to the local minimal path (instead of the global geodesic one), or it might take a very large number of iterations until the refinement area moves to the vicinity of the actual geodesic and the process converges. Reference [12] presented an iterative algorithm to compute a shortest geodesic between two points over a mesh. At each step it computes a new curve with smaller length. This is done by reducing locally the curve length at each vertex. It explores the fact that the intersection of a mesh face with a shortest geodesic is a line segment, and hence its vertices lie on mesh vertices or edges. However, iteration is not a good choice because it consumes more time and is difficult to end up. The second kind of methods define the approximate geodesic distance field over the whole mesh and then find out the required geodesics. The difference among these methods is the way they build the distance field. Surazhsky[7] defined the distance filed over the interval of edges, and this costs relatively more space and is difficult to manage. Novotni method[13] defined the geodesic distance field over the vertices which is efficient both in time and space. However, during the distance field generation, Novotni missed some cases which affect the output greatly. Other approximate geodesics include: Kimmel and Sethian[14] proposed an algorithm that runs in $O(n\log n)$ time. The approximate geodesics found by this method can be quite inaccurate, even for planar meshes. Polthier and Schmies[5] described a different definition of a geodesic path on meshes using a notion of "straightest" instead of "shortest". This notion may be inappropriate for some applications of geodesics.

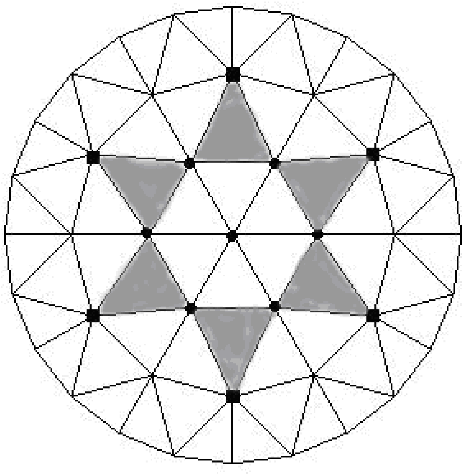


Fig. 1 Initialization course. The center vertex is the start point. Therefore the 1-ring vertices and the center vertex are fixed, and the square vertices lying on the 2-ring of the center vertex are computed, other vertices are unprocessed

3 Algorithm

3.1 Geodesic distance field

Novotni[13] proposed an approximation method to compute geodesic distances on triangulated domains in the three dimensional space. The particular approach is based on the Fast Marching Method for solving the Eikonal equation on triangular meshes. When computing the geodesic distance between two point, the algorithm proceeds by propagating a wavefront outwards from the start points. The advancing front can be thought of as a brush-fire advancing with constant velocity in all directions in which the mesh has not yet been "burnt". This is accomplished in a fashion very similar to the well known Dijkstra algorithm for computation of shortest paths in a graph. The geodesic distances of vertices are calculated propagatedly until the target vertex is met or all vertices of the mesh have their own geodesic distances.

The scenario is that given a start point and an end point on a mesh, compute the geodesic distance between these two points. The overall algorithm could be separated into initialization and computing courses. The initialization is as follows (see Fig. 1):

1) All vertices are initialized with the prescribed values, e.g. a zero.

2) All vertices are categorized into 3 groups: fixed, computed and unprocessed. If the start point is one of the vertices of the mesh, its 1-ring vertices and itself are categorized into fixed group because the geodesic distance of these vertices to the start point is definitely fixed. If the start point lies on the plane of a triangle, the vertices of this triangle are categorized into fixed.

3) Then the according distance values for all the vertices which are incident to triangles containing exactly two fixed vertices are computed (this will be discussed in the next section), and those vertices are included into the computed set indicating that the values of these vertices may change.

4) All the remaining vertices are included to the Unprocessed set.

The vertices of computed set are stored in a heap and sorted according to the geodesic distance. Thus the computing course proceeds as follows (see Fig. 2):

1) Pop up the vertex v with the smallest geodesic distance value. Add this vertex to the fixed set.

2) For each vertex in the 1-ring vertices of v, if it belongs to unprocessed and can be compute now (which means that it is incident to a triangle containing exactly two fixed vertices), remove it from unprocessed set and insert it to the computed set. If it belongs to the computed set and can be computed and the computed geodesic distance is less than its original value, then updating its geodesic value. Finally, resort the vertices in the computed set.

3) Return to 1) until the computed set is null or the end point is matched.

In the algorithm, the newly computed distance values cannot be smaller than the values supporting this computation. This monotonicity property ensures that the solution is always propagated outwards by selecting the vertex with smallest geodesic distance value. In other words, no values corresponding to vertices in the fixed set will have to be recomputed. Eventually, every non-fixed vertex will have to be chosen as the one with smallest geodesic distance value in the computed set, the complexity of the algorithm is therefore mainly influenced by this operation. In our implementation, we applied a pairing heap for this purpose, the above vertex location procedure took $O(\log n)$ time using this technique. Since all vertices in the domain have to be processed, the overall time complexity of the algorithm is $O(n\log n)$. Since the number of vertices in the heap at the same time is far less than the mesh vertice number, the algorithm runs fairly fast, which will be verified by experiments.

3.2 Virtual start point

When computing the geodesic distance of a vertex v_i from the start point, if the other two vertices of the triangle

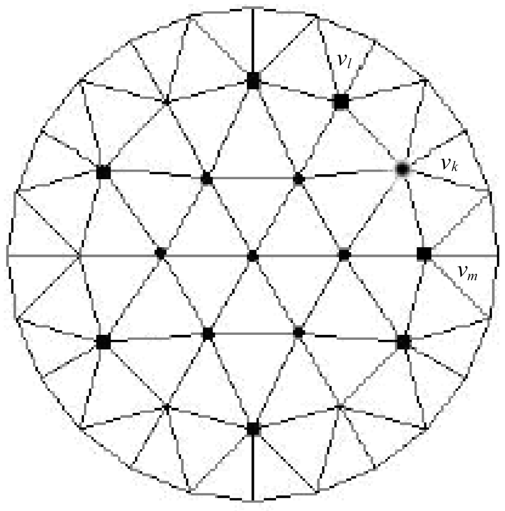


Fig. 2 Propagation course. Let us say $\boldsymbol{v}_k$ has the least geodesic distance from center point. Thus we add it into the fixed set. $\boldsymbol{v}_l$ originally belongs to the unprocessed set, but now its geodesic distance could be computed and we add it into the computed set. $\boldsymbol{v}_k$ originally belongs to computed, now we can compute its geodesic distance from a different direction, thus we update its geodesic distance value with the less one

in which v_i lies both have effective geodesic distances, the geodesic distance of v_i is computed using the virtual start point (as shown in Fig. 3), which is implemented as follows: Given a triangle $v_iv_{i+1}v_{i+2}$, if the geodesic distances of v_{i+1} and v_{i+2} are already calculated, say d_{i+1} and d_{i+2}, respectively, we could compose two circles using v_{i+1} and v_{i+2} as centers, d_{i+1} and d_{i+2} as radii, respectively. Denoting the intersect point far off from the v_i as v_s, the geodesic distance of v_i is the distance between v_i and v_s , which is $\|v_iv_s\|$.

If we locate the $v_iv_{i+1}v_{i+2}$ as in Fig. 3, the coordinates of v_s could be decided by

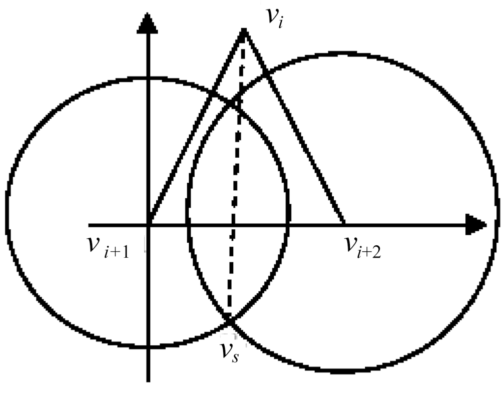


Fig. 3 The virtual start point

$$v_{sx}^2 + v_{sy}^2 = d_{i+1}^2$$
$$(v_{sx} - v_{(i+2)x})^2 + v_{sy}^2 = d_{i+2}^2.$$

The solution of the above equations is

$$v_{sx} = \frac{v_{(i+2)x}^2 + d_{i+1}^2 - d_{i+2}^2}{2v_{(i+2)x}}$$
$$v_{sy} = \pm\sqrt{d_{i+1}^2 - v_{sx}^2}.$$

From the definition we know that the v_{sy} which is less than 0 is the right solution. And thus we could calculate the distance between v_i and v_s, as well as the intersect point of v_iv_s and the x axis, which is used to find out the geodesic path.

Novotni′s method is fast and effective, but sometimes, the error is too big to accept. We find that the algorithm ignored two other cases when calculating the distance from the virtual start point. As shown in Fig. 4, Novotni′s method only considered the case of Fig. 4 (a), while missing the two cases of Fig. 4 (b) and Fig. 4 (c). Therefore, we improve the method as follows:

1) When the intersect falls in between v_{i+1} and v_{i+2}, the geodesic distance is $\|v_iv_s\|$ (see Fig. 4 (a))

2) When the intersect falls left to v_{i+1}, the geodesic distance is $d_{i+1}+\|v_iv_{i+1}\|$ (see Fig. 4 (b))

3) When the intersect falls right to v_{i+2}, the geodesic distance is $d_{i+2}+\|v_iv_{i+2}\|$ (see Fig. 4 (c))

3.3 Geodesic path

Novotni did not give how to find the geodesic path; but in some application such as tool path generation and remeshing, we need to find the explicit geodesic path. Hence, we propose a method to find the geodesic path based on the geodesic distance field generated in the last section.

During the building of geodesic distance field, for each fixed vertex, we store the two vertices ID on which the vertexs geodesic distance is computed and we call them the parents of the vertex. When creating geodesic path, like many other algorithms, we adopt a back-tracing method which starts form the end point and processes as follows:

Step 1. If the end point is one of the vertices of the mesh, find its parents. If the end point lies inside a triangle, calculate its geodesic distance on each two vertices of the triangle respectively, find out the shortest one.

Step 2. Calculate the intersect point of the current point and its virtual start point.

Step 3. If the intersect point lies in between the two vertices (see Fig. 4 (a)), add it to the geodesic path list, find out the vertex opposite the edge, calculate the geodesic distance of the intersect point using two pairs of the vertices respectively, find the shortest one. Replace the current point with the intersect point.

Step 4. If the intersect point lies outside the two vertices (see Fig. 4 (b) and Fig. 4 (c)), add the relevant vertex to the geodesic path list and replace the current point with the relevant vertex.

Step 5. Goto step 2 until 1) if the intersect point lies in between the two vertices and the vertex opposite the edge

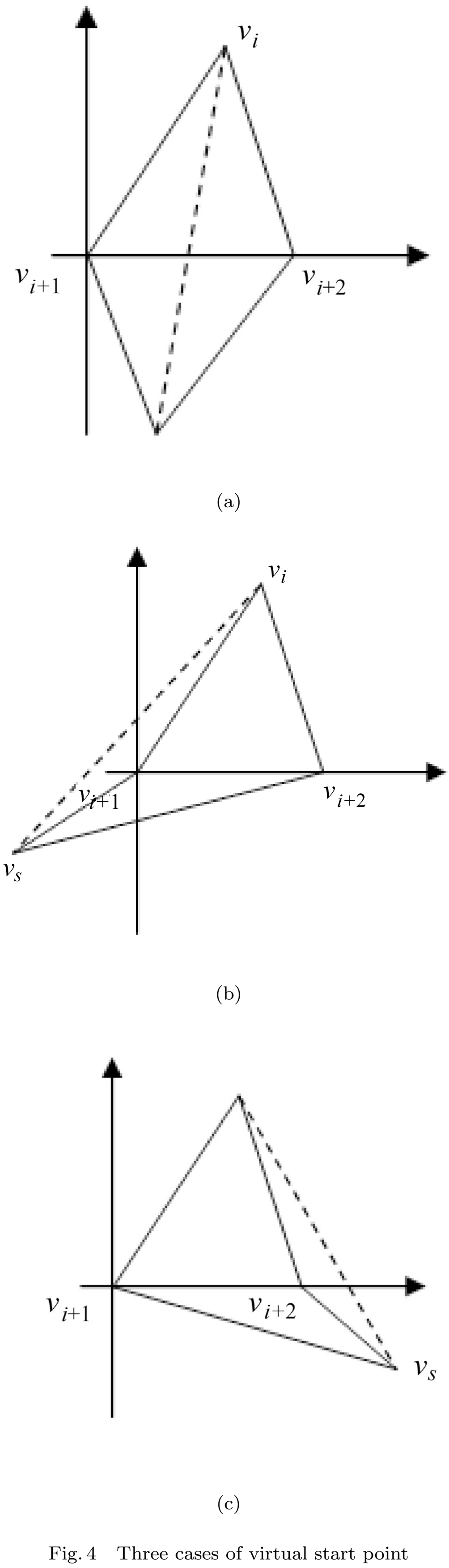


Fig. 4 Three cases of virtual start point

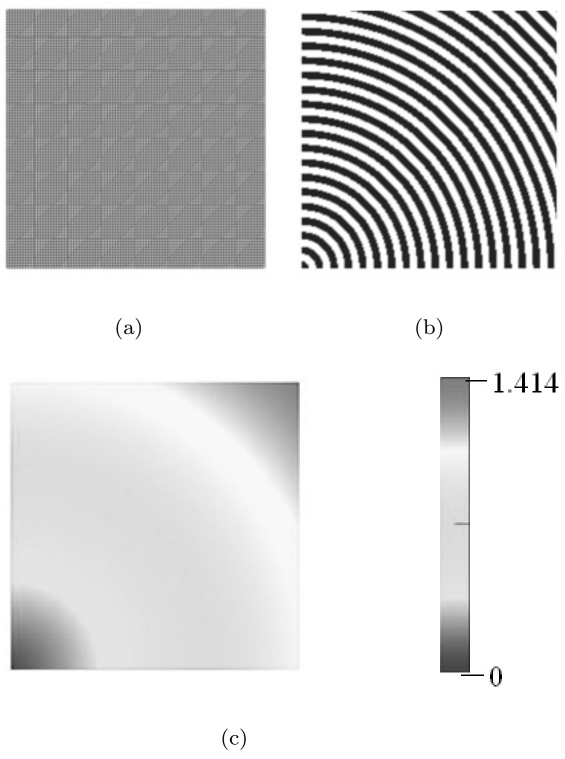


Fig. 5 Geodesic distance on a planar mesh

is the start point or 2) if the intersect point lies outside the two vertices and the parent vertices contain the start point.

Through the above method we could construct the geodesic path quickly.

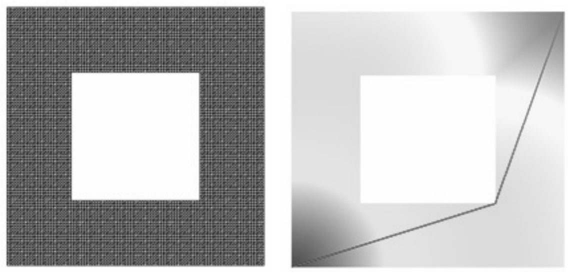
4 Results

In order to test our method, we did lots of experiments. Fig. 5 (a) is a dense planar rectangle mesh, which has 10 000 vertices and the edge length of the rectangle is 1. We calculated all geodesic distances from the left bottom vertex to others. Since the geodesic distance on a plane is just the Euclidean distance, we could easily find out the error of the algorithm. The result shows that the maximum error is 10∼15, the root mean square error is 10∼31, which is quite precise. Fig. 5 (b) is the iso-distance line from the left bottom vertex and Fig. 5 (c) is the colorful result.

Fig. 6 (a) is a dense planar rectangle mesh similar to Fig. 5 (a) but with a square hole in the center. The geodesic distance therefore is different. But since we considered the two cases in Fig. 4, our method could handle such a problem. Fig. 6 (b) is the colorful result and the geodesic path from the left bottom point to the up right point.

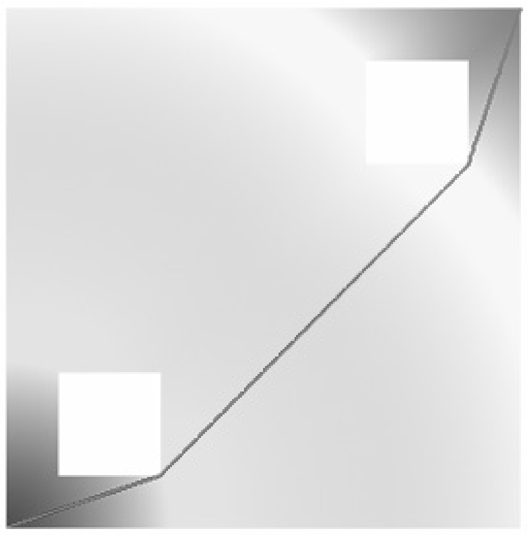
Fig. 7 explains the geodesic path more detailedly. In Fig. 7 (a), there are two holes and the result shows the geodesic path. In Fig. 7 (b), one more hole is added, thus the former path apparently is not the shortest one, therefore, the algorithm chooses the shortest path automatically.

Fig. 8 shows another more complicated result of our algorithm, in which we calculated all geodesic distances from

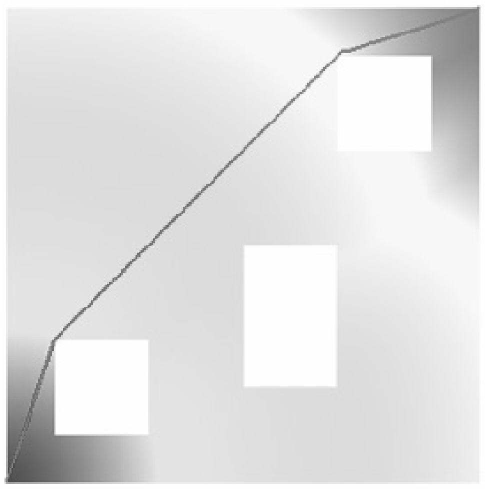


(a) (b)

Fig. 6 Geodesic distance on a planar mesh with a hole

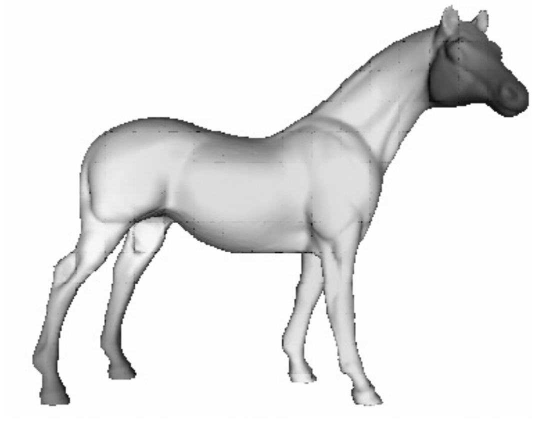


(a)

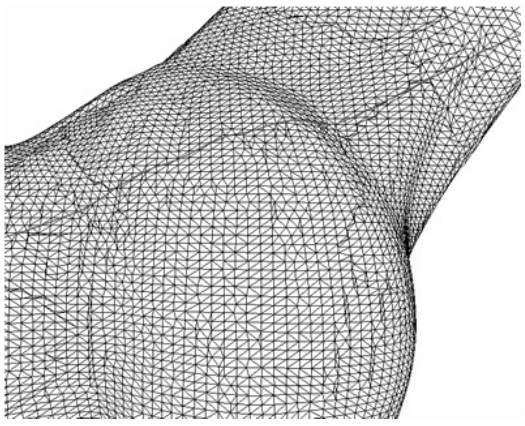


(b)

Fig. 7 Geodesic path on a planar mesh with more than one hole



(a)



(b)

Fig. 8 Geodesic distance on a horse

the top vertex on the nose of the horse to all other vertices.

Our method runs fast. We tested our method on a computer with 2.4 GHz CPU, 512 M RAM. For a mesh with nearly 20 000 vertices, we computed the geodesic distances from all other vertices to a start point in less than 5 s, while Novotni′s method processed a similar size problem on a 1 200 MHz AMD Athlon PC with 256 MB RAM in 7.5 s. Other method ran relatively slow. Another advantage of our method is that it costs less space. In fact, for each vertex, it just needs a float to store the geodesic and two ints to store parents ID, while Surazhskys method needs extra space to store the intervals.

5 Conclusions

In this paper we analyzed and improved the marching front method for geodesic distance computation proposed by Novotni. Especially, in the case of mesh with holes our results are better than the ones produced by the original algorithm. Also, we give a fast method to find the geodesic path. Experiments verify the effect and efficiency of our method. It is difficult to analyze the error bound of our method theoretically, so we test it by experiments, which we want to improve it in the future.

References

[1] S. Katz, A. Tal. Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts. *ACM Transactions on Graphics*. vol. 22, no. 3, pp. 954–961, 2003.

[2] P. Sander, Z. Wood, S. Gortler, J. Snyder, H. Hoppe. Multi-chart Geometry Images. In *Proceedings of Eurographics Symposium on Geometry Processing*, Aachen, Germany, pp. 146–155, 2003.

[3] G. Peyre, L. D. Cohen. Geodesic Remeshing Using Front Propagation. *International Journal of Computer Vision*, vol. 69, no. 1, pp. 145–156, 2006.

[4] M. Hilaga, Y. Shinagawa, T. Kohmura, T. L. Kunii. Topology Matching for Fully Automatic Similarity Estimation of 3d Shapes. In *Proceedings of ACM SIGGRAPH 2001*, Los Angeles, USA, pp. 203–212, 2001.

[5] K. Polthier, M. Schmies. Geodesic Flow on Polyhedral Surfaces. *Data Visualization*, Springer, Vienna, pp. 179-188, 1999.

[6] J. Mitchell, D. M. Mount, C. H. Papadimitriou. The Discrete Geodesic Problem. *SIAM Journal of Computing*, vol. 16, no. 4, pp. 647–668, 1987.

[7] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, H. Hoppe. Fast Exact and Approximate Geodesics on Meshes. In *Proceedings of ACM SIGGRAPH 2005*, Los Angeles, USA, pp. 553–560, 2005.

[8] J. Chen, Y. Han. Shortest Paths on a Polyhedron, Part I: Computing Shortest Paths. *International Journal of Computational Geometry & Applications*, vol. 6, no. 2, pp. 127–144, 1996.

[9] B. Kaneva, J. O'Rourke. An Implementation of Chen & Han's Shortest Paths Algorithm. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, New Brunswick, Germany, pp. 139–146, 2000.

[10] S. Kapoor. Efficient Computation of Geodesic Shortest Paths. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, Atlanta, Georgia, USA, pp. 770–779, 1999.

[11] T. Kanai, H. Suzuki. Approximate Shortest Path on a Polyhedral Surface and Its Applications. *Computer-Aided Design*, vol. 33, no. 11, pp. 801–811, 2001.

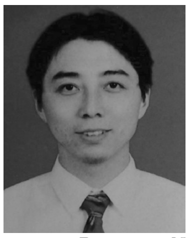
[12] Dimas Martinez, Luiz Velho, Paulo C. Carvalho. Computing Geodesics on Triangular Meshes. *Computers & Graphics*, vol. 29, no. 5, pp. 667–675, 2005.

[13] M. Novotni, R. Klein. Computing Geodesic Distances on Triangular Meshes. In *Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Bonn, Germany, pp. 341–347, 2002.

[14] R. Kimmel, J. Sethian. Computing Geodesic Paths on Manifolds. *Proceedings of the National Academy of Sciences of USA*, vol. 95, no. 15, pp. 8431–8435, 1998.

Jie Tang received his B. Sc. and M. Sc. degrees in mechanical engineering from the Southeast University, China, in 1994 and 1997, respectively, and the Ph.D. degree in computer application from Nanjing University of Aeronautics and Astronautics, China, in 2000. He was a senior system analyst at E1 Media Co. Ltd, HongKong before he became a postdoctoral researcher at Nanjing University, China in 2003. Currently, he is a associate professor in the Department of Computer Science and Technology at Nanjing University, China.

He has published about 20 referenced journal and conference papers. His current research interests include virtual reality, computer graphics and multimedia technologies.



Gang-Shan Wu received his B. Sc., M. Sc. and Ph.D. degrees in Computer Science from Nanjing University, China, in 1988, 1991 and 2000, respectively. He is now a professor of Department of Computer Science of Nanjing University, the director of multimedia computing teaching and researching group, director of Multimedia Computing Lab. He was the principal investigator or co-principal investigator of more than twenty projects sponsored by National 863 Hi-Tech Research and Development Program, National Natural Science Foundation of China and Key Project of Chinese Ministry of Education.

His research interests include multimedia information retrieval, Web information processing and Chinese information processing. He published more than thirty papers in these areas.

Prof. Wu has won second-class and third-class awards of Science and Technology Progress of Jiangsu Province respectively, a third-class award of Science and Technology Progress of the Chinese Ministry of Education.

Fu-Yan Zhang is a professor at the Department of Computer Science and Technology of Nanjing University, China. He is currently the chair of the Multimedia Computing Institute of Nanjing University.

His research interests include information processing and multimedia. He has published over 100 technical papers in these areas.

Ming-Min Zhang received her B. Sc. degree from Computer Science Dept, Nanjing University in 1990, and the M. Sc. degree from Computer Science and Engineering Department, Zhejiang University in 1995. She is now an associate professor of Computer and Engineering Department, Zhejiang University.

She has published more than 20 papers in recent years. She is the co-author of two books related to computer graphics and multimedia. Her research interests include virtual reality/virtual environment, multi-resolution modeling, real-time rendering, distributed VR, visualization, multimedia and image processing.