2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering

Isotropic Mesh Simplification by Evolving the Geodesic Delaunay Triangulation

Shi-Qing Xin Shuang-Min Chen Ying He
School of Computer Engineering
Nanyang Technological University
Singapore
Email: {sqxin|smchen|yhe}@ntu.edu.sg

Guo-Jin Wang
State Key Laboratory of CAD & CG
Department of Mathematics
Zhejiang University, China
Email: wanggj@zju.edu.cn

Xianfeng Gu Hong Qin
Department of Computer Science
Stony Brook University
New York, USA
Email: {gu|qin}@cs.sunysb.edu

***Abstract*—In this paper, we present an intrinsic algorithm for isotropic mesh simplification. Starting with a set of unevenly distributed samples on the surface, our method computes the geodesic Delaunay triangulation with regard to the sample set and iteratively evolves the Delaunay triangulation such that the Delaunay edges become almost equal in length. Finally, our method outputs the simplified mesh by replacing each curved Delaunay edge with a line segment. We conduct experiments on numerous real-world models of complicated geometry and topology. The promising experimental results demonstrate that the proposed method is intrinsic and insensitive to initial mesh triangulation.**

I. Introduction

Mesh simplification is the process of reducing the number of faces used in the surface while keeping the overall shape, volume and boundaries preserved as much as possible. Typically, mesh simplification is used to improve rendering speed or to minimize data size or compression requirements [1]. Topology-preserving and feature-preserving schemes are often desired.

Mesh simplification has been extensively studied in the past two decades. There exists a rich body of literatures in mesh simplification. The representative works include progressive meshes [2] and quadric error metrics scheme [3]. We refer the readers to [4], [5] for a complete survey.

In this paper, we present a new method for isotropic mesh simplification. Our method is different than the existing approaches in that geodesic Delaunay triangulation is employed. Therefore, the proposed method is intrinsic and independent of the embedding space. Starting with a set of initial sample points that are arbitrarily distributed on the input mesh, our method computes the geodesic Delaunay triangulation with regard to the sample set and iteratively evolves the Delaunay triangulation such that the Delaunay edges become almost equal in length. Our evolving step iteratively invokes the following operations:

1) Compute the geodesic Delaunay triangulation based on the sample set;
2) For each sample point, compute a standard deviation of the first order incident Delaunay edge lengths and predict a better sample point for substitute;
3) Update the sample point set.

The iterative algorithm terminates when the maximum standard deviation is less than a prescribed tolerance. Finally, our method outputs the simplified mesh by replacing each curved Delaunay edge with a line segment.

Compared with existing methods, our Delaunay evolving algorithm takes advantage of the geodesic Voronoi diagram, and therefore it is intrinsic, robust and insensitive to initial mesh triangulation. Furthermore, our algorithm refines the Delaunay edges to be as equal as possible in length, which leads to a simplified mesh of high triangulation quality. In addition, the evolving is only a local operation and thus can be parallelized. We apply our approach to real-world models with non-trivial topology and geometry. The experimental results demonstrate the efficacy of our algorithm. Figure 1 shows an example of our mesh simplification algorithm.

The remaining of the paper is organized as follows: Section II reviews the related work on mesh simplification, geodesic Delaunay triangulation, discrete geodesics and surface sampling. After that, Section III presents our algorithm for isotropic mesh simplification followed by the experimental results in Section IV. Finally, Section V draws the conclusion and discusses the future work.

II. Related Work

Our work is closely related to mesh simplification, geodesic Voronoi diagram/Delaunay triangulation, discrete geodesics, and surface sampling.

Mesh simplification The existing mesh simplification algorithms can be classified into the following categories:

- Volumetric approach [6] converts the input polygonal mesh into a volumetric description (e.g., voxels), then simplifies the mesh by 3D morphological operators. This approach is robust and flexible in that it can handle mesh degeneracies.
- Simplification envelopes [7] use a geometric construction to control the simplification, i.e., building a new surface inside the space formed by the two offsetting surfaces with regard to the user-specified threshold.

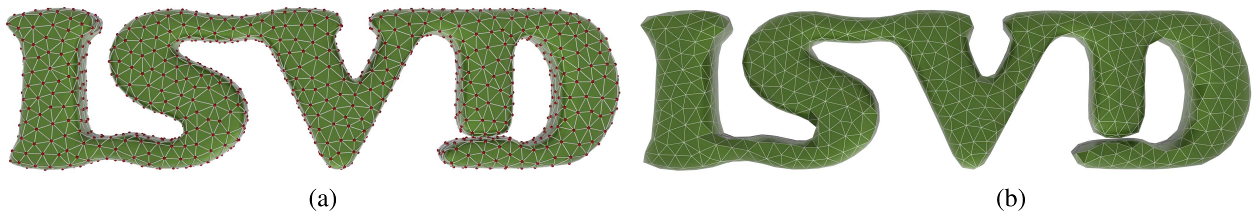


Figure 1. An example of running our algorithm on the "ISVD" logo with 50K vertices. (a) Given 800 arbitrarily sampled points on the given surface, our method locally optimizes the standard deviation of the geodesic Delaunay edge lengths, until all the points are distributed evenly on the surface. (b) By replacing the curved geodesic edge of the induced geodesic Delaunay triangulation with a straight line segment, our method can naturally generate an isotropic simplified mesh with the user-specified number of vertices.

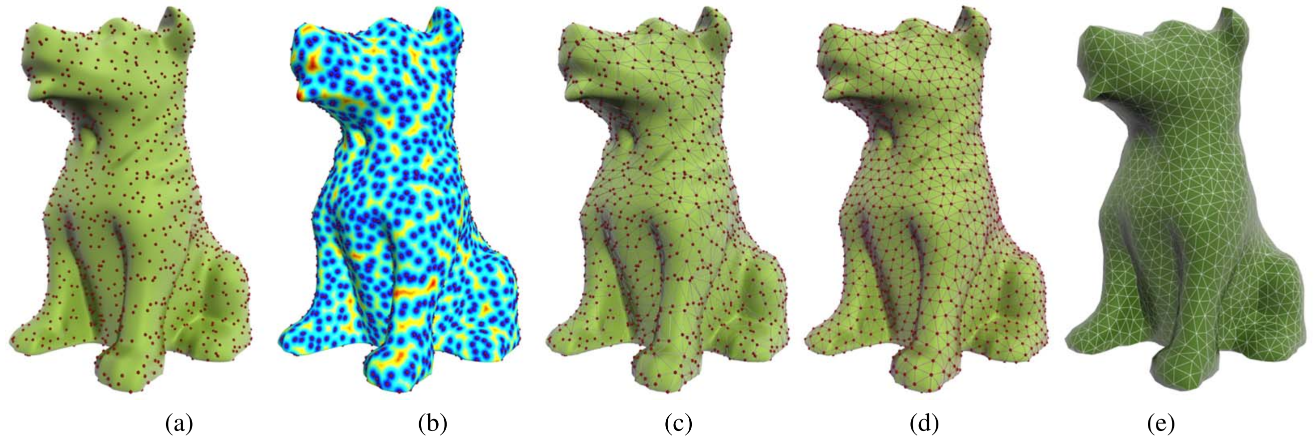


Figure 2. Algorithm overview: (a) The input Dog model with 50K vertices and 1800 randomly sampled points. (b) Take the samples as sources, and compute the multi-source-all-destination geodesic distance field. (c) The induced geodesic Delaunay triangulation of (b). (d) The updated geodesic Delaunay triangulation after 5 iterations. (e) The final simplified mesh.

- Wavelet surfaces method [8] begins with a wavelet description of the input model and results in a base shape plus a sequence of successively finer surface details [9].
- Vertex clustering [10] partitions the vertex set into a set of clusters and unify all vertices within the same cluster. Vertex clustering methods are fast and general. However, the simplification process itself is hard to control and gives poor quality approximations.
- Region Merging [11] partitions the surface into disjoint connected regions such that each region is as planar as possible.
- Vertex decimation [12], [13] is an iterative simplification algorithm. In each step of the decimation process, a vertex is selected for removal, all the faces adjacent to that vertex are removed from the model and then the resulting hole is re-triangulated.
- Edge contraction is based on the iterative contraction of vertex pairs (edges) [3], [11]. The fundamental operation is to iteratively merge two neighboring vertices to the same position. Hoppe [2] introduced the progressive mesh (PM) structure to encode a sequence of edge contraction. This technique also provides an effective technique for compressing the input geometry.

Geodesic Voronoi diagram Delaunay triangulation and Voronoi diagram in $\mathbb{R}^n$ are the most frequently employed techniques in computer graphics, e.g., mesh reconstruction [14], [15]. However, geodesic Voronoi diagram and geodesic Delaunay triangulation on polyhedral surfaces have not been widely studied due to the computational difficulty. Leibon and Letscher [16] proved that the geodesic distance based Voronoi diagram (GVD), as well as the geodesic-distance based Delaunay triangulation, exist on a manifold if the sampling points are sufficiently dense. Kimmel and Sethian [17] applied the Fast Marching method [18] to approximate GVD, while Xin and Wang [19] extended their exact geodesic algorithm [20] onto the GVD problem. Peyré and Cohen [21] took approximate GVD as a tool for remeshing and parametrization purpose. Fort and Sellarès [22] presented a method for computing generalized k-order Voronoi diagrams on triangulated surfaces. Yan *et al.* [23] successfully extended the idea of Centroidal Voronoi Tessellation (CVT) to isotropic remeshing. However, they used Restricted Voronoi Diagram (RVD) to construct CVT, rather than actual GVD, where RVD is defined as the intersection between a 3D Voronoi diagram and the input mesh surface.

Discrete geodesics In fact, the computation of geodesic Voronoi diagram and geodesic Delaunay triangulation depends heavily on algorithms for computing discrete geodesics. The discrete geodesic problem is first introduced by Sharir and Schorr [24], where an $O(n^3 \log n)$ algorithm was presented for convex polyhedra. Later, Mitchell *et al.* [25] (MMP) improved the time complexity bound to $O(n^2 \log n)$ by using the "continuous Dijkstra" technique, while Chen and Han [26] (CH) suggested building a binary tree to encode all the edge sequences that can possibly compute a shortest path, thereby improving the time complexity to $O(n^2)$. Surazhsky *et al.* [27] implemented the MMP algorithm and extended it to compute approximate geodesics with bounded error. Recently, Xin and Wang [20] improved the CH algorithm by exploiting a filtering theorem; their proposed method outperforms both the MMP and CH algorithms. Besides the exact geodesic algorithms, there are also many approximation algorithms. Polthier and Schmies introduced straightest geodesics on polyhedral surfaces [28] and developed a method to compute the evolution of distance circles on polyhedral surfaces using geodesic flow [29]. Xin and Wang [30] presented an algorithm to compute a locally exact geodesic between two distinct vertices on a polyhedral surface. In addition, the known approximation algorithms also include [31], [32], [33], [34]. Among these approximation algorithms, the Fast Marching Method [31] with an $O(n \log n)$ time complexity has been widely used in the computer graphics community.

Sampling Many remeshing algorithms [35], [36] depend on a high-quality sampling method. Poisson disk sampling [37] has been widely studied in recent years. The basic idea is to generate new sample points around existing points, and then check whether they can be added without disturbing the minimum distance constraint. Recent works have generalized Poisson disc sampling to arbitrary 3D surfaces [38] [39].

III. Algorithm for Isotropic Mesh Simplification

This section details our algorithm for isotropic mesh simplification. Section III-A gives the algorithm overview. In Section III-B, we skeleton Xin-Wang algorithm which plays a central role in computing geodesic Delaunay triangulation that will be discussed in Section III-C. Finally, Section III-D describes how to evolve sample points.

A. Overview

This subsection gives an overview of our simplification algorithm; see Figure 2 for an illustration. Our algorithm begins with a set of randomly sampled points. After that, we evolve the sample points by iteratively calling the following operations:

1) Compute the geodesic Delaunay triangulation with Xin-Wang's exact geodesic algorithm [20]; see Section III-B and Section III-C.
2) For each sample point, we compute the standard deviation of the lengths of 1-order adjacent Delaunay edges and update its location which tends to reduce the standard deviation; see Section III-D.
3) If the sample points do not change any more or the maximum standard deviation reaches a prescribed tolerance, we extract the Delaunay triangulation structure and output the simplified mesh by replacing each Delaunay edge with a straight line segment. Otherwise, we repeat 1) and 2).

B. Xin-Wang's Exact Geodesic Algorithm

The classic shortest path algorithm on graphs is due to Dijkstra's algorithm [40]. It propagates wavefronts from near to far. For the discrete geodesic problem on a polyhedral surface, the key to discretize the wavefronts is to construct *windows* such that each window encodes a set of shortest paths that share a common edge sequence. Windows fall into two categories: *pseudo-source* windows that end at a vertex, and *interval* windows that cover a continuous subset of an edge. The basic rules for computing the children of an existing window w are as follows (see Figure 3):

```
if $w$ is a pseudo-source window at vertex $v$ with geodesic distance $d$
    if $v$ is a saddle vertex (see Figure 3(a))
        for each adjacent vertex
            compute a pseudo-source window;
        end for
        for each opposite edge
            compute an interval window;
        end for
    else /*$v$ is a convex vertex*/
        No children need to be computed;
    end if
else /*$w$ is an interval window. Suppose $w$ covers an interval $[a,b]$ of the edge $\overline{v_1v_2}$. Let $v$ be the vertex opposite to $\overline{v_1v_2}$ and $I$ be the unfolded image of the last vertex passed through by $w$.*/
    if the line segment $\overline{Iv}$ is right to the interval $[a,b]$ (see Figure 3(b))
            Compute the only interval-window child on edge $\overline{v_1v}$;
    elseif the line segment $\overline{Iv}$ is left to the interval $[a,b]$ (see Figure 3(c))
            Compute the only interval-window child on edge $vv_2$;
    else /*$\overline{Iv}$ intersects the interval [a,b] at some point (see Figure 3(d)).*/
```

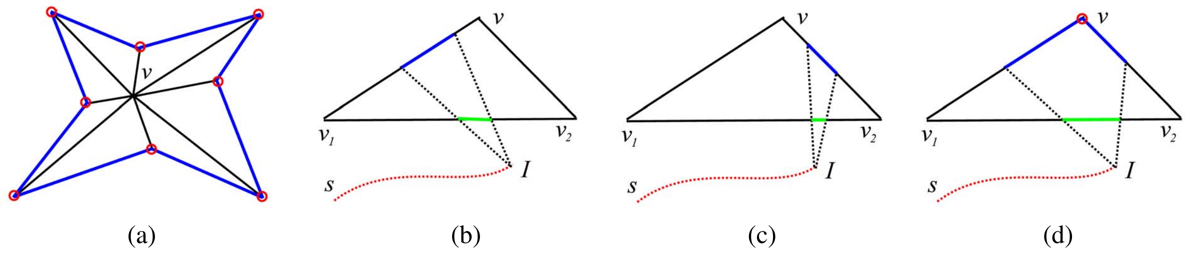


Figure 3. Computing the children of a window: (a) A pseudo-source window at a saddle vertex v may have children both on opposite edges and at adjacent vertices; (b-d) An interval window w on edge $\overline{v_1v_2}$ may have one or two interval-window children, depending on how the line segment $\overline{Iv}$ intersects w's interval.

```
        Compute two interval windows as the children, one on edge $v_1v$ and the other on edge $vv_2$;
        Compute a pseudo-source window at vertex $v$.
    end if
end if
```

The existing exact geodesic algorithms [25], [26], [20] differ in two key factors: one is how to check the validity of a new window and the other is how to control the priority of window propagation. The reason that Xin-Wang algorithm [20] outperforms the MMP algorithm [25] and the CH algorithm [26] is two-fold:

- Xin-Wang algorithm [20] checks the validity of a new window very strictly. On one hand, it inherits Chen and Han's "one angle one split" that allows at most one of the windows covering the same angle can have two children; see Figure 4(a). On the other hand, the algorithm discards an interval window if it cannot provide a shorter geodesic distance than one of the neighboring vertices; see Figure 4(b).
- A priority queue is maintained throughout the algorithm to guarantee that the wavefront propagates from near to far, like that achieved in Dijkstra's algorithm [40].

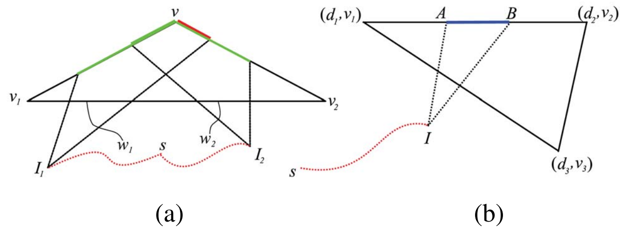


Figure 4. Xin-Wang algorithm [20] depends on two key theorems: (a) Chen and Han's "one angle one split" [26] states that at most one of the windows covering the same angle can have two children; and (b) Xin and Wang's filtering theorem [20] asserts that an interval window is no use if it cannot provide a shorter geodesic distance than one of the neighboring vertices.

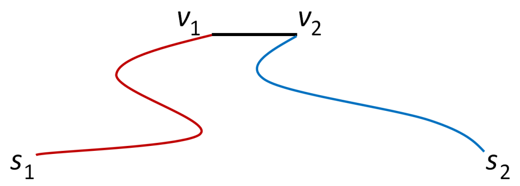


Figure 5. The key idea for computing geodesic Delaunay triangulation is that we take the sample points s_1 and s_2 as neighbors if they provide the geodesic distances for different endpoints of the same edge.

C. Geodesic Delaunay Triangulation

In this subsection, we describe how to build a geodesic Delaunay triangulation from a given set of sample points $S = \{s_i\}_{i=1}^m$:

Step 1. Use $s_i, i = 1, \cdots, m$, as source points and compute the geodesic distance field on the input mesh with Xin-Wang's geodesic algorithm [20]. So each vertex v is associated with a geodesic distance $d(v, s_i)$ where s_i is the closest sample point to v. Let $c(v)$ be the closest sample point of vertex v.

Step 2. Consider each mesh edge $e_{ij} = (v_i, v_j)$, where v_i and v_j are neighboring mesh vertices. If $c(v_i) \neq c(v_j)$, let $s_1 = c(v_i)$ and $s_2 = c(v_j)$ be the two sample points and mark them as neighbors; as shown in Figure 5.

Step 3. For every pair of sample points s_i and s_j which are marked as neighbors, find the geodesic path between s_i and s_j.

The above algorithm is inspired from such an observation: if there is a point q, such that $d(q, s_1) = d(q, s_2) < d(q, s_i), i \neq 1, 2$, then s_1 and s_2 are neighbors. Numerous experimental results show that the above algorithm gives a correct Delaunay triangulation as long as the input mesh is sufficiently dense.

We must point out that Step 3 is the most time-consuming step. However, it is not necessary for our simplification algorithm since only the neighboring structure is required. We list Step 3 here for completeness of geodesic Delaunay triangulation.

D. Evolving Sample Points

Yan *et al.* [23] successfully applied the idea of Centroidal Voronoi Tessellation (CVT) [41] onto isotropic remeshing. Furthermore, their algorithm uses a quasi-Newton method to acquire a fast convergence. In fact, the construction of CVTs can be viewed as an energy minimization process. In the case of Euclidean metric, the objective of CVT is to make the Voronoi cells to be similar in shape and basically equal in size.

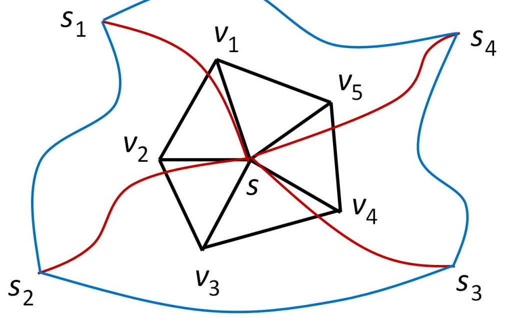


Figure 6. The adjacent vertices $v_i, i = 1, 2, 3, 4, 5$ serve as candidate sample points for the existing sample point s. We take v_i as the substitute of s if v_i can reduce the standard deviation of the lengths of the geodesic edges $\widehat{ss_j}, j = 1, 2, 3, 4$.

In this paper, we use another optimization technique for mesh simplification purpose. Suppose the sample point s is neighboring to $s_1, s_2, \cdots, s_k$ in the geodesic Delaunay triangulation. Let $\widehat{ss_i}$ be the geodesic path between s and s_i. Our basic idea is to make the geodesic Delaunay edges to be as equal as possible in length. Mathematically, we need to find a substitute s' for s in its neighborhood, such that the standard deviation of $\|\widehat{s's_1}\|, \|\widehat{s's_2}\|, \cdots, \|\widehat{s's_k}\|$ is minimized. Technically, we take the 1-order adjacent mesh vertices of the sample point, $v_1, v_2, \cdots, v_m$, as candidates and use $v_j (1 \leq j \leq m)$ to replace s if v_j can give the minimum standard deviation; see Figure 6. At each iteration, we find a new substitue for every sample point in S and thus update S into S'. If $S = S'$ or the maximum standard deviation is less than a prescribed tolerance, we extract the geodesic Delaunay triangulation structure of S' as the final simplified result.

For showing the difference between CVT and our Delaunay evolving technique, we take a collection of 2D points as the input and compare the two methods in Figure 7. It can be seen that by minimizing energy of CVT, the final Voronoi cells tend to be similar in shape and basically equal in size, while by locally minimizing the standard deviation of Delaunay edge lengths, the Delaunay edges tend to be equal in length. (We must point out that we add another constaint, for the case of Figure 7(c), that the sample points are required to be not too far away from the boundary. The constraint is not necessary for simplifying meshes.)

IV. Experimental Results

We implemented our mesh simplification algorithm and tested it on a PC with an Xeon 2.66GHz CPU and 4GB RAM. The experimental results show that our algorithm is robust, intrinsic to the geometry and insensitive to the input mesh triangulation. Therefore, it can be applied to real-world models with complicated geometry and topology. Figure 8 and Figure 9 show two models of non-trivial topology, the genus-5 Decocube model (50K vertices) and the genus-3 Two kids model (130K vertices). We simplified both models to approximate 2K vertices by our algorithm. In spite of the high simplification ratio, the resulting meshes still have a high-quality triangulation without loss of significant

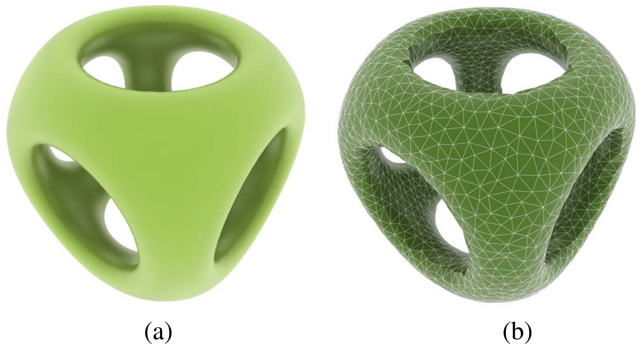


Figure 8. Test our algorithm on the genus-5 Decocube model: (a) the input Decocube model with 80K vertices and (b) the simplified mesh with only 2.2K vertices (3 iterations).

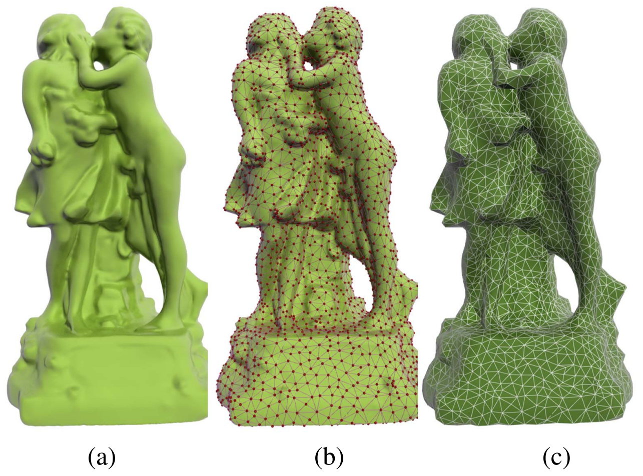


Figure 9. Test our algorithm on the genus-3 Two-kids model: (a) the input Two-kids model with 130K vertices, (b) the geodesic Delaunay triangulation after upon 4 iterations and (c) the simplified mesh with only 2.5K vertices.

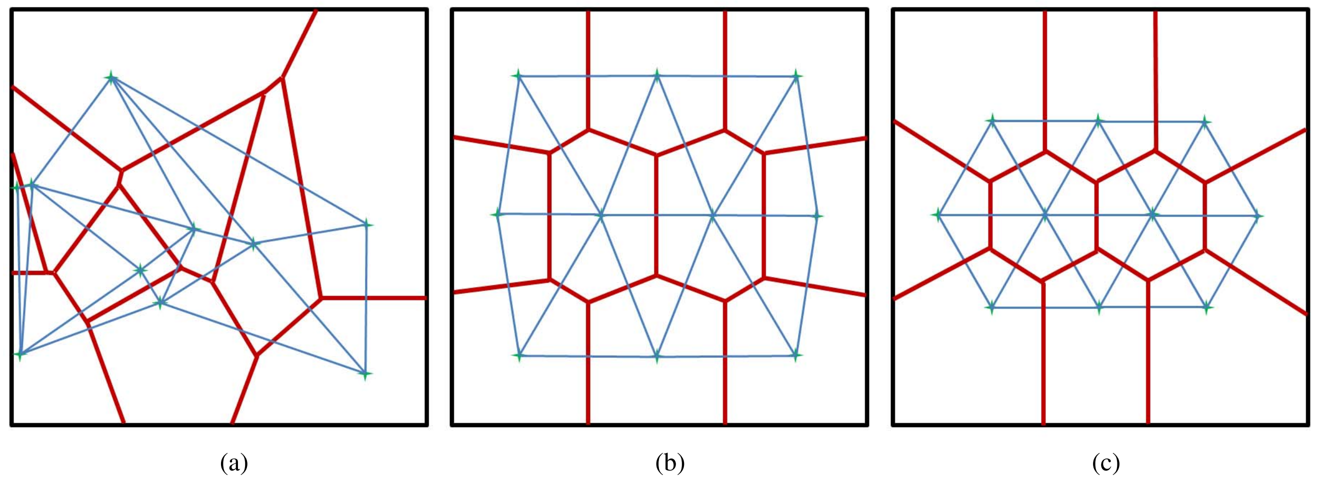


Figure 7. Comparison between CVT and our Delaunay evolving technique: (a) given a set of unevenly distributed sample points, the Voronoi diagram and Delaunay triangulation are drawn respectively in red and blue; (b) by minimizing energy of CVT, the final Voronoi cells are similar in shape and basically equal in size; and (c) after we evolve the Delaunay triangulation by locally minimizing the standard deviation, the Delaunay edges tend to be equal in length.

geometric features. As mentioned before, our algorithm takes advantage of the intrinsic property of discrete geodesic. Thus, the resulting geodesic Delaunay triangulation is intrinsic to the geometry, independent of the embedding space (see Figure 11). Moreover, our method is robust and insensitive to the initial mesh triangulation. Figure 10 shows an example of the Moai model with irregular triangulation. Our method can still guarantee a relatively good remeshing. The two popular mesh simplification algorithms, QSlim [3] and progressive meshes [2], heavily depend on the initial mesh triangulation and the embedding space. However, these approaches are more efficient than ours. The timing statistics of QSlim [3], progressive mesh [2] and our algorithm are 0.016 seconds, 0.012 seconds and 0.47 seconds, respectively.

Table I shows the timing statistics of our experimental results. We can clearly see that the running time for each iteration is basically linear to the mesh size if we fix the number of sample points. We also observe that the multi-source geodesic algorithm always runs several times faster than the single-source geodesic algorithm for the same input model. Taking the Decocube model for an example, the time for discrete geodesic is 1.794 seconds for the single-source case, while it reduces to 0.468 seconds for the case of 2,200 source points. This shouldn't be surprising since Xin-Wang's algorithm maintains a priority queue when the wavefronts propagates. For single-source geodesic, the wavefront is quite big and the algorithm terminates only when the wavefront touches all mesh vertices. For multi-source geodesic, each source initializes a wavefront propagation, and the wavefront is usually much smaller than the single-source case. This is the main reason that our evolving algorithm runs efficiently in practice.

Table I
TIMING STATISTICS (SECONDS) MEASURED ON A WORKSTATION WITH AN XEON 2.66GHZ CPU AND 4GB RAM.

Models	#vertices	#sample points	#iterations	Time per iteration (s)
ISVD (Figure 1)	24,998	600	3	0.297
Dog (Figure 2)	50,000	1,800	4	0.529
Decocube (Figure 8)	80,000	2,200	3	0.468
Dragon (Figure 12(b))	750,000	10,000	4	19.937
Dragon (Figure 12(c))	750,000	5,000	5	17.833
Dragon (Figure 12(d))	750,000	2,800	5	15.612
Two-kids (Figure 9)	130,000	2,500	5	0.805
Moai (Figure 10)	10,000	2,000	5	0.094

V. CONCLUSION AND FUTURE WORK

In this paper, we present an intrinsic mesh simplification algorithm. The key idea is to locally optimize the Delaunay triangulation such that the Delaunay edges are as equal as possible in length. Our algorithm is robust and insensitive to mesh triangulation due to the intrinsic feature of geodesic Voronoi diagram. Furthermore, as the main operations are local, the proposed algorithm is efficient and parallel in nature.

Our current implementation only utilized CPU based parallelization with OpenMP. We will develop GPU implementation to further boost the performance in the near future. We will also extend our framework to anisotropic mesh simplication.

ACKNOWLEDGMENT

This project was partially supported by NRF2008IDM-IDM-004-006 and AcRF Tier1 69/07.

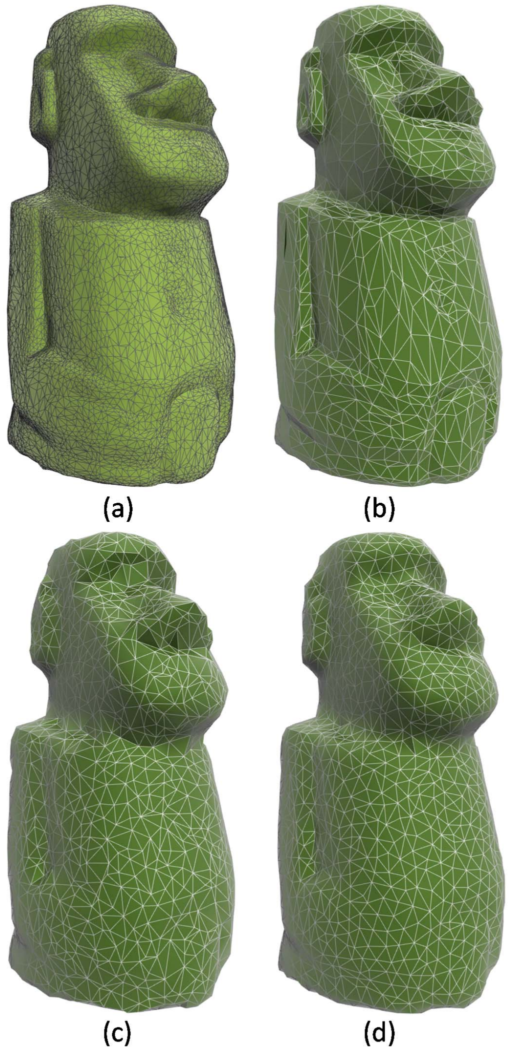


Figure 10. Comparison between quadric error metrics based simplification [3], progressive mesh [2] and our method: (a) the original Moai model with an irregular triangulation, (b) quadric error metrics based simplification [3], (c) progressive mesh result [2] and (d) our simplification result, where the target number of sample points is 2000 for all the three methods.

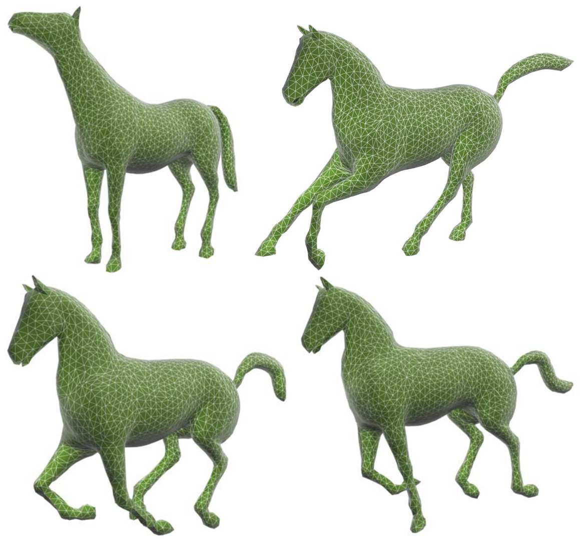


Figure 11. Our method is intrinsic to the geometry and independent of the embedding space.

References

[1] M. Franc, "Methods for polygonal mesh simplification," Ph.D. dissertation, University of West Bohemia, 2007.

[2] H. Hoppe, "Progressive meshes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96, 1996, pp. 99–108.

[3] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '97, 1997, pp. 209–216.

[4] H. P.S. and G. M., "Survey of surface simplification algorithms," Carnegie Mellon University - Dept. of Computer Science, Tech. Rep., 1997.

[5] J. O. Talton, "A short survey of mesh simplification algorithm," University of Illinois at Urbana-Champaign, Tech. Rep., 2004.

[6] F. S. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, pp. 191–205, 2003.

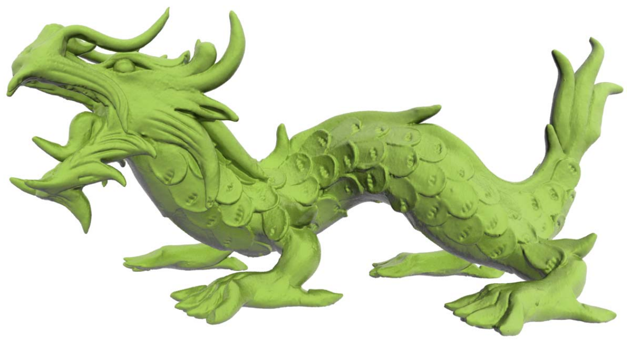
[7] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, "Simplification envelopes," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96, 1996, pp. 119–128.

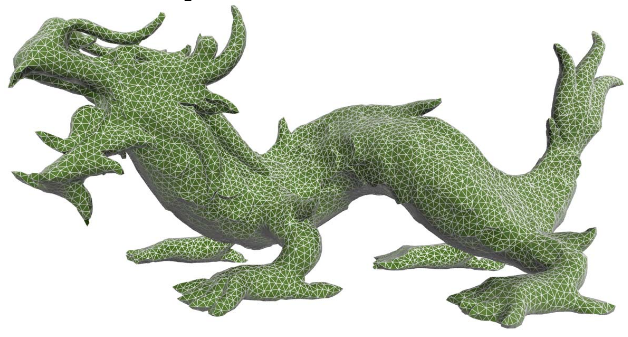
[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '95, 1995, pp. 173–182.

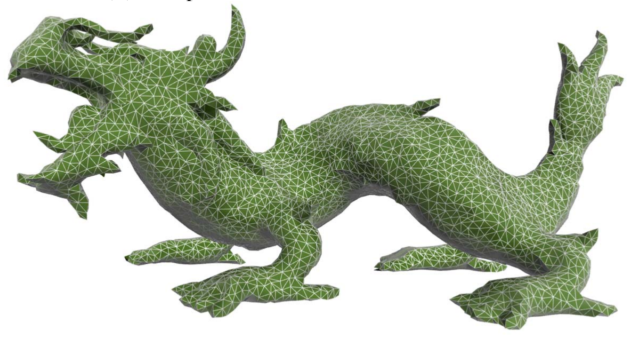
[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *SIGGRAPH '93 Proc.*, Aug. 1993, pp. 19–26.

[10] W. Schroeder, "A topology modifying progressive decimation algorithm," *Visualization Conference, IEEE*, vol. 0, p. 205, 1997.

[11] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes," *Visualization Conference, IEEE*, vol. 0, 2001.


(a) Original model with 750K vertices


(b) Simplified model with 10K vertices


(c) Simplified model with 5K vertices

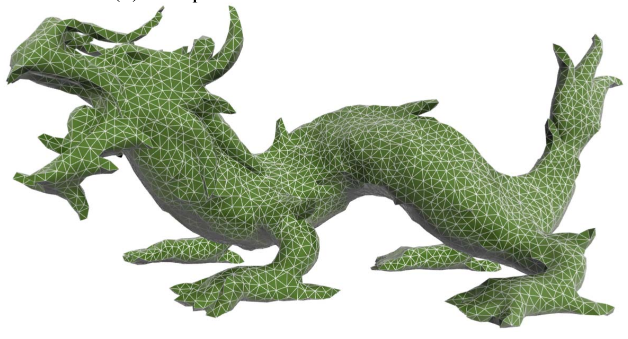

(d) Simplified model with 2.8K vertices

Figure 12. Large scale model

[12] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," *SIGGRAPH Comput. Graph.*, vol. 26, pp. 65–70, July 1992.

[13] K. J. Renze and J. H. Oliver, "Generalized unstructured decimation," *IEEE Comput. Graph. Appl.*, vol. 16, pp. 24–32, November 1996.

[14] X. Liu, J. G. Rokne, and M. L. Gavrilova, "A novel terrain rendering algorithm based on quasi delaunay triangulation," *Vis. Comput.*, vol. 26, pp. 697–706, June 2010.

[15] X. Liu, M. L. Gavrilova, and J. Rokne, "Incorporating object-centered sampling and delaunay tetrahedrization for visual hull reconstruction," *Vis. Comput.*, vol. 25, pp. 381–389, April 2009.

[16] G. Leibon and D. Letscher, "Delaunay triangulations and Voronoi diagrams for Riemannian manifolds," in *Proceedings of the sixteenth annual symposium on Computational geometry*. ACM, 2000, pp. 341–349.

[17] R. Kimmel and J. A. Sethian, "Fast voronoi diagrams and offsets on triangulated surfaces," in *Proc. of AFA Conf. on Curves and Surfaces*. University Press, 1999, pp. 193–202.

[18] J. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 4, p. 1591, 1996.

[19] S.-Q. Xin and G.-J. Wang, "Applying the improved chen and han's algorithm to different versions of shortest path problems on a polyhedral surface," *Comput. Aided Des.*, vol. 42, pp. 942–951, October 2010.

[20] ——, "Improving chen and han's algorithm on the discrete geodesic problem," *ACM Trans. Graph.*, vol. 28, pp. 104:1–104:8, September 2009.

[21] G. Peyré and L. Cohen, "Geodesic remeshing using front propagation," *International Journal of Computer Vision*, vol. 69, no. 1, pp. 145–156, 2006.

[22] M. Fort and J. A. Sellarès, "Computing generalized higher-order voronoi diagrams on triangulated surfaces," *Applied Mathematics and Computation*, vol. 215, no. 1, pp. 235–250, 2009.

[23] D. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," in *Computer graphics forum*, vol. 28, no. 5. Wiley Online Library, 2009, pp. 1445–1454.

[24] M. Sharir and A. Schorr, "On shortest paths in polyhedral spaces," *SIAM J. Comput.*, vol. 15, no. 1, pp. 193–215, 1986.

[25] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, “The discrete geodesic problem,” *SIAM J. Comput.*, vol. 16, no. 4, pp. 647–668, 1987.

[26] J. Chen and Y. Han, “Shortest paths on a polyhedron,” in *SCG ’90: Proceedings of the Sixth Annual Symposium on Computational Geometry*, 1990, pp. 360–369.

[27] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, “Fast exact and approximate geodesics on meshes,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 553–560, 2005.

[28] K. Polthier and M. Schmies, *Mathematical Visualization*. Springer Verlag, 1998, ch. Straightest geodesics on polyhedral surfaces, p. 391.

[29] ——, “Geodesic flow on polyhedral surfaces,” in *Proceedings of Eurographics-IEEE Symposium on Scientific Visualization ’99*, 1999, pp. 179–188.

[30] S.-Q. Xin and G.-J. Wang, “Efficiently determining a locally exact shortest path on polyhedral surfaces,” *Computer-Aided Design*, vol. 39, no. 12, pp. 1081–1090, 2007.

[31] R. Kimmel and J. A. Sethian, “Computing geodesic paths on manifolds,” in *Proc. Natl. Acad. Sci. USA*, 1998, pp. 8431–8435.

[32] S. Har-Peled, “Approximate shortest paths and geodesic diameter on a convex polytope in three dimensions,” *Discrete & Computational Geometry*, vol. 21, no. 2, pp. 217–231, 1999.

[33] P. K. Agarwal, S. Har-Peled, and M. Karia, “Computing approximate shortest paths on convex polytopes,” in *Symposium on Computational Geometry*, 2000, pp. 270–279.

[34] A. Spira and R. Kimmel, “Geodesic curvature flow on parametric surfaces,” in *Curve and Surface Design*, 2002, pp. 365–373.

[35] P. Alliez, É. de Verdière, O. Devillers, and M. Isenburg, “Isotropic surface remeshing,” in *Proceedings of shape modeling international*, vol. 49, 2003, p. 58.

[36] Y. Fu and B. Zhou, “Direct sampling on surfaces for high quality remeshing,” *Comput. Aided Geom. Des.*, vol. 26, pp. 711–723, August 2009.

[37] L.-Y. Wei, “Parallel poisson disk sampling,” in *ACM SIGGRAPH 2008 papers*, ser. SIGGRAPH ’08. New York, NY, USA: ACM, 2008, pp. 20:1–20:9.

[38] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, “Parallel poisson disk sampling with spectrum analysis on surfaces,” *ACM Trans. Graph.*, vol. 29, pp. 166:1–166:10, 2010.

[39] H. Li, L.-Y. Wei, P. V. Sander, and C.-W. Fu, “Anisotropic blue noise sampling,” *ACM Trans. Graph.*, vol. 29, pp. 167:1–167:12, December 2010.

[40] E. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[41] Q. Du, V. Faber, and M. Gunzburger, “Centroidal Voronoi tessellations: applications and algorithms,” *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.