Improving Chen and Han's Algorithm on the Discrete Geodesic Problem

SHI-QING XIN and GUO-JIN WANG
Zhejiang University, PR China

The computation of geodesic distances or paths between two points on triangulated meshes is a common operation in many computer graphics applications. In this article, we present an exact algorithm for the single-source all-vertices shortest path problem.

Mitchell et al. [1987] proposed an $O(n^2 \log n)$ method (MMP), based on Dijkstra's algorithm, where n is the complexity of the polyhedral surface. Then, Chen and Han [1990] (CH) improved the running time to $O(n^2)$. Interestingly Surazhsky et al. [2005] provided experimental evidence demonstrating that the MMP algorithm runs many times faster, in practice, than the CH algorithm.

The CH algorithm encodes the structure of the set of shortest paths using a set of windows on the edges of the polyhedron. Our experiments showed that in many examples over 99% of the windows created by the CH algorithm are of no use to define a shortest path. So this article proposes to improve the CH algorithm by two separate techniques. One is to filter out useless windows using the current estimates of the distances to the vertices, the other is to maintain a priority queue like that achieved in Dijkstra's algorithm. Our experimental results suggest that the improved CH algorithm, in spite of an $O(n^2 \log n)$ asymptotic time complexity, greatly outperforms the original CH algorithm in both time and space. Furthermore, it generally runs faster than the MMP algorithm and uses considerably less space.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Design and analysis of algorithms, computational geometry, shortest path problems

ACM Reference Format:

Xin, S.-Q. and Wang, G.-J. 2009. Improving Chen and Han's algorithm on the discrete geodesic problem. ACM Trans. Graph. 28, 4, Article 104 (August 2009), 8 pages. DOI = 10.1145/1559755.1559761 http://doi.acm.org/10.1145/1559755.1559761

1. INTRODUCTION

The *discrete geodesic problem* is to determine the shortest path between two points s and t on a polyhedral surface $\mathcal{S}$. It arises naturally in applications such as robotics, motion planning, geographic information systems, and navigation [Agarwal et al. 2000; Sethian 1999], and is an important topic in computational geometry. It is also central to computer graphics because the computation of geodesic paths is a common operation in many graphics problems [Surazhsky et al. 2005]. For example, cutting a mesh into several charts [Sander et al. 2003; Zhou et al. 2004] and establishing a surface distance metric [Zigelman et al. 2002; Peyré and Cohen 2005] often involve computation of shortest paths.

The "single source, all destinations" shortest path problem is to compute a data structure that allows the shortest path from the source to any destination point on the surface to be reported quickly. There are many exact algorithms for this problem. Sharir and Schorr [1986] were the first experimentalists to present a polynomial algorithm of time $O(n^3 \log n)$, although their algorithm only applies to a convex polytope, where n is the complexity of the surface. The time complexity was subsequently improved by Mount [1984] to $O(n^2 \log n)$. For an arbitrary polyhedral surface, Mitchell et al. [1987] (MMP) gave an $O(n^2 \log n)$ algorithm, which inherits the paradigm of Dijkstra's [1959] algorithm. Later, Chen and Han [1990] (CH) proposed an $O(n^2)$ algorithm based on a key observation of "one angle, one split." (Also, Kapoor [1999] announced a further improvement, but his proof has not yet been accepted by the research community.)

The CH algorithm consists of two phases. In the first phase, the shortest path from the source to each vertex is computed, along with a set of windows encoding information about the shortest paths from the source to points on the edges. In the second phase, the windows are used to compute a decomposition of the polyhedral surface, with which a shortest path to any destination can be reported in $O(\log n)$ time. For most applications in computer graphics, when the input mesh is composed of well-shaped triangles, it is sufficient to compute the shortest path to each vertex, making the second phase unnecessary. In this article we focus on the first phase of the CH

This work was supported by the National Basic Research Program of China (No. 2004CB719400) and the NNSF of China (No. 60873111).
Authors' address: S.-Q. Zin, G.-J. Wang (Corresponding Author), State Key Laboratory of CAD & CG, Department of Mathematics, Zhejiang University, Hangzhou 310027, PR China; email: wanggj@zju.edu.cn.

DOI 10.1145/1559755.1559761 http://doi.acm.org/10.1145/1559755.1559761

algorithm: the "single source, all vertices" shortest path problem. For a polyhedral surface with many long, thin triangles, we suggest storing all the windows that possibly determine a shortest path rather than throwing away those one-child windows, although this requires more space. Finally, for the destination point p, lying in a face interior, we consider all windows on the three edges bounding the face, and choose the window that provides the shortest distance to point p. This method comes from Surazhsky et al. [2005].

In fact, the discrete geodesic problem has many versions such as "single source, single destination" [Hershberger and Suri 1995; Kanai and Suzuki 2000; Pham-Trong et al. 2001; Morera et al. 2005; Polthier and Schmies 2006; Xin and Wang 2007], "single source, all destinations" [Mount 1984; Sharir and Schorr 1986; Mitchell et al. 1987; Chen and Han 1990; Agarwal et al. 1997b; Kimmel and Sethian 1998; Novotni and Klein 2002; Surazhsky et al. 2005] and "all pair" [Varadarajan and Agarwal 1997; Agarwal et al. 1997a; Har-Peled 1999a, 1999b; Agarwal et al. 2000; Aleksandrov et al. 2003]; see Mitchell [2000], Aleksandrov et al. [2003], and Mitchell and Sharir [2004] for a survey.

Kaneva and O'Rourke [2000] provided experimental results on the CH algorithm. However it should be noted that their implementation, as well as ours, only implements the first phase of Chen and Han's algorithm. Surazhsky et al. [2005] implemented the first phase of the MMP algorithm and induced an approximation algorithm at the same time. All the experimental results demonstrate that the MMP algorithm, of an $O(n^2 \log n)$ complexity, runs many times faster than the $O(n^2)$-time CH algorithm. In contrast, the weakness of the CH algorithm, as our experiments show, lies in the fact that it generates too many useless windows that don't contribute to any shortest path. Over 99% of the windows created by the CH algorithm can be abolished in many examples. So this article proposes to improve the CH algorithm by implementing two new techniques. The primary technique is to filter out useless windows using the current estimates of the distances to the vertices, and the other technique is to maintain a priority queue like that achieved in Dijksgra's algorithm. Our experimental results suggest that the improved CH algorithm, in spite of an $O(n^2 \log n)$ asymptotic time complexity, greatly outperforms the original CH algorithm in both time and space. Furthermore, our algorithm generally runs faster than the MMP algorithm and uses far less space.

The article is set out as follows. Section 2 gives some preparatory lemmas. Section 3 describes the techniques of the improved CH algorithm. We make comparisons among the CH algorithm, the MMP algorithm, and the improved CH algorithm in Section 4 and we draw conclusions in Section 5.

2. GEOMETRIC PRELIMINARIES

Let $\mathcal{S}$ be a triangulated polyhedral surface in $\mathcal{R}^3$, defined by a set of faces, edges, and vertices. Assume that the surface $\mathcal{S}$ has n faces and s is a vertex of the polyhedron. Our task is to compute a shortest path from the source vertex s to any vertex t with the path restricted on $\mathcal{S}$. We first borrow a terminology from Mitchell et al. [1987].

A *face sequence* $\mathcal{F}$ is defined by a list of adjacent faces $f_1, f_2, \ldots, f_{m+1}$ such that f_i and f_{i+1} share a common edge e_i; see Figure 1. We call the list of edges $\mathcal{E} = (e_1, e_2, \ldots, e_m)$ an *edge sequence*. Unless otherwise specified, the face sequences mentioned later are all *simple*: the faces within a face sequence are different from each other.

In studying the shortest path problem, a technique called *planar unfolding* is often used. An edge sequence $\mathcal{E} = (e_1, e_2, \ldots, e_m)$ is unfolded in this way: rotate f_1 around e_1 until its plane coincides with that of f_2 (f_1 and f_2 are kept on different sides of e_1), rotate

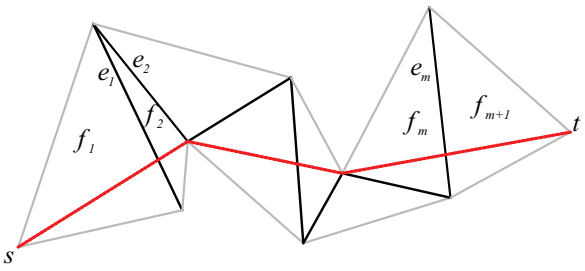


Fig. 1. Definitions of the face sequence $\mathcal{F}$ and the edge sequence $\mathcal{E}$. The thick polyline denotes the shortest path from the point s to the point t restricted on $\mathcal{F}$, while the light-gray polygon denotes the boundary of $\mathcal{F}$.

f_1 and f_2 around e_2 until their plane coincides with that of f_3, and repeat the process until all the faces $f_1, f_2, \ldots, f_m$ lie in the plane of f_{m+1} .

Obviously, a globally shortest path is, without doubt, a geodesic. More properties can be found in Sharir and Schorr [1986], Mount [1984], and Chen and Han [1990]. For example, two shortest paths from the same source cannot intersect each other except at the source point or the destination point or a saddle vertex. For description of the general form of a shortest path, we cite a lemma from Mitchell et al. [1987].

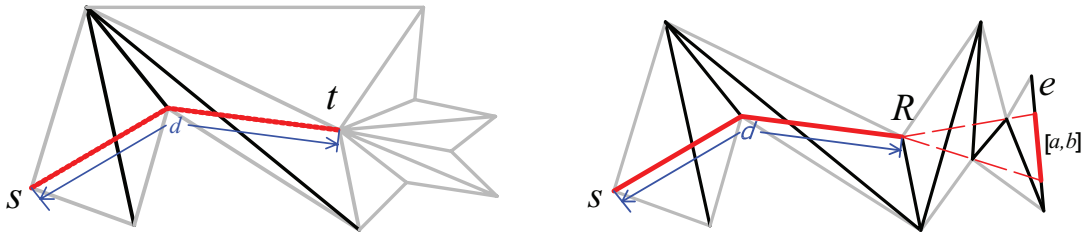
LEMMA 2.1. *The general form of a locally shortest path is a path that goes through an alternating sequence of vertices and edge sequences such that the unfolded image of the path along any edge sequence is a straight line segment and both the angles of the path passing through a vertex are greater than or equal to* π. *A globally shortest path is a geodesic, and it has the additional property that no edge can appear in more than one edge sequence and each edge sequence must be simple.*

Based on Lemma 2.1, we observe that a *geodesic sequence* Γ consists of a sequence of edges and vertices, beginning with the source s and ending with a vertex or an edge. Accordingly, we use two types of windows to encode the geodesic sequence information: *pseudo-source* windows and *interval* windows; see Figure 2. If Γ ends with a vertex v, we simply use a pair (d, v) to denote the key information, where d is the shortest distance from the source to vertex v restricted on Γ. If Γ ends with an edge e, then the last vertex r in Γ is defined to be the *root*. In the course of planar unfolding of the edge sequence next to vertex r, rotations are required, inducing a list of r's *images* $\{\mathcal{I}_i\}$, among which the last image $\mathcal{I}$ is located on the plane of edge e and the edge prior to e. In fact, the point set:

$$\{p | p \in e,\ \text{the shortest path from } s \text{ to } p \text{ can be unfolded into a straight line segment}\},$$

is an interval [Sharir and Schorr 1986], say, $[a, b]$. Then we can use a four-tuple $(d, \mathcal{I}, e, [a, b])$ as an interval window w to encode the key information, where d is the shortest distance from the source s to the root r restricted on Γ, as is shown in Figure 2(b).

During window propagation, a pseudo-source window at a saddle vertex v (the sum of the incident angles is greater than 2π) or a boundary vertex v can have children: an interval-window child on each edge opposite to v and a pseudo-source-window child at each vertex adjacent to v, while an interval window on edge e can have at most 3 children: two interval-window children on the two edges next to e and one pseudo-source-window child at the vertex opposite to e. According to the parent-child relationships, we can build a sequence tree $\mathcal{T}$ of an exponential size to contain the windows. Note that $\mathcal{T}$



(a) A pseudo-source window (b) An interval window

Fig. 2. A pseudo-source window and an interval window.

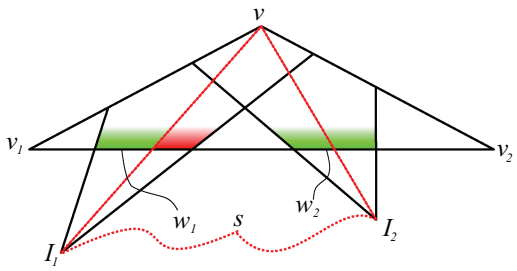


Fig. 3. The principle of one angle one split (see Lemma 2.2) implies that only three of the four possible edge sequences in this example can provide shortest paths. The three green windows belong to edge sequences that will continue, and the red one corresponds to an edge sequence that must stop because the shortest paths that it contains cannot continue to pass this triangle.

has at most n levels, where n is the total number of the faces. The CH algorithm [Chen and Han 1990] avoids exponential explosion based on the following Lemma 2.2, namely the property of "one angle one split" [Chen and Han 1990]. In Figure 3, two windows w_1, w_2, are entering $\triangle v_1v_2v$ from edge $\overline{v_1v_2}$ and both cover the vertex v. "One angle, one split" implies that at most one of w_1, w_2, can have two interval-window children during window propagation. We say that the two-child window *occupies* the angle $\angle v_1vv_2$ over the one-child one. Of course, $w_i (i = 1, 2)$ can also have a pseudo-source-window child at vertex v only if it can provide the shortest distance to v.

LEMMA 2.2. *Let w_1 and w_2 be two windows on the same directed edge $\overline{v_1v_2}$ of $\triangle v_1v_2v$, shown in Figure 3. If both of the windows cover the vertex v, then at most one of them can have two children which could be used to define a shortest sequence.*

3. THE IMPROVED CH ALGORITHM

The CH algorithm [Chen and Han 1990] is of an $O(n^2)$ time complexity, but in practice it runs very inefficiently, compared with the $O(n^2 \log n)$-time MMP algorithm [Mitchell et al. 1987]. This is demonstrated by previous experimental results [Kaneva and O'Rourke 2000; Surazhsky et al. 2005]. The inefficiency of the CH algorithm, we think, is because it generates considerably many useless windows that don't contribute to any shortest path. That is to say, the key idea in Chen and Han [1990], "one angle one split," is still too loose to effectively check the validity of a new window. We will describe the CH algorithm in Section 3.1. Section 3.2 presents a filtering theorem for checking a new window more strictly and Section 3.3 suggests that we should also maintain a priority queue throughout the algorithm. Finally, in Section 3.4 we propose a method for backtracing a shortest path from the source to any vertex, or even any surface point.

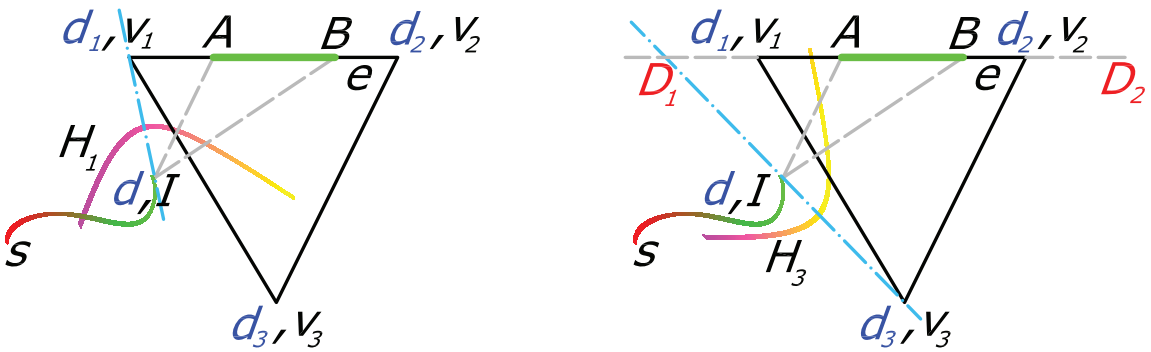
3.1 The CH Algorithm

We first need to conduct initialization: we set the distances at vertices, except the source, to be $+\infty$; then, we associate each angle with a null interval window and associate each vertex with a null pseudo-source window; finally, we introduce a first in first out queue $\mathcal{Q}$ to store pending events.

ALGORITHM 3.1. **The CH algorithm**

Assign the source s with distance 0, create a pseudo-source window w for s, and put w into the queue $\mathcal{Q}$;
While $\mathcal{Q}$ is not empty and the level size doesn't exceed the face number n
  Take out the head window w from $\mathcal{Q}$;
  If w is a pseudo-source window, say, $w = (d, v)$
    If d is less than the current distance estimate at vertex v
      Update the distance at v;
      If v is a saddle vertex
        Delete the old pseudo-source window at v and its subtrees;
        For each edge opposite to v, add a child window $(d, v, e, [0, 1])$ onto the tail of $\mathcal{Q}$;
        Update the distance of each vertex v' incident to v with w and add a pseudo-source window $(d + \|\overline{vv'}\|, v')$ to $\mathcal{Q}$ if $d + \|\overline{vv'}\|$ is less than the current distance at v';
  Else /*w is an interval window, say, $w := (d, \mathcal{I}, e, [a, b])$.*/
    If w has only one child on the left (right) edge, or w fails to occupy the opposite angle over the existing window w' according to Lemma 2.2, then
      Compute the only child and push it into $\mathcal{Q}$;
    Else /*w occupies the opposite angle over w'*/
      Delete the abolished subtree of w';
      Compute the two children of w and push them into $\mathcal{Q}$;
      Check if w can provide a shorter distance to the vertex v opposite to edge e; if true, update the distance estimate at v; and if v is a saddle vertex or a boundary vertex, we need also generate a pseudo-source window at v and insert it into the priority queue $\mathcal{Q}$.

Chen and Han [1990] proved that the total number of nodes generated—the abolished nodes included—is $O(n^2)$, rather than of an exponential size. But in order to achieve an $O(n)$ space complexity, they suggested only storing leaf nodes and branch nodes, while throwing away the one-child interval windows. So the sequence tree of the CH algorithm is a conceptual object rather than a real one. When a pseudo-source window w_v at vertex v is occupied over by



(a) $d + \|\overline{\mathcal{I}B}\| > d_1 + \|\overline{v_1B}\|$ (b) $d + \|\overline{\mathcal{I}A}\| > d_3 + \|\overline{v_3A}\|$

Fig. 4. Proof of Theorem 3.2.

another one, we need to delete the entire subtree rooted at w_v. But when an interval window w is occupied over, one of its subtrees should be deleted, leaving the other subtree, say, rooted at window w'. At the same time, we should update window w''s parent to be w's parent. Finally, we delete one-child window w. At the conclusion of Algorithm 3.1, the information encoded on vertices and angles is enough for backtracing the shortest paths to vertices.

3.2 Checking Windows with the Distances to Vertices

As we mentioned, in spite of the low complexity in time and space, the CH algorithm creates lots of useless windows. Therefore, the natural approach is to check new windows with a stricter rule. Consider the situation in Figure 4. The window $w = (d, \mathcal{I}, e, \overline{AB})$ encodes a set of shortest paths that enter the triangle through edge $\overline{v_1v_3}$ and exit within segment $\overline{AB}$. Thus we have the following theorem.

THEOREM 3.2. *Let w be a window that enters $\triangle v_1v_2v_3$ through edge $\overline{v_1v_3}$. Assume that d_1, d_2, d_3 are respectively the minimum-so-far distance at the three vertices v_1, v_2, v_3. Then w can't define a shortest sequence if*

$$d + \|\overline{\mathcal{I}B}\| > d_1 + \|\overline{v_1B}\|,$$

or

$$d + \|\overline{\mathcal{I}A}\| > d_2 + \|\overline{v_2A}\|,$$

or

$$d + \|\overline{\mathcal{I}A}\| > d_3 + \|\overline{v_3A}\|.$$

PROOF. Let $\mathcal{H}_i$ denote the half-hyperboloid determined by the equation:

$$\|\overline{pv_i}\| + d_i = \|\overline{p\mathcal{I}}\| + d, \ i = 1, 2, 3.$$

First, we consider the situation $d + \|\overline{\mathcal{I}B}\| > d_1 + \|\overline{v_1B}\|$; see Figure 4(a). If $d_1 \geq d$, then v_1 naturally lies in the *interior* of the half-hyperboloid $\mathcal{H}_1$, where the interior is defined as the convex part. Obviously, point B also lies in the interior of $\mathcal{H}_1$ if $d+\|\overline{\mathcal{I}B}\| > d_1+\|\overline{v_1B}\|$. Then the convexity of $\mathcal{H}_1$'s interior induces that segment $\overline{v_1B}$ lies entirely on the interior side of $\mathcal{H}_1$. Therefore $d+\|\overline{\mathcal{I}p}\| > d_1+\|\overline{v_1p}\|$ holds for each $p \in \overline{v_1B}$. If $d_1 < d$, then v_1 lies in the exterior of $\mathcal{H}_1$. At the same time, $d+\|\overline{\mathcal{I}B}\| > d_1+\|\overline{v_1B}\|$ implies that B also lies in $\mathcal{H}_1$'s exterior. Since the ray from focus v_1 either intersects $\mathcal{H}_1$ at a unique point or fails to intersect $\mathcal{H}_1$, segment $\overline{v_1B}$ must lie entirely on the exterior side of $\mathcal{H}_1$. Thus we also have $d + \|\overline{\mathcal{I}p}\| > d_1 + \|\overline{v_1p}\|$ for each $p \in \overline{v_1B}$. Putting them together, we conclude that w can't define a shortest sequence if $d + \|\overline{\mathcal{I}B}\| > d_1 + \|\overline{v_1B}\|$. Similar arguments apply to the cases of $d + \|\overline{\mathcal{I}A}\| > d_2 + \|\overline{v_2A}\|$.

Next, we consider the situation $d+\|\overline{\mathcal{I}A}\| > d_3+\|\overline{v_3A}\|$ as shown in Figure 4(b). The observation that the quadrilateral $v_1\mathcal{I}v_3v_2$ is convex shows $\overrightarrow{v_1v_2} \bullet \overrightarrow{\mathcal{I}v_3} > 0$. So if the straight line v_1v_2 and the line $\mathcal{I}v_3$ have an intersection point p, then p must be to the left of both segment $\overline{v_1v_2}$ and segment $\overline{\mathcal{I}v_3}$ or to the right of them. Then we define two points D_1, D_2 like this: let D_1 (respectively, D_2) be just the intersection point p if p is to the left (respectively, right) of segment $\overline{v_1v_2}$ and otherwise be an infinite point on the leftward (respectively, rightward) extension line of segment $\overline{v_1v_2}$. Since segment $\overline{D_1D_2}$ always lies on one side of the half-hyperboloid $\mathcal{H}_3$'s axis of symmetry $\overline{\mathcal{I}v_3}$, it intersects $\mathcal{H}_3$ in at most one point. Hence it can be inferred that point A and point D_2 always lie on the same side (interior or exterior) of $\mathcal{H}_3$ whether $d_3 \geq d$ or $d_3 < d$. Therefore, for each point $p \in \overline{AB}$, vertex v_3 gives a shorter distance than the image $\mathcal{I}$ does.

Summing up these two situations, we prove the theorem. □

In this way, we can clip off the overwhelming majority of worthless windows created by the CH algorithm, and also avoid computation of further propagation for them.

3.3 Maintaining a Priority Queue

Recall that many shortest path algorithms, for example, Dijkstra algorithm [Dijkstra 1959], the MMP algorithm [Mitchell et al. 1987] and Fast Marching Method [Sethian 1999], maintain a priority queue so that the nearest event can be handled first. To apply the paradigm, we need to define a distance measure $\|\cdot\|$ for windows and ensure $\|w\| \leq \|w'\|$ if w is a child of w'. In fact, for a pseudo-source window $w = (d, v)$, we simply define $\|w\| = d$; and for an interval window $w = (d, \mathcal{I}, e, [a, b])$, the following definition is enough (see Figure 5):

$$\|w\| = d + \min_{p\in[a,b]} \|\overline{\mathcal{I}p}\|.$$

We can further prove that if the windows $w_1, w_2, \ldots, w_k$ are obtained one by one in the planar unfolding of a certain geodesic sequence, then:

$$\|w_1\| \leq \|w_2\| \leq \cdots \leq \|w_k\|.$$

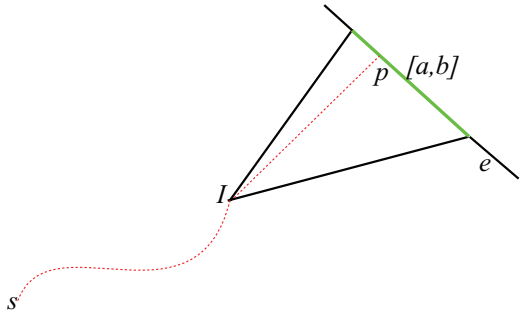


Fig. 5. $\|w\| = d + \min_{p\in[a,b]} \|\overline{\mathcal{I}p}\|$.

The CH algorithm performs window propagation level by level, rather than from near to far. Since the "by distance" manner is more compatible with the shortest path problem, maintaining a priority queue hopefully further optimizes the CH algorithm. Experimental results in Section 4 suggest maintaining a priority queue does further improve the CH algorithm greatly in both time and space. In spite of this, it must be noted that the theoretical asymptotic running time of the improved algorithm becomes $O(n^2 \log n)$ and the space complexity cannot be proved to be still $O(n)$.

3.4 Backtracing a Shortest Path from the Source to Any Destination Point

At the conclusion of the CH algorithm, the information encoded on vertices and angles is enough for backtracing the shortest paths to vertices. Consider a vertex v, whose shortest distance may be given by either an interval window occupying one of v's incident angles or a pseudo-source window occupying one of v's incident vertices, say, v'. If the neighbor vertex v' provides the shortest distance, then the shortest path to v consists of two parts: the shortest path to v' and segment $\overline{v'v}$. Otherwise, it is an interval window w on edge e that provides the shortest distance. According to the source image $\mathcal{I}$ encoded in window w and the position of vertex v, we can easily compute the entering point p on edge e and the entering direction, which enables us to continue backtracing the shortest path, as Figure 6 shows. Each shortest path will end with the source vertex s.

However, with some slight modification the information produced by our algorithm can be used to find the shortest path to a point in the interior of a triangle, using an idea similar to that proposed by Surazhsky et al. [2005]. As is shown in Figure 7(a), for point $p \in f$, they considered all windows on the three edges bounding the face f, and minimize $\|p - p'\| + D(p')$ over all points p' within these windows. After we choose the best window, the backtracing process can continue in the same way. However, Chen and Han [1990] suggested throwing away those one-child interval windows to ensure the $O(n)$ space complexity, making it not informative to compute the shortest path to a point in a face interior if we only implement the first phase of the CH algorithm. So we suggest that the one-child interval windows (useless windows excluded) should be kept on the corresponding edges. Thus for a destination point $p \in f$, we can also backtrace the shortest path in the same way.

Although we get different information at the conclusion of the MMP algorithm and the improved CH algorithm, as Figure 7 shows, both of the algorithms can be used to backtrace shortest paths to surface points. In fact, for the improved CH algorithm, it is possible that some directed edges are only partly covered by interval windows, but the information is still enough to backtrace the shortest path to any surface point. It is at the expense of more space. The correctness

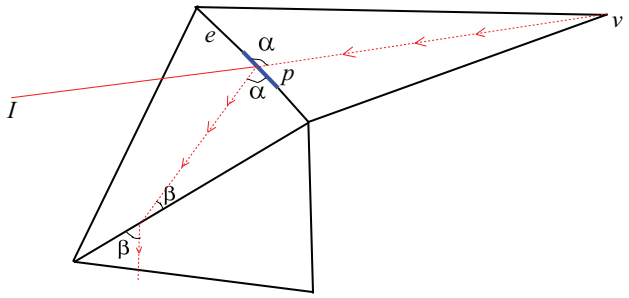


Fig. 6. Backtracing shortest paths to vertices.

is based on two observations: (1) at the conclusion of the original CH algorithm, each directed edge is covered by a list of interval windows if we don't abolish the one-child interval windows; furthermore, these windows encode all shortest edge sequences; and (2) Theorem 3.2 does nothing but filter out those totally useless windows.

As regards the complexity of backtracing a shortest path, whether used with the MMP algorithm or the improved CH algorithm, traversing the windows of a triangle might require $O(n)$ time to find the distance from a destination point p to the source s. This is because a triangle, in either algorithm, might have $O(n)$ windows on its boundary. (Surazhsky et al. [2005] report that most triangles seem to have $O(\sqrt{n})$ windows on average.) A complete implementation of either the CH algorithm or the MMP algorithm would give a data structure that could report the distance from an arbitrary surface point p to the source s in $O(\log n)$ time. But a complete implementation would probably be impractical and unnecessary. (Here we must thank an anonymous referee who gave this accurate and acute observation.)

4. EXPERIMENTAL RESULTS

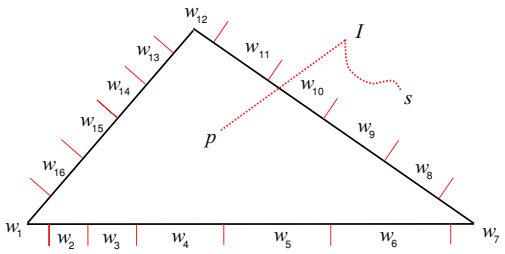
In this section, we provide some experimental results. Our experiments are made on an HP Compaq dc7800 computer with the following configuration:

—Intel(R) Core(TM)2 Duo CPU;
—E8400 @3.00GHz;
—2.99GHz, 2.98GB of RAM;
—Microsoft Windows XP Professional SP3.

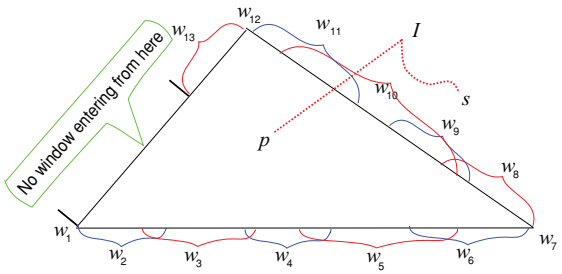
In this research, we compare exact algorithms including

(1) the MMP algorithm [Mitchell et al. 1987] implemented by Danil Kirsanov, one of the authors of Surazhsky et al. [2005]: Note that Kirsanov's code was not used for that paper;
(2) the original CH algorithm [Chen and Han 1990];
(3) the ICH1 algorithm: with the filtering theorem (Theorem 3.2) to improve the CH algorithm;
(4) the ICH2 algorithm: further maintaining a priority queue.[1]

[1]The MMP implementation by Kirsanov is available at the following website:
http://research.microsoft.com/en-us/um/people/hoppe/proj/geodesics/default.htm
Our implementation of the other three algorithms, including CH, ICH1 and ICH2, can be freely obtained at:
http://sites.google.com/site/xinshiqing/knowledge-share
We use some typical models as test objects from http://www.cs.princeton.edu/gfx/proj/sugcon/models/ and http://www-static.cc.gatech.edu/projects/large_models/.

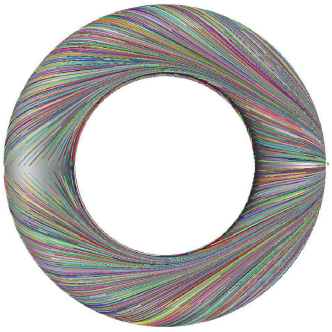


(a) At the conclusion of the MMP algorithm, each directed edge is covered by a list of disjoint interval windows.

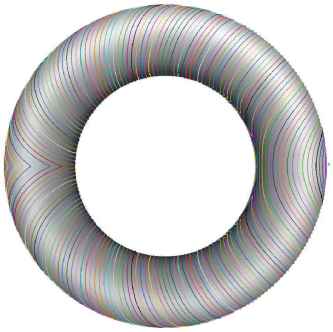


(b) At the conclusion of the improved CH algorithm, it is possible that some edges are only partly covered by interval windows.

Fig. 7. Backtracing shortest paths to a surface point.

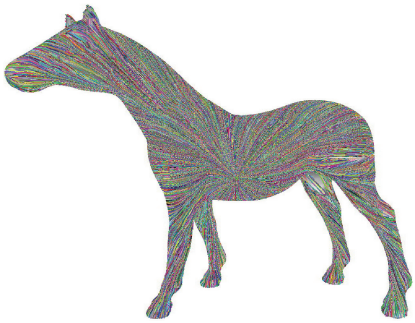


(a) Torus; shortest paths

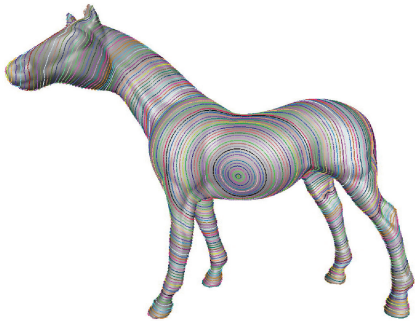


(b) Torus; isolines

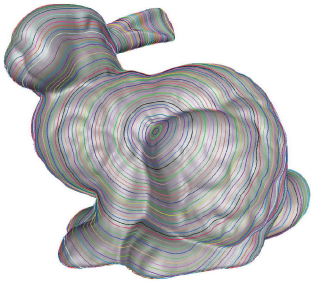
(g) Lucy; isolines



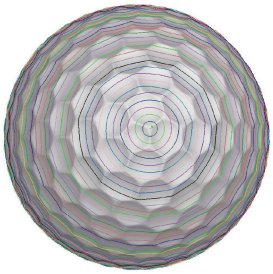
(c) Horse; shortest paths



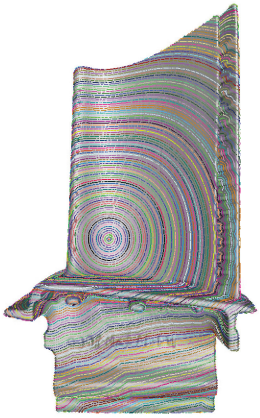
(d) Horse; isolines



(e) Bunny; isolines



(f) Golfball; isolines



(h) Blade; isolines

Fig. 8. Test models.

Table I. Comparison Between Various Exact Shortest Path Algorithms on the Models of Figure 8: ICH1 Denotes the Algorithm that Uses Only Theorem 3.2 to Improve Chen and Han's Algorithm; ICH2 is the Improved CH Algorithm that Implements Both of the Techniques in Section 3; "Space" Indicates the Peak Memory Cost—the Storage of the Model Itself Excluded—Used in Performing These Algorithms; "Windows" is Used to Reduce the Total Number of Windows (Including Trimmed or Filtered Windows), and "Levels" Represents the Maximum Depth of the Resulting Sequence Tree

Model	Faces	Algorithms	Time (s)	Space (M)	Windows	Levels
Torus	9,600	CH	281.23	0.81	401,536,708	9,600
		MMP	0.39	13.18	318,893	166
		ICH1	0.63	0.81	554,591	163
		ICH2	0.27	0.15	246,123	163
Horse	96,964	CH	54,853.38	8.14	33,226,338,112	96,964
		MMP	7.45	186.15	4,837,866	558
		ICH1	32.226	8.14	19,528,130	512
		ICH2	4.70	1.48	4,526,478	516
Bunny	144,046	CH	*- Unfinished in 24 hours -*			144,046
		MMP	*- 2 components; Kirsanov's code fails to support -*			
		ICH1	50.31	12.09	31,407,079	577
		ICH2	6.97	2.20	6,853,459	577
Golfball	245,760	CH	*- Unfinished in 24 hours -*			245,760
		MMP	30.78	715.69	19,028,255	718
		ICH1	193,547	20.63	87,500,689	700
		ICH2	30.56	3.75	26,065,948	706
Lucy	525,814	CH	*- Unfinished in 24 hours -*			525,814
		MMP	28.05	724.04	18,808,447	1318
		ICH1	417.00	44.13	170,308,953	1172
		ICH2	18.39	8.02	17,424,771	1202
Blade	1,765,388	CH	*- Unfinished in 24 hours -*			1,765,388
		MMP	*- Out of memory -*			
		ICH1	4,713.81	148.17	967,402,026	1760
		ICH2	130.75	26.95	101,702,706	1812

Figure 8 illustrates shortest paths or isolines on these models. The detailed experimental results can be seen in Table I, where the performance indicators include:

Time (s): the time spent in window propagation;

Memory (M): the peak memory;

Windows: the total number of windows generated, deleted, or trimmed; or filtered windows included;

Levels: the maximum depth of the sequence tree.

Our experimental results suggest that the improved CH algorithm, in spite of an $O(n^2 \log n)$ asymptotic time complexity, greatly outperforms the original CH algorithm in both time and space. Furthermore, it generally runs faster than the MMP algorithm and uses considerably less space. We think that the main reasons why the improved CH algorithm runs faster than the MMP algorithm are: (1) the improved CH algorithm, with the filtering rule, generates considerably fewer useless windows; generally speaking, the improved CH algorithm and the MMP algorithm generate almost the same amount of windows (both useful and useless windows included); and (2) with either the original CH algorithm or the improved CH algorithm, little computation is required when we perform window propagation, while the MMP algorithm involves a complicated computation (e.g., locating a window and trimming a window) during window propagation. Furthermore, the improved CH algorithm has an absolute advantage over the MMP algorithm since we follow Chen and Han's suggestion of throwing away one-child windows.

We also take notice that there are generally slightly more windows created by the improved CH algorithm than those created by the MMP algorithm. This is because we use only the three distance values at the nearest vertices as a filter to check the validity of a new window. Perhaps better checking rules will be available.

5. CONCLUSIONS

In this article, we discuss the "single source, any vertex" shortest path problem. The main aim of this research is to show how to filter out useless windows using the current estimates of the distances to the vertices, and, to explore another technique, involving maintaining a priority queue. The improved CH algorithm greatly outperforms the original version in both time and space, and even runs faster than the MMP algorithm in far less space. Furthermore, it can also be used to backtrace the shortest path from a source to any surface point. We believe that the improved CH algorithm could be a good choice for many applications in computer graphics.

ACKNOWLEDGMENT

This work would not have been possible without the help of Prof. Yijie Han, who gave us many valuable suggestions in overcoming difficulties. We also wish to thank the anonymous reviewers for their detailed comments and helpful feedback.

REFERENCES

AGARWAL, P. K., ARONOV, B., O'ROURKE, J., AND SCHEVON, C. A. 1997a. Star unfolding of a polytope with applications. *SIAM J. Comput. 26,* 6, 1689–1713.

AGARWAL, P. K., HAR-PELED, S., AND KARIA, M. 2000. Computing approximate shortest paths on convex polytopes. In *Proceedings of the Symposium on Computational Geometry*. 270–279.

AGARWAL, P. K., HAR-PELED, S., SHARIR, M., AND VARADARAJAN, K. R. 1997b. Approximating shortest paths on a convex polytope in three dimensions. *J. Assoc. Comput. Mach. 44*, 567–584.

ALEKSANDROV, L., MAHESHWARI, A., AND SACK, J.-R. 2003. An improved approximation algorithm for computing geometric shortest paths. In *Proceedings of Foundations of Computation Theory (FCT)*. 246–257.

CHEN, J. AND HAN, Y. 1990. Shortest paths on a polyhedron. In *Proceedings of the 6th Annual Symposium on Computational Geometry (SCG)*. ACM Press, New York, 360–369.

DIJKSTRA, E. 1959. A note on two problems in connection with graphs. *Numerische Mathematik 1*, 269–271.

HAR-PELED, S. 1999a. Approximate shortest paths and geodesic diameter on a convex polytope in three dimensions. *Discrete Computat. Geom. 21,* 2, 217–231.

HAR-PELED, S. 1999b. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput. 28,* 4, 1182–1197.

HERSHBERGER, J. AND SURI, S. 1995. Practical methods for approximating shortest paths on a convex polytope in r3. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Philadelphia, 447–456.

KANAI, T. AND SUZUKI, H. 2000. Approximate shortest path on polyhedral surface based on selective refinement of the discrete graph and its applications. In *Proceedings of the Geometric Modeling and Processing (GMP)*. IEEE Computer Society, Washington, DC. 241.

KANEVA, B. AND O'ROURKE, J. 2000. An implementation of Chen & Han's shortest paths algorithm. In *Proceedings of the 12th Canadian Conference on Computational Geometry (CCCG)*.

KAPOOR, S. 1999. Efficient computation of geodesic shortest paths. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, New York, 770–779.

KIMMEL, R. AND SETHIAN, J. 1998. Computing geodesic paths on manifolds. *Proc. Nat. Acad. Sci. 95,* 15, 8431–8435.

MITCHELL, J. S. B. 2000. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, Eds. Elsevier, Chapter 15, 633–701.

MITCHELL, J. S. B., MOUNT, D. M., AND PAPADIMITRIOU, C. H. 1987. The discrete geodesic problem. *SIAM J. Comput. 16,* 4, 647–668.

MITCHELL, J. S. B. AND SHARIR, M. 2004. New results on shortest paths in three dimensions. In *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG)*. ACM Press, New York, 124–133.

MORERA, D. M., VELHO, L., AND CARVALHO, P. C. 2005. Computing geodesics on triangular meshes. *Comput. Graph. J. 29,* 5, 667–675.

MOUNT, D. M. 1984. On finding shortest paths on convex polyhedra. Tech. rep. 1495. Computer Science Department, University of Maryland, College Park, MD.

NOVOTNI, M. AND KLEIN, R. 2002. Computing geodesic distances on triangular meshes. In *Proceedings of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*.

PEYRÉ, G. AND COHEN, L. 2005. Geodesic computations for fast and accurate surface remeshing and parameterization. In *Progress in Nonlinear Differential Equations and Their Applications 63*, 157–171.

PHAM-TRONG, V., SZAFRAN, N., AND BIARD, L. 2001. Pseudo-geodesics on three-dimensional surfaces and pseudo-geodesic meshes. *Numer. Algor. 26,* 4, 1017–1398.

POLTHIER, K. AND SCHMIES, M. 2006. Straightest geodesics on polyhedral surfaces. In *ACM SIGGRAPH Courses*. ACM, New York, 30–38.

SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., AND HOPPE, H. 2003. Multi-chart geometry images. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP)*. Eurographics Association, 146–155.

SETHIAN, J. 1999. Fast marching methods. *SIAM Rev. 41,* 2, 199–235.

SHARIR, M. AND SCHORR, A. 1986. On shortest paths in polyhedral spaces. *SIAM J. Comput. 15,* 1, 193–215.

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph. 24,* 3, 553–560.

VARADARAJAN, K. R. AND AGARWAL, P. K. 1997. Approximating shortest paths on a nonconvex polyhedron. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 182–191.

XIN, S.-Q. AND WANG, G.-J. 2007. Efficiently determining a locally exact shortest path on polyhedral surfaces. *Comput.-Aid. Design 39,* 12, 1113–1119.

ZHOU, K., SYNDER, J., GUO, B., AND SHUM, H.-Y. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP)*. ACM Press, New York, 45–54.

ZIGELMAN, G., KIMMEL, R., AND KIRYATI, N. 2002. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Trans. Vis. Comput. Graph. 8,* 1, 198–207.

Received July 2006, revised December 2007, September 2008, March 2009, accepted March 2009