A Survey of Geodesic Paths on 3D Surfaces[☆]

Prosenjit Bose[a], Anil Maheshwari[a], Chang Shu[b], Stefanie Wuhrer[b]

[a]*School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6*
[b]*National Research Council of Canada, Ottawa, Ontario, Canada K1A 0R6*

Abstract

This survey gives a brief overview of theoretically and practically relevant algorithms to compute geodesic paths and distances on three-dimensional surfaces. The survey focuses on three-dimensional polyhedral surfaces. The goal of this survey is to identify the most relevant open problems, both theoretical and practical.

Key words: Computational Geometry, Shortest Paths, Three Dimensional Shortest Paths, Polyhedral Surfaces.

1. Introduction

Finding shortest paths and shortest distances between points on a surface S in three-dimensional space is a well-studied problem in differential geometry and computational geometry. The shortest path between two points on S is called a *geodesic path* on the surface and the shortest distance between two points on S is called a *geodesic distance*. In this survey, we consider the case where a discrete surface representation of S is given. Namely, S is represented as a polyhedron P in three-dimensional space [16, p. 64]. Since discrete surfaces are not differentiable, methods from differential geometry to compute geodesic paths and distances cannot be applied in this case. However, algorithms from differential geometry can be discretized and extended. Furthermore, the discrete surface can be viewed as a graph in three-dimensional space. Therefore, methods from graph theory and computational geometry have been applied to find geodesic paths and distances on polyhedral surfaces.

[☆]This work was partially supported by NSERC.

The general problem of computing a shortest path among polyhedral obstacles in 3D was shown to be NP-hard by Canny and Reif [13] using a reduction from 3-SAT. Refer to the article by Papadimitriou [51] for an overview on various types of shortest path problems. Computing a geodesic path on a polyhedral surface is an easier problem and it is solvable in polynomial time.

Computing geodesic paths and distances on polyhedral surfaces is applied in various areas such as robotics, geographic information systems (GIS), circuit design, mesh morphing, radiation treatment in bio-medicine, dentistry and computer graphics. For example, geodesic path problems can be applied to finding the most efficient path a robotic arm can trace without hitting obstacles, analyzing water flow, studying traffic control, texture mapping and morphing, and face recognition. A survey related to geodesic paths in two- and higher-dimensional spaces can be found in the Handbook of Computational Geometry [43].

Note that if we know a geodesic path between two points on P then the geodesic distance can easily be computed by measuring the (weighted) length of this geodesic path. Hence, we concentrate on the problem of computing geodesic paths on P.

Problems on finding geodesic paths and distances depending on the number of source and destination points have been studied. The three most commonly studied problems are (a) finding the geodesic path from one source vertex $s \in P$ to one destination vertex $d \in P$, (b) finding the geodesic paths from one source vertex $s \in P$ to all destination vertices in P, or equivalently, finding the geodesic paths from all source vertices in P to one destination vertex $d \in P$, known as *single source shortest path (SSSP) problem*, and (c) finding the geodesic paths between all pairs of vertices in P, known as *all-pairs shortest path (APSP) problem*.

This survey focuses primarily on solving the SSSP problem on polyhedral surfaces. We summarize literature scattered in different fields and bring it together in a common theme. One of the main features of this survey is the identification of important directions for future work.

The algorithms reviewed in this survey are compared by means of the following five categories:

- Accuracy of the computed geodesic path.
- Cost metric used to compute the geodesic path. The cost metric can be the Euclidean distance or a more general weighted distance function that is potentially anisotropic (for example charging more for going uphill than for going downhill).
- Space complexity of the algorithm.
- Time complexity of the algorithm.
- Applicability of the algorithm by surveying if the algorithm has been implemented and tested in practice.

Approximation algorithms are compared according to their *approximation ratio* (or *approximation factor*) k. An algorithm that finds approximations to a geodesic path with approximation ratio k returns a path of length at most k times the exact geodesic path.

To solve the problem of computing geodesic paths on discrete surfaces, two different general approaches can be used. First, the polyhedral surface can be viewed as a graph and algorithms to compute shortest paths on graphs can be extended to find geodesic paths on polyhedral surfaces. Algorithms following this approach are reviewed in Section 2. Second, the polyhedral surface can be viewed as a discretized differentiable surface and algorithms from differential geometry can be extended to find geodesic paths on polyhedral surfaces. Algorithms following this approach are reviewed in Section 3. At the end of each section, open problems related to the section are summarized.

2. Graph-Based Algorithms

This section reviews algorithms to compute geodesic shortest paths that can be viewed as extensions of graph theoretic algorithms. To obtain a good understanding of the reviewed algorithms, we first review some well-known graph theoretic algorithms.

Dijkstra [17] proposed an algorithm to solve the SSSP problem on a directed weighted graph $G(V, E)$ with n vertices, m edges, and positive weights. Dijkstra's algorithm proceeds by building a list of processed vertices for which the shortest path to the source point s is known. The algorithm iteratively decreases estimates on the shortest paths of non-processed vertices, which are stored in a priority queue. In each iteration of the algorithm, the closest unprocessed vertex from the source is extracted from the priority queue and processed by relaxing all its incident edges. The notion of relaxation underlines the analogy between the length of the shortest path and the length of an extended tension spring. When the algorithm starts, the length of the shortest path is overestimated and can be compared to an extended spring. In each iteration, a shorter path is found, which can be compared to relaxing the spring. Although the original implementation used $O(n^2)$ time, the running time was decreased to $O(n \log n + m)$ by using Fibonacci heaps [21]. Thorup [61] presented an $O(m)$-time algorithm for the case where each edge is assigned a positive integer weight. The algorithm runs on a RAM model and assumes that all weights and distances fit in a word each. The main idea is to use a hierarchical bucketing structure to avoid the bottleneck caused by sorting the vertices in increasing order from s.

The length of a path on S depends on the employed cost metric. Hence, the shortest or geodesic path on S depends on this cost metric. In Section 2.1, geodesic path algorithms with Euclidean cost metric are reviewed. In Section 2.2, geodesic path algorithms on weighted surfaces are reviewed. Using a weighted cost metric implies that different faces of S can be weighted differently. Clearly, any algorithm that can solve a shortest path problem using a weighted cost metric can also solve the same problem using the Euclidean cost metric.

2.1. Euclidean Cost Metric

When using the Euclidean distance as a cost metric, shortest paths along a polyhedron P consist of straight line segments that cross faces of the polyhedron. An approach to compute shortest paths on P is to compute a superset of all the possible edge sequences on P and use this information to compute shortest paths. Since all of the algorithms using this strategy establish bounds on the number of possible edges sequences of shortest paths on P [34], they are mainly of theoretical interest. Hence, these algorithms are not discussed in detail in this survey. The algorithms pursuing this approach are less efficient than the ones surveyed. A good overview of the algorithms finding edge sequences is given by Lanthier [34, p. 30–35].

We first review algorithms that only operate on the surface of a convex polyhedron. Second, we review algorithms that operate on the surface of any (convex or non-convex) polyhedron.

2.1.1. Convex Polyhedra

This section discusses algorithms that operate on the surface of a convex polyhedron P with n vertices. Shortest paths according to the Euclidean cost metric are considered.

Sharir and Schorr [57] proposed an algorithm that computes the exact shortest path between points on the surface of P. The proposed algorithm is mainly based on three observations. First, any shortest path intersecting an edge e of P enters and leaves e under the same angle. Second, no shortest path on a convex polygon P can pass through a vertex p of P unless p is the source or destination of the shortest path. Third, if the sequence of edges of P intersected by the shortest path between s and p is known, the shortest path can be computed as the straight line joining s and p after unfolding the faces adjacent to the edge sequence to a plane. The authors aim to subdivide P with respect to a given source point s, such that the shortest path from s to any other point in P can be found efficiently. They define *ridge points* x of P as points that have the property that there exists more than one shortest path from s to x and prove that the ridge points can be represented by $O(n^2)$ straight line segments. (The dash-dotted lines in Figure 1 show some ridge points on a surface.) The algorithm partitions the boundary of P into at most n connected regions called *peels* not containing any vertices or ridge points of P. The boundaries of peels contain only ridge points, vertices, and edges of P. The algorithm to construct the peels is similar to Dijkstra's graph search algorithm. The peels are then iteratively unfolded to the plane. The algorithm first preprocesses P by constructing the peels with respect to s in $O(n^3 \log n)$ time. The algorithm stores the computed peels in a tree called slice tree that can then be used to determine the shortest path between an arbitrary point on P and s in $O(n)$ time. The slice tree data structure uses $O(n^2)$ space.

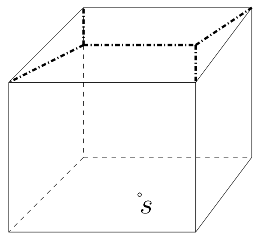

Figure 1: *A set of ridge points on a convex polyhedron.*

Mount [47] improved the algorithm by Sharir and Schorr both in terms of time and space complexity. The main observation by Mount is that the peels defined by Sharir and Schorr can be viewed as Voronoi regions of a point set R containing the planar unfolding of the source point s. Note that R contains at most n points per face of P because there are at most n peels intersecting a face of P. Mount observes that the shortest path from s to any point x on P is at most the shortest path from x to any point $r \in R$ plus the distance between r and s along the planar unfolding of the path. This observation depends on the convexity of P and on the fact that all shortest paths unfold to polygonal chains consisting of straight line segments. Mount uses this observation to prove that the Voronoi regions of R are identical to the peels of P with s as source point. An algorithm following the outline of Dijkstra's algorithm is used to compute the point set R and simultaneously, the Voronoi regions of R. Using this approach, P can be preprocessed with respect to s in $O(n^2 \log n)$ time and $O(n^2)$ space. The space requirement to store the data structure after building it can be reduced to $O(n \log n)$ by storing $O(n)$ different but similar lists of size $O(n)$ each in an efficient way to avoid redundancy. Note that building the data structure still uses $O(n^2)$ space. The query time to compute a shortest path from an arbitrary point $p \in P$ to s is reduced to $O(k + \log n)$, where k is the number of faces of P intersected by the shortest path by using an output sensitive point location data structure. Mount [48] reduced the space requirement to build the data structure storing the Voronoi diagram to $O(n \log n)$ by building a hierarchical structure on the intersections between edges of P and geodesic paths starting from s. The data structure stores for each edge e of P a tree whose leaves contain the intersections between e and geodesic paths crossing e in order. Common sub-trees of different edges are shared to reduce the space complexity.

To avoid the high time complexity of finding geodesic paths, Hershberger and Suri [27] proposed an algorithm that finds an approximate shortest path between two points on the surface of P. The algorithm takes only $O(n)$ time and has an approximation factor of 2. The main idea of the algorithm is to extend the notion of bounding boxes to a general simplified representation of P and to compute the shortest path between two points on this simplified shape. First, all faces of P are labeled positive or negative based on the sign of the dot product between the positive z-axis and the outer normal of the face. The set of edges that forms the boundary between the positive and the negative faces are called horizon edges. The vertical planes through the horizon edges are called horizon planes. There are $O(n)$ horizon planes. To compute a shortest path between s and t, the faces containing the source and destination points are extended into planes, and each horizon plane is added separately to the two planes to obtain a wedge consisting of three planes. Figure 2 shows three possible types of wedges. This yields $O(n)$ wedges. The shortest paths between the two points are computed on each of these wedges in $O(1)$ time as a wedge has constant description size. The shortest path that was found is shown to approximate the shortest path on P within a factor of 2. The algorithm can be extended to approximately solve the SSSP problem in $O(n \log n)$ time. That is, starting from one source point, the algorithm computes approximations with approximation ratio 2 to all other points on P.

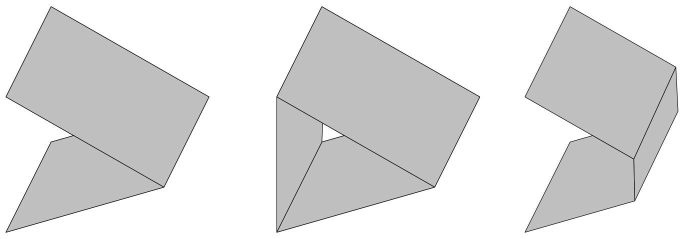

Figure 2: *Three possible types of wedges.*

Har-Peled et al. [26] extended the algorithm by Hershberger and Suri to obtain an approximation ratio of $(1 + \epsilon)$ for $0 < \epsilon < 1$. The algorithm is based on the approximation scheme by Dudley [18] that approximates the minimum number of sets required to approximate every set as ϵ-approximation. The algorithm by Har-Peled et al. proceeds by expanding P by a factor related to ϵ and to the approximation obtained by Hershberger and Suri's algorithm. Denote the expanded polyhedron by P'. The shortest path between two vertices on P is approximated on a grid lattice between the boundaries of P and P'. Since P is convex and since the path is not in the interior of P, the length of the path cannot be shorter than the true shortest path. The path obtained by this method can be projected to P while ensuring that the length of the path does not grow. The running time of the algorithm is $O(n \min(\frac{1}{\epsilon^{1.5}}, \log n) + \frac{1}{\epsilon^{4.5}} \log \frac{1}{\epsilon})$ and hence depends both on n and ϵ. As with the algorithm by Hershberger and Suri, this algorithm can be extended to solve the SSSP problem approximately. The running time of the extended algorithm is $O(\frac{n}{\epsilon^{4.5}}(\log n + \log \frac{1}{\epsilon}))$. Although the theory used by Har-Peled et al. is rather technical, the algorithm itself is simple. Agarwal et al. [2] improved the running time of the algorithm by Har-Peled et al. to $O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^3})$. This improves the running time of the algorithm to solve the SSSP problem approximately to $O(\frac{n}{\epsilon^3} + \frac{n}{\epsilon^{1.5}} \log n)$. Har-Paled [24] presented a further improvement of the running time of this algorithm. After preprocessing the convex polytope in $O(n)$ time, a $(1+\epsilon)$-approximation of the shortest path between two vertices is reported in $O(\frac{\log n}{\epsilon^{1.5}} + \frac{1}{\epsilon^3})$ time. This improves the running time of the algorithm to approximately solve the SSSP problem to $O(n(1 + \frac{\log n}{\epsilon^{1.5}} + \frac{1}{\epsilon^3}))$.

Recently, Schreiber and Sharir [53] proposed an exact solution to the SSSP problem on convex polyhedra in 3-dimensional space. The algorithm extends Dijkstra's algorithm to allow continuous updates. That is, a wavefront is propagated from the source s along the boundary of P and the wavefront is updated at events that change the topology of the wavefront. Note that a similar technique of continuous Dijkstra updates was used by Mount [47]. The general idea of the continuous Dijkstra technique was formally described by Mitchell et al. [45] and is reviewed later in this survey. An implicit representation of the solution is computed in optimal time $O(n \log n)$. The implicit representation is stored using $O(n \log n)$ space. This representation is computed using the so-called *conforming surface subdivision*, which is computed by simulating the growth-process of a cube around each vertex of P until the union of cubes becomes connected and by intersecting this union with the boundary of P. Using this representation, the shortest path from the source to any point on P can be reported in $O(\log n + k)$ time, where k is the number of faces of P crossed by the path.

Agarwal et al. [1] propose an algorithm to compute a $(1 + \epsilon)$-approximation of the shortest path between two vertices that uses $O(\frac{n}{\sqrt{\epsilon}})$ time and $O(\frac{1}{\epsilon^4})$ space. The approach proceeds by constructing a graph, computing the shortest path on this graph, and projecting the computed graph onto the surface of P. Agarwal et al. implemented and tested this algorithm and the algorithm by Hershberger and Suri [27] for artificial data sets with up to almost 100000 faces. This shows that the algorithm is suitable for large-scale datasets.

Schreiber [52] extended the previous approach by Schreiber and Sharir [53] to so-called *realistic polyhedra*. Realistic polyhedra are defined as three classes of non-convex polyhedra. The first class of realistic polyhedra have a boundary that forms a terrain whose maximal facet slope is bounded by a constant. The second class of realistic polyhedra has the property that each axis parallel square with edge length l that has distance at least l from any vertex of P is intersected by at most a constant number of faces of P. The third class of realistic polyhedra has the property that for each edge e of P of length $|e|$, there are at most a constant number of faces within shortest path distance $O(|e|)$.

The following problems related to computing Euclidean shortest paths on the surface of a convex polyhedron remain unsolved:

Open Problem 1. *Can the SSSP problem on convex polyhedra be solved in $O(n \log n)$ time using $O(n)$ space (Schreiber et al. [53])?*

Open Problem 2. *Can an efficient trade off between the query time and the space complexity be established (Schreiber et al. [53])?*

2.1.2. General Polyhedra

This section discusses algorithms that operate on the surface of a simple polyhedron P with combinatorial complexity n. Note that P need not be convex. Shortest paths according to the Euclidean cost metric are considered. The main difficulty that arises when considering non-convex polyhedra is that geodesic paths from s to t on P may pass through a vertex p of P. The difficulty arises because the unfolding is not well defined at a vertex.

O'Rourke et al. [50] extended the algorithms by Sharir and Schorr [57] and Mount [47] to obtain an algorithm that finds the exact geodesic path between two vertices of an arbitrary polyhedron in polynomial time. Both the source and the destination point of P are considered to be vertices of P. The algorithm considers the problem in two steps. First, the straight-line distances between all pairs of vertices of P are found. This is achieved by extending the technique to compute peels in [57]. Second, the shortest distance between the source and the destination vertex is found on the graph induced by the vertices of P. The algorithm takes $O(n^5)$ time to compute a shortest path on P. Since the complexity of the running time is high, the algorithm is not relevant for practical purposes and has not been implemented.

Mitchell et al. [45] formalized the technique called *continuous Dijkstra* previously used by Mount [47] to find shortest paths from a source point s on the surface of a convex polyhedron. Unlike the approach by Mount, this algorithm is suitable to solve the SSSP problem on non-convex polyhedra. The algorithm traverses the graph induced by P similarly to the graph exploration of a graph G in Dijkstra's algorithm [17]. Edges of P behave like nodes in G. Since the distance from s on P to an edge e is not unique, e is labeled by a function describing the distance from s to e. The algorithm keeps track of a subdivision of e into intervals with the property that for two points p and q in the same interval of e, the shortest paths from s to p and from s to q pass through the same sequence of vertices and edges of P. Mitchell et al. observed that these subdivisions of e resemble the peels used by Sharir and Schorr [57]. However, special care needs to be taken when computing this subdivision, since geodesic paths emanating from s can pass through a vertex p of P. In this case, p is treated as a *pseudo-source*. The pseudo-source p is labeled by the geodesic distance from s to p. For any point x of P, the geodesic distance is the minimum of the shortest distance from s to x not passing through a vertex of P and the geodesic distance from the nearest pseudo-source of P to x plus the label of the pseudo-source. This observation allows one to compute the subdivision of s and to store for each interval of the subdivision the distance to the nearest pseudo-source. For a given source vertex s, the algorithm computes a subdivision of

P in $O(n^2 \log n)$ time and $O(n^2)$ space. Once the subdivision has been computed, the distance from s to any other point on P can be computed in $O(\log n)$ time. Reporting the shortest path between s and any other point on P takes $O(\log n + k)$ time, where k is the number of faces of P crossed by the shortest path. If the algorithm is initialized with more than one source point, the subdivision obtained after the continuous Dijkstra algorithm represents the Voronoi diagram of the source points. The continuous Dijkstra technique can be used with different update schemes to obtain new algorithms, as we saw for convex polyhedra [53].

Although the contribution by Mitchell et al. is technical, the algorithm is practical as well. Recently, Surazhkhy et al. [60] implemented and tested the algorithm on data sets obtained using a laser-range scanner. Although the worst-case running time of the algorithm is $O(n^2 \log n)$, Surazhsky et al. found the algorithm's average running time in their experiments to be much lower and suitable for objects with hundreds of thousands of triangles. The exact algorithm by Mitchell et al. is then modified to obtain an algorithm that solves the SSSP problem with approximation ratio $(1+\epsilon)$. Surazhsky et al. derive from their experiments that an average running time of $O(n \log n)$ can be expected in practice for bounded approximations from one source point to all the other points of the mesh. Figure 3 shows Surazhsky et al.'s result for solving the SSSP problem. The algorithm is then compared to the popular fast marching method [32] (see Section 3). This algorithm runs almost as fast as fast marching while having significantly higher accuracy.

Chen and Han [14] developed an algorithm to compute geodesic distances from a source point s on a non-convex polyhedron that does not use the continuous Dijkstra technique. The algorithm constructs a tree called the *sequence tree* that can be viewed as an extension of the dual graph of the tree containing ridge points used by Sharir and Schorr [57] to non-convex polyhedra. In the case of a convex polyhedron, the sequence tree T contains nodes consisting of an edge e of P, the image of s in the local coordinate system of the face incident to e, and the projection of the image onto e. Chen and Han prove that T has a linear number of nodes, contains all of the shortest paths, and can be built in $O(n^2)$ time. For non-convex polyhedra, T contains additional leaves representing pseudo-sources of P (as defined by Mitchell et al. [45]) and the

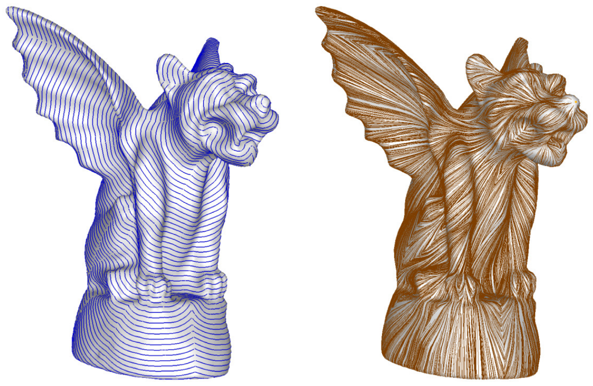
Figure 3: *Result by Surazhkhy et al. [60] to solve the SSSP problem. The left figure shows the isolines and the right figure shows the geodesic paths.*

distances of pseudo-sources from s. This augmentation of T adds at most $O(n)$ nodes. Hence, the algorithm builds a sequence tree in $O(n^2)$ time and $O(n)$ space. After T is computed, the geodesic distance between s and any point in P can be reported in $O(\log n)$ time. The geodesic path can be reported in $O(\log n + k)$ time, where k is the number of faces of P crossed by the path. Kaneva and O'Rourke [29] implemented and tested the algorithm on synthetic data. The implementation confirms the quadratic time complexity and the linear space complexity in practice. The experiments show that roundoff errors are not a serious problem for this algorithm. Kaneva and O'Rourke found the space complexity to be the bottleneck of the algorithm. The reason for this is that Chen and Han's algorithm assumes that unfolding the faces crossed by a path has constant complexity. Data sets with tens of thousands of points were used to test the algorithm. See also Xin and Wang [67] for variations and experimental study on Chen and Han's algorithm.

Har-Peled [25] extended his previous approach to compute $(1+\epsilon)$-approximations of geodesic paths on convex polyhedra [24] to work in case of general polyhedra. Given a source point s, the algorithm computes a subdivision of P based on Voronoi diagrams of faces of P. The subdivision has size $O\left(\frac{n}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$ and is computed in time $O\left(n^2\log n + \frac{n}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{n}{\epsilon}\right)\right)$. In the special case of convex polyhedra, the preprocessing time becomes $O\left(\frac{n}{\epsilon^3}\log\left(\frac{1}{\epsilon}\right) + \frac{n}{\epsilon^{1.5}}\log\left(\frac{1}{\epsilon}\right)\log n\right)$. After this preprocessing step, a $(1+\epsilon)$-approximation of the shortest path between s and any point p on P can be reported in $O\left(\log\left(\frac{n}{\epsilon}\right)\right)$ time. This implies that a $(1+\epsilon)$-approximation to the SSSP problem can be obtained in $O\left(n^2\log n + \frac{n}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{n}{\epsilon}\right) + n\log\left(\frac{n}{\epsilon}\right)\right)$ time.

Varadarajan and Agarwal [63] proposed an algorithm that answers the question of whether it is possible to compute an approximate shortest path between two points, s and t, on a polyhedron in sub-quadratic time. The proposed algorithm only works for polyhedra of genus zero. Two algorithms using the same general technique were proposed. The first algorithm computes an approximation to the shortest path with approximation ratio $7(1+\epsilon)$ in $O(\frac{n^{5/3}}{\epsilon}\log^{2/3} n(\log n + \log\frac{1}{\epsilon}))$ time. The second algorithm takes $O(\frac{n^{8/5}}{\epsilon}\log^{3/5} n(\log n + \log\frac{1}{\epsilon}))$ time, but the approximation ratio increases to $15(1+\epsilon)$. The main idea of the algorithm is to partition the boundary of the simple polyhedron P into regions, consisting of faces of P, using partitioning results of Frederickson [20]. On the boundary of each region, Steiner points are introduced so that they can approximate the actual shortest path within the desired accuracy. The efficient computation of the placement of these Steiner points depends upon efficiently computing lower and upper bounds to the actual shortest path distance. This is computed by finding the smallest cube centered at s, such that there is a path lying strictly in the interior of the cube, that connects s and t. The dimension of the cube establishes a lower bound to the shortest path distance. Once the Steiner points are determined, their algorithm uses similar ideas as is used in several shortest path algorithms in planar graphs (see [20]). Although this is the first paper to break the quadratic time complexity, the contribution is mainly of theoretic interest because the algorithm is involved. Hence, the algorithm has not been implemented.

Kapoor [30] presented an algorithm that claims to solve the problem of computing the exact shortest path between a pair of points on P in sub-quadratic time. The algorithm follows the continuous Dijkstra technique by Mitchell et al. [45] and propagates a wavefront over the surface of P starting from a source point s. The algorithm maintains the wavefront as a collection of circular arcs with centres at s and pseudo-sources of P. Furthermore, the algorithm maintains all of the edges of P that have not yet been reached by the wavefront. The algorithm takes $O(n\log^2 n)$ time and $O(n)$ space. According to O'Rourke [49], the details of the algorithm are *"formidable"*. The algorithm and its analysis are quite complex; some issues with the details and the claims are pointed out by Schreiber and Sharir [53].

Kanai and Suzuki [28] proposed an iterative approximation algorithm to compute the shortest path between pairs of points on P. The algorithm is based on Dijkstra's algorithm and iteratively refines the mesh in regions where the path can pass. The refinement proceeds by placing Steiner points on edges of P and then repeating Dijkstra's algorithm on the augmented graph. The user gives two thresholds related to the accuracy of the approximation. The first threshold defines the number of times the algorithm iterates. The second threshold is related to the number of Steiner points placed on an edge of P. The authors implement the algorithm and compare it to an implementation of Chen and Han's algorithm. They find their algorithm outperforms Chen and Han's algorithm both in terms of time and space complexity. A result of four iterations of their iterative refinement is shown in Figure 4.

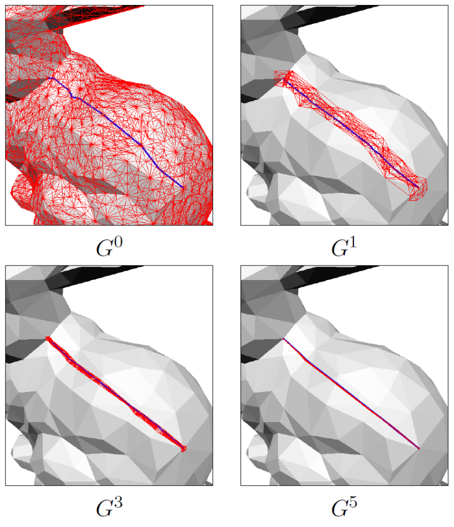

Figure 4: *Result by the iterative refinement approach proposed by Kanai and Suzuki [28].*

The following problems related to computing Euclidean shortest paths on the surface of a possibly non-convex polyhedron remain unsolved:

Open Problem 3. *Can the exact shortest path between a pair of vertices on a nonconvex polyhedron P be computed in $O(n \log n)$ time using $O(n)$ space (Mitchell [43])?*

Open Problem 4. *Can the SSSP problem on a nonconvex polyhedron P be solved in*

$O(n \log n)$ time and $O(n)$ space (Mitchell [43])? (Note that this is a generalization of Open Problem 1.)

2.2. Weighted Cost Metric

This section discusses algorithms that operate on the surface of a possibly non-convex polyhedron P with combinatorial complexity n in 3-dimensional space. Unlike in Section 2.1, the length of the shortest path is not measured by its Euclidean length. Instead, a positive weight w_i is associated with each face f_i of P. The length of a path crossing f_i is its Euclidean length multiplied by w_i. The weights can be used to model the difficulty of the path. For example, it is harder to walk on an uneven terrain than on an asphalt road. A good overview of algorithms related to weighted shortest paths can be found in [6, 34].

Mitchell and Papadimitriou [46] presented an algorithm to compute the shortest path distance between two arbitrary points on a simple polyhedron with n edges. They note that shortest paths obey Snell's Law of refraction at edges of the subdivision. The algorithm is based on this observation and the continuous Dijkstra technique formalized in [45]. Therefore, the authors note that the algorithm can be extended to compute weighted shortest paths on the surface of possibly non-convex polyhedra. The algorithm takes $O(ES)$ time, where E is the number of events in the continuous Dijkstra algorithm and S is the time complexity of performing a numerical search to solve the problem of finding a $(1+\epsilon)$ shortest path passing though a given sequence of k edges. It is shown that $E = O(n^4)$. To perform the numerical search, a form of binary search is developed and used. This binary search takes $O(k^2 \log(nN))$ time, where $k = O(n^2)$, N is the largest integer coordinate of any vertex in the subdivision. Hence, the algorithm finds an approximation of the shortest path with approximation ratio $(1+\epsilon)$ using $O(n^8 \log(nN\frac{W}{w\epsilon}))$ time and $O(n^4)$ space, where $\frac{W}{w}$ is the ratio between the maximum and the minimum weight. Parts of this algorithm have been implemented [44]. Note that the time complexity of $O(n^8 \log(nN\frac{W}{w\epsilon}))$ is a pessimistic estimate; this method may perform well in practice.

Mata and Mitchell [40] proposed a different approach to compute shortest paths between two arbitrary points on a polyhedral surface P. They construct a graph that can be searched to obtain an approximate shortest path on P. The algorithm is claimed to find a path of approximation ratio $(1 + \frac{W}{wk\Theta})$, where $\frac{W}{w}$ is the ratio between the maximum and the minimum weight, Θ is the smallest interior angle of a face of the polyhedron, and k is a user-defined threshold to control the accuracy of the algorithm. This algorithm has been implemented and tested using both real-life data sets with up to 1000 vertices and synthetic data with up to 20000 vertices. The authors demonstrate that their algorithm outperforms several other heuristics. Note that the authors in [5] make a remark that the analysis of the approximation factor as presented in the proof of Proposition 1 of [40] has gaps.

Lanthier et al. [35, 36] presented an approach to construct a weighted graph G that can be searched to obtain an approximate shortest path on P. Without loss of generality, they assume that P is triangulated, i.e., each face is a triangle. In each face f_i of P, a *face graph* G_i is computed as follows. First, place m Steiner points evenly along each edge of f_i, for some positive integer m. Nodes of G_i are all the Steiner points on the boundary of f_i as well as the three vertices of f_i. A node pair u and v is connected in G_i by an edge if either u and v lie on two different edges of f_i or u and v are adjacent to each other on the same edge of f_i. The *weight* of the edge uv is set to $w_i * |uv|$. The graph G is obtained by taking the union of face graphs over all the faces of P. It is shown in Claim 3.1 in [36] that for any segment s' crossing a face f_i, there is an edge in e in the face graph G_i, whose weight is within an additive term of the weighted length of s'. More precisely, $||e|| \leq ||s'|| + w_i \frac{|L|}{m+1}$, where L is the longest edge in P and $||.||$ denotes the weighted length. Given this, it can be shown that a shortest path between two vertices s and t in P, denoted by $\pi(s,t)$, can be approximated by a path in G, where each segment of $\pi(s,t)$ is approximated by an edge in the corresponding face graph. In [46] it has been shown a shortest path may contain $\Theta(n^2)$ segments, where n is the number of triangles in P. By setting $m = \Theta(n^2)$, we obtain that the graph G approximates the shortest path on P between two vertices within an additive factor of $W|L|$, where W denotes the maximum among all w_i's. The following four results are presented in [36] using this approach.

Their first algorithm computes an approximate shortest path between two arbitrary points on P by finding shortest paths in G in $O(n^5)$ time. The computed path is at most

$W|L|$ times longer than the true weighted shortest path on P. Second, a faster and less accurate algorithm is presented to compute the shortest path between two arbitrary points on P by replacing each face graph by its spanner, and then finding the shortest path in the subgraph. The computed shortest path has length at most $\beta(||\pi(s,t)|| + W|L|)$, where $\beta > 1$ is a constant. Third, algorithms are presented to process P for queries asking for shortest paths between a fixed source point s in P and an arbitrary point q in P. The query time is proportional to $\log n$ and the accuracy of the shortest path. Fourth, algorithms are presented to process P for queries asking for shortest paths between two arbitrary points in P. The authors implemented and tested all of the algorithms on both real-life and synthetic data sets. Experiments show that, in practice, only on average a constant number (≤ 6) of Steiner points per edge suffice to obtain high quality results. In the unweighted case, the algorithm seems to converge very rapidly in terms of accuracy, with a significant improvement in run time over the algorithm of Chen and Han [14].

Aleksandrov et al. [4] presented an algorithm to compute an approximation of a weighted shortest path on arbitrary polyhedra with approximation ratio $(1+\epsilon)$. The algorithm is similar to [35, 36] in that $m = O(\log \frac{|L|}{r})$ Steiner points are placed on each edge of P, where r is $min(\frac{\epsilon}{2+3W/w}, \frac{1}{6})$ times the minimum distance of a vertex of P to the boundary of the union of its incident faces. Steiner points are placed in a geometric progression along the edge.

This scheme was further extended in [5]. The approximation ratio remains the same $(1+\epsilon)$ though a fewer number of Steiner points are required. Steiner points are placed in a geometric progression along each edge with the property that if a and b are two consecutive Steiner points on an edge e of P, then for any point x on the boundary of the union of the triangles neighboring e, the angle $\angle axb \leq \frac{\pi}{2}\epsilon$. While executing Dijkstra's SSSP algorithm, properties of Snell's law of refraction are used to prune the search. Assume that Dijsktra's algorithm has finished processing the node u and let v be the next node that is extracted out from the priority queue. Instead of performing the relax operation on all the edges incident to v, it is only performed on all the potential edges which lie in a *geodesic cone* in the direction of uv and satisfy the (discrete version of) Snell's law of refraction. This is closely related to the pruning step that is used by

Mitchell and Papadimitriou [46]. Their algorithm takes $O(\frac{n}{\epsilon}\log\frac{1}{\epsilon}(\frac{1}{\sqrt{\epsilon}}+\log n))$ time, for $0<\epsilon<1$, and $O(n\log n)$ time, for $\epsilon\geq 1$, to compute the shortest path between two arbitrary vertices on P.

Sun and Reif [59] improved the algorithm by Aleksandrov et al. [5] to run in $O(\frac{n}{\epsilon}\left(\log\frac{1}{\epsilon}+\log n\right)\log\frac{1}{\epsilon})$ time. This improvement is achieved by solving the SSSP problem on the graph using a new algorithm called the *Bushwhack algorithm*. It is similar to Dijkstra's algorithm. However, for each Steiner point it maintains a small set of incident edges that are likely to be used to improve the current shortest path. The main observation is that the geodesic cones used in [5] for pruning Dijkstra's algorithm can be replaced by a set of non-overlapping intervals. Sun and Reif implemented and tested their algorithm. They found that when $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ Steiner points are inserted per edge, the hidden constant in the O-notation is significant. Also, they made a clever observation which removed the dependence of the factor W/w that was present in the analysis of the algorithm in [5].

Aleksandrov et al. [6] further improved the running time of the algorithm to $O(\frac{n}{\sqrt{\epsilon}}\log\frac{n}{\epsilon}\log\frac{1}{\epsilon})$ by discretizing P differently. In this algorithm, Steiner points are placed in a geometric progression along the three bisectors of triangles of P. Let l be the bisector between two edges e_1 and e_2 of P, that are incident at vertex v forming an angle α. At most $m=\frac{1.61}{\sin\alpha}\log_2\frac{2|l|}{r(v)}$ Steiner points are placed on l, where $r(v)$ is defined to be the weighted radius of v (see Definition 2.1 in [6]). Furthermore, let x_1 and x_2 be points on e_1 and e_2, respectively. If p is the Steiner point closest to the intersection between segment (x_1x_2) and l, then $|x_1p|+|px_2|\leq(1+\frac{\epsilon}{2})|x_1x_2|$. All the Steiner points and vertices in P form the nodes of the approximation graph G. Let V denote the node set of the graph. It is shown that $|V|=C(P)\frac{n}{\sqrt{\epsilon}}\log_2\frac{2}{\epsilon}$, where $C(P)<4.83\Gamma\log_2 2L$ and L is the maximum of the ratios $|l(v)|/r(v)$, among all vertices $v\in P$, and Γ is the average of the reciprocals of the sines of angles in P. The edges in G connect Steiner points within neighboring faces. It is shown that shortest paths in G can be computed in $O(|V|\log|V|)$ time using the geometric properties associated with these paths. Note that the time complexity of this algorithm depends on the geometry of P, and, in particular, assumes that L is constant.

We note that the face graphs in the Steiner point based algorithms can be computed on the fly - and hence it needs to be computed only for those triangles where the SSSP propagates. Furthermore, the constants hidden in the O-notation in the complexities of these discretization schemes depend upon the geometry of the given surface - especially the aspect ratio of the triangular faces. The constants in [6] are evaluated precisely, and they are less than $5\Gamma \log 2L$, as stated above. If none of the angles in P is less than ten degrees, then Γ is less than 5.

Mehlhorn and Ziegelmann [41], as part of their package on constrained network optimization, have reimplemented and enhanced the algorithms of Lanthier et al. [36]. In this version, two weights are associated to each face and the problem is to compute a minimal path with respect to the first weight while satisfying the limit given by the second weight. This problem is known as the lexicographic weighted region problem and it was posed and solved by Gewali et al. [22].

Lanthier et al. [37] implemented a parallel algorithm to compute approximations of ratio $(1+\epsilon)$ for weighted shortest paths. As in previous approaches, the approach proceeds by constructing a graph and by computing the shortest path between vertices of a graph. The computation of the shortest paths is based on Dijkstra's algorithm and can be broken down into three components: preprocessing to find a graph G, executing Dijkstra's algorithm on G, and backtracking the path. The algorithm uses a spatial indexing structure called *multidimensional fixed partition* that achieves load balancing and reduces the idle time of processors. The algorithm can solve the SSSP and the APSP problems. The algorithm was tested on a network of workstations, on a Beowulf cluster, and on a symmetric multiprocessing architecture. The tests were performed for six geographic data sets with up to one million triangles and achieved acceptable running times.

Aleksandrov et al. [3] preprocess P in $O\left(\frac{n}{\sqrt{\epsilon}} \log \frac{1}{\epsilon} \log \frac{n}{\epsilon}\right)$ time using a data structure of $O\left(\frac{n}{\sqrt{\epsilon}} \log \frac{1}{\epsilon}\right)$ size, by adapting the algorithm in [6] and building certain kind of additively weighted Voronoi diagrams based on geodesic distances. An approximate shortest path query between any point q on P and a given source point s on P can be answered in $O\left(\log \frac{1}{\epsilon}\right)$ time using this data structure. It is assumed that the query consists of the query point as well as the face containing it. These types of queries are called *single source queries* (SSQs). Next, they preprocess P in $O\left(\frac{(g+1)n^2}{\epsilon^{3/2}q}\log\frac{n}{\epsilon}\log^4\frac{1}{\epsilon}\right)$ time and $O\left(\frac{(g+1)n^2}{\epsilon^{3/2}q}\log^4\frac{1}{\epsilon}\right)$ space, such that an approximate shortest path between any pair of points on P can be computed in $O(q)$ time, where g is the genus of P and $q \in (\frac{1}{\sqrt{\epsilon}}\log^2\frac{1}{\epsilon}, \frac{(g+1)^{2/3}n^{1/3}}{\sqrt{\epsilon}})$ is an input parameter. These types of queries are called *all pairs query* (APQs). This result is based on partitioning of the discrete graph G. The partitioning theorem presented in [3] is generalization of the classical planar separator theorem of Lipton and Tarjan [38]. It partitions the graph into several components, such that the weight of each of the component is small, the cost of the boundary of each of the component is small, as well as the graph can be partitioned fairly quickly. (For a precise statement, see Theorem 2 in [3].)

If the weights used for the weighted distances are restricted to be in the range $[1, \rho] \cup \{\infty\}$, $\rho \geq 1$, Cheng et al. [15] present an algorithm to compute an approximation of ratio $(1+\epsilon)$ of the shortest path from s to t that runs in $O(\frac{\rho\log\rho}{\epsilon}n^3\log\frac{\rho n}{\epsilon})$ time. The advantage of this algorithm is that the running time does not depend on the geometry of P. It uses the knowledge that the shortest path between two points s and t lies in the region given by the intersection of P with an ellipse E. The focii of E are s and t, and its diameter is at most $\frac{4}{3}\rho$ times the (unweighted)-geodesic distance between s and t. This sets up a bound on the length of the longest edge within this region. With the knowledge that a weighted shortest path contains at most $O(n^2)$ links, the relative error in the discretization can be bounded, by quantifying the error with respect to each link in the path. This idea is very similar to finding a lower and an upper bound to the shortest path distance as used in Agarwal and Varadarajan [63].

The following problems related to computing weighted shortest paths on the surface of a possibly non-convex polyhedron remain unsolved:

Open Problem 5. *No exact algorithm for computing weighted shortest paths exists to our knowledge.*

Open Problem 6. *How can m Steiner points be placed on each face such that the best approximation accuracy is obtained [34]? Is this problem NP-hard?*

Open Problem 7. *To our knowledge, while many Steiner point schemes have been* *used to compute weighted shortest paths in practice, there is no extensive experimental comparison of the different algorithms to compute weighted shortest paths.*

Note that Open Problem 5 may be related to the problem of computing a shortest path between two points in $\mathbb{R}^3$ that touches a given sequence of lines in a given order [46, 12]. The approach is based on the observation that this shortest path is a geodesic path in a length space of non-positive curvature.

3. Sample-Based Algorithms

This section reviews algorithms for computing shortest paths on discretized smooth surfaces. We focus on the case where the discretization at hand is given as a polyhedron. Unlike the above-mentioned algorithms, the algorithms reviewed in this section generalize algorithms from differential geometry to compute geodesic paths on smooth surfaces to operate on discretized surfaces. The research area concerned with these problems is *discrete differential geometry*. For a more extensive survey of this approach, refer to Kirsanov [33].

Kimmel and Kiryati [31] assume that a discretized surface is given as a voxel representation. That is, space is divided into a cubical grid and each grid point is labeled as located inside the surface, on the surface, or outside of the surface. The approach proposed by Kimmel and Kiryati has two stages. In the first stage, a 3D length estimator is used in a graph search on the graph defined by the surface voxels to find a global approximation of the shortest path. This approximation is then refined using local information. The refinement is done using a discrete version of geodesic curvature shortening flow. This way, an approximation of a shortest path between two grid points can be found. The approximation ratio is not shown to be bounded. However, since the underlying surface is assumed to be smooth, the approximation is the best that can be obtained with the available voxel grid size. The algorithm has been implemented and tested thoroughly.

A popular method to solve the single source shortest path problem is the *fast marching method* [54, 56]. The approach proceeds by solving a discretized version of the Eikonal equation [7, p.2-3] over a regular grid. An algorithm similar to fast marching was developed independently by Tsitsiklis [62] based on a different discretization of the Eikonal equation. The Eikonal equation is a partial differential equation measuring the first arrival time of a wavefront propagated over the grid. Let $u(t)$ be a curve that spreads with velocity $v(t) > 0$, where t denotes time. That is, after time t_i, all the points at distance at most $\int_{t=0}^{t_i} v(t)dt$ from $u(0)$ are contained in $u(t)$. The Eikonal equation is given by $\|\nabla u(t)\| = \frac{1}{v(t)}$. As $v(t) > 0$, the wavefront propagates in a directed way and the direction of propagation is called the *upwind direction*. If we set $v(t) \equiv 1$, then solving the Eikonal equation is equivalent to computing Euclidean geodesics. When a regular grid domain is considered [54], solving the Eikonal equation results in the arrival time $T(x, y)$ for each grid point (x, y). The algorithm takes $O(N \log N)$ time, where N denotes the number of grid cells.

The fast marching method was extended to triangulated surfaces [32], unstructured meshes [55], implicit surfaces [42], parametric surfaces [58], and regularly sampled parametric surfaces [64]. As this survey focuses on shortest paths on polyhedral surfaces, we only review methods operating on triangular surfaces.

Kimmel and Sethian [32] presented an approach called the fast marching method on triangular domains (FMM) that extends the fast marching method to solve the SSSP problem on a triangular grid with n vertices. In this case, the arrival time $T(x, y, z)$ is computed for each vertex (x, y, z) of the mesh. The algorithm is similar to Dijkstra's algorithm [17] in that the vertices are processed iteratively when their estimated distance to the source is smallest. To process a vertex p, we need to update the distance from p to the source s based on the processed neighbors of p. If p is the only unprocessed vertex of a triangle t, then we can update the distance from p to the s through t by solving a quadratic equation. However, if p is a vertex of an obtuse triangle t, it is possible that we need to update the distance from p to s although only one vertex of t is processed. In this case, Kimmel and Sethian compute numerical support for t by locally unfolding triangles opposite the obtuse angle to the plane. The obtuse angle is split and the split is propagated through the neighboring unfolded triangles until a processed vertex is found. This processed vertex is then used to compute the update of p through the obtuse triangle. The intersection of a shortest path and a face of P is always a line segment. The algorithm results in shortest paths that cut through faces of the triangulation and yields consistent results. However, the shortest paths found using the FMM method only approximate the true geodesic distances on the triangular mesh due to numerical inaccuracies introduced by the numerical support of obtuse triangles. The accuracy of the approach depends on the quality of the underlying triangulation; namely on the longest edge e_{max} and the widest angle Φ_{max} in the triangular mesh. The error is in the order of $O(e_{max}/(\pi - \Phi_{max}))$, which can be unbounded for near-degenerate triangulations. The algorithm takes $O(e^2_{max}/(\Phi_{min} h^2_{min}(\pi - \Phi_{max})^3) n \log n)$ time and $O(n)$ space, where Φ_{min} is the smallest acute angle of the triangulation and h_{min} is the triangle altitude with minimum length. Since the algorithm is easy to implement and performs well in practice, several implementations of the algorithm exist. Figure 5 shows an example of the output of the fast marching algorithm. As mentioned in the previous section, FMM has been compared to the implementation by Surazhsky et al. [60] of the algorithm by Mitchell et al. [45]. Although Surazhsky's implementation is almost as fast as FMM while producing results of higher accuracy, FMM is still the most popular algorithm in practice because is it easy to implement. For instance, FMM has been used extensively for mesh processing tasks [19, 10, 65].

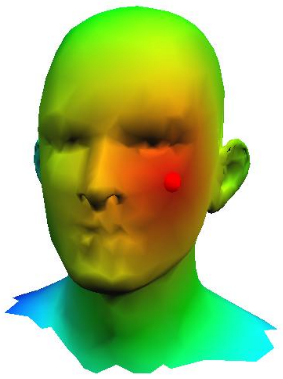

Figure 5: *Approach by Kimmel and Sethian [32]. We visualize the SSSP distances found by the fast marching algorithm. The red point is the source. Isolines have identical color.*

Various results exist to improve the accuracy of the fast marching method. Kirsanov [33] introduced a novel update rule for FMM during the march. This update rule yields a higher numerical accuracy of the resulting shortest paths. Martinez et al. [39] presented a way to iteratively improve an existing estimate of a geodesic path between two vertices of a triangulated surface. Starting from a path computed via FMM, the path can be refined to yield a better approximation. Similar to Kimmel and Kiryati [31], a discrete geodesic curvature flow is used to iteratively improve the approximation. Martinez et al. showed that the iterative scheme converges to a local minimum. While it is proven that the approximation is improved until convergence, no quantification of the improvement or the convergence rate is given. Xin and Wang [66] presented another iterative method to improve the path found by the fast marching method. The algorithm first improves the initial fast marching method by classifying the edges of P into different types and by treating different edge types differently during the wave front propagation. Second, the algorithm iteratively improves the resulting shortest path until the exact locally shortest path is found. Clearly, the iterative methods are less efficient in terms of time than FMM.

Bronstein et al. [11] presented a scheme to compute the geodesic distance between two arbitrary points s and t (not necessarily vertices) of P. The scheme finds the triangles of P containing s and t and interpolates the geodesic distances between the triangle vertices.

Bertelli et al. [9] consider solving the APSP problem using FMM. Their goal is to take advantage of the redundant computation in different passes of the SSSP algorithm to obtain a more efficient approach than simply running FMM n times with each vertex as source point. Although the algorithm is shown to achieve higher efficiency in experiments, the worst case running time of the algorithm remains $O(n^2 \log n)$. Giard and Macq [23] propose an alternative way to solve the APSP problem on polyhedral surfaces. The approach unfolds the mesh to the plane in a preprocessing step. The unfolding is performed for each of a number of chosen reference vertices. The unfolding stage proceeds by rotating all of the edges traversed by Dijkstra's algorithm with a given reference point as source. The edges are rotated into the tangent plane of the reference point in order. Note that this unfolding is different than the one used by Chen and Han [14], where triangles are rotated into the same plane around their common edge. In the query phase, the new source is located in the unfolding of its closest reference point. The Euclidean distances in this unfolding are used to approximate the geodesic map of the new source. The algorithm was implemented and is shown to perform well in practice.

Ben Azouz et al. [8] consider the problem of computing a geodesic distance between two vertices s and t of a mesh P that may contain holes. The approach computes the canonical form [19] of a set of sample points of the mesh in a preprocessing phase. The canonical form is computed by embedding the intrinsic geometry of P into a Euclidean space. To compute the canonical form, multi-dimensional scaling is used with the geodesic distances computed via FMM as dissimilarities. The dissimilarities are weighted according to the proportion of the FMM path that does not pass by a hole of P. To compute the geodesic distance between s and t, the approach adds s and t to the canonical form and reports the Euclidean distance as an estimate. A worst case optimal upper bound of the estimate is obtained. The algorithm was implemented and is shown to perform well in practice.

The following problems related to sample-based geodesic computations remain unsolved:

Open Problem 8. *Graph-based algorithms find globally optimal paths that may not be locally optimal if the graph is based on samples obtained from a smooth surface. Algorithms from differential geometry can be discretized to find locally shortest paths. However, these algorithms can often get trapped in local insignificant minima. Can graph-based algorithms be combined with algorithms from discrete differential geometry to yield efficient globally convergent algorithms to compute a bounded approximation of the geodesic distance on a sample set obtained from a smooth surface [33]?*

Open Problem 9. *Can FMM be generalized to solve the APSP problem in $o(n^2 \log n)$ time (recall that a solution in $O(n^2 \log n)$ was presented by Bertelli et al. [9])?*

4. Summary

To summarize this survey, Table 1 gives the reviewed results on shortest path problems on polyhedral surfaces. The table differentiates the algorithms based on theoretical time complexity and approximation ratio. Since the practical improvements of FMM do not improve the time complexity or approximation ratio, they are not listed in the table.

Polyhedral Surface	Cost Metric	Approx. Ratio	Time Complexity	Geom. Dep.	Ref.
Graph-based					
Convex	Euclidean	1	$O(n^3 \log n)$	No	[57]
Convex	Euclidean	1	$O(n^2 \log n)$	No	[47]
Convex	Euclidean	2	$O(n)$	No	[27]
Convex	Euclidean	$1+\epsilon$	$O(n \min(\frac{1}{\epsilon^{1.5}}, \log n) + \frac{1}{\epsilon^{4.5}} \log \frac{1}{\epsilon})$	No	[26]
Convex	Euclidean	$1+\epsilon$	$O(n \log \frac{1}{\epsilon} + \frac{1}{\epsilon^3})$	No	[2]
Convex	Euclidean	1	$O(n \log n)$	No	[53]
Graph-based					
Non-convex	Euclidean	1	$O(n^5)$	No	[50]
Non-convex	Euclidean	1	$O(n^2 \log n)$	No	[45]
Non-convex	Euclidean	1	$O(n^2)$	No	[14]
Non-convex	Euclidean	$1+\epsilon$	$O\left(n^2 \log n + \frac{n}{\epsilon} \log\left(\frac{1}{\epsilon}\right) \log\left(\frac{n}{\epsilon}\right)\right)$	No	[25]
Non-convex	Euclidean	$7(1+\epsilon)$	$O(\frac{n^{\frac{5}{3}}}{\epsilon} \log^{\frac{2}{3}} n(\log n + \log \frac{1}{\epsilon}))$	No	[63]
Non-convex	Euclidean	$15(1+\epsilon)$	$O(\frac{n^{\frac{8}{5}}}{\epsilon} \log^{\frac{3}{5}} n(\log n + \log \frac{1}{\epsilon}))$	No	[63]
Non-convex	Euclidean	1	$O(n \log^2 n)$	No	[30]*
Graph-based					
Non-convex	Weighted	$1+\epsilon$	$O(n^8 \log(nN \frac{W}{w\epsilon}))$	Yes: TC	[46]
Non-convex	Weighted	Additive	$O(n^5)$	Yes: TC	[36]
Non-convex	Weighted	$1+\epsilon$	$O(nmr \log r + \frac{(nm)^2}{r} \log \frac{nm}{\sqrt{r}} + \frac{(nm)^2}{\sqrt{r}})$	Yes: TC	[4]
Non-convex	Weighted	$1+\epsilon$	$O(\frac{n}{\epsilon} \log \frac{1}{\epsilon}(\frac{1}{\sqrt{\epsilon}} + \log n))$	Yes: TC	[5]
Non-convex	Weighted	$1+\epsilon$	$O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\epsilon} \log \frac{1}{\epsilon})$	Yes: TC	[6]
Non-convex	Weighted	$1+\epsilon$	$O(\frac{n}{\epsilon} \left(\log \frac{1}{\epsilon} + \log n\right) \log \frac{1}{\epsilon})$	Yes: TC	[59]
Non-convex	Weighted	$1+\epsilon$	$O(\frac{\rho^2 \log \rho}{\epsilon^2} n^3 \log \frac{\rho n}{\epsilon})$	No	[15]
Non-convex	Weighted	$1+\epsilon$	$O\left(\frac{n}{\sqrt{\epsilon}} \log \frac{1}{\epsilon} \log \frac{n}{\epsilon}\right)$	Yes: TC	[3]
Sample-based					
Non-convex	Euclidean	Unbounded	$O\left(\frac{e_{max}^2}{\Phi_{min} h_{min}^2 (\pi - \Phi_{max})^3} n \log n\right)$	Yes: A TC	[32]

Table 1: Results on Shortest Paths on a Polyhedral Surface P with n vertices. The constant $\epsilon > 0$ is the desired accuracy of the shortest path. In the weighted case, N is the largest integer coordinate of any vertex in the subdivision and $\frac{W}{w}$ is the ratio between the maximum and the minimum weight. The symbol m denotes the number of Steiner points placed along one edge. The symbol r denotes $min(\frac{\epsilon}{2+3W/w}, \frac{1}{6})$ times the minimum distance of a vertex of P to the boundary of the union of its incident faces. The constant $\rho > 1$ is the largest weight assigned to a face of P. Furthermore, e_{max} is the longest edge, Φ_{min} and Φ_{max} are the smallest and largest angle, respectively, and h_{min} is the triangle altitude with minimum length. The geometry dependence column indicates whether the time complexity (TC) or the accuracy (A) of the approach depend on the geometry of the surface.
* Time bound is claimed but contains gaps [53].

5. Acknowledgements

Authors acknowledge valuable inputs from Lyudmil Aleksandrov, Joseph Mitchell, Joseph O'Rourke, Jörg-Rüdiger Sack, and the anonymous referee. Authors thank Vitaly Surazhsky for permission to use Figure 3 and Takashi Kanai for permission to use Figure 4.

References

[1] Pankaj Agarwal, Sariel Har-Peled, and Meetesh Karia. Computing approximate shortest paths on convex polytopes. *Algorithmica*, 33(2):227–242, 2002.

[2] Pankaj Agarwal, Sariel Har-Peled, Micha Sharir, and Kasturi Varadarajan. Approximating shortest paths on a convex polytope in three dimensions. *Journal of the ACM*, 44(4):567–584, 1997.

[3] Lyudmil Aleksandrov, Hristo Djidjev, Hua Guo, Anil Maheshwari, Doron Nussbaum, and Jörg-Rüdiger Sack. Approximate shortest path queries on weighted polyhedral surfaces. *Mathematical Foundations of Computer Science (to appear in Discrete and Computational Geometry)*, 4162:98–109, 2006.

[4] Lyudmil Aleksandrov, Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. An ϵ-approximation for weighted shortest paths on polyhedral surfaces. In *Scandinavian Workshop on Algorithm Theory*, pages 11–22, 1998.

[5] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximation algorithms for geometric shortest path problems. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 286–295, 2000.

[6] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM*, 52(1):25–53, 2005.

[7] Vladimir Arnold. *Lectures on Partial Differential Equations, Second Edition*. Springer, 2004.

[8] Zouhour Ben Azouz, Prosenjit Bose, Chang Shu, and Stefanie Wuhrer. Approximations of geodesic distances for incomplete triangular manifolds. In *Canadian Conference on Computational Geometry*, pages 177–180, 2007.

[9] Luca Bertelli, Baris Sumengen, and B. S. Manjunath. Redundancy in all pairs fast marching method. In *IEEE International Conference on Image Processing 2006 (ICIP)*, 2006.

[10] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Three-dimensional face recognition. *International Journal of Computer Vision*, 64(1):5–30, 2005.

[11] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM Journal on Scientific Computing*, 28(5):1812–1836, 2006.

[12] Dima Burago, Dima Grigoriev, and Anatol Slissenko. Approximating shortest path for the skew lines problem in time doubly logarithmic in 1/epsilon. *Theoretical Computer Science*, 315(2-3):371–404, 2004.

[13] John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *IEEE Conference on Foundations of Computer Science*, pages 39–48, 1987.

[14] Jindong Chen and Yijie Han. Shortest paths on a polyhedron; Part I: computing shortest paths. *International Journal of Computational Geometry & Applications*, 6(2):127–144, 1996.

[15] Siu-Wing Cheng, Hyeon-Suk Na, Antoine Vigneron, and Yajun Wang. Approximate shortest paths in anisotropic regions. *SIAM Journal on Computing*, 38(3):802–824, 2008.

[16] Mark de Berg, Otfried Cheong, Marc van Krefeld, and Mark Overmars. *Computational Geometry Algorithms and Applications, Third Edition*. Springer, 2008.

[17] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[18] Richard M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10:227–236, 1974. Erratum in Journal of Approximation Theory 26:192-193, 1979.

[19] Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[20] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[21] Michael Fredman and Robert Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[22] Laxmi Gewali, Alex Meng, Joesph Mitchell, and Simeon Ntafos. Path planning in $0/1/\infty$ weighted regions with applications. In *Proceedings of the fourth annual symposium on Computational geometry*, pages 266–278, 1988.

[23] Joachim Giard and Benoit Macq. From mesh parameterization to geodesic distance estimation. In *European Workshop on Computational Geometry*, 2009.

[24] Sariel Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *Discrete and Computational Geometry*, 21:216–231, 1999.

[25] Sariel Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM Journal on Computing*, 28(4):1182–1197, 1999.

[26] Sariel Har-Peled, Micha Sharir, and Kasturi R. Varadarajan. Approximating shortest paths on a convex polytope in three dimensions. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry*, pages 329–338, 1996.

[27] John Hershberger and Subhash Suri. Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$. *Computational Geometry Theory and Applications*, 10(1):31–46, 1998.

[28] Takashi Kanai and Hiromasa Suzuki. Approximate shortest path on a polyhedral surface and its applications. *Computer Aided Design*, 33(11):801–811, 2001.

[29] Biliana Kaneva and Joseph O'Rourke. An implementation of Chen & Han's shortest paths algorithm. In *Proceedings of the 12th Canadian Conference on Computational Geometry*, pages 139–146, 2000.

[30] Sanjiv Kapoor. Efficient computation of geodesic shortest paths. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 770–779, 1999.

[31] Ron Kimmel and Nahum Kiryati. Finding shortest paths on surfaces by fast global approximation and precise local refinement. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(6):643–656, 1996.

[32] Ron Kimmel and James Sethian. Computing geodesic paths on manifolds. *National Academy of Sciences of the USA*, 95:8431–8435, 1998.

[33] Danil Kirsanov. *Minimal Discrete Curves And Surfaces*. PhD thesis, Harvard University, Applied Mathematics, 2001.

[34] Mark Lanthier. *Shortest path problems on polyhedral surfaces*. PhD thesis, Carleton University, School of Computer Science, 1999.

[35] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 274–283, 1997.

[36] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximating weighted shortest paths on polyhedral surfaces. *Algorithmica*, 30(4):527–562, 2001.

[37] Mark Lanthier, Doron Nussbaum, and Jörg-Rüdiger Sack. Parallel implementation of geometric shortest path algorithms. *Parallel Computing*, 29:1445–1479, 2003.

[38] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[39] Dimas Martinez, Luiz Velho, and Paulo Cezar Carvalho. Computing geodesics on triangular meshes. *Computers and Graphics*, 29:667–675, 2005.

[40] Christian Mata and Joseph Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions (extended abstract). In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 264–273, 1997.

[41] Kurt Mehlhorn and Mark Ziegelmann. Cnop - a package for constrained network optimization. In Adam L. Buchsbaum and Jack Snoeyink, editors, *ALENEX*, volume 2153 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2001.

[42] Facundo Memoli and Guillermo Sapiro. Distance functions and geodesics on point clouds. Technical Report 1902, University of Minnesota, Institute of Mathematics and its Applications, 2002.

[43] Joseph Mitchell. *Geometric shortest paths and network optimizations.* In: Jörg-Rüdiger Sack and Jorge Urrutia (Editors). The handbook of computational geometry. Chapter 15, pages 633–701. Elsevier Science, 2000.

[44] Joseph Mitchell. Personal Communication, 2010.

[45] Joseph Mitchell, David Mount, and Christoph Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16:647–668, 1987.

[46] Joseph Mitchell and Christos Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, 1991.

[47] David Mount. On finding shortest paths in convex polyhedra. Technical Report 1495, University of Maryland, Baltimore, Department of Computer Science, 1985.

[48] David Mount. Storing the subdivision of a polyhedral surface. *Discrete & Computational Geometry*, 2:153–174, 1987.

[49] Joseph O'Rourke. Computational geometry column 35. *SIGACT News*, 30(2):31–32, 1999.

[50] Joseph O'Rourke, Subhash Suri, and Heather Booth. Shortest paths on polyhedral surfaces. In *Symposium on Theoretical Aspects in Computer Science*, pages 243–254, 1985.

[51] Christos Papadimitriou. Shortest-path motion. In *Proceedings of the sixth conference on Foundations of software technology and theoretical computer science*, pages 144–153, 1986.

[52] Yevgeny Schreiber. Shortest paths on realistic polyhedra. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 74–83, 2007.

[53] Yevgeny Schreiber and Micha Sharir. An optimal-time algorithm for shortest paths on a convex polytope in three dimensions. *Discrete and Computational Geometry*, 39:500–579, 2008.

[54] James Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[55] James Sethian and Alexander Vladimirsky. Fast methods for the eikonal and related hamilton-jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703, 2000.

[56] James A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[57] Micha Sharir and Amir Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15:193–215, 1986.

[58] Alon Spira and Ron Kimmel. An efficient solution to the eikonal equation on parametric manifolds. *Interfaces and Free Boundaries*, 6(4):315–327, 2004.

[59] Zheng Sun and John H. Reif. On finding approximate optimal paths in weighted regions. *Journal of Algorithms*, 58(1):1–32, 2006.

[60] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics*, 24(3):553–560, 2005.

[61] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, 1999.

[62] John Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[63] Kasturi Varadarajan and Pankaj Agarwal. Approximating shortest paths on an nonconvex polyhedron. *SIAM Journal on Computing*, 30(4):1321–1340, 2000.

[64] Ofir Weber, Yohai Devir, Alexander Bronstein, Michael Bronstein, and Ron Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics*, 27(4):1–16, 2008.

[65] Stefanie Wuhrer, Chang Shu, Zouhour Ben Azouz, and Prosenjit Bose. Posture invariant correspondence of incomplete triangular manifolds. *International Journal of Shape Modeling*, 13(2):139–157, 2007.

[66] Shi-Qing Xin and Guo-Jin Wang. Efficiently determining a locally exact shortest path on polyhedral surfaces. *Computer Aided Design*, 39(12):1081–1090, 2007.

[67] Shi-Qing Xin and Guo-Jin Wang. Improving Chen and Han's algorithm on the discrete geodesic problem. *ACM Trans. Graph.*, 28(4), 2009.