Approximate Point-to-Face Shortest Paths in \( \mathcal{R}^3 \)

Yam Ki Cheung and Ovidiu Daescu

Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080, USA
{ykcheung,daescu}@utdallas.edu

Abstract. We address the point-to-face approximate shortest path problem in \( \mathcal{R}^3 \): Given a set of polyhedral obstacles with a total of \( n \) vertices, a source point \( s \), an obstacle face \( f \), and a real positive parameter \( \epsilon \), compute a path from \( s \) to \( f \) that avoids the interior of the obstacles and has length at most \((1 + \epsilon)\) times the length of the shortest obstacle avoiding path from \( s \) to \( f \). We present three approximation algorithms that take \( O\left(\frac{n^4(L + \log(n/\epsilon))^2}{\epsilon^2} + \frac{n^2(L + \log(n/\epsilon))/\epsilon^2}{\epsilon^2} + \frac{n^2\log(n/p)\log(n\log \rho)}{\epsilon^2}\right) \) time, \( O\left(\frac{T_{p-p}(n)\lambda(n)}{\epsilon^2}\right) \) time, and \( O\left(\frac{n^2\lambda(n)\log(n/p)}{\epsilon^2} + \frac{n^2\log(n/p)\log(n\log \rho)}{\epsilon^2}\right) \) time, respectively, where \( L \) is the precision of the integers used, \( T_{p-p}(n) \) is the time complexity of the point-to-point shortest path algorithm used, \( p \) is the ratio of the length of the longest obstacle edge to the Euclidean distance between \( s \) and \( f \), and \( \lambda(n) \) is a very slowly-growing function related to the inverse of the Ackermann’s function.

1 Introduction

The Euclidean shortest path problem among obstacles in the plane or space is one of the oldest and well-known problems in computational geometry. It has been intensively studied, see [1, 5–13, 15–18, 20, 22], as well as the survey by Mitchell [14]. In general the problem is stated as: Given a set of obstacles in the d-dimensional space \( \mathcal{R}^d \) with a total of \( n \) vertices, find a shortest (Euclidean) path between a source point \( s \) and a target point \( t \) while avoiding the interior of the obstacles. There are two commonly studied versions of this problem. One is the single pair version, that asks to find a shortest path between two given query points. The other one is the single source version, which first constructs a shortest path map with respect to a source point \( s \). After that, for any given query point \( t \), a shortest path between \( s \) and \( t \) can be found based on the shortest path map.

In \( \mathcal{R}^2 \), shortest paths are polygonal and turn only at the vertices of the polygonal obstacles. Sharir and Schorr [22] have developed an \( O(n^2 \log n) \) time algorithm based on discrete graph searching and the visibility graph of the obstacles, where \( n \) is the number of the obstacle vertices. Various studies, e.g. [7, 9, 17].

* This research was supported in part by NSF award CCF-0635013.
improved the time to quadratic in worst case. Kapoor, Maheshwari, and Mitchell \cite{Kapoor10} gave an interesting $O(n + h^2 \log n)$ time and $O(n)$ space algorithm based on the visibility graph approach, where $h$ is the number of holes (obstacles) of the given input. This algorithm is the only algorithm known to be linear in $n$ in both time and space. However, the time dependence on $h$ is quadratic, so the algorithm does not perform well if $h$ is not relatively small compared to $n$. Avoiding the visibility graph approach, Mitchell \cite{Mitchell92} developed a version of the continuous Dijkstra method and obtained the first subquadratic, $O(n^{3/2+\epsilon})$ time algorithm. Subsequently, based on the same technique, this result was improved by Hershberger and Suri \cite{Hershberger95} to $O(n \log n)$ time and $O(n \log n)$ space.

In $\mathbb{R}^3$, shortest paths among polyhedral obstacles are polygonal and turn only on obstacle edges or vertices. However, unlike the case in $\mathbb{R}^2$, shortest paths need not lie on any discrete graph. Sharir and Schorr \cite{Sharir89} have shown that shortest paths in $\mathbb{R}^3$ are geodesic, i.e. paths must enter and leave an edge at the same angle. Given a distinct sequence of edges, the local optimal path between two points can be unfolded at each edge to form a straight line, and the local optimal path can be uniquely identified. Nevertheless, the problem is still significantly harder than in $\mathbb{R}^2$. For algebraic considerations, Bajaj \cite{Bajaj88} has shown that the algebraic complexity is exponential, since comparing the lengths of two paths may require exponentially many bits. Considering the combinatorial aspect of the problem, Canny and Reif \cite{Canny92} have shown that the shortest path problem in $\mathbb{R}^3$ is NP-hard.

An interesting special case of the shortest path problem is that in which the path is restricted to the surface of a single polytope. The first significant study of this special case in computational geometry is by Sharir and Schorr \cite{Sharir89}. They gave an $O(n^3 \log n)$ time algorithm for convex polytopes by exploiting the special structure of geodesic paths along the surface of a convex polytope. Mount \cite{Mount92} gave an improved algorithm for convex polytopes with running time $O(n^2 \log n)$. For general nonconvex polytopes, Mitchell, Mount and Papadimitriou \cite{Mitchell95} presented an $O(n^2 \log n)$ algorithm extending the technique of Mount \cite{Mount92}. See also \cite{Bajaj88,Mitchell95,Sharir89,Sharir92} for more studies for this topic.

Our real interest is in the general case in $\mathbb{R}^3$. Several papers \cite{Bajaj88,Mitchell95,Sharir89,Sharir92} have presented polynomial time approximation algorithms for computing an $\epsilon$-approximate path, which has length at most $(1 + \epsilon)$ times the length of the shortest path between two query points, where $\epsilon$ is a real parameter defining the quality of the approximation. We will discuss those relevant to this paper in Section \ref{sec:approximation}.

Throughout the paper we use the following notations.

$S$ denotes a set of polyhedral obstacles;
$E$ denotes the set of edges of $S$;
$P_{st}$ denotes a shortest path from $s$ to $t$ that avoids the interior of the obstacles;
$Q_{st}$ denotes an approximate path of $P_{st}$ (see next section);
$|P|$ denotes the length of a given path $P$;
$st$ denotes the line segment between points $s$ and $t$;
$|st|$ denotes the Euclidean distance between $s$ and $t$. 
1.1 Our Results

In this paper, we address the point-to-face shortest path problem in $\mathbb{R}^3$: Given a set of polyhedral obstacles with a total of $n$ vertices, a source point $s$, and a destination face $f$ of some obstacle in the input, compute an $\epsilon$-approximate path from $s$ to $f$ that avoids the interior of the obstacles. We assume the obstacle surfaces are triangulated, so $f$ is a triangle in $\mathbb{R}^3$.

The problem has multiple applications, including robot navigation, path planning and structural proteomics. For example, in structural proteomics, after a protein surface has been segmented and pockets identified, various descriptors can be associated with key points of the pocket. Given a key point $s$, one can measure the pocket depth of $s$ by computing various distance measures from $s$ to the “caps” of the pocket, the most natural of which is the shortest collision free distance from $s$ to the caps (a cap is a face of the convex hull of the protein).

We present three approximation algorithms for the point-to-face shortest path problem that take $O\left(n^4 (L + \log(n/\epsilon))^2 / \epsilon^2 + \sqrt{n^2 (L + \log(n/\epsilon))/\epsilon} + n^2 \log(n \log \rho) \log(n \log \rho)\right)$ time, respectively, where $L$ is the precision of the integers used, $T_{p-p}(n)$ is the time complexity of the point-to-point shortest path algorithm used, $\rho$ is the ratio of the length of the longest obstacle edge to the Euclidean distance between $s$ and $f$, and $\lambda(n)$ is a very slowly-growing function related to the inverse of the Ackermann’s function. The main contribution of this paper is in the third algorithm.

1.2 Previous Work in $\mathbb{R}^3$

In the approximate shortest path problem, an additional real positive parameter $\epsilon > 0$, which defines the quality of the approximation, is given as part of the input, and the goal is to find a path between two given points $s$ and $t$ that avoids (the interior of) the obstacles and has length at most $(1 + \epsilon)$ times the length of the shortest obstacle avoiding path between those two points. Such an approximate path is referred to as an $\epsilon$-approximate path or an $\epsilon$-approximation of the shortest path. In this paper, we assume $\epsilon \leq 1$. There is also a more general approach, where given a point $s$, a shortest path map with respect to $s$ is constructed. In $\mathbb{R}^3$, this approach has been investigated for shortest paths on polytopes, as well as for shortest paths among obstacles [11].

Papadimitriou [15] gave the first fully polynomial time approximation scheme for the general shortest path problem in $\mathbb{R}^3$. The time complexity of the algorithm is $O(n^4 (L+\log(n/\epsilon))^2 / \epsilon^2)$, where $n$ is the complexity of the set of obstacles $S$, i.e. the number of edges, and $L$ is the precision of the integers used, that is, the number of bits in the largest integer describing the coordinates of any scene element.

The approach is relatively simple and thus it could be implemented in practice. It discretizes the problem by breaking every edge into a number of small
segments. A visibility graph $G$ is constructed, in which each node represents a segment. A link between two nodes is created if the two segments represented are visible to each other. The weight of the link is set to the Euclidean distance between the mid-points of the segments (see Fig. 1 for an illustration).

To partition one edge $e$ of $S$, a coordinate system is chosen such that $e$ lies on the x-axis and the origin is the closest point on $e$ to $s$. A sequence of points is added on $e$ with coordinates $\{x_i = a(1 + \epsilon_1)^{i-1} : i = 1, 2, 3, \ldots\}$, where $a$ is the Euclidean distance from $s$ to $e$ and $\epsilon_1$ is a real positive parameter. As a result, the length of each segment is no more than $\epsilon_1$ times the distance from $s$ to the segment. It is shown in [18] that applying Dijkstra’s algorithm on $G$ gives a path which is at most $(1 + 4\epsilon_1)$ times the length of the shortest path. Taking $\epsilon_1 = \epsilon/4n$ gives an $\epsilon$-approximate path.

Given a source point $s$, a real positive parameter $\epsilon$, and a set of polyhedral obstacles $S$ in $\mathbb{R}^3$, with a total of $n$ vertices, Har-Peled [11] presented a technique to construct approximate shortest path maps in $O(T_{p-p}(n) + (1/\epsilon^2 \log 1/\epsilon))$ time on each face and in $O(T_{p-p}(n) + (1/\epsilon^2 \log 1/\epsilon)n^2)$ time total in $\mathbb{R}^3$, where $T_{p-p}(n)$ is the time complexity of the point-to-point shortest path algorithm used. In [11], the point-to-point shortest path algorithm used is that of Clarkson [6], which we will discuss later. Once the map is constructed, it takes $O(\log n/\epsilon)$ time for each query, that is, given a query point $t \in \mathbb{R}^3$, the length of a $(1 + \epsilon)$-approximate path from $s$ to $t$ can be reported in $O(\log(n/\epsilon))$ time.

To construct a shortest path map for $s$ on an obstacle face $f$, Har-Peled’s algorithm places a set of $O(1/\epsilon^2 \log 1/\epsilon)$ points on $f$. A weighted Voronoi diagram is then constructed on those points. The weight of each point is the length of an $\epsilon/8$-approximate path from $s$ to the point, obtained by any existing shortest path algorithm. For a point $t \in f$, the distance function $f(t)$ is defined as $f(t) = w_p + |pt|$, where $p$ is the point in the weighted Voronoi diagram closest to $t$, $w_p$ is the weight of $p$, and $|pt|$ is the Euclidean distance between $p$ and $t$ (see Fig. 2). It was shown in [11] that if the points are placed “carefully”, $f(t)$ is at most $(1 + \epsilon)$ times the length of the shortest path between $s$ and $t$. 

**Fig. 1.** A shortest path and its approximation.
The distance function $f(t)$ is defined as $f(t) = w_p + |pt|$

Clarkson [6] gave an $O(n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n/\epsilon) \log(n \log \rho))$ algorithm for computing an $\epsilon$-approximation of the shortest obstacle avoiding path between two given points $s$ and $t$ in $\mathbb{R}^3$, where $\rho$ is the ratio of the length of the longest obstacle edge to the Euclidean distance between $s$ and $t$, and $\lambda(n)$ is a very slowly-growing function related to the inverse of the Ackermann’s function.

Let $S$ be the set of obstacles. The general idea of Clarkson’s approach is to construct a visibility graph $V$. $V$ contains $s$, $t$, and the points on $S$ that $P_{st}$ might pass through. To limit the number of edges per node in $V$, a cone structure $C$ is applied on every node $x \in V$ (see Fig. 3). In a cone $C_a$, with apex $x$, if $y$ is the closest node visible to $x$ then an edge $e$ between $x$ and $y$ is added to $V$. The weight of $e$ is $|xy|$. Notice that there could be many nodes visible to $x$ in $C_a$, but at most one edge is added. Clarkson showed that to ensure that the path found by running a single source shortest path algorithm on $V$ is an $\epsilon$-approximate path of $P_{st}$, the size of the cone structure needed is $O(1/\epsilon^{d-1})$, where $d$ is the dimension of the problem (i.e., $d = 3$ in $\mathbb{R}^3$). Hence, there are $O(1/\epsilon^d)$ edges incident to a node in $V$. In the three-dimensional case, $P_{st}$ could make a turn at any point on any edge $e \in E$. Let $a$ and $b$ be the end points of an edge $e \in E$. A point $x \in e$ can be expressed as a function $x = \beta a + (1-\beta)b$, where $0 \leq \beta \leq 1$. For a cone $C_a$ with apex $x = \beta a + (1-\beta)b$, define the distance function $h^{C_a}(\beta) = |xy|$, where $y$ is the closest node in $C_a$ visible to $x$. One important observation Clarkson made is that $h^{C_a}$ is piecewise linear. Given a predefined cone structure $C$, each $e \in E$ can be divided into segments according to this piecewise linearity. The set of line segments formed is called the combinatorial characterization of $V$, and is denoted as $K_\epsilon$. According to the study of Davenport-Schinzel sequences [21], the size of $K_\epsilon$ is $O(n^2 \lambda(n)/\epsilon^2)$, where $n$ is the number of obstacle edges, and $\lambda(n)$ is a very slowly-growing function (see above). Hence the connectivity relations of $V$ can be represented by $K_\epsilon$.

$V$ contains a set of carefully selected Steiner points chosen as follows. All endpoints of segments in $K_\epsilon$ are Steiner points. Additional Steiner points are
Fig. 3. A simple cone structure with 8 cones.

added to further divide each edge into segments no longer than $\frac{\epsilon}{4n}B$, where $B$ is a lower bound on $|P_{st}|$.

The algorithm for computing an $\epsilon$-approximate path consists of two phases. In the first phase, $B$ is set to $|st|$, $\epsilon$ is set to 1/2, and a simpler cone structure is used to compute a 1/2-approximate path. This provides a better lower bound on $|P_{st}|$ for the second phase, which gives the final approximation.

2 Approximate Point-to-Face Shortest Paths

In this section we present three algorithms for finding approximate point-to-face shortest paths among polyhedral obstacles in $\mathbb{R}^3$.

2.1 A General Approach

The main idea used is to place (a grid of) Steiner points on the target face $f$ such that there is at least one point close enough to $t$ to give us a good approximation. Let $B$ and $D$ denote a lower bound and an upper bound on the shortest path between $s$ and $f$, respectively.

Lemma 1 below is an application of Lemma 2.10 of [11], which proved a more general claim.

**Lemma 1.** Let $x$ be any point on the target face $f$. Let $h$ be the point on $f$ closest to the source $s$ with respect to the Euclidean distance. We have $|P_{sh}| \leq 2|P_{sx}|$.

**Proof.** Refer to Fig. 4. Let $s'$ be the projection of $s$ on the plane supporting $f$. If $s' \in f$, then $h = s'$, otherwise $h$ lies on the boundary of $f$. It is obvious that $|hx| \leq |sx| \leq |P_{sx}|$. We have $|P_{sh}| \leq |P_{sx}| + |hx| \leq 2|P_{sx}|$. \qed
Let $t \in f$ be the end point of the shortest path between $s$ and $f$. Hence the shortest path between $s$ and $f$ can also be expressed as $P_{st}$. By Lemma 1, we have $|P_{sh}| \leq 2|P_{st}|$. By computing an $\epsilon$-approximate path between $s$ and $h$, i.e. $Q_{sh}$, we can obtain the lower bound and upper bound on $|P_{st}|$ as $B = |Q_{sh}|/(2(1+\epsilon))$ and $D = |Q_{sh}|$.

2.2 The First Algorithm

Our first algorithm for finding approximate point-to-face shortest paths extends Papadimitriou’s algorithm for the point-to-point version. We proceed as follows. The lower bound $B$ and the upper bound $D$ of the shortest path can be computed by identifying the point $h$ and computing $Q_{sh}$ using Papadimitriou’s algorithm. Obviously $|ht| \leq P_{st} \leq D$, which implies $t$ must be within distance $D$ of $h$ (see Fig. 5).

We define a sample grid as a uniform grid of unit length $\epsilon B$, which is applied on the target face $f$ within distance $D$ of $h$. The total number of sample points is $O(1/\epsilon^2)$.

Lemma 2. There exists one sample point $t'$ that gives a $3\epsilon$-approximate path.

Proof. Let $t'$ be the sample point closest to $t$. We have $|tt'| \leq \epsilon B \leq \epsilon |P_{st}|$ and

$$|P_{st'}| \leq |P_{st}| + |tt'|$$

$$|Q_{st'}| \leq (1+\epsilon)|P_{st'}|$$

$$\leq (1+\epsilon)(|P_{st}| + |tt'|)$$

$$\leq (1+\epsilon)(|P_{st}| + \epsilon |P_{st}|)$$

$$\leq (1 + 3\epsilon) |P_{st}|$$

where $Q_{st'}$ is the approximating path and $P_{st}$ is the shortest path. \qed
In Papadimitriou’s algorithm, each edge of $S$ is split into at most $O((L + \log(1/\epsilon_1))/\epsilon_1)$ segments. The number of visibility graph edges resulting from this subdivision of edges of $S$ is $O((n(L + \log(1/\epsilon_1))/\epsilon_1)^2)$. With $\epsilon_1 = \epsilon/4n$ we have $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2)$ edges. We also have $O(n^2(L+\log(n/\epsilon))/\epsilon*(1/\epsilon^3))$ graph edges corresponding to visibility edges between the segments on edges of $S$ and the sample points on $f$. The number of vertices of the graph is $O(n^4(L+\log(n/\epsilon))/\epsilon^2 + n^2(L+\log(n/\epsilon))/\epsilon^3)$. Dijkstra’s algorithm takes $O(N \log N + M)$ time, where $M$ is the number of graph edges and $N$ is the number of graph nodes. With $M = O(n^4(L+\log(n/\epsilon))^2/\epsilon^2 + n^2(L+\log(n/\epsilon))/\epsilon^3)$ and $N = n^2(L+\log(n/\epsilon))/\epsilon^2 + 1/\epsilon^3)$, Dijkstra’s algorithm takes $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2 + n^2(L+\log(n/\epsilon))/\epsilon^3)$ time to compute a $3\epsilon$-approximate path between $s$ and $f$ by Lemma 2.

In order to have an $\epsilon$-approximate path between $s$ and $f$, we simply use a new parameter $\epsilon'' = \epsilon/3$ to construct the visibility graph. The new parameter does not affect the complexity of the algorithm.

**Theorem 1.** Given a point $s$, a face $f$, a set $S$ of obstacles, with a total of $n$ vertices, and a real positive parameter $\epsilon$, an $\epsilon$-approximation of the shortest obstacle-avoiding path from $s$ to $f$ can be computed in $O(n^4(L+\log(n/\epsilon))^2/\epsilon^2 + n^2(L+\log(n/\epsilon))/\epsilon^3)$ time, where $n$ is the complexity of $S$ and $L$ is the number of bits in the largest integer describing the coordinates of any scene element.

### 2.3 The Second Algorithm

Our second algorithm builds upon Har-Peled’s point-to-point approximate shortest path algorithm. Har-Peled’s algorithm constructs an approximate shortest path map on a given face $f$ with respect to a source $s$ in $O(T_{p-p}(n) \cdot (1/\epsilon^2 \log 1/\epsilon))$ time, where $T_{p-p}(n)$ is the complexity of the point-to-point shortest path algorithm used. As mentioned previously, a full map on $f$ is unnecessary as $t$ must locate within $D$ radius of $h$. Hence only a partial map within $D$ radius of $h$ is
Theorem 2. Given a query point \( s \), a face \( f \) and a real positive parameter \( \epsilon \), an \( \epsilon \)-approximation of the shortest obstacle-avoiding path from \( s \) to \( f \) can be computed in \( O((n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n\rho) \log(n \log \rho)) \ast (1/\epsilon^2)) \) time, where \( \rho \) is the ratio of the length of the longest obstacle edge to the Euclidean distance between \( s \) and \( t \), and \( \lambda(n) \) is a very slowly-growing function related to the inverse of the Ackermann’s function.

This second algorithm is very similar to the first one. Both algorithms need a sample grid of size \( O(1/\epsilon^2) \) and execute a point-to-point shortest path algorithm on each point, which means we have to run a point-to-point shortest path algorithm \( O(1/\epsilon^2) \) times. In the next subsection, we show how to obtain a point-to-face shortest path algorithm which has the same asymptotic complexity as Clarkson’s point-to-point version.

2.4 The Third Algorithm

Theorem 3. Given a query point \( s \), a face \( f \), a set of obstacles \( S \) and a real positive parameter \( \epsilon \), we can find an \( \epsilon \)-approximate path of the shortest obstacle-avoiding path from \( s \) to \( f \) in time \( O(n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n\rho) \log(n \log \rho)) \ast (1/\epsilon^2) \), where \( \rho \) is the ratio of the length of the longest obstacle edge to the Euclidean distance between \( s \) and \( t \), and \( \lambda(n) \) is a very slowly-growing function related to the inverse of the Ackermann’s function.

Notice that in Theorem 3 the point-to-face approximation has the same complexity as Clarkson’s point-to-point approximation [6]. Indeed, our solution builds upon Clarkson’s point-to-point solution.

We will use a different approach to prove Theorem 3. Instead of building a sample grid on \( f \) first, we will construct a visibility graph first and then decide the additional Steiner points needed on \( f \) based on the visibility graph constructed. All additional Steiner points on \( f \) can be added directly to the visibility graph. Hence, we only need to execute Dijkstra’s algorithms once for all Steiner points placed on \( f \). We will explain how additional Steiner points and the corresponding visibility graph edges are produced, and give a count on the total number of Steiner points.

Clarkson’s algorithm consists of two phases. In the first phase, a coarse approximation of the shortest path between the source and destination is computed, which gives a better lower bound for the second phase. We set \( h \), which is the closest point on \( f \) to \( s \) in Euclidean distance, as the destination point and find a \( \frac{1}{2} \)-approximation of the shortest path between \( s \) and \( h \) using the first phase of
Clarkson’s approach. Let $t \in f$ be the end point of the shortest path between $s$ and $f$. Set the length of the $\frac{1}{2}$–approximate path as the upper bound $D$ of $P_{st}$ and let $B = D/2(1 + 1/2)$ as the lower bound, following Lemma. Obviously, $P_{st}$ cannot travel beyond the sphere centered at $h$ with radius $2D$ (see Fig. 6). In the second phase, following Clarkson’s approach, we only need to partition obstacle edges within the sphere into segments of length $\epsilon B/4n$ and then apply the cone structure on each Steiner point created. With this, we have the following lemma.

**Lemma 3.** If $t$ is on an edge $e \in f$ then there exists a Steiner point $t' \in e$ such that $|Q_{st'}| \leq (1 + 5/4\epsilon)|P_{st}|$.

**Proof.** On $e$, choose the Steiner point closest to $t$ as $t'$. See Fig. 7 for an illustration. Since each segment is no longer than $\frac{\epsilon}{4n}|P_{st}|$, $|tt'| \leq \frac{\epsilon}{8n}|P_{st}|$. By triangle inequality, we have $|P_{st'}| \leq |P_{st}| + |tt'|$ and

\[
|Q_{st'}| \leq (1 + \epsilon)|P_{st'}| \quad \text{(from Clarkson’s Steiner point placement)}
\]

\[
\leq (1 + \epsilon)(|P_{st}| + |tt'|)
\]

\[
\leq (1 + \epsilon)|P_{st}| + (1 + \epsilon)(\frac{\epsilon}{8n}|P_{st}|)
\]

\[
\leq (1 + \epsilon + \frac{\epsilon}{8n} + \frac{\epsilon^2}{8n})|P_{st}|
\]

\[
\leq (1 + 5/4\epsilon)|P_{st}| \quad \text{(since $\epsilon \leq 1$ and $n \geq 1$)}
\]

To handle the case when $t$ is in the interior of $f$ we need to add two sets of Steiner points. We do this as follows. For the first set, for each obstacle edge $e$,
we add point \( u \in e \) as a new Steiner point, if \( u \) satisfies:
1) its projection \( u' \) in the plane containing \( f \) is in the interior of \( f \), and
2) \( u \) is visible from \( u' \), and
3) \( uu' \) is tangent to some obstacle, which does not contain \( e \).
We apply the cone structure (following Clarkson’s algorithm) on \( u \) and add the corresponding edges to the visibility graph. Since there are at most \( n^2 \) such points and the number of Steiner points in the original visibility graph is \( \Omega(n^2) \), the complexity of the visibility graph is unchanged.

To obtain the second set of Steiner points, we proceed as follows. For each existing Steiner point \( v \) on an obstacle edge, let \( v' \) be the projection of \( v \) in the plane containing \( f \). We add \( v' \) and edge \( v'v \) to the visibility graph if \( v' \) is in the interior of \( f \) and \( v \) is visible from \( v' \). Note that we do not apply the cone structure on \( v' \). Observe that, we at most double the number of Steiner points and each additional Steiner point introduces exactly one edge. The complexity of the visibility graph remains the same. We will discuss later how to efficiently determine the visibility between a point and its projection on a plane.

**Lemma 4.** Let the contact point of \( P_{st} \) on the last obstacle edge before reaching \( f \) be \( c_1 \), that is, \( c_1t \) is the last segment of \( P_{st} \). If \( t \) is an interior point of \( f \) then \( c_1t \) must be perpendicular to \( f \).

**Proof.** We make the proof by contradiction. Suppose \( t \) is in the interior of \( f \) and \( c_1t \) is not perpendicular to \( f \). We will construct a new path from \( c_1 \) to some point \( t' \in f \) such that the new path is shorter than \( |c_1t| \).

Let \( c'_1 \) be the projection of \( c_1 \) on the plane \( \pi \) containing \( f \). Consider the plane formed by \( c_1, c'_1 \) and \( t \). Let \( t' = t \). We shift \( t' \) along the segment \( tc'_1 \) until either \( c'_1 \) is reached or the segment \( c_1t' \) intersects some edge \( e \) at a point \( c_2 \). We have \( |c_1t'| < |c_1t| \), a contradiction.

\[\square\]
Lemma 5. If $t$ is an interior point of $f$, in the modified visibility graph there exists at least one Steiner point $t'$ on $f$ and such that $|Q_{st'}| \leq (1 + 5/4\epsilon)|P_{st}|$.

Proof. By Lemma 4, the last segment $c_1t$ of $P_{st}$ is perpendicular to $f$, since $t$ is an interior point of $f$. Assume $c_1$ is not a Steiner point, since otherwise we are done. On the edge containing $c_1$, let $d_1$ be one of the two Steiner points neighboring $c_1$, specifically, one visible from its projection on $f$. Such a Steiner point exists from our placement of Steiner points (set one). We have $|c_1d_1| \leq \frac{\epsilon}{4n}|P_{st}|$. Let the projection of $d_1$ on $f$ be $t'$. Use an alternative path from $s$ to $t'$ to approximate $P_{st}$ (See Fig. 9 for an illustration). We have

$$|Q_{st'}| = |Q_{sd_1}| + |d_1t'| \leq |P_{sd_1}| + \epsilon|P_{st}| + |d_1t'| \leq |P_{sd_1}| + \epsilon|P_{st}| + |c_1t| + |c_1d_1| \leq |P_{st}| + \epsilon|P_{st}| + \frac{\epsilon}{4n}|P_{st}| \leq (1 + 5/4\epsilon)|P_{st}|$$

Following Lemma 3 and Lemma 5 the new visibility graph gives us a $5/4\epsilon$-approximate path from $s$ to $f$. To conclude the proof of Theorem 3 we need to show how to find the additional Steiner points and visibility edges within the given time bound.

Visibility Computation We now show how to compute the additional Steiner points on obstacle edges as well as on the target obstacle face $f$. Since addition of Steiner points does not change the complexity of the visibility graph, the cost of the shortest path computation is $O(n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n^2) \log(n \log \rho))$. We would like to keep the computation of additional Steiner points and edges that capture the visibility between Steiner points on obstacle edges and their projections on $f$ within this time bound.
A naive approach to compute the visibility between two points $u$ and $u'$ would require $O(n)$ time by checking whether the line segment $uu'$ intersects any obstacle. Over all points, this will exceed the time bound above.

We can try to reduce the time to compute the visible pairs by using a ray shooting data structure. As mentioned in [23], the general ray shooting problem in three dimensions is still far from being fully solved. However, our problem is a special case of the ray shooting problem. We need to determine the visibility between a point and its orthogonal projection on the target plane $\pi$ supporting $f$, instead of the visibility between any two arbitrary points, i.e., the direction for ray shooting is always perpendicular to $\pi$. On each obstacle edge $e$, all Steiner points and their projections on $\pi$ are coplanar. Let $\pi'$ be the plane passing through $e$ and perpendicular to $\pi$. By computing $G = \pi' \cap S$ for each edge of $S$ (see Fig. 10), we can transform the three dimensional visibility problem into $O(n)$ subproblems in dimension two, where $n$ is the number of obstacle edges.

Let edge $e'$ be the projection of an edge $e$ onto $\pi$. Notice that all line segments between Steiner points on $e$ and their projections on $\pi$ (that is, on $e'$) are parallel. We can perform a plane sweeping on $G$ with a horizontal line and maintain a data structure which stores edges intersecting with the sweep line at a certain time instance. The data structure is updated when the sweep line passes a vertex and each insertion and deletion process requires $O(\log n)$ time [19]. During this process, we could partition $e$ into $O(n)$ open segments, such that all points on each segment have the same visibility answer with respect to their corresponding projections on $e'$. Furthermore, the end points of segments that are visible from their projection on $f$ are the new Steiner points to be added on obstacle edges. The total time of adding Steiner points is $O(n^2 \log n + N)$, where $N$ is the number of Steiner points in the original visibility graph.

We can simplify this process as follows. Since the order of the edges intersecting with the sweep line is not important, we can avoid the plane sweep procedure altogether while still partitioning each edge into $O(n)$ segments in
Fig. 10. The sweep line intercepts edges e, 1, 2, 3, 4, and q

$O(n \log n)$ time. Recall that we assume the obstacles are triangulated. For each triangle $\Delta$, find the intersection segment $r = \pi' \cap \Delta$. If at least one vertex of $r$ is inside of the quadrilateral formed by the endpoints of $e$ and $e'$, we project $r$ on $e$. The projection of $r$ on $e$ is the region in which the visibility of Steiner points to their counterparts are blocked by $r$. Hence we can divide $e$ into $O(n)$ segments in $O(n \log n)$ time. Again, the total time is $O(n^2 \log n + N)$.

Note that the cost of visibility computation is dominated by the cost of the shortest path computation, i.e. the overall time to approximate the shortest path from $s$ to $f$ is $O(n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n\rho) \log(n \log \rho))$.

This concludes the proof of Theorem 8.

3 Conclusions

In this paper we discussed three point-to-face approximate shortest path algorithms in $\mathcal{R}^3$. It is interesting to notice that the point-to-face shortest path must end on $f$ within a certain range from $h$, where $h$ is the point on $f$ closest to $s$ in Euclidean distance. That range can be obtained by applying known point-to-point shortest path algorithms between $s$ and $h$. After placing a sample grid near $h$, we proved we can find an $\epsilon$-approximate path between $s$ and $f$ in $O(n^4 (L + \log(n/\epsilon))^2/\epsilon^2 + n^2 (L + \log(n/\epsilon))/\epsilon^3)$ time by extending Papadimitriou’s algorithm or in $O((n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n\rho) \log(n \log \rho)) \cdot (1/\epsilon^2))$ time by extending Har-Peled’s algorithm. However, this approach still requires to execute a point-to-point shortest path algorithm $O(1/\epsilon^2)$ times. Finally we showed that Clarkson’s point-to-point shortest path approach can be extended to solve the problem by adding additional Steiner points directly to the visibility graph, without changing the asymptotic complexity of the algorithm, resulting in an $O(n^2 \lambda(n) \log(n/\epsilon)/\epsilon^4 + n^2 \log(n\rho) \log(n \log \rho))$ time algorithm for finding an $\epsilon$-approximate path between a source point $s$ and an obstacle face $f$. 
References

1. P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan. Approximate shortest paths on a convex polytope in three dimensions. *J. Assoc. Comput. Mach.*, 44:567-584, 1997.
2. C. Bajaj. The algebraic complexity of shortest paths in polyhedral spaces. tech. rept. CSD-TR-523, Computer Science Dept., Purdue University, June 1985.
3. C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete Comput. Geom.*, 3:177-191, 1988.
4. J. Canny, and R.H. Reif. New lower bound techniques for robot motion planning problems. *Proc. 28th IEEE Sympos. Found. Comput. Sci.*, pp. 49-60, 1987.
5. J. Chen, and Y. Han. Shortest paths on a polyhedron; Part I: computing shortest paths. *Internat. J. Comput. Geom. Appl.*, 6: 127-144, 1996.
6. K.L. Clarkson. Approximation algorithms for shortest path motion planning. *Proc. 19th ACM Sympos. Theory Comput.*, New York, pp.56-65, 1987.
7. S.K. Ghosh, and D.M. Mount. An output sensitive algorithm for computing visibility graphs. *Proc. 28th Annual IEEE Sympos. on Foundations of Computer Science*, pp. 11-19, 1987.
8. S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *Discrete & Computational Geometry*, 21(2): 217-231, 1999.
9. S. Kapoor and S.N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. *Proc. 4th Annual ACM Sympos. on Computational Geometry*, pp. 172-182, 1988.
10. S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geom.*, 18:377-383, 1997.
11. S. Har-Peled. Constructing Approximate Shortest Path Maps in Three Dimensions. *SIAM J. Comput.*, 28(4): 1182-1197, 1999.
12. J. Hershberger, and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215-2256, 1999.
13. J. S. B. Mitchell. Shortest paths among obstacles in the plane. *Internat. J Comput. Geom. Appl.*, 6:309-332, 1996
14. J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. pp. 633-701. Elsevier Publishers B.V. North-Holland, Amsterdam, 2000.
15. D.M. Mount. On finding shortest paths on convex polyhedra. Tech. Rept., Computer Science Dept., Univ. Maryland, College Park, Octobor, 1984.
16. J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647-668, 1987
17. M.H. Overmars and E. Welzl. New methods for computing visibility graphs. *Proc. 4th Annual ACM Sympos. on Computational Geometry*, pp. 164-171, 1988.
18. C.H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20:259-263, 1985.
19. F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
20. Y. Schreiber and M. Sharir. An optimal-time algorithm for shortest paths on a convex polytope in three dimensions. *Discrete & Computational Geometry (DCG)*, 39(1-3):500-579, 2008.
21. M. Sharir, R. Cole, K. Kedem, D. Leven, R. Pollack, and S. Sifrony. Geometric application of Davenport-Schinzel sequences. *Proc. 27th IEEE FOCS*, pp. 77-86, 1986.
22. M. Sharir and A. Schorr. On Shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193-215, 1986.

23. M. Sharir, and H. Shaul, Ray shooting and stone throwing with near-linear storage. *Comput. Geom.*, 30(3): 239-252, 2005.