Constructing Approximate Shortest Path Maps in Three Dimensions[*]

Sariel Har-Peled[†]

November 7, 2005

Abstract

We present a new technique for constructing a data-structure that approximates shortest path maps in $\mathbb{R}^d$. By applying this technique, we get the following two results on approximate shortest path maps in $\mathbb{R}^3$.

(i) Given a polyhedral surface or a convex polytope $\mathcal{P}$ with n edges in $\mathbb{R}^3$, a source point s on $\mathcal{P}$, and a real parameter $0 < \varepsilon \leq 1$, we present an algorithm that computes a subdivision of $\mathcal{P}$ of size $O((n/\varepsilon)\log(1/\varepsilon))$ which can be used to answer efficiently approximate shortest path queries. Namely, given any point t on $\mathcal{P}$, one can compute, in $O(\log(n/\varepsilon))$ time, a distance $\Delta_{\mathcal{P},s}(t)$, such that $d_{\mathcal{P},s}(t) \leq \Delta_{\mathcal{P},s}(t) \leq (1+\varepsilon)d_{\mathcal{P},s}(t)$, where $d_{\mathcal{P},s}(t)$ is the length of a shortest path between s and t on $\mathcal{P}$.

The map can be computed in $O(n^2 \log n + (n/\varepsilon)\log(1/\varepsilon)\log(n/\varepsilon))$ time, for the case of a polyhedral surface, and in $O((n/\varepsilon^3)\log(1/\varepsilon) + (n/\varepsilon^{1.5})\log(1/\varepsilon)\log n)$ time if $\mathcal{P}$ is a convex polytope.

(ii) Given a set of polyhedral obstacles $\mathcal{O}$ with a total of n edges in $\mathbb{R}^3$, a source point s in $\mathbb{R}^3 \setminus int \bigcup_{O \in \mathcal{O}} O$, and a real parameter $0 < \varepsilon \leq 1$, we present an algorithm that computes a subdivision of $\mathbb{R}^3$, which can be used to answer efficiently approximate shortest path queries. That is, for any point $t \in \mathbb{R}^3$, one can compute, in $O(\log(n/\varepsilon))$ time, a distance $\Delta_{\mathcal{O},s}(t)$ that ε-approximates the length of a shortest path from s to t that avoids the interiors of the obstacles. This subdivision can be computed in roughly $O(n^4/\varepsilon^6)$ time.

1 Introduction

The *three-dimensional Euclidean shortest-path problem* is defined as follows: Given a set of pairwise-disjoint polyhedral objects in $\mathbb{R}^3$ and two points s and t, compute the shortest path between s and t which avoids the interiors of the given polyhedral 'obstacles'. This problem has received considerable attention in computational geometry. It was shown to be NP-hard by Canny and Reif [CR87], and the fastest available algorithms for this problem

[*]This work has been supported by a grant from the U.S.–Israeli Binational Science Foundation. This work is part of the author's Ph.D. thesis, prepared at Tel-Aviv University under the supervision of Prof. Micha Sharir.

[†]School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel; `sariel@math.tau.ac.il`

run in time that is exponential in the total number of obstacle vertices (which we denote by n) [RS94, Sha87]. The apparent intractability of the problem has motivated researchers to develop polynomial-time algorithms for computing approximate shortest paths and for computing shortest paths in special cases.

In the *approximate three-dimensional Euclidean shortest-path* problem, we are given an additional parameter $\varepsilon > 0$, and the goal is to compute a path between s and t that avoids the interiors of the obstacles and whose length is at most $(1 + \varepsilon)$ times the length of the shortest obstacle-avoiding path (we call such a path an *ε-approximate path*). Approximation algorithms for the three-dimensional shortest path problem were first studied by Papadimitriou [Pap85], who gave an $O(n^4(L + \log(n/\varepsilon))^2/\varepsilon^2)$-time algorithm for computing an ε-approximate shortest path, where L is the number of bits used in each computation. A rigorous analysis of Papadimitriou's algorithm was recently given by Choi et al. [CSY94]. A different approach was taken by Clarkson [Cla87], resulting in an algorithm with roughly $O(n^2/\varepsilon^4)$ running time (the precise result is stated in Theorem 4.2).

The problem of computing a shortest path between two points along the surface of a single convex polytope is an interesting special case of the three-dimensional Euclidean shortest-path problem. Sharir and Schorr [SS86] gave an $O(n^3 \log n)$ algorithm for this problem, exploiting the property that a shortest path on a polyhedron *unfolds* into a straight line. Mitchell et al. [MMP87] improved the running time to $O(n^2 \log n)$; their algorithm works for non-convex polyhedra (or polyhedral surfaces) as well. Chen and Han [CH96] gave another algorithm with an improved running time of $O(n^2)$. It is a rather long-standing and intriguing open problem whether the shortest path on a convex polytope can be computed in subquadratic time. This has motivated the problem of finding near-linear algorithms that produce only an approximation of the shortest path. The first result in this direction is by Hershberger and Suri [HS95]. They present a simple algorithm that runs in $O(n)$ time, and computes a path whose length is at most $2d_P(s, t)$. Using the algorithm of [HS95], Agarwal et al. [AHSV97] present a relatively simple algorithm that computes an ε-approximate shortest path (i.e., a path on ∂P between two points $s, t \in \partial P$ whose length is at most $(1+\varepsilon)d_P(s, t)$), for any prescribed $0 < \varepsilon \leq 1$, where the running time of the algorithm is $O(n \log(1/\varepsilon)+1/\varepsilon^3)$. In a companion paper [Har99], we present an improved algorithm, with $O(n)$ preprocessing time, that answers two-points ε-approximate shortest-path queries in $O((\log n)/\varepsilon^{1.5} + 1/\varepsilon^3)$ time, for *any* pair of points $s, t \in \partial P$. Recently, Varadarajan and Agarwal [VA97] gave a subquadratic time algorithm that computes a constant approximation to the shortest path on a polyhedral terrain. Other recent works, by Mata and Mitchell [MM97], and also by Lanthier at al. [LMS97] implement various heuristics for computing approximate shortest paths on weighted terrains (i.e., each face f is being assigned a weight w_f, such that the distance between any two points $a, b \in f$ is $w_f \cdot |ab|$). Those programs give satisfactory results in practice, which are within an order of magnitude better than their worst case analysis.

In this paper, extending our work in [Har99], we present a new general technique for constructing a data-structure that one can use to answer ε-approximate shortest-path queries, for a source point s and approximation factor $\varepsilon > 0$ fixed in advance. Using this technique, we solve two problems involving approximate shortest path maps in $\mathbb{R}^3$.

Approximate shortest path maps. The exact algorithms of [MMP87, SS86] receive as input a convex polytope or a polyhedral surface $\mathcal{P}$, and a fixed source point s on $\mathcal{P}$, and compute a map (i.e., a subdivision of $\mathcal{P}$) of complexity $\Theta(n^2)$, that can be used to answer (exact) shortest path queries from s to any point on $\mathcal{P}$ (along $\mathcal{P}$) in $O(\log n)$ time (such a query reports the *length* of the shortest path; reporting the path itself might require more time). This shortest path map can be stored in linear space, for the case of a convex polytope, by using a persistent data-structure, see [Mou87]. However, the time required to compute this compact representation of the shortest path map is quadratic in the worst case. This raises the problem of computing a map of near-linear size for approximate shortest-path queries from s. We show in Section 3 that this is indeed possible: Given a polyhedral surface $\mathcal{P}$ with n edges in $\mathbb{R}^3$, a source point $s \in \mathcal{P}$, and a prescribed $0 < \varepsilon \leq 1$, there exists a map (a subdivision of $\mathcal{P}$) of complexity $O((n/\varepsilon)\log(1/\varepsilon))$, such that for any $t \in \mathcal{P}$, one can compute the length of an ε-approximate shortest path between s and t on $\mathcal{P}$ in $O(\log(n/\varepsilon))$ time, by locating t in the map.

We present an algorithm that constructs such an approximation map in $O(n^2 \log n + (n/\varepsilon) \log(1/\varepsilon) \log(n/\varepsilon))$ time, for the case of a polyhedral surface, and in $O((n/\varepsilon^3)\log(1/\varepsilon) + (n/\varepsilon^{1.5}) \log(1/\varepsilon) \log n)$ time, for the case of a convex polytope. Note that if $\mathcal{P}$ is a convex polytope than our previous result [Har99] provides an alternative structure with similar properties. However, the dependence of the query time on ε is much better in the method we present here.

Approximate spatial shortest path maps. In Section 4, we present a similar result for ε-approximate shortest paths among polyhedral obstacles in $\mathbb{R}^3$. Let $\mathcal{O}$ be a set of polyhedral obstacles in $\mathbb{R}^3$ with a total of n edges, s a source point in $\mathbb{R}^3$, and $0 < \varepsilon \leq 1$ a parameter. We show that there exists a spatial subdivision $\mathcal{M}$ of $\mathbb{R}^3$, such that for any $t \in \mathbb{R}^3$, one can compute, in $O(\log(n/\varepsilon))$ time, the length of an ε-approximate shortest path between s and t, that avoids the interiors of the obstacles, by performing a spatial point-location query with t in $\mathcal{M}$. The space needed to compute and preprocess $\mathcal{M}$ for spatial point-location is $O(n^2/\varepsilon^{4+\delta})$, for any $\delta > 0$, and the preprocessing time is

$$O\left(\frac{n^4}{\varepsilon^2}\left(\frac{\beta(n)}{\varepsilon^4}\log\frac{n}{\varepsilon} + \log(n\rho)\log(n\log\rho)\right)\log\frac{1}{\varepsilon}\right),$$

where ρ is the ratio of the length of the longest edge in $\mathcal{O}$ to the Euclidean distance between s and t, $\beta(n) = \alpha(n)^{O(\alpha(n))^{O(1)}}$, and $\alpha(n)$ is the extremely slowly growing inverse of the Ackermann function. This algorithm uses the algorithm of Clarkson [Cla87], that computes an ε-approximate shortest path, between two given points, in $O\left(\frac{n^2}{\varepsilon^4}\beta(n)\log\frac{n}{\varepsilon} + n^2\log(n\rho)\log(n\log\rho)\right)$ time.

The paper is organized as follows. Section 2 introduces the notion of a *distance function*, and show how to compute a "small-size" additive weighted Voronoi diagram that enables us to ε-approximate this function. We present two applications of this result. In Section 3, we present the algorithm mentioned above for constructing a map for approximate shortest-path queries from a fixed source on a convex polytope or a polyhedral surface in $\mathbb{R}^3$. In Section 4 we present the algorithm mentioned above for constructing a spatial subdivision for approximate shortest-path queries from a fixed source among polyhedral obstacles in $\mathbb{R}^3$. We conclude in Section 5 by mentioning a few open problems.

2 Approximating a Distance Function by a Weighted Voronoi Diagram

In this section, we introduce the notion of a distance function, and show how to compute a "small-size" additive weighted Voronoi diagram that approximates it up to a factor of $1+\varepsilon$. We use this result in Sections 3 and Section 4 to derive our two main results.

Definition 2.1 Let $\mathcal{I}$ be a subset of $\mathbb{R}^d$. A function $f : \mathcal{I} \to \mathbb{R}$ is a *distance function* on $\mathcal{I}$ if:

- $f(x) + f(y) \geq |xy|$, for any $x, y \in \mathcal{I}$.
- $f(x) + |xy| \geq f(y)$, for any straight segment $xy \subseteq \mathcal{I}$,

where $|xy|$ denotes the Euclidean distance between x and y.

Thus, a distance function has to satisfy two types of triangle inequalities. Since these inequalities are satisfied by the Euclidean distance from any fixed point, a distance function can be regarded as a certain generalization of the Euclidean distance.

Example 2.2 Figure 1 illustrates some geometric restriction imposed on a univariate distance function.

Example 2.3 (i) Let $\mathcal{P}$ be a polyhedral surface in $\mathbb{R}^3$, and let s be a source point on $\mathcal{P}$. For any $t \in \mathcal{P}$, let $d_{\mathcal{P},s}(t)$ denote the length of a shortest path between s and t on $\mathcal{P}$. It is easy to verify that $d_{\mathcal{P},s}(t)$ is a distance function on $\mathcal{P}$.

(ii) Let $\mathcal{O}$ be a collection of pairwise-disjoint polyhedral obstacles in $\mathbb{R}^3$, and let s be a point in $FP(\mathcal{O}) = \mathbb{R}^3 \setminus \bigcup_{O \in \mathcal{O}} int\, O$. Let $FP(\mathcal{O}, s)$ denote the connected component of $FP(\mathcal{O})$ that contains s (i.e., the set of all the points in $\mathbb{R}^3$ that can be connected to s by a path that avoids the interiors of the obstacles of $\mathcal{O}$).

For any $t \in FP(\mathcal{O}, s)$, we denote by $d_{\mathcal{O},s}(t)$ the length of a shortest path between s and t, that avoids the interior of the obstacles of $\mathcal{O}$. Clearly, $d_{\mathcal{O},s}$ is a distance function on $FP(\mathcal{O}, s)$.

Definition 2.4 A pair $\mathcal{S} = (S, w)$ is a *weighted set of points* if $S = \{p_1, \ldots, p_m\}$ is a finite set of points in $\mathbb{R}^d$, and $w(\cdot)$ is a function assigning non-negative weights to the points of S. We define the distance of a point p from the point p_i to be $V_{(p_i, w(p_i))}(p) = |pp_i| + w(p_i)$. We define $V_{\mathcal{S}}(p) = \min_{i=1}^m V_{(p_i, w(p_i))}(p)$. The function $V_{\mathcal{S}}(p)$ induces a natural subdivision $\mathcal{V}_{\mathcal{S}}$ of $\mathbb{R}^d$ into cells, known as the (additive) *weighted Voronoi diagram* of $\mathcal{S}$, such that the i-th cell is the locus of all points closest to p_i in this distance function. As is well known, in the planar case, $\mathcal{V}_{\mathcal{S}}$ has complexity $O(m)$, and it can be computed in $O(m \log m)$ time (see [For87]).

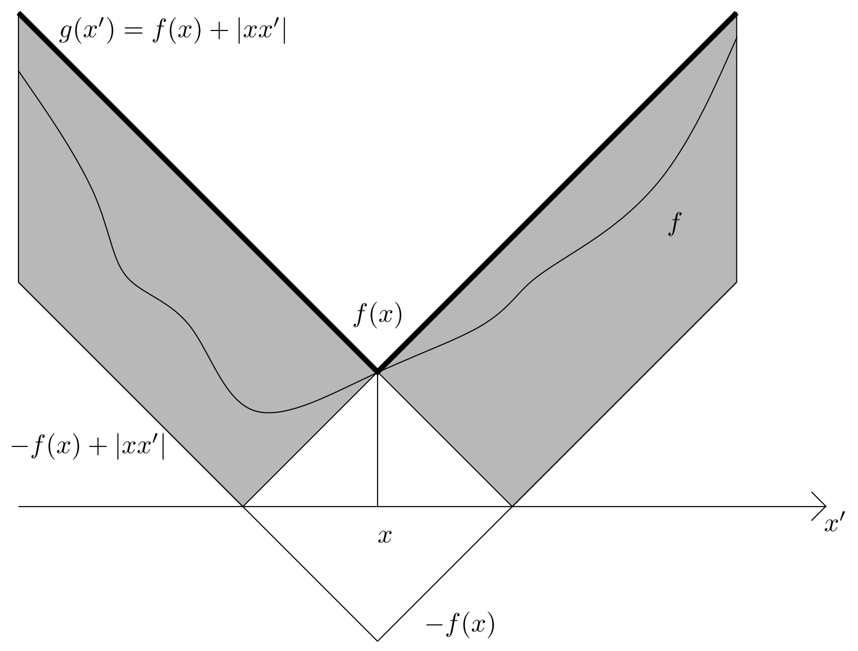

Figure 1: The graph of the distance function f must lie inside the gray area. This implies that the function $g(x') = f(x) + |xx'|$ approximates $f(x')$ "well" in a small neighborhood of x, and for sufficiently large values of x'.

Remark 2.5 For $d \geq 3$, the complexity of an additive weighted Voronoi diagram of m points in $\mathbb{R}^d$ is $O\big(m^{\lfloor d/2 \rfloor + 1}\big)$. This follows by reducing the computation of the diagram to the computation of a convex hull in $d + 2$ dimensions. Furthermore, one can compute the diagram in $O(m^{\lfloor d/2 \rfloor + 1})$ time. See [Aur91].

We next show how to approximate a distance function $f(\cdot)$ by a weighted Voronoi diagram. First, we compute a global minimum p_0 of f. As illustrated in Figure 1, the function $V_{(p_0, f(p_0))}(p)$ approximates $f(p)$ "well" for p sufficiently close to, or sufficiently far from p_0. In other words, f is well-approximated in these regions by the weighted Voronoi diagram of the single site p_0 with weight $f(p_0)$. By adding extra sites to the diagram, we can make the distance induced by the resulting diagram an ε-approximation to f everywhere, as will be shown next.

Definition 2.6 Given a point $p \in \mathbb{R}^d$, and $r \geq 0$, let $B(p, r)$ denote the closed ball of radius r centered at p, and let $\overline{B}(p, r)$ denote the set $\mathbb{R}^d \setminus B(p, r)$. For $r' > r$, let $\mathcal{A}(p, r, r')$ denote the annulus (or shell) $B(p, r') \setminus B(p, r)$.

The following sequence of technical lemmas provide the basis for approximating a given distance function by a weighted Voronoi diagram. The following lemmas are stated for arbitrary dimension d. We will apply them with $d = 1, 2$, or 3.

Lemma 2.7 *Let $\mathcal{I}$ be a convex subset of $\mathbb{R}^d$, f a distance function defined over $\mathcal{I}$, and $\mathcal{S} = (S, w)$ a weighted set, such that $S \subseteq \mathcal{I}$ and $f(x) \leq w(x)$, for all $x \in S$. Then $f(t) \leq V_{\mathcal{S}}(t)$, for all $t \in \mathcal{I}$.*

Proof: Let t be any point of $\mathcal{I}$, and let x denote the point of S realizing $V_{\mathcal{S}}(t)$. Then $V_{\mathcal{S}}(t) = w(x) + |tx| \geq f(x) + |tx| \geq f(t)$. ■

Formalizing the intuition behind Figure 1, we have the following:

Lemma 2.8 *Let $\mathcal{I}$ be a convex subset of $\mathbb{R}^d$, f a distance function defined over $\mathcal{I}$, $0 < \varepsilon \leq 1$ a parameter, p a point in $\mathcal{I}$, and w a real number such that $f(p) \leq w \leq (1+\varepsilon/8)f(p)$. Then $f(t) \leq V_{(p,w)}(t) \leq (1+\varepsilon)f(t)$, for all t in*

$$\mathcal{I} \cap \left(B\left(p, \frac{\varepsilon f(p)}{8}\right) \cup \overline{B}\left(p, \frac{6f(p)}{\varepsilon}\right)\right).$$

Proof: The first inequality $f(t) \leq V_{(p,w)}(t)$ follows immediately from Lemma 2.7.

As for the other inequality, for $t \in \mathcal{I} \cap B\left(p, \frac{\varepsilon f(p)}{8}\right)$, we have

$$\frac{w}{(1+\varepsilon/8)} - \frac{\varepsilon w}{8} \leq f(p) - |pt| \leq f(t) \leq V_{(p,w)}(t) = w + |pt| \leq w + \frac{\varepsilon w}{8}.$$

However,

$$\frac{w + \varepsilon w/8}{\frac{w}{1+\varepsilon/8} - \varepsilon w/8} = \frac{1+\varepsilon/8}{\frac{1}{1+\varepsilon/8} - \varepsilon/8} \leq \frac{1+\varepsilon/8}{1 - \varepsilon/8 - \varepsilon/8} = \frac{1+\varepsilon/8}{1-\varepsilon/4} \leq 1+\varepsilon,$$

since $\varepsilon \leq 1$. Thus $V_{(p,w)}(t) \leq (1+\varepsilon)f(t)$, for all $t \in \mathcal{I} \cap B(p, \varepsilon f(p)/8)$.

For $t \in \mathcal{I} \cap \overline{B}(p, 6f(p)/\varepsilon)$, we have

$$|pt| - w \leq |pt| - f(p) \leq f(t) \leq V_{(p,w)}(t) = |pt| + w.$$

However,

$$\frac{|pt| + w}{|pt| - w} = 1 + \frac{2w}{|pt| - w} \leq 1 + \frac{2w}{3w/\varepsilon - w} \leq 1 + \frac{2w}{2w/\varepsilon} = 1 + \varepsilon,$$

since $|pt| \geq 6f(p)/\varepsilon \geq 3w/\varepsilon$. Thus $V_{(p,w)}(t) \leq (1+\varepsilon)f(t)$, for any such t. ■

Lemma 2.9 *Let $\mathcal{I}$ be a convex subset of $\mathbb{R}^d$, f a distance function defined over $\mathcal{I}$, $0 < \varepsilon \leq 1$ a parameter, and p a point in $\mathcal{I}$. Then for any $t \in \mathcal{I} \cap B(p, \varepsilon f(p)/9)$ and any number w_t such that $f(t) \leq w_t \leq (1+\varepsilon/8)f(t)$, we have $f(p) \leq V_{(t,w_t)}(p) \leq (1+\varepsilon)f(p)$.*

Proof: Since $|pt| \leq \varepsilon f(p)/9$, it follows that $f(t) \geq f(p) - |pt| \geq f(p)(1-\varepsilon/9)$.

Thus,

$$\frac{\varepsilon f(t)}{8} \geq \frac{\varepsilon}{8} f(p)\left(1 - \frac{\varepsilon}{9}\right) \geq \frac{\varepsilon f(p)}{9} \geq |pt|,$$

implying that $p \in B(t, \varepsilon f(t)/8)$. By Lemma 2.8, we have $f(p) \leq V_{(t,w_t)}(p) \leq (1+\varepsilon)f(p)$.

■

The preceding lemmas suggest the following strategy for constructing an approximation of a distance function f over (a convex portion of) $\mathbb{R}^d$: Pick a point p, such that $f(p)$ is close to the global minimum of f. The Voronoi diagram $\mathcal{V}_{(p,w)}$ approximates f "well" near p and outside a larger ball centered at p, where w is an approximation of $f(p)$. We approximate f in the space between those two balls by partitioning it into concentric spherical shells whose radii form an increasing geometric progression, and by covering each shell by a uniform grid (whose unit length increases with the radius of the shell). In this manner, the number of points needed is only a function of ε (the approximation factor) and d.

When approximating a distance function on a convex subset $\mathcal{I}$ of $\mathbb{R}^d$, we have to cope with the possibility that sites might be placed outside $\mathcal{I}$. We overcome this by projecting all such sites onto the boundary of $\mathcal{I}$.

Definition 2.10 Let $\mathcal{I}$ be a convex subset in $\mathbb{R}^d$, and let x be a point in $\mathbb{R}^d$. Let $\nu(x, \mathcal{I})$ denote the projection of x onto $\mathcal{I}$; that is, $\nu(x, \mathcal{I})$ is the closest point (in the Euclidean distance) in $\mathcal{I}$ to x. Clearly, if $x \in \mathcal{I}$ then $\nu(x, \mathcal{I}) = x$.

When x is fixed, we call $\nu(x, \mathcal{I})$ the *hook point* of $\mathcal{I}$.

Definition 2.11 Let $r \geq r' > 0$ be real numbers, let p be a point in $\mathbb{R}^d$, and let $\mathcal{I}$ be a convex set in $\mathbb{R}^d$. We denote by $S(p, \mathcal{I}, r, r')$ the set $\mathcal{I}_{r'} \cap B(p, r) \cap ((r'/\sqrt{d})\mathbb{Z}^d)$, where $\mathbb{Z}^d$ is the integer lattice, and $\mathcal{I}_{r'} = \cup_{q \in \mathcal{I}} B(q, r')$ is the set of all point in $\mathbb{R}^d$ that are at distance at most r' from some point of $\mathcal{I}$. Clearly, $|S(p, \mathcal{I}, r, r')| = O((r/r')^d)$ (with a constant of proportionality depending on d).

The following technical lemma shows how to pick the sites of the additive weighted Voronoi diagram, so that it ε-approximates a given distance function.

Lemma 2.12 *Let $\mathcal{I}$ be a convex subset of $\mathbb{R}^d$, $f : \mathcal{I} \to \mathbb{R}$ a distance function, $0 < \varepsilon \leq 1$ a parameter, c a positive constant, and p a point in $\mathcal{I}$ such that $f(t) \geq f(p)/c$, for all $t \in \mathcal{I}$. Then one can compute a set S in $\mathcal{I}$ of size $O((1/\varepsilon)^d \log(1/\varepsilon))$ (the constant of proportionality depends on c), such that $p \in S$, and for any weight function w on S satisfying $f(x) \leq w(x) \leq (1 + \varepsilon/8)f(x)$, for all $x \in S$, we have $f(t) \leq V_{\mathcal{S}}(t) \leq (1 + \varepsilon)f(t)$, for all $t \in \mathcal{I}$, where $\mathcal{S} = (S, w)$.*

Proof: Let w_p be any number satisfying $f(p) \leq w_p \leq (1 + \varepsilon/8)f(p)$.

Let $r_i = (2^i + 1)w_p$, for $i = 1, \ldots, m$, where $m = \lceil \log_2(6/\varepsilon) \rceil$. Let $\mathcal{A}_1 = B(p, r_1)$, let $\mathcal{A}_i = \mathcal{A}(p, r_{i-1}, r_i)$, for $i = 2, \ldots, m$, and let $\mathcal{A}_{m+1} = \overline{B}(p, r_m)$. Clearly, $\mathcal{I} = \cup_{i=1}^{m+1}(\mathcal{I} \cap \mathcal{A}_i)$.

Let $r'_1 = \varepsilon w_p/(18c)$, and let $r'_i = \varepsilon 2^{i-1} w_p/9$, for $i = 2, \ldots, m$. Let $S'_i = \mathcal{A}_i \cap S(p, \mathcal{I}, r_i, r'_i)$, for $i = 1, \ldots, m$. Let $S = \{p\} \cup \bigcup_{i=1}^m S_i$, where $S_i = \left\{ \nu(x, \mathcal{I}) \,\middle|\, x \in S'_i \right\}$, for $i = 1, \ldots, m$. See Figure 2 for an illustration of the set S.

Let w be any weight function, such that $f(x) \leq w(x) \leq (1 + \varepsilon/8)f(x)$, for any $x \in S$, and let $\mathcal{S} = (S, w)$.

We claim that $\mathcal{S}$ is the required weighted set. Indeed, let $t \in \mathcal{I}$. If $t \in \mathcal{A}_{m+1}$ then $|pt| \geq (2^m + 1)w_p \geq 6f(p)/\varepsilon$. Thus $t \in \overline{B}(p, 6f(p)/\varepsilon)$ and by Lemmas 2.7 and 2.8, we have $f(t) \leq V_{\mathcal{S}}(t) \leq V_{(p,w_p)}(t) \leq (1 + \varepsilon)f(t)$.

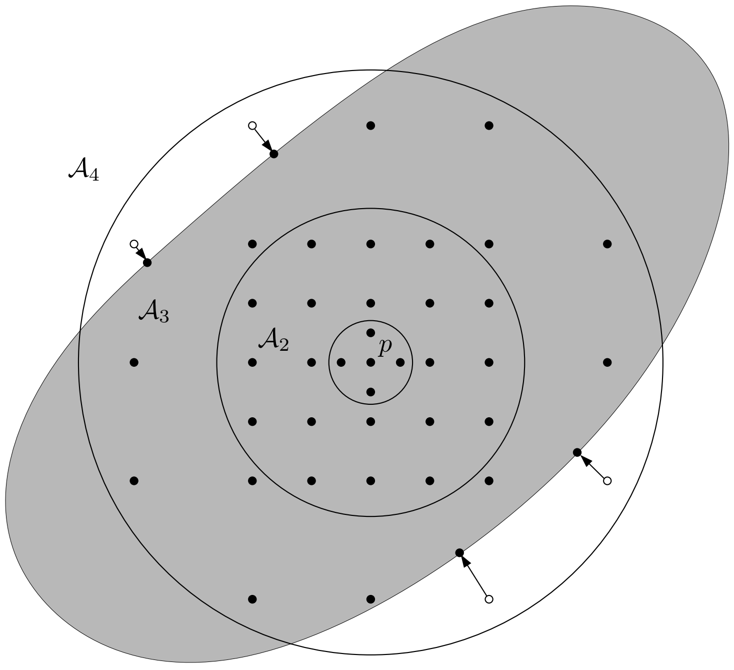

Figure 2: Illustrating the proof of Lemma 2.12. We pick our sites to be on a uniform grid inside each concentric shell around p.

If $t \in B(p, r_m)$, let $\mathcal{A}_i$ be the shell containing t. Let x' be the closest point to t in S'_i. Let $x = \nu(x', \mathcal{I})$. By the definition of ν and S'_i and by the convexity of $\mathcal{I}$, we have $|tx| \le |tx'| \le r'_i$ (see Figure 3).

If $i = 1$ then the inequality $f(t) \ge f(p)/c \ge w_p/2c$ implies that $|tx| \le r'_1 = \varepsilon w_p/(18c) \le \varepsilon f(t)/9$. Thus $x \in B(t, \varepsilon f(t)/9)$.

If $i > 1$ then we also have $|tx| \le r'_i = \varepsilon 2^{i-1} w_p/9 \le \varepsilon f(t)/9$, since $f(t) \ge |pt| - f(p) \ge (2^{i-1}+1)w_p - f(p) \ge 2^{i-1} w_p$. Thus $x \in B(t, \varepsilon f(t)/9)$.

By Lemma 2.9 and Lemma 2.7, we have $f(t) \le V_{\mathcal{S}}(t) \le V_{(x,w(x))}(t) \le (1+\varepsilon) f(t)$.

As for the size of S, we have

$$|S| = O\left(\sum_{i=1}^{m} \left(\frac{r_i}{r'_i}\right)^d\right) = O\left(\left(\frac{3w_p}{\varepsilon w_p/(18c)}\right)^d + \sum_{i=2}^{m} \left(\frac{(2^i+1)w_p}{\varepsilon 2^{i-1} w_p/9}\right)^d\right) = O\left(\frac{1}{\varepsilon^d} \log \frac{1}{\varepsilon}\right).$$

■

To approximate a distance function $f(\cdot)$ using the constructive proof of Lemma 2.12, we need to find a point which is, within a constant factor, a global minimum of f over the given range. The following lemma shows that this can be easily done if f has a known zero point outside the given range.

Lemma 2.13 *Let $\mathcal{I}'$ be a subset of $\mathbb{R}^d$, $f : \mathcal{I}' \to \mathbb{R}$ a distance function, $0 < \varepsilon \le 1$ a parameter, $\mathcal{I}$ a convex subset of $\mathcal{I}'$, and s a point in $\mathcal{I}' \setminus \mathcal{I}$ such that $f(s) = 0$. Then $f(t) \ge f(h)/2$, for all $t \in \mathcal{I}$, where $h = \nu(s, \mathcal{I})$.*

Proof: Let t be any point in $\mathcal{I}$. Since h is the closest point in $\mathcal{I}$ to s and $\mathcal{I}$ is convex, it easily follows that $|st| \ge |ht|$, see Figure 3. Since $f(t) = f(t) + f(s) \ge |st|$, we have $f(t) \ge$

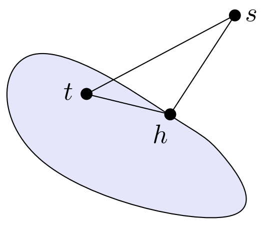

Figure 3: $|st| \geq |ht|$ for $h = \nu(s, \mathcal{I})$

$|ht|$. Moreover, the segment ht is a contained in $\mathcal{I}$, implying that $f(h) \leq f(t) + |ht| \leq 2f(t)$. ■

Remark 2.14 The lemma also holds when $s \in \mathcal{I}$, but then it only yields the trivial bound $f(t) \geq 0$ for all $t \in \mathcal{I}$. Then, of course, s is the required global minimum.

Remark 2.15 Let $\mathcal{I}'$ be a set in $\mathbb{R}^d$, let $\mathcal{I}$ be a convex subset of $\mathcal{I}'$, $f : \mathcal{I}' \to \mathbb{R}$ a distance function, $0 < \varepsilon \leq 1$ a parameter, and s a point of $\mathcal{I}'$ such that $f(s) = 0$. Computing a weighted set $\mathcal{S} = (S, w)$ such that the weighted Voronoi diagram induced on $\mathcal{I}$ approximates f up to a factor of $(1 + \varepsilon)$, can be accomplished by following the proof of Lemma 2.12, in four stages:

(i) Compute the point $p = \nu(s, \mathcal{I})$. By Lemma 2.13, p is "almost" a minimum of f on $\mathcal{I}$.

(ii) Compute an $(\varepsilon/8)$-approximation w_p to $f(p)$ (as prescribed in the lemma).

(iii) Construct the set S of points in $\mathcal{I}$, as prescribed in the proof.

(iv) Approximate the distance function values of all the points of S, up to a factor of $1+\varepsilon/8$, and use these values as the weights for the points of S.

Remark 2.16 The set of points S produced in the proof of Lemma 2.12 is made out of $O(\log(1/\varepsilon))$ subsets (i.e., $S_1, \ldots, S_m$) such that $f(x) \leq cf(t)$, for all $x, t \in S_i$, for $1 \leq i \leq m$, where c is an appropriate constant. This property enables us, in the case of shortest paths on a convex polytope, to approximate the value of the distance function to all the points of S_i simultaneously, yielding a more efficient algorithm. See Remark 3.5 for the details.

Remark 2.17 Let $\mathcal{I}'$ be a subset of $\mathbb{R}^d$, and let f, g be two distance functions defined over $\mathcal{I}'$. It is easy to verify that $h(x) = \max(f(x), g(x))$ is also a distance function. This implies that the distance function induced by any furthest neighbor Voronoi diagram of a finite set of points in $\mathbb{R}^d$, is a distance function in $\mathbb{R}^d$. Hence, by Lemma 2.12, we have:

Corollary 2.18 *Any furthest neighbor Voronoi diagram of points in $\mathbb{R}^d$, can be ε-approximated by a (nearest neighbor) weighted Voronoi diagram, having $O((1/\varepsilon^d)\log(1/\varepsilon))$ sites.*

The following is a strengthening of Lemma 2.12, by noticing that in the cases we are going to apply it to, our distance function is the length of a shortest path from a fixed source point to the given point of $\mathcal{I}$. In such cases, if the source point lies outside $\mathcal{I}$, then a shortest path connecting any point in $\mathcal{I}$ to our source point, must first pass through the boundary $\partial\mathcal{I}$. We next show that if we are able to ε-approximate the distance on the boundary of $\mathcal{I}$, then we can trivially ε-approximate the distance function to all the points of $\mathcal{I}$.

Definition 2.19 Let $\mathcal{I}$ be a convex polytope in $\mathbb{R}^d$. We call a function $f : \mathcal{I} \to \mathbb{R}$ *boundary-induced* on $\mathcal{I}$, if f is a distance function, and for any $t \in \mathcal{I}$, there exists a point $x \in \partial\mathcal{I}$ such that $f(t) = f(x) + |tx|$.

Definition 2.20 Given a convex polytope $\mathcal{I}$ in $\mathbb{R}^d$, we denote by $\phi(\mathcal{I})$ the set of all the facets ($(d-1)$-faces) of $\mathcal{I}$.

Lemma 2.21 *Let $\mathcal{I}$ be a convex polytope in $\mathbb{R}^d$, $f : \mathcal{I} \to \mathbb{R}$ a boundary-induced distance function, and $0 < \varepsilon \leq 1$ a parameter. For any facet F of $\mathcal{I}$, let $\mathcal{S}(F) = (S_F, w_F)$ be a weighted set of points in F, such that $f(t) \leq V_{\mathcal{S}(F)}(t) \leq (1+\varepsilon)f(t)$, for all $t \in F$. Then $f(t) \leq V_{\mathcal{S}}(t) \leq (1+\varepsilon)f(t)$, for all $t \in \mathcal{I}$, where $\mathcal{S} = (S, w) = \cup_{F\in\phi(\mathcal{I})}\mathcal{S}(F)$.*

Proof: For any $t \in \mathcal{I}$, let x be the point in $\partial\mathcal{I}$ satisfying $f(t) = f(x) + |tx|$, and let F be a facet of $\mathcal{I}$ that contains x. By Lemma 2.7, we have

$$\begin{aligned} f(x) + |tx| = f(t) \leq V_{\mathcal{S}}(t) &\leq V_{\mathcal{S}}(x) + |tx| \leq V_{\mathcal{S}(F)}(x) + |tx| \leq (1+\varepsilon)f(x) + |tx| \\ &\leq (1+\varepsilon)(f(x) + |tx|) = (1+\varepsilon)f(t). \end{aligned}$$

■

Remark 2.22 Let $\mathcal{I}'$ be a set in $\mathbb{R}^d$, let $\mathcal{I}$ be a convex subset of $\mathcal{I}'$, $f : \mathcal{I}' \to \mathbb{R}$ a boundary-induced distance function, $0 < \varepsilon \leq 1$ a parameter, and s a point of $\mathcal{I}'$ such that $f(s) = 0$. Computing a weighted set $\mathcal{S} = (S, w)$ such that the weighted Voronoi diagram induced on $\mathcal{I}$ approximates f up to a factor of $(1+\varepsilon)$, can be done by applying the algorithm described in Remark 2.15 for each facet of $\mathcal{I}$. By Lemma 2.21, the union of all these weighted sets has the required properties.

3 Approximate Shortest-Path Map on a Polyhedral Surface in $\mathbb{R}^3$

Let $\mathcal{P}$ be a given polyhedral surface in $\mathbb{R}^3$ with n edges, let s be a source point on $\mathcal{P}$, and let $0 < \varepsilon \leq 1$ be a given parameter. In this section, we give an algorithm for constructing an approximation map on $\mathcal{P}$ of complexity $O((n/\varepsilon)\log(1/\varepsilon))$, such that given any $t \in \mathcal{P}$, one can compute in $O(\log(n/\varepsilon))$ time a distance $\Delta_{\mathcal{P}}(s,t)$ satisfying $d_{\mathcal{P}}(s,t) \leq \Delta_{\mathcal{P}}(s,t) \leq (1+\varepsilon)d_{\mathcal{P}}(s,t)$.

Although the following description is rather technical, one has to bear in mind that it is a straightforward implementation of the technique of Section 2. Namely, for each edge of our domain (polyhedral terrain, or a convex polytope) we compute a "small" set of points, we approximate the (geodesic) distance from the source point to all those points, and we construct the weighted additive Voronoi diagram that those points induce on each face of the domain.

Definition 3.1 A *polyhedral surface* $\mathcal{P}$ in $\mathbb{R}^3$ is the union of a collection of planar polygonal faces, with their edges and vertices, such that each edge is incident to at most two faces and any pair of faces intersect either at a common edge, a common vertex, or not at all. A face is a simple closed polygon (i.e., it contains its boundary), and an edge is a closed segment (i.e., it contains its endpoints). Without loss of generality we assume that all the faces are triangular (since simple polygons may be triangulated in linear time [Cha91] and the number of new edges introduced by the triangulation is linear in the number of vertices). We also assume that $\mathcal{P}$ is connected.

A *polyhedral terrain* is a polyhedral surface that intersects every vertical line in at most a single point.

Definition 3.2 Given a polyhedral surface $\mathcal{P}$ in $\mathbb{R}^3$, and any two points s, t on $\mathcal{P}$, we denote by $d_{\mathcal{P},s}(t)$ the length of a shortest path between s and t on $\mathcal{P}$.

As noted in Example 2.3 (i), $d_{\mathcal{P},s}(\cdot)$ is a distance function on $\mathcal{P}$. Moreover, if F is a face of $\mathcal{P}$ and $s \notin F$ then $d_{\mathcal{P},s}$ is boundary induced on F. (If $s \in F$ then $d_{\mathcal{P},s}(t)$ is the Euclidean distance $|st|$)

The following theorem is the main result of this section.

Theorem 3.3 *Let $\mathcal{P}$ be a polyhedral surface in $\mathbb{R}^3$ with n edges, s a source point on $\mathcal{P}$, and $0 < \varepsilon \leq 1$ a real parameter. Then there exists a subdivision Π of $\mathcal{P}$ of complexity $O((n/\varepsilon)\log(1/\varepsilon))$, which facilitates ε-approximate shortest path queries from s on $\mathcal{P}$. That is, for any query point t on $\mathcal{P}$, one can compute, in $O(\log(n/\varepsilon))$ time, a distance $\Delta_{\mathcal{P}}(s,t)$, such that $d_{\mathcal{P},s}(t) \leq \Delta_{\mathcal{P}}(s,t) \leq (1+\varepsilon)d_{\mathcal{P},s}(t)$.*

The map can be computed in $O(n^2\log n + (n/\varepsilon)\log(1/\varepsilon)\log(n/\varepsilon))$ time, if $\mathcal{P}$ is an arbitrary polyhedral surface, and in $O((n/\varepsilon^3)\log(1/\varepsilon) + (n/\varepsilon^{1.5})\log(1/\varepsilon)\log n)$ time, if $\mathcal{P}$ is a convex polytope.

The space used by the algorithm is $O((n/\varepsilon)\log(1/\varepsilon))$, if $\mathcal{P}$ is either a convex polytope or a polyhedral terrain, and $O(n^2 + (n/\varepsilon)\log(1/\varepsilon))$ otherwise.

Proof: For each face F of $\mathcal{P}$ that does not contain s, we construct a weighted Voronoi diagram that approximates $d_{\mathcal{P},s}$ on F. By Remark 2.22, this can be done by constructing a weighted Voronoi diagram on each edge of $\mathcal{P}$, as outlined in Remark 2.15.

For the general case, we compute the exact shortest-path map of s on $\mathcal{P}$, using the algorithm of [MMP87], in $O(n^2\log n)$ time. The exact map enables us to compute the shortest distance from s to any point of $\mathcal{P}$ in $O(\log n)$ time. Thus, computing the distances from s to the n hooks of the edges of $\mathcal{P}$ takes additional $O(n\log n)$ time. The hook point of an edge is the closest point of the edge to s, and it can be computed in $O(1)$ time.

For the convex case, we approximate the distances on $\mathcal{P}$ to all the hooks on the edges of $\mathcal{P}$, up to a factor of $(1+\varepsilon/8)$. This takes $O(n/\varepsilon^3 + (n/\varepsilon^{1.5})\log n)$ time, using the algorithm of [AHSV97, Sec. 6].

For each edge e of $\mathcal{P}$, we compute a set S_e of $O((1/\varepsilon)\log(1/\varepsilon))$ points on e, as specified in the proof of Lemma 2.12, taking the corresponding p to be the hook of e and w_p to be the approximated (exact in the non-convex case) distance along $\mathcal{P}$ from s to p. Let $S = \cup_e S_e$, taken over all edges e of $\mathcal{P}$.

We now compute (or approximate) the distances from s to all the points in S. For the non-convex case, this can be done in $O((n/\varepsilon)\log(1/\varepsilon)\log n)$ time, using the exact shortest path map.

For the convex case, we compute approximate distances from s to all points of S, up to a factor of $(1+\varepsilon/8)$. Using the observation of Remark 2.16, we partition S into $O(n\log(1/\varepsilon))$ sets, each of size $O(1/\varepsilon)$, such that the required distances to the points in each such set are within a fixed constant factor of each other (namely, for each edge e of $\mathcal{P}$, the set S_e is decomposed into $O(\log(1/\varepsilon))$ sets, as in the proof of Lemma 2.12). Using the algorithm described in Remark 3.5 below, we can compute the distances from s to all the points of S in $O((n/\varepsilon^3)\log(1/\varepsilon) + (n/\varepsilon^{1.5})\log(1/\varepsilon)\log n)$ time.

Next, we compute, for each face F of $\mathcal{P}$, the weighted Voronoi diagram induced by the weighted points of $\mathcal{S}$ that lie on ∂F. This takes $O((n/\varepsilon)\log^2(1/\varepsilon))$ overall time (see [For87]), since each face contains $O((1/\varepsilon)\log(1/\varepsilon))$ points of $\mathcal{S}$. Let Π be the resulting map, consisting of the union of all those facial Voronoi diagrams.

By Lemma 2.12 and Lemma 2.21, the map Π on $\mathcal{P}$ has the required properties. Moreover, we can preprocess each face F of $\mathcal{P}$ in $O((1/\varepsilon)\log^2(1/\varepsilon))$ time, such that point location queries on F can be answered in $O(\log(1/\varepsilon))$ time (see [O'R94]). Overall, this preprocessing takes $O((n/\varepsilon)\log^2(1/\varepsilon))$ time.

To answer a approximate shortest path query for a query point q, the algorithm must locate the face of $\mathcal{P}$ containing q. If $\mathcal{P}$ is a polyhedral terrain, we project the terrain into the xy-plane and preprocess it, in $O(n\log n)$ time, for planar point location. If $\mathcal{P}$ is a convex polytope, it can be preprocessed in linear time to answer point location queries in $O(\log n)$ time (see [DK85]). Otherwise, we preprocess $\mathcal{P}$ for spatial point-location in $O(n^2\log n)$ time, and $O(n^2)$ space, with $O(\log n)$ query time, using the algorithm of [THI90].

Given any query point q on $\mathcal{P}$, the algorithm computes the face F of $\mathcal{P}$ that contains q in $O(\log n)$ time. Locating the face of the subdivision Π that contains q takes an additional $O(\log(1/\varepsilon))$ time. Thus, ε-approximate shortest path queries for $\mathcal{P}$ can be answered in $O(\log(n/\varepsilon))$ time. (If the face containing q is already known, the query time reduces to $O(\log(1/\varepsilon))$.) ∎

Definition 3.4 Let P be a convex body in $\mathbb{R}^3$. An *outer path* of P is a curve γ connecting two points on ∂P and disjoint from the interior of P.

Remark 3.5 Let P be a convex polytope in $\mathbb{R}^3$, s a source point on P, T a set of points on P, and $0 < \varepsilon \leq 1$ a prescribed parameter. One can ε-approximate the length of the shortest path from s to all the points of T on P, in $O((n+|T|)/\varepsilon^3 + ((n+|T|)/\varepsilon^{1.5})\log(n+|T|))$ time, by adding the points of T as vertices to P and by using the algorithm of [AHSV97, Sec. 6].

The algorithm of [AHSV97] works by computing an approximation polytope for each point of T, and by computing the exact distance from s to the point on this polytope.

Moreover, suppose that T can be partitioned into m sets $T_1, \ldots, T_m$, such that $d_{P,s}(t) \leq c \cdot d_{P,s}(t')$, for all $t, t' \in T_i$ and for each $i = 1, \ldots, m$, where c is a prescribed constant, and all the points of T_i belong to the same edge of P, for any fixed $i = 1, \ldots, m$. Then it is possible to speed up the above algorithm, as follows. Instead of constructing an approximation polytope for each destination point separately, we construct an approximation polytope that can be used to approximate the distances from s to all the points of T_i, for $i = 1, \ldots, m$. This is done by ensuring that all the points of T_i lie on the boundary of the approximation polytope calculated by the algorithm, which can be enforced by intersecting it with a supporting plane of P passing through the edge containing the points of T_i (adding at most one new face to the approximation polytope). We also need to use a more refined approximation polytope, so as to achieve the claimed error bound, but since c is a constant this does not change the asymptotic complexity of the algorithm. See [AHSV97] for the technical details.

This improves the running time to

$$O\left(n + \frac{m}{\varepsilon^3} + \frac{m}{\varepsilon^{1.5}} \log n + \frac{|T|}{\varepsilon^{1.5}}\right),$$

by constructing an approximation polytope for the points of T_i (in $O(\frac{m}{\varepsilon^{1.5}} \log n)$ time), computing the exact distance map from the source point on the approximation polytope (in $O(1/\varepsilon^3)$ time), and extracting the shortest path to each point of T_i, repeating all this, for $T_1, \ldots, T_m$. Moreover, for each point $t \in T_i$, the algorithm computes a polygonal outer path of P, made out of $O(1/\varepsilon^{1.5})$ segments, that realizes the approximated distance.

Remark 3.6 The algorithm of [MMP87] works for arbitrary polyhedral surfaces; in particular, it is not restricted to polyhedral terrains. Thus, the algorithm of Theorem 3.3 also works for general polyhedral surfaces.

Remark 3.7 For a convex polytope P with n edges in $\mathbb{R}^3$, one can compute an approximation map that can be used to compute an outer path that realizes the approximate distance. This is done by modifying the algorithm of Theorem 3.3, such that it stores an outer path from the source point to each of the constructed sites, where the outer path realizes its ε-approximate distance. Such a path is readily available from the procedure used to compute the approximate distance to the site, and the complexity of such a path is $O(1/\varepsilon^{1.5})$ (See [AHSV97]). The space needed to store the extended approximation map is $O(n/\varepsilon^{2.5} \log(1/\varepsilon))$, and the computation time remains $O((n/\varepsilon^3) \log(1/\varepsilon) + (n/\varepsilon^{1.5}) \log(1/\varepsilon) \log n)$.

The new map can be used to answer approximate shortest path queries, in $O(\log(n/\varepsilon))$ time, and also compute, in additional $O(1/\varepsilon^{1.5})$ time, an outer path of the convex polytope realizing this distance. Such an outer path can be projected onto the boundary of the convex polytope, in additional $O(n \log(1/\varepsilon) + 1/\varepsilon^3)$ time, resulting in a path on ∂P which is an ε-approximation to the shortest path, see [AHSV97]. Note however that the performance of the enhanced data structure is poorer both in terms of storage and query time.

4 Constructing Spatial Approximate Shortest-Path Maps in $\mathbb{R}^3$

Let $\mathcal{O}$ be a collection of pairwise-disjoint polyhedral obstacles in $\mathbb{R}^3$, s a source point in $\mathbb{R}^3 \setminus int \bigcup \mathcal{O}$, and $0 < \varepsilon \leq 1$ a parameter. In this section, we present an algorithm for preprocessing $\mathcal{O}$ such that for any point in $\mathbb{R}^3$ (or, more precisely, for any 'free' point that can be reached from s without penetrating into an obstacle) one can compute, in $O(\log(n/\varepsilon))$

time, a distance $\Delta_{\mathcal{O},s}(t)$ satisfying $d_{\mathcal{O},s}(t) \le \Delta_{\mathcal{O},s}(t) \le (1+\varepsilon)d_{\mathcal{O},s}(t)$, where $d_{\mathcal{O},s}(t)$ is the length of a shortest path between s and t that avoids the interiors of the obstacles.

The preprocessing time of the algorithm is roughly $O(n^4/\varepsilon^6)$, which shows that the problem of approximating the distance from a single source to all the 'free' points in $\mathbb{R}^3$, is not much harder (computationally) than approximating the distance between any specific pair of points (which can be done in roughly $O(n^2/\varepsilon^4)$ time, see [Cla87]). In fact, for a fixed source point and many destination points, our algorithm will actually be faster. The problem of computing the *exact* distance between two points in $\mathbb{R}^3$ among polyhedral obstacles is NP-hard, as shown by Canny and Reif [CR87], and the fastest available algorithms for this problem run in time that is exponential in the total number of obstacle vertices [RS94, Sha87, SS86].

Definition 4.1 Let $\mathcal{O}$ be a collection of pairwise-disjoint polyhedral obstacles with a total of n edges in $\mathbb{R}^3$, and s a source point in $FP(\mathcal{O}) = \mathbb{R}^3 \setminus int \bigcup_{O\in\mathcal{O}} O$. Let $FP(\mathcal{O}, s)$ denote the set of all points in $FP(\mathcal{O})$ that can be connected to s by a path that avoids the interiors of the obstacles of $\mathcal{O}$.

For any $t \in FP(\mathcal{O}, s)$, we denote, as above, by $d_{\mathcal{O},s}(t)$ the length of a shortest path between s and t, that avoids the interiors of the obstacles of $\mathcal{O}$.

As noted in Example 2.3 (ii), $d_{\mathcal{O},s}(\cdot)$ is a distance function over $FP(\mathcal{O}, s)$, and for any convex set $\mathcal{I} \subseteq FP(\mathcal{O}, s)$ such that $s \notin \mathcal{I}$, the function $d_{\mathcal{O},s}(\cdot)$ is boundary induced over $\mathcal{I}$.

Theorem 4.2 (Clarkson [Cla87]) *Given a set $\mathcal{O}$ of polyhedral obstacles in $\mathbb{R}^3$, and points s and t, an ε-approximate path between s and t that does not penetrate into any obstacle in $\mathcal{O}$ can be computed in*

$$O\left(\frac{n^2}{\varepsilon^4}\beta(n)\log\frac{n}{\varepsilon} + n^2\log(n\rho)\log(n\log\rho)\right)$$

time, where n is the number of obstacle edges, and ρ is the ratio of the length of the longest edge in $\mathcal{O}$ to the Euclidean distance between s and t, $\beta(n) = \alpha(n)^{O(\alpha(n))^{O(1)}}$, and $\alpha(n)$ is the inverse of the Ackermann function.

The following theorem is the main result of this section.

Theorem 4.3 *Let $\mathcal{O}$ be a collection of pairwise-disjoint polyhedral obstacles with n edges in $\mathbb{R}^3$, s a source point in $FP(\mathcal{O})$, and $0 < \varepsilon < 1$ a parameter. Then a subdivision $\mathcal{M}$ of $FP(\mathcal{O}, s)$ of complexity $O(n^2/\varepsilon^{4+\delta})$, for any $1 > \delta > 0$, can be computed in*

$$O\left(\frac{n^4}{\varepsilon^2}\left(\frac{\beta(n)}{\varepsilon^4}\log\frac{n}{\varepsilon} + \log(n\rho)\log(n\log\rho)\right)\log\frac{1}{\varepsilon}\right)$$

time, where ρ, and $\beta(n)$ are as above.

For any query point $t \in FP(\mathcal{O}, s)$, one can compute in $O(\log(n/\varepsilon))$ time a distance $\Delta_{\mathcal{O},s}(t)$, such that $d_{\mathcal{O},s}(t) \le \Delta_{\mathcal{O},s}(t) \le (1+\varepsilon)d_{\mathcal{O},s}(t)$.

Proof: First, we partition $FP(\mathcal{O})$ into $O(n^2)$ vertical prisms. This can be easily done by erecting a vertical wall from each edge of the obstacles. For an edge e of the obstacles, such a wall is the set of all points in $FP(\mathcal{O})$ that lie on vertical rays emanating from the edge, and not intersecting the obstacles. Let $\mathcal{M}'''$ denote the resulting partition of $FP(\mathcal{O})$. It is easy to verify that the complexity of $\mathcal{M}'''$ is $O(n^2)$, and that it can be computed in $O(n^2 \log n)$ time.

We refine $\mathcal{M}'''$, by further partitioning each cell of $\mathcal{M}'''$ into vertical triangular prisms. This is done by projecting each cell of $\mathcal{M}'''$ into the xy-plane, and by triangulating the resulting polygon, in $O(m \log m)$ time (see [O'R94]), where m is the number of vertices of the polygon. For each new edge created, we erect a corresponding vertical wall inside the cell. Let $\mathcal{M}''$ be the resulting subdivision of $FP(\mathcal{O})$. Clearly, the complexity of $\mathcal{M}''$ remains $O(n^2)$, and it can be computed in additional $O(n^2 \log n)$ time.

Let T_v be the vertical prism in $\mathcal{M}''$ that contains s. We construct an adjacency graph G on the vertical prisms of $\mathcal{M}''$. By computing the connected component of G that contains T_v, one obtains the subdivision $\mathcal{M}' = \mathcal{M}'' \cap FP(\mathcal{O}, s)$.

Each cell $\mathcal{I}$ of $\mathcal{M}'$ is a vertical prism, having at most 5 faces. We can approximate the distance function $d_{\mathcal{O},s}(t)$ inside $\mathcal{I}$ by computing a weighted set $\mathcal{S}_\mathcal{I} = (S_\mathcal{I}, w_\mathcal{I})$, as specified in the proofs of Lemma 2.12 and Lemma 2.21. To do so, it is necessary to $(\varepsilon/8)$-approximate the value of $d_{\mathcal{O},s}(\cdot)$ for $O((1/\varepsilon^2)\log(1/\varepsilon))$ points (i.e., the points of $\mathcal{S}_\mathcal{I}$). By Theorem 4.2, this takes

$$O\left(\left(\frac{\beta(n)}{\varepsilon^4}\log\frac{n}{\varepsilon} + \log(n\rho)\log(n\log\rho)\right)\frac{n^2}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$$

time. The weighted Voronoi diagram $\mathcal{V}_{\mathcal{S}_\mathcal{I}}$ induced by $\mathcal{S}_\mathcal{I}$ inside $\mathcal{I}$ approximates $d_{\mathcal{O},s}$ inside $\mathcal{I}$ up to a factor of $1+\varepsilon$.

Let $\mathcal{S} = \cup_{\mathcal{I}\in\mathcal{M}'}\mathcal{S}_\mathcal{I}$. Clearly, one can $(\varepsilon/8)$-approximate the distance between s and all the sites of $\mathcal{S}$ in

$$O\left(\frac{n^4}{\varepsilon^2}\left(\frac{\beta(n)}{\varepsilon^4}\log\frac{n}{\varepsilon} + \log(n\rho)\log(n\log\rho)\right)\log\frac{1}{\varepsilon}\right)$$

time.

Let $\mathcal{M}$ be the subdivision $\bigcup_{\mathcal{I}\in\mathcal{M}'}(\mathcal{V}_{\mathcal{S}_\mathcal{I}} \cap \mathcal{I})$. We preprocess $\mathcal{M}$ for spatial point location, by constructing a two-level spatial point location data structure. First, we preprocess $\mathcal{M}'$ for point location in $O(n^2 \log n)$ time, using the algorithm of [THI90]. Next, we preprocess each cell $\mathcal{I}$ of $\mathcal{M}'$ for nearest neighbor queries for the weighted set $\mathcal{S}_\mathcal{I}$. By Lemma 4.5 below, performing this preprocessing for all the cells of $\mathcal{M}'$ takes a total of $O\big(n^2/\varepsilon^{4+\delta}\big)$ randomized expected time and space, for any $\delta > 0$.

For any query point $t \in FP(\mathcal{O}, s)$, we can compute in $O(\log n + \log(1/\varepsilon)) = O(\log(n/\varepsilon))$ time, the cell of $\mathcal{M}$ that contains t; that is, in $O(\log(n/\varepsilon))$ time, one can compute a distance $\Delta_{\mathcal{O},s}(t)$, such that $d_{\mathcal{O},s}(t) \le \Delta_{\mathcal{O},s}(t) \le (1+\varepsilon)d_{\mathcal{O},s}(t)$. ■

To complete the description and analysis of the algorithm, we next show how to preprocess a weighted set in $\mathbb{R}^3$ so that one can perform efficient nearest neighbor queries in the additive weighted Voronoi diagram that it induces.

Definition 4.4 Let $\mathcal{S} = (S, w)$ be a weighted set in $\mathbb{R}^3$. We decompose the weighted Voronoi diagram $\mathcal{V}_\mathcal{S}$ into "simpler" cells in the following way: For each cell C in $\mathcal{V}_S$, we compute the spherical map S_C of the cell, by projecting the boundary of the cell onto the sphere of directions centered at p_C, where p_C is the site of C in S. (We use here the well-known property that C is star-shaped with respect to p_C.) We decompose S_C into pseudo-vertical subcells on the sphere of directions, by drawing a meridian arc upwards and downwards from each vertex S_C, and from each locally longitude-extremal point on any arc of S_C, and by extending each of these meridian arcs until it hits another arc of S_C or, failing this, all the way to the poles of the sphere of directions. Clearly, the complexity of S_C is linear in the complexity of the cell C.

We project each "vertical" trapezoid in S_C back into C, to obtain the portion within C of the cone with apex p_C spanned by the trapezoid. This defines a decomposition of C into simple subcells, such that each subcell is uniquely defined by at most 6 points of S. We decompose all the cells of $\mathcal{V}_S$ in a similar manner, and let $\mathcal{C}(\mathcal{S})$ denote the resulting subdivision. We call $\mathcal{C}(\mathcal{S})$ the *spherical decomposition* of $\mathcal{V}_\mathcal{S}$.

For a weighted set $\mathcal{R} \subseteq \mathcal{S}$ and a subcell $\mathcal{T} \in \mathcal{C}(\mathcal{R})$, a weighted point $(p, w_p) \in \mathcal{S}$ *conflicts with* $\mathcal{T}$ if there exists a point $t \in \mathcal{T}$, such that $V_{(p,w_p)}(t) < V_\mathcal{R}(t)$. Let $K(\mathcal{S}, \mathcal{T})$ denote the set of all the points of $\mathcal{S}$ that conflict with $\mathcal{T}$. The *conflict size* of $\mathcal{T}$ is $w(\mathcal{S}, \mathcal{T}) = |K(\mathcal{S}, \mathcal{T})|$.

Lemma 4.5 *Let $\mathcal{S} = (S, w)$ be a weighted set of m points in $\mathbb{R}^3$, and $\delta > 0$ be a parameter. One can compute, in $O(m^{2+\delta})$ randomized expected time, a data structure for nearest-neighbor queries, of size $O(m^{2+\delta})$, such that for any point $p \in \mathbb{R}^3$, one can compute, in $O(\log m)$ time, the cell of $\mathcal{V}_\mathcal{S}$ that contains p; that is, the point in $\mathcal{S}$ realizing the distance $V_\mathcal{S}(p)$.*

Proof: We construct the data-structure using a randomized divide and conquer algorithm. We randomly pick a subset R of S of size r, where r is a parameter to be specified later. One can compute the weighted Voronoi diagram of $\mathcal{R} = (R, w)$, in $O(r^2)$ time, by Remark 2.5, and construct the spherical decomposition $\mathcal{C}(\mathcal{R})$ in $O(r^2 \log r)$ additional time, using plane sweeping techniques on the sphere of directions (see [O'R94]).

For each subcell $\mathcal{T}$ in $\mathcal{C}(\mathcal{R})$, we compute its conflict size $w(\mathcal{S}, \mathcal{T})$. Each subcell in $\mathcal{C}(\mathcal{R})$ is uniquely defined by at most 6 sites in $\mathcal{R}$, and if $K(\mathcal{S}, \mathcal{T}) \cap \mathcal{R} \neq \emptyset$ then $\mathcal{T} \notin C(\mathcal{R})$. We can thus apply the analysis of Clarkson and Shor. By [CS89, Corollary 3.8], $w(\mathcal{S}, \mathcal{T}) \leq c \cdot (n \log r)/r$, for all $\mathcal{T} \in \mathcal{C}(\mathcal{R})$, with probability at least $1/2$, where $c > 0$ is an appropriate constant. We sample $\mathcal{R}$ from $\mathcal{S}$ repeatedly until we get a sample that fulfills this condition. Overall, this stage takes $O(mr^2 + r^2 \log r)$ expected running time. For each cell $\mathcal{T} \in \mathcal{C}(\mathcal{R})$, we construct recursively a data-structure for point-location in the Voronoi diagram $\mathcal{V}_{K(\mathcal{S},\mathcal{T})}$.

For any query point p, locating the subcell $\mathcal{T}$ in $\mathcal{C}(\mathcal{R})$ that contains p is done by a brute force search inside $\mathcal{C}(\mathcal{R})$, in $O(r^2)$ time. Then, we compute the point realizing $V_\mathcal{S}(p)$ by recursively performing a nearest neighbor query in the data-structure computed for $\mathcal{V}_{K(\mathcal{S},\mathcal{T})}$. Thus, a query takes $Q(m) = Q(c(m \log r)/r) + O(r^2)$ time, and the data-structure can be computed, in randomized expected time

$$T(m) = T(r) + O(r^2) T\left(\frac{cm \log r}{r}\right) + O\left(mr^2 + r^2 \log r\right).$$

Choosing r to be a sufficiently large constant, we have $Q(m) = O(\log m)$, and $T(m) = O(m^{2+\delta})$ (where the constants of proportionality depend on δ). A similar bound holds for the space required by the algorithm. ■

Remark 4.6 The only stage in the algorithm of Theorem 4.3 that uses randomization is the construction of the spatial point-location data described in Lemma 4.5. This can be replaced by a deterministic data-structure as follows.

We observe that each spherical cell, in the decomposition described above, can be parametrized by 24 parameters (6 sites and their respective weights). Thus, we define a range space $(\mathcal{S}, \mathfrak{R})$, where $\mathfrak{R}$ is the set of all possible subsets of $\mathcal{S}$ that are contained inside such a spherical cell. It is easy to verify that this is a range space having finite VC-dimension. By a result of Matoušek [Mat95], we can compute, in $O(mr^{O(1)})$ time, a subset $\mathcal{R}$ of $\mathcal{S}$ having $O(r \log r)$ points, which is $(1/r)$-net of $(\mathcal{S}, \mathfrak{R})$. In particular, the set $\mathcal{R}$ can replace the random sample in the proof of Lemma 4.5, see [Mat95]. This yields a deterministic algorithm with the same time/space complexity as in Lemma 4.5.

Alternatively, one can naively preprocess the Voronoi diagram $\mathcal{V}_{\mathcal{S}}$ for spatial point-location directly, see [THI90]. However, this approach is considerably less efficient than the approach proposed above.

5 Conclusions

In this paper we have presented two results for computing approximate maps that facilitate shortest paths queries on the surface of a convex polytope or on a polyhedral surface in 3-space, or among polyhedral obstacles in 3-space. We conclude by mentioning the following open problems.

- Can an ε-approximate shortest path between two points on a polyhedral terrain, or on the surface of a nonconvex polyhedron, be computed in time that is near-linear in the number of edges? A recent subquadratic solution has been obtained by Varadarajan and Agarwal [VA97], but it only computes a constant-factor approximation to the shortest path.
- Can the exact shortest path between two points on a convex polyhedron be computed in near-linear time? in subquadratic time?
- Can the methods and techniques used in this paper be extended to handle shortest path queries for weighted surfaces (as in [LMS97, MM97])?

Acknowledgments

The author wishes to thank Pankaj Agarwal and Micha Sharir for helpful discussions concerning the problems studied in this paper and related problems. Micha Sharir has also suggested Lemma 4.5. The author also wishes to thank the referees for their comments and suggestions.

References

[AHSV97] 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.

[Aur91] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[CH96] J. Chen and Y. Han. Shortest paths on a polyhedron; Part I: computing shortest paths. *Int. J. Comput. Geom. & Appl.*, 6(2):127–144, 1996.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[Cla87] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 56–65, 1987.

[CR87] J. F. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 49–60, 1987.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[CSY94] J. Choi, J. Sellen, and C. K. Yap. Approximate Euclidean shortest path in 3-space. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 41–48, 1994.

[DK85] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. Algorithms*, 6:381–392, 1985.

[For87] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[Har99] S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *Discrete Comput. Geom.*, 21:216–231, 1999.

[HS95] J. Hershberger and S. Suri. Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$. In *Proc. 6th ACM-SIAM Sympos. Discrete Algorithms*, pages 447–456, 1995.

[LMS97] M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 274–283, 1997.

[Mat95] J. Matoušek. Approximations and optimal geometric divide-and-conquer. *J. Comput. Syst. Sci.*, 50(2):203–208, 1995.

[MM97] C. Mata and J. S.B. Mitchell. A new algorithm for computing the shortest paths in planar subdivisions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 264–273, 1997.

[MMP87] J. S.B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[Mou87] D. M. Mount. Storing the subdivision of a polyhedral surface. *Discrete Comput. Geom.*, 2:153–174, 1987.

[O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, Cambridge, 1994.

[Pap85] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Inform. Process. Lett.*, 20:259–263, 1985.

[RS94] J. H. Reif and J. A. Storer. A single-exponential upper bound for finding shortest paths in three dimensions. *J. Assoc. Comput. Mach.*, 41(5):1013–1019, 1994.

[Sha87] M. Sharir. On shortest paths amidst convex polyhedra. *SIAM J. Comput.*, 16:561–572, 1987.

[SS86] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986.

[THI90] X.-H. Tan, T. Hirata, and Y. Inagaki. Spatial point location and its applications. In *Proc. 1st Annu. SIGAL Internat. Sympos. Algorithms*, volume 450 of *Lect. Notes in Comp. Sci.*, pages 241–250. Springer-Verlag, 1990.

[VA97] K. R. Varadarajan and P. K. Agarwal. Approximating shortest paths on a non-convex polyhedron. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 182–191, 1997.