Sublinear Geometric Algorithms*

Bernard Chazelle
Department of Computer Science
Princeton University
and NEC Laboratories America, Inc.
chazelle@cs.princeton.edu

Ding Liu
Department of Computer Science
Princeton University
dingliu@cs.princeton.edu

Avner Magen
Department of Computer Science
University of Toronto
avner@cs.toronto.edu

ABSTRACT

We initiate an investigation of sublinear algorithms for geometric problems in two and three dimensions. We give optimal algorithms for intersection detection of convex polygons and polyhedra, point location in two-dimensional Delaunay triangulations and Voronoi diagrams, and ray shooting in convex polyhedra, all of which run in time $O(\sqrt{n}\,)$, where n is the size of the input. We also provide sublinear solutions for the approximate evaluation of the volume of a convex polytope and the length of the shortest path between two points on the boundary.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations; I.3.5 [**Computational Geometry and Object Modeling**]: Geometric algorithms, languages, and systems

General Terms

Algorithms theory

Keywords

Sublinear algorithms, approximate shortest paths, polyhedral intersection

1. INTRODUCTION

An outgrowth of the recent work on property-testing, the study of sublinear algorithms has emerged as a field unto itself and great strides have been made in the context of graph and combinatorial problems [27]. Large geometric datasets often call for algorithms that examine only a small fraction of the input, but it is fair to say that sublinear computational geometry is still largely unchartered territory. If preprocessing is allowed, then of course this is an entirely different story [3, 20]. For example, checking whether a point lies in a convex 3-polyhedron can be done in logarithmic time with linear preprocessing. However, little of this technology is of any use with massive datasets, since examining the whole input—let alone preprocessing it—is out of the question. Sublinear algorithms have been given for dynamic problems [15] or in situations where a full multidimensional data structure is available [9]. There has also been work on geometric property testing, both in an approximate [10, 11, 16] and exact [21] setting.

In this paper, sublinearity is understood differently. The input is taken to be in any standard representation with no extra assumptions. For example, a planar subdivision or a polyhedron is given in classical edge-based fashion (eg, DCEL, winged-edge), with *no extra preprocessing*. This implies that we can pick an edge at random in constant time, but we cannot sample randomly among the neighbors of a given vertex in constant time. Our motivation is two-fold: (i) we seek the minimal set of computational assumptions under which sublinearity is achievable; (ii) the assumptions should be realistic and nonrestrictive. Note, for example, that sublinear separation algorithms for convex objects are known [6, 13], but all of them require preprocessing, so they fall outside our model. Under these conditions one might ask whether there exist any interesting "offline" problems that can be solved in sublinear time. The answer is yes. Note that randomization is a necessity because, in a deterministic setting, most problems in computational geometry require looking at the entire input. There has been some (but little) previous work on sublinear geometric algorithms as we define them, specifically point location in two- and three-dimensional Delaunay triangulations of sets of random points [12, 24]. As far as we know, however, these are the only works that fall inside of our model. Here is a summary of our results. In all cases, n denotes the input size and all polyhedra are understood to be in $\mathbf{R}^3$:

- An optimal $O(\sqrt{n}\,)$ time algorithm for checking whether two convex polyhedra intersect. The algorithm reports an intersection point if they do and a separating plane if they don't.
- Optimal $O(\sqrt{n}\,)$ time algorithms for point location in

*This work was supported in part by NSF grant CCR-998817, ARO Grant DAAH04-96-1-0181, and NEC Research Institute.

STOC'03, June 9–11, 2003, San Diego, California, USA.

two-dimensional Delaunay triangulations and Voronoi diagrams, and ray shooting in convex polyhedra.

In contrast with property-testing, it is important to note that our algorithms never err. All the algorithms are of the Las Vegas type, and randomization affects the running time but not the correctness of the output.[1] In [12] Devroye, Mücke, and Zhu showed that a simple technique for point location in two-dimensional Delaunay triangulations, namely random sampling then walking from the nearest sample to the query, has expected running time $O(n^{1/3})$ for *n random* input points and a random query. This does not contradict the optimality of our $O(n^{1/2})$ bound because the points must be chosen randomly in [12].

We also consider optimization problems for which approximate solutions are sought. We give:

- An $O(\varepsilon^{-1}\sqrt{n}\,)$ time algorithm for approximating the volume of a convex polytope with arbitrary relative error $\varepsilon > 0$.
- An $O(\varepsilon^{-5/4}\sqrt{n}\,)+f(\varepsilon^{-5/4})$ time algorithm for approximating the length of the shortest path between two points on the boundary of a convex polyhedron with arbitrary relative error $\varepsilon > 0$. Here, $f(n)$ denotes the complexity of the *exact* version of problem. This implies that the complexity of our algorithm is $O(\sqrt{n}\,)$, for any fixed $\varepsilon > 0$.

The shortest path problem for polyhedral surfaces has been extensively studied, drawing its motivation from applications in route planning, injection molding, computer assisted surgery [1, 18, 23]. In the convex case (the one at hand), an $O(n^3 \log n)$ algorithm was given by Sharir and Schorr [29], later improved by Mitchell et al. [22] to $O(n^2 \log n)$ and by Chen and Han [7] to $O(n^2)$; therefore, it is known that $f(n) = O(n^2)$. More recently, Kapoor [19] has announced a proof that $f(n) = O(n \log^2 n)$, which would make our algorithm run in time $O(\varepsilon^{-5/4}\sqrt{n}\,)$. This improves on Agarwal et al.'s algorithm [2], which runs in $O(n \log \varepsilon^{-1} + \varepsilon^{-3})$ time , for any $\varepsilon > 0$.

Our method makes progress on an important geometric problem of independent interest.

- Given a convex polytope P of n vertices, how many vertices must an enclosing polytope Q have if it is to approximate any (large enough) shortest path on ∂P with relative error at most ε? We reduce to $O(\varepsilon^{-5/4})$ the best previous bound of $O(\varepsilon^{-3/2})$, due to Agarwal et al. [2].

A Flavor of the Techniques

As a warmup exercise, consider the classical *successor searching* problem: Given a sorted (doubly-linked) list of n keys and a number x, find the smallest key $y \geq x$ (the *successor* of x) in the list or report that none exists. Although we could not find a reference, the following algorithm is probably folklore. Choose $\sqrt{n}$ list elements at random, and find the predecessor and successor of x among those (perhaps only one exists). This provides an entry point into the list, from which a naive search takes us to the successor. To make random sampling possible, we may assume that the list element are stored in consecutive locations (say, in a table). However—and this is the key point—no assumption is made on the ordering of the elements in the table (otherwise we could do a binary search).

Lemma 1.1. *Successor searching can be done in $O(\sqrt{n}\,)$ expected time per query, which is optimal.*

Proof. Let Q_c be the set of all elements that are at most $c\sqrt{n}$ away from the answer on the list (in either direction). The probability of not hitting Q_c after $\sqrt{n}$ random choices of the list elements is $2^{-\Omega(c)}$ and so the expected distance of the answer to the random sample is $O(\sqrt{n}\,)$. This immediately implies that the expected time of the algorithm is $O(\sqrt{n}\,)$.

For the lower bound, we use Yao's minimax principle. We fix a distribution on the input, and we lower-bound the expected complexity of any deterministic algorithm. The input is a linked list containing the numbers 1 through n in sorted order. In our model, the list is represented by a table $T[1 \cdots n]$, with the i-th element in the list stored in location $\sigma(i)$ of the table; hence, $T[\sigma(i)] = i$. The input distribution is formed by choosing the permutation σ uniformly from the symmetric group on n elements. The query is set to be n. In other words, the problem is to locate the last element in the list. A deterministic algorithm can be modeled as a sequence of steps of the form: (A) pick a location $T[k]$ already visited and look up the next (or previous) item, ie, $T[\sigma(i \pm 1)]$, where $k = \sigma(i)$; (B) compute a new index k and look up $T[k]$. Each step may involve the consideration of every piece of information gathered so far. In particular, in a B-step we may not consult either one of the adjacent items in the list before computing k (unless, of course, these items were visited earlier). In this way, $\sigma^{-1}(k)$ is equally likely to lie anywhere in the portion of the list still unvisited. For this reason, after a A-steps and b B-steps, there is a probability at least $\left(1 - \frac{\sqrt{n}+a+b}{n}\right)^b$ that none of the last $\sqrt{n}$ elements in the list has been visited in a B-step. Right after the last B-step, either the total number of A- and B-steps exceeds $\sqrt{n}$ or, with constant nonzero probability, at least $\sqrt{n}$ A-steps (some of which may have already been taken) are required to reach the last element in the list. This immediately implies that the expected time of any deterministic algorithm is $\Omega(\sqrt{n}\,)$ $\Box$

We can generalize these ideas to polygon intersection. Given two convex polygons P and Q, with n vertices each, determine whether they intersect or not and, if they do, report one point in the intersection. Note that if one polygon consists of a single point, then it is easy to express the problem as successor searching and solve with $O(\sqrt{n}\,)$ CCW tests. Conversely, it is trivial to embed any successor problem as a convex polygon intersection problem. This shows that $\Theta(\sqrt{n}\,)$ is the correct bound in a model where the answer must be not just yes/no, but the address of the list node containing an intersected polygon edge where the intersection takes place.

Choose a random sample of r vertices from each polygon, and let $R_p \subseteq P$ and $R_q \subseteq Q$ denote the two corresponding convex hulls. By linear programming, we can test R_p and R_q for intersection without computing them explicitly. This can be done probabilistically (or even deterministically) in

[1] Throughout this paper, unless specified otherwise, the running times are understood in the expected sense.

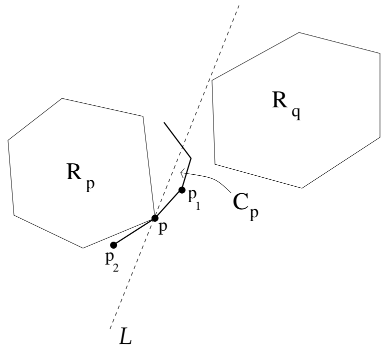

Figure 1: Intersecting two convex polygons

linear time. There are many ways of doing that (see [5] for references). It is easy to modify the algorithm (of, say, [28]) in $O(r)$ time so that it reports a point of intersection if there is one, and a bi-tangent separating line $\mathcal{L}$ otherwise (Figure 1). Let p be the vertex of R_p in $\mathcal{L}$, and let p_1, p_2 be their two adjacent vertices in P. If neither of them is on the R_q side of $\mathcal{L}$, then define C_p to be the empty polygon. Otherwise, by convexity exactly one of them is; say, p_1. We walk along the boundary of P starting at p_1, away from p, until we cross $\mathcal{L}$ again. This portion of the boundary, clipped by the line $\mathcal{L}$, forms a convex polygon, also denoted by C_p. A similar construction for Q leads to C_q.

It is immediate that $P \cap Q \neq \emptyset$ if and only if P intersects C_q or Q intersects C_p. We check the first condition and, if it fails, check the second one. We restrict our explanation to the case of $P \cap C_q$. First, we check whether R_p and C_q intersect, again using a standard linear time algorithm, and return with an intersection point if they do. Otherwise, we find a line $\mathcal{L}'$ that separates R_p and C_q and, using the same procedure described above, we compute the part of P, denoted C'_p, on the other side of $\mathcal{L}'$. Finally, we test C'_p and C_q for intersection in linear time. Correctness is immediate. The running time is $O(\sqrt{n} + |C_p| + |C'_p| + |C_q| + |C'_q|)$. It follows from a standard union bound that $\mathbf{E}\,|C_p| = O(n/r)\log n$, but a more carefuly analysis shows that in fact $\mathbf{E}\,|C_p| = O(n/r)$. (The three-dimensional case discussed below will subsume this result, so there is no need for a proof now.) The overall complexity of the algorithm is $O(r + n/r)$, and choosing $r = \lfloor \sqrt{n} \rfloor$ gives the desired bound of $O(\sqrt{n}\,)$.

THEOREM 1.2. *To check whether two convex n-gons intersect can be done in $O(\sqrt{n}\,)$ time, which is optimal.*

To put Theorem 1.2 in perspective, recall that the intersection of two convex polygons can be determined in logarithmic time if the vertices are stored in an array in cyclic order [6]. The key point of our result is that, in fact, a linked list is sufficient for sublinearity. Similarly, if polyhedra are preprocessed à la Dobkin-Kirkpatrick then fast intersection detection is possible [13]. What we show below is that sublinearity is achievable even with no preprocessing at all. Again, we use a two-stage process: In the first stage we break up the problem into r subproblems of size roughly n/r, and then identify which ones actually need to be solved; in the second stage we solve these subproblems in standard (ie, non-sublinear) fashion. Their number is constant; hence the square root complexity. What prevents us from solving these subproblems recursively is the model's restriction to *global* random sampling. In other words, one can sample efficiently for the main problem but *not* for the subproblems.

2. CONVEX POLYHEDRAL INTERSECTIONS

Given two n-vertex convex polyhedra P and Q in $\mathbf{R}^3$, determine whether they intersect or not and, if they do, report one point in the intersection. Choose a random sample of $r = \lfloor \sqrt{n} \rfloor$ edges from each one, and let R_p and R_q denote their respective convex hulls. We do not compute them. Instead, by linear programming, we test R_p and R_q for intersection in $O(r)$ time [5, 28]. We stop with a point of intersection if there is one. Otherwise, we find a separating plane $\mathcal{L}$ that is tangent to both R_p and R_q. It is important to choose the plane $\mathcal{L}$ in a canonical fashion. To do that, we set up the linear program so as to maximize, say, the coefficient a in the equation[2] $ax + by + cz = 1$ of $\mathcal{L}$.

Next, choose a plane π normal to $\mathcal{L}$ and consider projecting P and Q onto it. (Of course, we do not actually do it.) Let p be a vertex of R_p in $\mathcal{L}$ (there could be two of them, but not more if we assume general position between P and Q) and let p^* be its projection onto π. Let $p_1^*, p_2^*, \ldots, p_k^*$ be the set of vertices adjacent to p^* in the projection of P onto π. We test to see if any of them is on the R_q side of $\mathcal{L}$, and identify one such point, p_1, if the answer is yes (more on that below). If none of them is on the R_q side, then we define C_p to be the empty polyhedron. This is because in this case, P is completely on the other side of $\mathcal{L}$. Otherwise, we construct the portion of P, denoted C_p, that lies on the R_q side of $\mathcal{L}$. Note that C_p is a convex polytope, not just the boundary of P cut off by $\mathcal{L}$. We compute C_p by using a standard flooding mechanism. Beginning at p_1, we perform a depth-first search through the facial structure of P, restricted to the relevant side of $\mathcal{L}$. Because C_p is convex, the edges form a single connected component, so we never need to leave C_p. This allows us to build the entire facial representation of C_p in time proportional to its number of edges. From then on, the algorithm has the same structure as its polygonal counterpart, ie, we compute C_p, C'_p, C_q, C'_q and perform the same sequence of tests.

The question is now: how do we find p_1 (if it exists)? To simplify the analysis, once we have p, we resample by picking r edges in P at random; let E be the subset of those incident to p. To find p_1, we project on π all of the edges of E. If there exists an edge of E that is on the R_q side of $\mathcal{L}$, then we identify its endpoint as p_1. Otherwise all the edges of E lie on one side of $\mathcal{L}$. We then identify the two extreme ones (e and f in Figure 2); being extreme means that all the other projected edges of E lie in the wedge between e and f in π. Assume that e and f are well defined and distinct. Consider the cyclic list V of edges of P incident to p. The edges of E break up V into blocks of consecutive edges. It is not hard to prove that pp_1 lies in blocks starting or ending with e or f, if such a p_1 (as defined above) exists. So, we examine each of these relevant blocks (at most four) exhaustively. If e and f are not both distinct and well defined, we may simply search for p_1 by checking every edge of P incident to p.

[2]With perturbation techniques, we can always assume general position, and hence avoid having a solution passing through the origin. We will also assume that the relative position of P and Q is general.

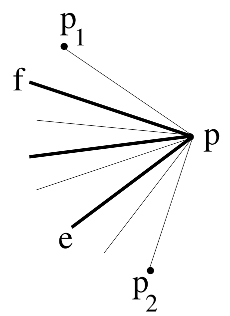

Figure 2: The edges of P incident to p; the thick lines form the random sample.

THEOREM 2.1. *Two convex n-vertex polyhedra in $\mathbf{R}^3$ can be tested for intersection in $O(\sqrt{n})$ time; this is optimal.*

PROOF. Optimality was already discussed in the polygonal case and correctness follows from elementary convex geometry, so we limit our discussion to the complexity of the algorithm. Because of the resampling, the expected sizes of the blocks next to e and f (or alternatively the expected size of the neighborhood of p if the blocks are not distinct) are trivially $O(n/r)$, so the running time is $O(r+n/r+\mathbf{E}\,|C_p|)$, where $|C_p|$ denotes the number of edges of C_p. We may exclude the other terms $|C'_p|$, $|C_q|$, and $|C'_q|$, since our upper bound on $\mathbf{E}\,|C_p|$ will apply to them as well. The naive bound of $O((n/r)\log n)$ on $\mathbf{E}\,|C_p|$ can be improved to $O(n/r)$. Here is how.

We modify the sampling distribution a little. Then we argue that reverting back to the original setting does not change the asymptotic value of the upper bound. The modification is two-fold: (i) we view P as a multiset M where each vertex appears as many times as its number of incident edges; (ii) R_p is formed by picking each point of M independently with probability r/n. With respect to the modified distribution, $|C_p|$ is proportional to the number of constraints in M that violate the linear program $\mathcal{P}(R_p, R_q)$ used to define $\mathcal{L}$ (with each point of R_p and R_q defining a linear constraint). Consider a subset $M' \subseteq M$ such that the solution of $\mathcal{P}(M', R_q)$ (if it exists) has exactly k violations in M. We distinguish between the solutions with one point in M and two in R_q and those with two points in M and one in R_q. (Assuming general position between P and Q, these are the only possibilities.) Let f_k and g_k count the number of solutions of the first and second type respectively, and let $f_{\le k} = f_0 + \cdots + f_k$ (with the same definition for g). For example, we have $f_0 + g_0 = 1$ and $f_{|M|} = g_{|M|} = 0$. We can prove by standard arguments [8, 25] that

$$f_{\le k} = O(k) \qquad \text{and} \qquad g_{\le k} = O(k^2). \tag{1}$$

To see why, form a random sample S by picking each point of M independently with probability s/n. Obviously, the number of solutions of $\mathcal{P}(S, R_q)$ is one; therefore, so is its expected value. This gives us

$$\sum_{j\le |M|} f_j\Big(\frac{s}{n}\Big)\Big(1-\frac{s}{n}\Big)^j + \sum_{j\le |M|} g_j\Big(\frac{s}{n}\Big)^2\Big(1-\frac{s}{n}\Big)^j = 1.$$

Choosing $s = n/k$ we have,

$$\sum_{j\le k} f_j\Big(\frac{s}{n}\Big)\Big(1-\frac{1}{k}\Big)^j + \sum_{j\le k} g_j\Big(\frac{s}{n}\Big)^2\Big(1-\frac{1}{k}\Big)^j \le 1.$$

it follows easily that

$$\Big(\frac{s}{n}\Big)f_{\le k} + \Big(\frac{s}{n}\Big)^2 g_{\le k} = O(1),$$

which proves (1).

Returning to our modified distribution, which assigns probability r/n to each point of M, we now have

$$\mathbf{E}\,|C_p| = O(1)\sum_{k\le |M|}\Big\{kf_k\Big(\frac{r}{n}\Big)\Big(1-\frac{r}{n}\Big)^k + kg_k\Big(\frac{r}{n}\Big)^2\Big(1-\frac{r}{n}\Big)^k\Big\}$$

The first $k_0 = O(n/r)$ summands add up to

$$\Big(\frac{r}{n}\Big)k_0 f_{\le k_0} + \Big(\frac{r}{n}\Big)^2 k_0 g_{\le k_0} = O\Big(\frac{n}{r}\Big).$$

Setting $u_k = k\Big(1-\frac{r}{n}\Big)^k$, we can use summation by parts to bound the contribution of the last $|M| - k_0$ summands. This gives an upper bound of

$$\Big(\frac{r}{n}\Big)\sum_{k_0\le k<|M|}\Big\{(u_k - u_{k+1})f_{\le k}\Big\} + \Big(\frac{r}{n}\Big)u_{|M|}f_{\le |M|}$$

$$+\Big(\frac{r}{n}\Big)^2\sum_{k_0\le k<|M|}\Big\{(u_k - u_{k+1})g_{\le k}\Big\} + \Big(\frac{r}{n}\Big)^2 u_{|M|}g_{\le |M|}.$$

By (1) and the inequality

$$u_k - u_{k+1} \le \frac{rk}{n}\,e^{-kr/n},$$

this is also bounded by

$$O\Big(\frac{r}{n}\Big)^2\sum_{k_0\le k<|M|}k^2 e^{-kr/n} + O\Big(\frac{r}{n}\Big)^3\sum_{k_0\le k<|M|}k^3 e^{-kr/n} + O(1)$$

$$= O\Big(\frac{n}{r}\Big).$$

This implies that

$$\mathbf{E}\,|C_p| = O\Big(\frac{n}{r}\Big). \tag{2}$$

Let $\mathcal{D}$ be the original distribution (the one used by the actual algorithm) with r replaced by $7r$. Of course, by (2) this scaling has no asymptotic effect on the upper bound for $\mathbf{E}\,|C_p|$. We define an intermediate distribution $\mathcal{D}_1$ by going through each edge (u,v) of P twice, selecting it with probability r/n, and then throwing into the sample both u and v, provided that the edge (u,v) has not been selected yet. (Note that this implies that u and v are kept out with probability $(1-r/n)^2$.) There are at most $3n$ edges in P, so the probability that a sample from $\mathcal{D}_1$ is of size less than $7r$ is overwhelmingly high. Since all equal-size subsets of edges are equally likely to be chosen, $\mathbf{E}_{\mathcal{D}}\,|C_p|$ is nonincreasing with the sample size, and so, $\mathbf{E}_{\mathcal{D}}\,|C_p| = O(\mathbf{E}_{\mathcal{D}_1}\,|C_p|)$. Let $\mathcal{D}_2$ denote the modified distribution used in the calculations. Observe that $\mathcal{D}_2$ is derived from $\mathcal{D}_1$ by picking only u if (u,v) is chosen the first time it is considered for selection, and then only v if it is picked the second time around. By monotonicity, we then have $\mathbf{E}_{\mathcal{D}_1}\,|C_p| = O(\mathbf{E}_{\mathcal{D}_2}\,|C_p|)$. This proves that (2) holds in the distribution used by the algorithm. $\square$

Within the same amount of time we can report a separating plane if the two convex polyhedra do not intersect and a point in the intersection otherwise. In fact, we can always find a point of intersection on the boundary of at least one of the polyhedra. From this it is immediate to derive the following:

COROLLARY 2.2. *Given a convex polyhedron with n vertices and a line, we can compute their intersection explicitly in optimal $O(\sqrt{n}\,)$ time.*

This allows us to perform ray shooting toward a convex polyhedron in sublinear time. This gives us useful ammunition for all sorts of location problems.

3. RAY SHOOTING APPLICATIONS

Given the Delaunay triangulation $\mathcal{T}$ of a set S of n points in the plane and a query point q, consider the problem of locating q, ie, retrieving the triangle of $\mathcal{T}$ that contains it. The Delaunay triangulation can be given in any classical edge-based data structure (e.g. DCEL) that supports $O(1)$ time access to a triangle from a neighboring triangle. We use the close relationship between Delaunay triangulations and convex hulls given by the mapping $h : (x, y) \mapsto (x, y, x^2+y^2)$. As is well known, the Delaunay triangulation of S is facially isomorphic to the lower hull of $h(S)$ (ie, the part of the convex hull that sees $z = -\infty$). In this way, point location in $\mathcal{T}$ is equivalent to ray shooting towards the convex hull, where the ray originates from the query point q and shoots in the positive z direction. Obviously, any facial feature of the convex hull can be retrieved in constant time from its corresponding feature in the Delaunay triangulation. (The one exception is the set of faces outside the lower hull: we can simplify matters by adding a dummy vertex to the hull at $z = \infty$.)

The same argument can be used for point location in Voronoi diagrams. Recall that each point (p_x, p_y) is now lifted to the plane $Z = 2p_xX + 2p_yY - (p_x^2 + p_y^2)$, which is tangent to the paraboloid $Z = X^2 + Y^2$. The Voronoi diagram of S is isomorphic to the lower envelope of the arrangement formed by the n tangent planes. Note that any vertex (resp. edge) of the envelope can be derived in constant time from the three (resp. two) faces incident to the corresponding vertex (resp. edge).

THEOREM 3.1. *Point location in the Delaunay triangulation or Voronoi diagram of n points in the plane can be done in optimal $O(\sqrt{n}\,)$ time.*

We consider the following problem, which will arise in our subsequent discussion of volume approximation and shortest paths algorithms. Given a convex polyhedron P with n vertices and a point q, let $n_P(q)$ denote the (unique) point of P that is closest to q. Of course, we can assume that q does not lie inside P, which we can test by using the previous algorithm. To compute $n_P(q)$ we extract a sample polyhedron R_p, as we did before, and find $n_{R_p}(q)$ by exhaustive search. This also gives us a plane $\mathcal{L}$ tangent to R_p at $n_{R_p}(q)$ and normal to the segment $qn_{R_p}(q)$. Next, we compute the intersection C_p of P with the halfspace bounded by $\mathcal{L}$ that contains q. We already explained how to do that in the previous section. In fact, a similar analysis shows that the expected size of C_p is just $O(n/r)$. Obviously, $n_P(q) = n_{C_p}(q)$, so we can finish the work by exhaustive search in C_p. The entire algorithm takes time $O(r+n/r)$, so we set $r = \lfloor \sqrt{n}\, \rfloor$.

THEOREM 3.2. *Given a convex polyhedron P with n vertices and a point q, the nearest neighbor of q in P can be found in $O(\sqrt{n}\,)$ time.*

We can compute a related function by similar means. Given a directed line ℓ, consider an orthogonal system of coordinates with ℓ as one of its axes (in the positive direction), and define $\xi_P(\ell)$ to be any point of P with maximum ℓ-coordinate. If we choose a point q at infinity on ℓ, then $\xi_P(\ell)$ can be chosen as $n_P(q)$, and so we can apply Theorem 3.2. Another function we can compute in this fashion maps a plane $\mathcal{L}$ and a direction ℓ in $\mathcal{L}$ to the furthest point of P in $\mathcal{L}$ along ℓ: in other words, $\xi_P(\mathcal{L}, \ell) = \xi_{\mathcal{P}\cap\mathcal{L}}(\ell)$. We summarize our results.

COROLLARY 3.3. *Given a convex polyhedron P with n vertices, a directed line ℓ, and a plane π, the points $\xi_P(\ell)$ and $\xi_P(\pi, \ell)$ can be found in $O(\sqrt{n}\,)$ time.*

4. VOLUME APPROXIMATION

We seek to approximate the volume of a convex polytope P. We proceed in two stages. First, we compute a large enough enclosed ellipsoid, which we use to rescale P affinely. This is intended to make P *round* enough so that good Hausdorff distance approximation yields good volume approximation. Second, we use a standard construction of Dudley [14] to find, via the methods of the previous section, an enclosing polytope of $O(1/\varepsilon)$ vertices whose boundary is at Hausdorff distance at most ε from P.

STAGE 1: We begin by computing, in $O(\sqrt{n}\,)$ time, a polytope $P' \subseteq P$, such that $\mathtt{vol}\,(P') \geq c_0\,\mathtt{vol}\,(P)$ for some constant $c_0 > 0$. Compute the six points $\xi_P(\ell)$, for $\ell = \pm x$, $\pm y$, $\pm z$. These points come in pairs, so let w_1, w_2 be the pair forming the largest distance. Given a point w on the line $\mathcal{L}$ passing through w_1 and w_2, let P_w denote the intersection of P with the plane through w that is orthogonal to $\mathcal{L}$. Let w_0 be the midpoint of w_1w_2 (Figure 3). We first show that if S is a set of points in P_{w_0} such that

$$\mathtt{area}\,(\mathtt{conv}\,(S)) \geq c_1\,\mathtt{area}\,(P_{w_0}), \tag{3}$$

for some constant $c_1 > 0$, then $\mathtt{vol}\,(\mathtt{conv}\,(S \bigcup \{w_1, w_2\})) \geq c_2\mathtt{vol}\,(P)$, for some other constant $c_2 > 0$. Therefore, we can take $P' = \mathtt{conv}\,(S \bigcup \{w_1, w_2\})$ to achieve our goal. Indeed, assume we have such a set S. As a straightforward consequence of Pythagoras' theorem, we find that $\mathtt{diam}\,(P) \leq \sqrt{3}\,d(w_1, w_2)$; therefore, the orthogonal projection of P on $\mathcal{L}$ is a segment $v_1v_2 \supseteq w_1w_2$ of length at most $\sqrt{3}\,d(w_1, w_2)$. This implies that, for any w in $\mathcal{L}$, $\mathtt{area}\,(P_w) \leq 2\sqrt{3}\,\mathtt{area}\,(P_{w_0})$. To see why, observe that if, say, $w \in v_1w_0$, then, by convexity, P_w is enclosed in the cone with apex w_2 and base P_{w_0}. Therefore, P_w lies in a copy of P_{w_0} scaled by at most $d(v_0, w_2)/d(w_0, w_2) \leq 2\sqrt{3}$, which proves our claimed upper bound on $\mathtt{area}\,(P_w)$. Of course, the same argument can be repeated if $w \in w_0v_2$. Because $\mathtt{vol}\,(P) = \int_{v_1}^{v_2} \mathtt{area}\,(P_w)\,dw$, we can conclude that the 4 quantities

$$\mathtt{vol}\,(P), \qquad \mathtt{vol}\,(\mathtt{conv}\,(P_{w_0} \cup \{v_1, v_2\})),$$

$$\mathtt{vol}\,(\mathtt{conv}\,(P_{w_0} \cup \{w_1, w_2\})), \qquad \mathtt{vol}\,(\mathtt{conv}\,(S \cup \{w_1, w_2\}))$$

are all equal up to within constant factors.

We now show how to find a set S satisfying (3). Let a, b be two mutually orthogonal vectors both normal to $\mathcal{L}$, and

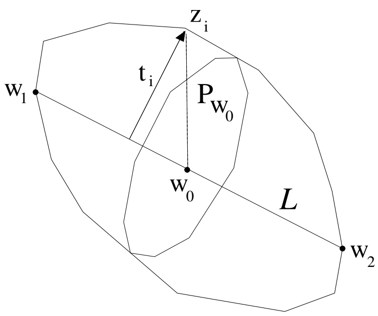

Figure 3: Approximating P from within.

let z_1, z_2, z_3, z_4 be the four points $\xi_P(\ell)$, for $\ell = a, -a, b, -b$. We define $t_i = \overrightarrow{w_0 z_i} - u_i$, where u_i is the projection of $w_0 z_i$ on $\mathcal{L}$. Now, for $i = 1, 2, 3, 4$, we compute the intersection of ∂P with the line passing through w_0 with direction t_i. This gives us four points on the boundary of P_{w_0}. Let s denote the furthest one from w_0. It can be shown that $d(w_0, s)$ is a constant-factor approximation of $\mathtt{diam}\,(P_{w_0})$. Pick the midpoint m of $w_0 s$, and compute the intersection of ∂P with the line in P_{w_0} passing through m with direction normal to $w_0 s$, which gives us two points m_1, m_2. We omit the proof that the quadrilateral with vertex set $S = \{w_0, s, m_1, m_2\}$ has an area at least proportional to $\mathtt{area}\,(P_{w_0})$, and thus satisfies (3). We note that a similar approach to the one we described above was used by Barequet and Har-Peled in [4]. The difference is that they approximate the volume of a convex polytope from outside by a bounding box, whereas we approximate it from within.

Let $\mathcal{E}$ be the largest ellipsoid enclosed in $\mathtt{conv}\,(P')$, also known as the Löwner-John ellipsoid. It is computable in constant time within any fixed relative error by solving a constant-size quadratic program [17]. As is well known, its volume is at least $(1/\mathrm{dim})^2$ times that of the enclosing polytope; therefore,

$$\mathtt{vol}\,(\mathcal{E}) \geq \frac{1}{9}\,\mathtt{vol}\,(P') \geq c \cdot \mathtt{vol}\,(P),$$

for some constant $c > 0$. Make the center of the ellipsoid the origin of the system of coordinates, and use the ellipsoid's positive semidefinite matrix to rescale P. To do that, we consider the linear transformation that takes the ellipsoid into a ball of the same volume. Specifically, if $x^T A^T A x \leq 1$ is the equation of the ellipsoid, then we consider the transformation $T = A/(\det A)$. The polytope TP has the same volume as P, but it is *round*, namely it contains a ball B of volume in $\Omega(\mathtt{vol}\,(TP))$. Thus, we might as well assume that P has this property to begin with. Note that P is also enclosed in a concentric ball B' that differs from B by only a constant-factor scaling. (If not then TP would contain a point p so far away from B that the convex hull of p and B, although contained in P, would have volume much larger than $\mathtt{vol}\,(B)$ and hence $\mathtt{vol}\,(P)$, which would give a contradiction.) Finally, by rescaling we can also assume that P is enclosed in the unit ball and its volume is bounded below by a positive constant. By Corollaries 2.2 and 3.3, all of the work in Stage 1 can be done in $O(\sqrt{n}\,)$ time.

Stage 2: We implement Dudley's construction [14] of a convex polytope Q such that: (i) $Q \supseteq P$; (ii) $Q \subset P_\varepsilon$, where P_ε is the Minkowski sum of P with a ball of radius ε; (iii) Q has $O(1/\varepsilon)$ vertices. Dudley's result was used constructively in [2]. The difference here is that our implementation is sublinear. We compute an $\sqrt{\varepsilon}$-net on the unit sphere,[3] and project this net down to ∂P, using the nearest-neighbor function n_P as a projection map. Finally, we form Q as the intersection of the $O(1/\varepsilon)$ halfspaces bounded by the appropriate tangent planes passing through the vertices of the projected net. With suitable use of the nearest neighbor algorithm of Theorem 3.2, we can implement the entire construction in time $O(\varepsilon^{-1}\sqrt{n}\,)$ for the projection construction (since the facial representations of P and TP are the same, the algorithm can use TP as though it had its full facial representation at its disposal) and $O(\varepsilon^{-1}\log\varepsilon^{-1})$ for intersecting the halfspaces needed to form Q. Since we can obviously assume that Q does not have more vertices than P, there is no need for ε to be smaller than, say, $1/n^2$. This implies that the entire construction time is in fact $O(\varepsilon^{-1}\sqrt{n}\,)$. Because of (i) and (ii), $\mathtt{vol}\,(Q) = (1 + O(\varepsilon))\mathtt{vol}\,(P)$.

We sketch a proof: Recall that P is "sandwiched" between two concentric balls B and B' such that $rad(B') = 1$ and $rad(B) = \Omega(1)$. We assume that B and B' are centered at the origin. We define a polytope P_ε^+ as follows: for each face f of P, let f_ε^+ be the plane parallel to f that avoids the interior of P, such that the distance between f and f_ε^+ is ε. Let H_f be the halfspace bounded by f_ε^+ that contains P. Then $P_\varepsilon^+ = \cap_{f \in P} H_f$. It is easy to show that $Q \subset P_\varepsilon \subset P_\varepsilon^+$. On the other hand we can show, using elementary geometry and the fact that $B \subset P \subset B'$, that $P_\varepsilon^+ \subset (1 + O(\varepsilon))P$. This shows that $\mathtt{vol}\,(Q) \leq \mathtt{vol}\,(P_\varepsilon^+) = (1 + O(\varepsilon))\mathtt{vol}\,(P)$.

Theorem 4.1. *Given any $\varepsilon > 0$, it is possible to approximate the volume of an n-vertex convex polytope with arbitrary relative error $\varepsilon > 0$ in time $O(\varepsilon^{-1}\sqrt{n}\,)$.*

5. APPROXIMATE SHORTEST PATHS

Given a convex polyhedron P with n vertices and two points s and t on its boundary ∂P, find the shortest path between s and t outside the interior of P. It is well known that the shortest path lies on the boundary ∂P. In fact, it is easy to construct instances where any reasonable approximation of the shortest path on ∂P involves $\Omega(n)$ edges. This rules out sublinear algorithms, unless we are willing to follow paths outside of P. We show how to compute a path between s and t whose length exceeds the minimum by a factor of at most $1 + \varepsilon$, for any $\varepsilon > 0$.

Our algorithm relies on a new result of independent interest. Let $d_P(s,t)$ denote the length of the shortest path between s and t in ∂P. Given a point $v \in \partial P$, let H_v be the supporting plane of P at v (or any such plane if v is a vertex), and let H_v^+ denote the halfspace bounded by H_v that contains P. Given $\varepsilon > 0$, we say that a convex polytope Q is an *ε-wrapper* of P if: (c_0 is an absolute constant discussed below.)

(i) Q encloses P;

(ii) the Hausdorff distance between ∂P and ∂Q is $O(\varepsilon\,\mathtt{diam}\,(P))$;

(iii) given any $s, t \in \partial P$ such that $d_P(s,t) \geq c_0\,\mathtt{diam}\,(P)$, $d_{\widehat{Q}}(s,t) \leq (1+\varepsilon)d_P(s,t)$, where $\widehat{Q} = Q \cap H_s^+ \cap H_t^+$.

[3]This is a collection of $O(\varepsilon^{-1})$ points on the sphere such that any spherical cap of radius $\sqrt{\varepsilon}$ contains at least one of the points.

Lemma 5.1. *Any convex 3-polytope has an ε-wrapper of size $O(1/\varepsilon)^{5/4}$, for any $\varepsilon > 0$.*

This result improves on the $O(1/\varepsilon)^{3/2}$ bound of Agarwal et al. [2]. The use of a wrapper is self-evident. First, we clip the polytope to ensure that $d_P(s,t) \geq c_0\, \mathtt{diam}\,(P)$ (Section 5.1). Next, we compute an ε-wrapper (Section 5.2) and approximate the shortest path between s and t by computing the shortest path between the two points in $\partial\widehat{Q}$. This can be done in quadratic time by using an algorithm by Chen and Han [7]. The resulting path, which is of length $(1 + O(\varepsilon))d_P(s,t)$, can be shortened to $(1+\varepsilon)d_P(s,t)$ by rescaling ε suitably. Note that in (iii) the condition on s and t being sufficiently far apart is essential. It is a simple exercise to show that no variant of a wrapper can accommodate all pairs (s,t) simultaneously. If $f(n)$ denotes the complexity of the *exact* version of problem, then we have,

Theorem 5.2. *Given any $\varepsilon > 0$ and two points s,t on the boundary of a convex polytope P of n vertices, it is possible to find a path between s and t outside P of length at most $(1+\varepsilon)d_P(s,t)$ in time $O(\varepsilon^{-5/4}\sqrt{n}\,) + f(\varepsilon^{-5/4})$.*

We refer the reader back to the introduction for a discussion of the implication of this result in view of the state-of-the-art on the function $f(n)$.

5.1 Computing Short Paths

Given two points $s,t \in \partial P$, our first task is to ensure that $d_P(s,t) \geq c_0\, diam\,(P)$, for some constant $c_0 > 0$. To do this, we first compute a value δ such that $\delta \leq d_P(s,t) \leq 8\delta$. We will substitute for P the intersection P' of P with a box centered at s of side length 16δ. Obviously, the shortest path between s and t relative to P and P' are identical. The only computational primitive we need is the nearest neighbor function of Theorem 3.2. It is clear that if we can compute the function relative to P, then we can do it with respect to P' with only constant-time overhead.

To compute a constant-factor approximation for $d_P(s,t)$, we adapt an algorithm of Har-Peled [18] to our sublinear setting. All that is needed is an implementation of the following primitive: Given two rays r_1, r_2 from a fixed point $p \in P$, let H be the plane spanned by these two rays and let C denote the two-dimensional cone in H wedged between r_1 and r_2. Given an additional query ray $r \in H$ (not necessarily emanating from p), we need to compute $\xi_{C\cap P}(H,r)$. By Corollary 3.3, this can be done in $O(\sqrt{n}\,)$ time.

5.2 The ε-Wrapper Construction

Assuming without loss of generality that $\mathtt{diam}\,(P) = 1$, it suffices to prove the following:

Theorem 5.3. *Given any $\varepsilon > 0$ and a convex polytope P of n vertices, there exists a convex polytope Q with $O(\varepsilon^{-5/4})$ vertices such that: (i) $Q \supseteq P$; (ii) the Hausdorff distance between ∂P and ∂Q is $O(\varepsilon)$; and (iii) given any $s,t \in \partial P$ such that $d_P(s,t) \geq c_0$ for some constant c_0, $d_{\widehat{Q}}(s,t) \leq (1+O(\varepsilon))d_P(s,t)$, where $\widehat{Q} = Q \cap H_s^+ \cap H_t^+$.*

We first show how to construct Q. Let S be a sphere of radius 2 centered at some arbitrary point in P. Draw a grid G of longitudes and latitudes on S, so that each cell is of length $\sqrt{\varepsilon}$ by $\sqrt{\varepsilon}$ (with an exception made for the last latitude and longitude, if $\sqrt{\varepsilon}$ does not divide π). All lengths

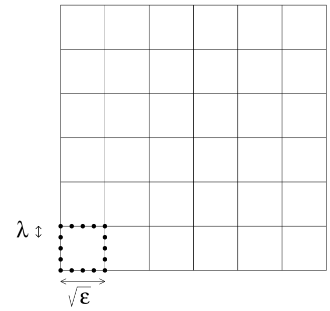

Figure 4: The grid G.

in this discussion are Euclidean, *except* in this case where the length of a circular arc refers to its corresponding angle. We choose a parameter $\lambda = \varepsilon^{3/4}$ and subdivide each side of a cell into sub-arcs of length λ (Figure 4). In this way each cell has $O(\sqrt{\varepsilon}/\lambda)$ vertices, and the whole construction defines a set V of $O(1/\lambda\sqrt{\varepsilon}\,)$ vertices. For each point $v \in V$, we compute $n_P(v)$, its nearest neighbor in ∂P, and define

$$Q = \bigcap \{\, H^+_{n_P(v)} \mid v \in V \,\}. \tag{4}$$

It is immediate from our choice of λ that Q has $O(\varepsilon^{-5/4})$ vertices. Every point of the sphere S has at least one vertex of G at distance $O(\sqrt{\varepsilon}\,)$. By a result of Dudley [14], this implies part (ii) of Theorem 5.3. Since (i) is obvious, it remains for us to prove (iii).

Borrowing terminology from Agarwal et al. [2], we say that a pair $(\sigma, \mathcal{H})$ forms a *supported path* of P if $\sigma = p_1,q_1,p_2,q_2,\ldots, q_{m-1},p_m$ is a polygonal line disjoint from the interior of P and $\mathcal{H} = H_{p_1},\ldots,H_{p_m}$ is a sequence of supporting planes of P, such that $q_{i-1}p_i$ and p_iq_i both lie in H_{p_i}, with $q_0 = p_1$ $q_m = p_m$ (Figure 5). For $0 < i < m$, the *folding angle* α_i at q_i is the dihedral angle of the wedge between H_{p_i} and $H_{p_{i+1}}$ (the one outside P). The folding angle of σ is defined as $\alpha(\sigma) = \sum_{0<i<m} \alpha_i$.

Lemma 5.4. **(Agarwal et al. [2])** *Given $s,t \in \partial P$, there exists a supported path σ of P with $O(1/\varepsilon)$ edges, joining s and t, such that:*

$$d_P(s,t) \leq |\sigma| \leq (1+\varepsilon)d_P(s,t) \qquad \textit{and} \qquad \alpha(\sigma) = O(\varepsilon^{-1/2}).$$

To help build intuition for the remainder of our discussion, it is useful to sketch the proof of the lemma. Mapping the grid G to P via the nearest neighbor function n_P creates a grid $n_P(G)$ on ∂P (with curved, possibly degenerate edges). It is convenient to think of P as a smooth manifold by infinitesimally rounding the vertices and edges. It does not much matter how we do that as long as the end result endows each point $p \in \partial P$ with an (outward) unit normal vector η_p that is a continuous function of p. Note that in this way, for any $u \in S$, the vectors $un_P(u)$ and $\eta_{n_P(u)}$ are collinear, and the function n_P is a bijection. The fundamental property of the nearest-neighor function is that it is non-expansive. We need only a weak version of that fact, which follows directly from Lemmas 4.3 and 4.4 in [14].

Lemma 5.5. **(Dudley [14])** *Given two points $p,q \in \partial P$, $|pq|$ and $\angle(\eta_p,\eta_q)$ are both in $O(\,|n_P^{-1}(p)n_P^{-1}(q)|\,)$.*

This implies that, for any two points $p,q \in \partial P$ in the same cell of the mapped grid $n_P(G)$, both $|pq|$ and $\angle(\eta_p,\eta_q)$

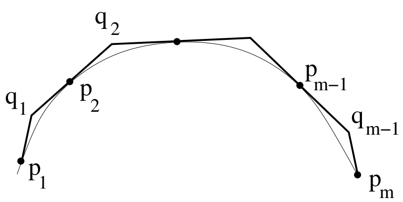

Figure 5: The path σ.

are in $O(\sqrt{\varepsilon})$. We shortcut the shortest path on ∂P from s to t to form a supported path σ that passes through each cell at most once. In this manner, we identify $O(1/\varepsilon)$ points $p_1, \ldots, p_m$ on ∂P, where p_i (resp. p_{i+1}) is the entry (resp. exit) point of the path through the i-th cell in the sequence. The points p_i's lie on the edges of $n_P(G)$. There are two exceptions, $p_1 = s$ and $p_m = t$, which might lie in the interior of the cell. Next, we connect each pair (p_i, p_{i+1}) by taking the shortest path on $H_{p_i} \cup H_{p_{i+1}}$. The path intersects $H_{p_i} \cap H_{p_{i+1}}$ at a point denoted q_i. (Note that q_i might be infinitesimally close to p_i.) This forms a supported path σ with $O(1/\varepsilon)$ vertices $s = p_1, q_1, p_2, q_2, \ldots, q_{m-1}, p_m = t$. The only real difference with the proof in [2] is that we skip the final "trimming" step and keep the points p_i's unchanged. We mention two useful, immediate consequences of Lemma 5.5.

- The folding angle at q_i is $O(\sqrt{\varepsilon})$.
- For each $1 \le i \le m$, the point p_i belongs to ∂P and, for $i \ne 1, m$, there exists a point $w_i = n_P(v_i)$, where $v_i \in V$, such that both $|p_i w_i|$ and $\angle(\eta_{p_i}, \eta_{w_i})$ are in $O(\lambda)$.

From σ we build a curve σ' of length $(1+O(\varepsilon))|\sigma|$ that joins s and t outside the interior of $\widehat{Q}$. The classical result below shows that the shortest path on $\partial\widehat{Q}$ from s to t cannot be longer than σ', which proves Theorem 5.3.

THEOREM 5.6. **(Pogorelov [26])** *Given a convex body C, let γ be a curve joining two points $s, t \in \partial C$ outside the interior of C. Then the length of γ is at least that of the shortest path joining s and t on ∂C.*

We now explain how to construct σ'. For $0 < i < m$, let (p_i, η_{p_i}) and (q_i, η_{p_i}) be the rays emanating from p_i and q_i, respectively, in the direction normal to H_{p_i} away from P. Together with the segments $p_i q_i$ and $q_i p_{i+1}$, the four rays (p_i, η_{p_i}), (q_i, η_{p_i}), $(q_i, \eta_{p_{i+1}})$, and $(p_{i+1}, \eta_{p_{i+1}})$ define a polyhedral surface Σ_i, which consists of two unbounded rectangles, Σ_i^1 and Σ_i^3, joined together at q_i by an unbounded triangle, Σ_i^2 (Figure 6). Note that the surface is in general nonplanar but Σ_i^2 is always normal to the line $H_{p_i} \cap H_{p_{i+1}}$. Out of Σ_i we carve a polyhedral strip S_i as follows. Fix a large enough constant $c > 0$, and let K_i denote the plane $H_{p_i} + c\lambda^2 \eta_{p_i}$. In other words, K_i is a parallel copy of H_{p_i} translated by $c\lambda^2$ away from P: As usual, the superscripted K_i^+ denotes the halfspace enclosing P. Recall that w_i is the nearest neighbor of v_i defined earlier. We need to consider

$$S_i = \Sigma_i \cap \Big\{ (K_i^+ \cap K_{i+1}^+) \cup (H_{w_i}^+ \cap H_{w_{i+1}}^+) \Big\}.$$

Again, we have two exceptions for $i = 1, m-1$, where we use $H_{p_1}^+$ instead of $H_{w_1}^+$ and $H_{p_m}^+$ instead of $H_{w_m}^+$.

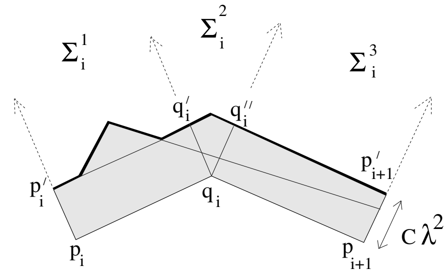

Figure 6: The curve σ_i'.

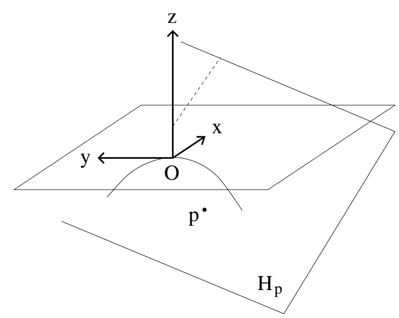

Figure 7: How H_p intersects the xz plane.

Let $p_i p_i'$ be the edge of S_i incident to p_i collinear with η_{p_i}. We denote by σ_i' the portion of ∂S_i between p_i' and p_{i+1}', and define σ' as $\bigcup_{0<i<m} \sigma_i'$. To provide a connection to s and t, we also add to σ' the segments $p_1 p_1'$ and $p_m p_m'$. To show that σ' is a connected curve outside the interior of $\widehat{Q}$ of length $(1+O(\varepsilon))|\sigma|$ requires a simple technical lemma.

LEMMA 5.7. *Given an orthogonal system of reference (O, xyz), assume that P is tangent to the xy plane at O and lies below it. Given a point p on ∂P, if $|n_P^{-1}(O) n_P^{-1}(p)| < \delta$, for some small enough $\delta > 0$, then the intersection of H_p with the xz plane has for equation, $Z = aX + b$, where $|a| = O(\delta)$ and $0 \le b = O(\delta^2)$.*

PROOF. By Lemma 5.5, the normal to H_p forms a small angle $\theta = O(\delta)$ with the z axis, so the plane H_p, being nonparallel to the z axis, can be expressed as $Z = aX + cY + b$. The cross product between the normal $(a, c, -1)$ and the z-axis vector is the vector $(c, -a, 0)$. By the cross product formula, its length, which is $\sqrt{a^2+c^2}$, is also equal to $\sqrt{a^2+c^2+1}\,\sin\theta$. It follows that $a^2 + c^2 = O(a^2 + c^2 + 1)\delta^2$; therefore,

$$a^2 + c^2 = \frac{O(\delta^2)}{1 - O(\delta^2)} = O(\delta^2), \tag{5}$$

and hence $|a| = O(\delta)$. By convexity of P, the plane H_p intersects the nonnegative part of the z axis, and p_z, the z coordinate of p, is nonpositive. By (5) and $|Op| = O(\delta)$, it follows that

$$0 \le b = p_z - a p_x - c p_y \le \sqrt{a^2 + c^2}\,\sqrt{p_x^2 + p_y^2} = O(\delta^2).$$

□

We examine each σ_i' separately, omitting the cases $i = 1, m-1$, which are trivial modifications of the general case

$1 < i < m-1$. The curve σ_i' lies outside the interior of $H_{w_i}^+ \cap H_{w_{i+1}}^+$ and hence of $\widehat{Q}$. It is naturally broken up into three parts, $\sigma_i^j \subset \Sigma_i^j$ ($j = 1, 2, 3$), each one of them being a polygonal curve whose edges lie in any one of four planes: K_i, K_{i+1}, H_{w_i}, and $H_{w_{i+1}}$. Applying Lemma 5.7 with $(p_i, \overrightarrow{p_iq_i}, \overrightarrow{p_ip_i'})$ in the role of (O, x, z) and w_i in the role of p, we find that H_{w_i} intersects the segment p_ip_i', for c large enough; similarly, $H_{w_{i+1}}$ intersects $p_{i+1}p_{i+1}'$. This shows that p_i' is the intersection of the ray (p_i, η_{p_i}) with the plane K_i; therefore, p_i' is *the same point* in the definition of σ_i' and σ_{i-1}', thus proving that the curve σ' is, indeed, connected. (The danger was having p_i' defined by $H_{w_{i+1}}$.) We now bound the length of σ_i'.

- By Lemma 5.7 the slopes of the edges of σ_i^1 are chosen among: 0 for K_i; $O(\sqrt{\varepsilon})$ for K_{i+1}; $O(\lambda)$ for H_{w_i}; and $O(\sqrt{\varepsilon})$ for $H_{w_{i+1}}$. It follows that $|\sigma_i^1| \le |p_iq_i|/\cos\theta$, where $\theta = O(\sqrt{\varepsilon})$; therefore, $|\sigma_i^1| = (1 + O(\varepsilon))|p_iq_i|$. The same argument shows that $|\sigma_i^3| = (1+O(\varepsilon))|q_ip_{i+1}|$.

- Let q_i', q_i'' be the endpoints of the curve σ_i^2 (Figure 6), and let a, a', b, b' be the distances along the ray (q_i, η_{p_i}) from q_i to K_i, K_{i+1}, H_{w_i}, and $H_{w_{i+1}}$, respectively. By definition of S_i,

  $$|q_iq_i'| = \max\Big\{ \min\{a, a'\}, \min\{b, b'\} \Big\}.$$

  Obviously, $a = c\lambda^2$ and, by Lemma 5.7, $b = O(\lambda|p_iq_i| + \lambda^2)$. This implies that $|q_iq_i'| = O(\lambda|p_iq_i| + \lambda^2)$ and, by the same argument,

  $$|q_iq_i'| + |q_iq_i''| = O(\lambda(|p_iq_i| + |q_ip_{i+1}|) + \lambda^2).$$

  Within Σ_i^2, the curve σ_i^2 is a polygonal line consisting of at most a constant number of edges. It is easy to see that for any vertex v of σ_i^2 (including q_i' and q_i''), the angle between q_iv and edges of σ_i^2 incident to v is $\pi/2 \pm O(\sqrt{\varepsilon})$. This follows from a simple geometric observation: given any plane H whose normal makes with q_iv an angle at most α, the angle formed by q_iv and any line on H lies in the range $[\pi/2 - \alpha, \pi/2 + \alpha]$. Since any of the edges of σ_i^2 lies on one of four planes: K_i, K_{i+1}, H_{w_i} and $H_{w_{i+1}}$, and the normal of each of them makes an angle of $O(\sqrt{\varepsilon})$ with q_iv, the claim follows. Because the folding angle of $O(\sqrt{\varepsilon})$ can be assumed to be less than, say, $\pi/2$, this implies that the curve σ_i^2 lies entirely at a distance $O(|q_iq_i'|+|q_iq_i''|)$ from q_i. It follows that $|\sigma_i^2| = O(|q_iq_i'| + |q_iq_i''|)\sqrt{\varepsilon}$.

Putting everything together we find that

$$|\sigma_i'| = (1 + O(\varepsilon) + O(\lambda\sqrt{\varepsilon}))(|p_iq_i| + |q_ip_{i+1}|) + O(\lambda^2\sqrt{\varepsilon}).$$

In view of the fact that $|p_1p_1'| = |p_mp_m'| = c\lambda^2$, summing up over all $|\sigma_i'|$'s (there are $O(1/\varepsilon)$ of them),

$$\begin{aligned} |\sigma'| &= (1 + O(\varepsilon) + O(\lambda\sqrt{\varepsilon}))|\sigma| + O(\lambda^2/\sqrt{\varepsilon}) \\ &= (1 + O(\varepsilon))|\sigma| + O(\varepsilon) \\ &= (1 + O(\varepsilon))|\sigma|, \end{aligned}$$

which completes the proof of Theorem 5.3. Note that the setting of λ is made to ensure that the additive term $O(\lambda^2/\sqrt{\varepsilon})$ is $O(\varepsilon)$. □

6. ACKNOWLEDGMENTS

We wish to thank Pankaj Agarwal, Funda Ergun, Sariel Har-Peled, Joe Mitchell, and Ronitt Rubinfeld for several helpful discussions.

7. REFERENCES

[1] Agarwal, P.K., Har-Peled, S., Karia, M. *Computing approximate shortest paths on convex polytopes*, Algorithmica 33 (2) 1999, 227-242.

[2] Agarwal, P.K., Har-Peled, S., Sharir, M., Varadarajan, K. *Approximating shortest paths on a convex polytope in three dimensions*, J. ACM 44 (1997), 567–584.

[3] Agarwal, P.K., Erickson, J. *Geometric range searching and its relatives*, in "Advances in Discrete and Computational Geometry," eds. Chazelle, B., Goodman, J.E., Pollack, R., Contemporary Mathematics 223, Amer. Math. Soc., 1999, pp. 1–56.

[4] Barequet, G., Har-Peled, S. *Efficiently approximating the minimum-volume bounding box of a point set in three dimensions*, J. Algorithms 38 (2001), 91–109.

[5] Chazelle, B. *The Discrepancy Method: Randomness and Complexity*, Cambridge University Press, 2000; paperback version 2001.

[6] Chazelle, B., Dobkin, D.P. *Intersection of convex objects in two and three dimensions*, J. ACM 34 (1987), 1–27.

[7] Chen, J., Han, Y. *Shortest paths on a polyhedron*, Proc. 6th SOCG (1990), 360–369.

[8] Clarkson, K.L., Shor, P.W. Applications of random sampling in computational geometry, II, *Disc. Comput. Geom.* 4 (1989), 387–421.

[9] Czumaj, A., Ergun, F., Fortnow, L., Magen, A., Newman, I., Rubinfeld, R., Sohler, C. *Sublinear-time approximation of Euclidean minimum spanning tree*, Proc. 14th SODA (2003), to appear.

[10] Czumaj, A., Sohler, C. *Property testing with geometric queries*, Proc. 9th ESA (2001), 266–277.

[11] Czumaj, A., Sohler, C., Ziegler, M. *Property testing in computational geometry*, Proc. 8th ESA (2000), 155–166.

[12] Devroye, L., Mücke, E.P., Zhu, B. *A note on point location in Delaunay triangulations of random points*, Algorithmica 22 (1998), 477-482.

[13] Dobkin, D.P., Kirkpatrick, D.G. *Determining the separation of preprocessed polyhedra – a unified approach*, Proc. 17th ICALP (1990), 400–413.

[14] Dudley, R.M. *Metric entropy of some classes of sets with differentiable boundaries*, J. Approx. Theory 10 (1974), 227–236.

[15] Eppstein, D. *Dynamic Euclidean minimum spanning trees and extrema of binary functions*, Disc. Comput. Geom. 13 (1995), 111–122.

[16] Ergun, F., Kannan, S., Kumar, S. Ravi, Rubinfeld, R., Viswanathan, M. *Spot-checkers*, Proc. STOC (1998), 259–268.

[17] Grötschel, M., Lovász, L., Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[18] Har-Peled, S. *Approximate shortest-path and geodesic diameter on convex polytopes in three dimensions*, Disc.

Comput. Geom. 21 (1999), 217–231.

[19] Kapoor, S., *Efficient computation of geodesic shortest paths*, Proc. 31rd STOC (1999), 770–779.

[20] Matoušek, J. *Geometric range searching*, ACM Comput. Surv. 26 (1994), 421–461.

[21] Mehlhorn, K., Naher, S., Schilz, T., Schirra, S., Seel, M., Seidel, R., Uhrig, C. *Checking geometric programs or verification of geometric structures*, Comput. Geom.: Theory and Appl. 12 (1999), 85–103.

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

[23] Mitchell, J.S.B. *An algorithmic approach to some problems in terrain nevigation*, Autonomous Mobile Robots: Perception, Mapping and Nevigation, IEEEE computer society press, Los Alamitos, CA (1991) 408–427.

[24] Mücke, E.P., Saias, I., Zhu, B. *Fast randomized point location without preprocessing in two and three-dimensional Delaunay triangulations*, Proc. 12th SOCG (1996), 274–283.

[25] Mulmuley, K. *Output sensitive and dynamic constructions of higher order Voronoi diagrams and levels in arrangements*, JCSS 47 (1993), 437–458.

[26] Pogorelov, A.V. *Extrinsic geometry of convex surfaces*, Volume 35 of *Translations of Mathematical Monographs*, American Mathematical Society, Providence, RI, 1973.

[27] Ron, D. *Property testing*, a tutorial, to appear in "Handbook on Randomization."

[28] Seidel, R. *Small-dimensional linear programming and convex hulls made easy*, Disc. Comput. Geom. 6 (1991), 423–434.

[29] Sharir, M., Schorr, A. *On shortest paths in polyhedral spaces*, SIAM J. Comput. 15 (1986), 193–215.