Randomized Incremental Constructions of Three-Dimensional Convex Hulls and Planar Voronoi Diagrams, and Approximate Range Counting[*]

Haim Kaplan[†] Micha Sharir[‡]

August 17, 2005

Abstract

We present new algorithms for approximate range counting, where, for a specified $\varepsilon > 0$, we want to count the number of data points in a query range, up to *relative* error of ε. We first describe a general framework, adapted from Cohen [12], for this task, and then specialize it to two important instances of range counting: halfspaces in $\mathbb{R}^3$ and disks in the plane. The technique reduces the approximate range counting problem to that of finding the minimum rank of a data object in the range, with respect to a *random* permutation of the input.

A major technical step in our analysis, which we believe to be of independent interest, is a bound of $O(n \log n)$ on the expected complexity of the overlay of all the Voronoi faces that are generated during a randomized incremental construction of the Voronoi diagram of n points in the plane. The same bound holds for the expected complexity of the overlay of all the faces of the minimization diagram of the lower envelope of n planes in $\mathbb{R}^3$, or for the expected complexity of the overlay of all the normal (or Gaussian) diagram faces of the convex hull of n points in $\mathbb{R}^3$, that are generated during a randomized incremental construction of the lower envelope or of the hull, respectively. All these bounds are tight in the worst case.

The first bound leads to an algorithm that, for a query point $x \in \mathbb{R}^2$, efficiently retrieves the sequence of nearest neighbors of x in P, over the random insertion process. A query takes $O(\log n)$ expected time, and the expected storage size is $O(n \log n)$. Similarly, the other bounds lead to an algorithm that, for a query direction $\omega \in \mathbb{S}^2$, efficiently retrieves the sequence of the convex hull vertices that are touched by the planes with outward direction ω that support the convex hull during the random insertion process. Again, a query takes $O(\log n)$ expected time, and the expected storage size is $O(n \log n)$.

These algorithms are used as the main component in the approximate range counting technique that we present, for ranges that are halfspaces in $\mathbb{R}^3$ or disks in the plane. Our algorithms have the best performance known to date; they slightly improve upon the previous technique of Aronov and Har-Peled [5], and appear to be conceptually simpler.

[*]Work by Haim Kaplan was partially supported by the German Israeli Foundation (GIF) grant no. 2051-1156-6/2002. Work by Micha Sharir was partially supported by NSF Grant CCR-00-98246, by a grant from the U.S.-Israeli Binational Science Foundation, by a grant from the Israel Science Fund, Israeli Academy of Sciences, for a Center of Excellence in Geometric Computing at Tel Aviv University, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

[†]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: `haimk@post.tau.ac.il`

[‡]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. E-mail: `michas@post.tau.ac.il`

1 Introduction

Approximate range counting. Let P be a finite set of points in $\mathbb{R}^d$, and $\mathcal{R}$ a set of ranges (certain subsets of $\mathbb{R}^d$, e.g., halfspaces, balls, etc.). The *range counting problem* for $(P, \mathcal{R})$ is to preprocess P into a data structure that supports efficient queries of the form: Given a range $r \in \mathcal{R}$, count the number of points in $r \cap P$.

Unfortunately, the best algorithms for solving the *exact* range counting problem are not very efficient. For example, consider the case where the ranges are halfspaces in $\mathbb{R}^d$. If we wish to answer queries in logarithmic or polylogarithmic time, the best solution requires $O(n^d)$ storage, and if we allow only linear or near-linear storage, the best known query time is $O(n^{1-1/d})$ [23]. The case $d = 3$, addressed in this paper, thus requires $O(n^{2/3})$ time for a counting query, with near-linear storage.

It is therefore desirable to find improved algorithms that can answer *approximate range counting* queries, in which we specify the maximum *relative* error $\varepsilon > 0$ that we allow, and, for any range $r \in \mathcal{R}$, we want to quickly estimate $n_r = |r \cap P|$, so that the answer n' that we produce satisfies

$$(1-\varepsilon)n_r \le n' \le (1+\varepsilon)n_r.$$

In particular, if $n_r < \frac{1}{\varepsilon}$, it has to be counted *exactly* by the algorithm. Specializing this still further, the case where $n_r = 0$ (*range emptiness*) has to be detected exactly by the algorithm.

Using ε-approximations. There is a simple well-known method that almost achieves this goal. That is, choose a random sample E of $\frac{c}{\varepsilon^2} \log \frac{1}{\varepsilon}$ points of P, for some sufficiently large absolute constant c (that depends on the so-called VC-dimension of the problem [9]). Then, with high probability, E is an ε-*approximation* for P (see, e.g., [9]), in the sense that, with high probability, we have, for each $r \in \mathcal{R}$,

$$\left| \frac{|E \cap r|}{|E|} - \frac{|P \cap r|}{|P|} \right| \le \varepsilon.$$

This allows us to approximate $|P \cap r|$ by $|E \cap r| \cdot \frac{|P|}{|E|}$, where $|E \cap r|$ is obtained by brute force, in $O(|E|)$ time. However, the *additive* error bound is $\varepsilon|P|$, rather than $\varepsilon|P \cap r|$. If $|P \cap r|$ is proportional to $|P|$, then an appropriate re-scaling of ε turns this absolute error into the desired relative error. However, if $|P \cap r|$ is small, the corresponding re-scaling of ε will require $|E|$ to grow significantly to ensure relative error of ε, and the approach will become inefficient. In particular, range emptiness cannot be detected exactly by this method, unless we take $E = P$.

Cohen's technique. In this paper we present a different approach to approximate range counting, and demonstrate it on two instances: halfspace range counting in $\mathbb{R}^3$, and disk range counting in the plane. Our technique is an adaptation of a general method, introduced by Cohen [12], which estimates the number of data objects in a given range S as follows. One assigns to each data object, independently, a random *weight*, drawn from an exponential distribution with density function e^{-x}, sorts the objects by their weights into a random permutation, and then finds the *minimum rank* in that permutation of the objects in the query range S. One then repeats this experiment $O\left(\frac{1}{\varepsilon^2} \log n\right)$ times, computes the average μ of the *weights* of the minimum elements, and approximates $|S|$ by $1/\mu$. (Cohen [12] also proposes several other estimators that have similar properties.) As shown in [12], this approximate count lies, with high probability, within relative error ε of $|S|$. If only $\frac{1}{\varepsilon^2}$ experiments are conducted, the *expected* relative error remains at most ε. See [12] for more details.

To apply this machinery for approximate halfspace range counting in $\mathbb{R}^3$, say, we need to solve the following problem: Let P be a set of n points in $\mathbb{R}^3$ in general position,[1] and let π be a random permutation of P. (It is easily verified that the sorted order of the points of P according to their randomly drawn weights is indeed a random permutation; see [12].) We want to construct a data structure that can answer efficiently *halfspace-minimum range queries* of the form: Given a query halfspace h, find the point of $p \in P \cap h$ of minimum rank in π (i.e., minimum value of $\pi(p)$).

Similarly, for approximate disk range counting in the plane, we need to solve an analogous problem, for a set P of n points in the plane, where each query specifies a disk, and seeks the point of P of minimum rank that lies in the disk.

Our results. We present efficient algorithms that perform these minimum-rank range searching tasks. The expected storage that they use is $O(n \log n)$, a query takes $O(\log n)$ expected time, and the expected preprocessing time is $O(n \log n)$. Plugging these algorithms into the general approximate range counting framework of Cohen [12], we obtain algorithms that use $O\left(\frac{1}{\varepsilon^2} n \log^2 n\right)$ expected storage and preprocessing time, and answer a query in $O\left(\frac{1}{\varepsilon^2} \log^2 n\right)$ expected time. (This should be compared to the $O(n^{2/3})$ cost of exact range counting queries with near-linear storage.) If the approximate count only has to hold in expectation, we can improve each of these bounds by a factor of $\log n$. Moreover, a simple modification of the algorithm that we detail below brings the storage down to $O\left(\frac{1}{\varepsilon^2} n \log n\right)$, without affecting the bound on the query time.

The overlay of minimization diagrams. A major technical step in our analysis, which we believe to be of independent interest, is a bound of $O(n \log n)$ on the expected complexity of the *overlay* of all the Voronoi faces that are generated during a randomized incremental construction of the Voronoi diagram of n points in the plane (as in [18]). The same bound holds for the expected complexity of the overlay of all the normal (or Gaussian) diagram faces that are generated on the unit sphere $\mathbb{S}^2$ during a randomized incremental construction of the 3-dimensional convex hull of a set of n points in $\mathbb{R}^3$, or for the expected complexity of the overlay of all the faces of the minimization diagram that are generated during a randomized incremental construction of the lower envelope of a set of n planes in $\mathbb{R}^3$. (We note that Voronoi diagrams are a special case of such minimization diagrams [16].) In all these cases, the $O(n \log n)$ bound is tight in the worst case.

The first bound leads to an algorithm that, for a query point $x \in \mathbb{R}^2$, efficiently retrieves the *entire sequence* of nearest neighbors of x in P, over the random insertion process. A query takes $O(\log n)$ expected time, and the expected storage size and preprocessing time is $O(n \log n)$. Similarly, the bounds in $\mathbb{R}^3$ lead to (a) an algorithm that preprocesses a set P of n points in $\mathbb{R}^3$ and, for a query direction $\omega \in \mathbb{S}^2$, efficiently retrieves the sequence of the convex hull vertices that are touched by the planes with outward direction ω that support the convex hull during the random insertion process; and (b) an algorithm that preprocesses a set H of n planes in $\mathbb{R}^3$ and, for a query point $x \in \mathbb{R}^2$, efficiently retrieves the sequence of the planes that attain the lower envelope at x during the random insertion process. Again, in both cases, a query takes $O(\log n)$ expected time, and the expected storage size and preprocessing time is $O(n \log n)$. Using this machinery, finding the minimum rank of a point in a query disk, in the first case, or the minimum rank of a point in a query halfspace, in the second case, can easily be done, using the same resources.

Our new bound on the complexity of the overlay of diagrams of these kinds is related to the work of Guibas et al. [18] on randomized incremental construction of Voronoi diagrams. They build a

[1]To simplify the presentation, we assume throughout the paper that the data objects (points or planes) are in general position, in the sense discussed, e.g., in [7].

data structure containing the Voronoi regions that are generated during a randomized incremental construction of the diagram, such that one can efficiently obtain the entire sequence of Voronoi regions that contain a query point. The time it takes to answer a query, using their structure, is $O(\log^2 n)$ (we briefly discuss below the difficulty in reducing this cost). Guibas et al. [18] pose it as an open problem to improve the query time to $O(\log n)$. As far as we know, almost 15 years since this problem has been posed, it is still open.

The main technical result of our paper provides a partial solution to this problem. Namely, we modify the structure, so that it can answer a point location query of this kind in $O(\log n)$ time, at the cost of increasing the expected storage size to $O(n \log n)$.

Background. Except for the alternative approach that uses ε-approximations, as discussed earlier, there are two recent results that present other alternative solutions to the approximate range counting problem. The first result is due to Aronov and Har-Peled [5], who reduce the problem to range emptiness. Since their result competes with ours, we describe it in some detail.

First, one may assume that $|P \cap r| = \Omega(1/\varepsilon)$. Otherwise we can find $|P \cap r|$ exactly, using a range reporting mechanism, for disks in the plane or for halfspaces in $\mathbb{R}^3$, in $O\left(\log n + \frac{1}{\varepsilon}\right)$ time, using $O(n \log n)$ preprocessing time, and $O(n \log n)$ space. For this purpose we can use a recent algorithm of Chan [8] that preprocesses n points in $O(n \log n)$ expected time into a data structure of size $O(n \log n)$, such that a range reporting query can be answered in $O(\log n + k)$ expected time, where k is the number of points reported. An improved data structure of Ramos [27] reduces the space bound to $O(n \log \log n)$ and makes the query time worst case.

For larger values of $|P \cap r|$, Aronov and Har-Peled perform binary search on this quantity. At each step, they draw a random sample R of P, with the property that, for $|P \cap r|$ in the middle of the current size range I (which is known to contain $|P \cap r|$), the expected number of points of $P \cap r$ that are chosen in the sample is 1. They then test $R \cap r$ for emptiness, and repeat this step for $O\left(\frac{1}{\varepsilon^2} \log n\right)$ different random samples (drawn and preprocessed in advance). If the range turns out to be empty (resp., nonempty) in most trials, then the high (resp., low) quarter portion of the size range I can be eliminated (with high probability), and the algorithm keeps iterating, until the size of I becomes sufficiently small to guarantee a relative error of ε.

In more detail, this step is implemented as follows. The main routine is a random sampling scheme that, given a possible range $I = [a, b]$ for $|P \cap r|$ where $b \geq (1 + \varepsilon)a$, returns, with high probability, (i) UP, if $|P \cap r| \geq \frac{a+3b}{4}$ (i.e., it lies in the upper quarter of the range), (ii) DOWN, if $|P \cap r| \leq \frac{3a+b}{4}$ (the lower quarter of I), and (iii) either UP or DOWN, when $|P \cap r|$ lies between these two thresholds. This scheme works by sampling $O\left(\frac{1}{\varepsilon^2} \log n\right)$ random subsets of P, each of expected size $\frac{2n}{a+b}$, and building a data structure for range emptiness queries over each subset. To perform the binary search step for a query r, one queries each of the emptiness data structures, and if most subsets have empty intersection with r the answer is DOWN, and otherwise the answer is UP.

Aronov and Har-Peled apply this sampling scheme for the intervals $[a_i, b_i]$, where a_1 is $\Theta(1/\varepsilon)$, $b_1 = (1 + \varepsilon)a_1$, and, for $i \geq 1$, $a_{i+1} = b_i$ and $b_{i+1} = (1 + \varepsilon)a_i$. The last interval is the first one for which $b_i \geq n$, and it is then truncated to fit inside the range $[1, n]$. It is easy to check that the number of intervals is $O\left(\frac{1}{\varepsilon} \log n\right)$. To approximate the count in a range r where $|P \cap r| = \Omega(1/\varepsilon)$, we perform binary search over the intervals $[a_i, b_i]$, querying the data structure of each interval $[a_i, b_i]$ that we access, and returning the left endpoint of the interval with the smallest index for which the answer is DOWN (which is also the right endpoint of the largest interval for which the answer is UP).

In the two cases that we consider (which are also the two cases to which Aronov and Har-Peled apply their technique), they show that their data structure uses $O\left(\frac{1}{\varepsilon^3} n \log^2 n\right)$ storage, and answers a query in $O\left(\frac{1}{\varepsilon^2} \log^2 n \log\left(\frac{1}{\varepsilon} \log n\right)\right)$ time: One has to perform $O\left(\log\left(\frac{1}{\varepsilon} \log n\right)\right)$ binary search steps, each involving $O\left(\frac{1}{\varepsilon^2} \log n\right)$ range emptiness queries, each of which takes $O(\log n)$ time (in the special cases of disks in the plane or halfspaces in $\mathbb{R}^3$).

It seems however that their analysis can be tightened and that their data structure in fact requires only $O\left(\frac{1}{\varepsilon^2} n \log n\right)$ space: At the i-th interval $[a_i, b_i]$, one has to sample an expected number of

$$\frac{2n}{a_i + b_i} = \frac{2n}{\frac{1}{\varepsilon}(1+\varepsilon)^{i-1}(2+\varepsilon)}$$

elements, and then preprocess them into a linear-size range emptiness data structure. Since this has to be repeated $O\left(\frac{1}{\varepsilon^2} \log n\right)$ times at each interval, the total storage is

$$O\left(\frac{1}{\varepsilon^2} \log n\right) \cdot O\left(\sum_i \frac{n\varepsilon}{(1+\varepsilon)^i}\right) = O\left(\frac{1}{\varepsilon^2} \log n\right) \cdot O\left(\frac{n\varepsilon}{1 - \frac{1}{1+\varepsilon}}\right) = O\left(\frac{1}{\varepsilon^2} n \log n\right).$$

This makes their space bound asymptotically the same as ours, for the enhanced version of our algorithm.

Our data structure improves the query time of Aronov and Har-Peled by a $O\left(\log\left(\frac{1}{\varepsilon} \log n\right)\right)$ factor, which seems a rather marginal improvement. Nevertheless we believe that the main merit of our work is in (a) the technique that we develop (which is in fact Cohen's technique, but it is the first time, as far as we know, that it is being applied in a geometric context), and (b) the tight $O(n \log n)$ bound on the complexity of the overlay of Voronoi and minimization diagrams. We also strongly believe that these findings will find additional geometric applications.

Another recent result is due to Aronov and Sharir [6]. In this work in progress, they take the partition-tree data structure of Matoušek [22], or its recent extension by Sharir and Shaul [29], which facilitates efficient range emptiness or range reporting queries for *shallow* ranges,[2] and modify the structure by adding to each node of the tree an ε-approximation subset of the set that it stores. The query range is then fed into the structure, and visits some of its nodes. As long as it is shallow, with respect to the subset stored at the current node, the query proceeds in the standard recursive (and efficient) manner. When the query range is detected not to be shallow, the algorithm counts it approximately, using the ε-approximation stored at the current node. (Recall from the earlier discussion that ε-approximations produce good relative error when the ranges are large.) By fine-tuning the relevant parameters, one obtains a performance that is comparable with that of the corresponding range-emptiness or range-reporting algorithms, and is significantly faster than that of exact range counting. (The analysis in [6] caters mainly to approximate range counting in higher dimensions, and thus does not directly compete with our technique.)

Another result related to our complexity bound for the overlay of minimization diagrams is due to Agarwal et al. [3], who have developed a kinetic binary space partitioning (BSP) technique for a set of moving interior-disjoint segments in the plane. To obtain this result, they consider the overlay of the vertical decompositions of prefixes of a random insertion sequence of the segments. They give a very simple proof that the expected complexity of this overlay is $O(n \log n)$: Let $\sigma_1, \ldots, \sigma_k$ be the segments that intersect the vertical ray ρ emanating upwards from an endpoint of a segment s, in increasing order of their y-coordinates. Then σ_i *crosses* (a portion of) ρ in the overlay if and only if s is inserted before $\sigma_1, \ldots, \sigma_i$. Since we add the segments in random order, the probability that ρ crosses σ_i is $1/(i+1)$. Therefore the expected number of segments crossing ρ is at most

[2] That is, more efficient than the partition-tree structure for range counting, which handles *arbitrary* ranges.

$\sum_{i=1}^{n} 1/(i+1) = O(\log n)$, and repeating it for each segment endpoint and each incident vertical ray yields the asserted bound.

However, in our case, a crossing in the overlay of, say, Voronoi regions is determined by *four* of the inserted points, whereas a crossing of the kind studied by [3] is determined by only *two* segments. This tends to make the analysis considerably more intricate.

Agarwal, Ericksen, and Guibas [2], still in the context of developing kinetic BSP structures, extend the result of Agarwal et al. [3] to intersecting segments. In this case the bound on the complexity of the overlay of the vertical decomposition of the segments is $O(n \log n + k)$, where k is the number of segment intersections. In three dimensions, they prove a bound of $O(n \log^2 n + k)$ on the complexity of the overlay of the vertical decomposition of n triangles, where k is the number of intersections between the projections of the edges of the triangles onto the xy-plane.

2 Problem Definition and Main Results

Consider the following minimum range searching problem. Given a random permutation π of a set H of n planes in $\mathbb{R}^3$, and a query point $\kappa^* = (\xi, \eta, \zeta) \in \mathbb{R}^3$, we want to find the plane $p \in H$ of minimum rank $\pi(p)$ that passes below κ^*. For this, we insert the planes of H one at a time, in order of increasing rank, maintaining their *lower envelope* after each insertion. The goal is then to extract from this construction a data structure that, given the projection $(\xi, \eta) \in \mathbb{R}^2$ of the query point, can report the sequence of planes that attain the lower envelope at (ξ, η) during the incremental construction. The heights of these planes become progressively lower at (ξ, η), and the first plane whose height at (ξ, η) is lower than the height of κ^* is the desired plane of minimum rank. Our main result is stated in the following theorem. (In all the results stated below, the expectation is with respect to the random choice of π.)

Theorem 2.1 *Let $\pi = (h_1, \ldots, h_n)$ be a random permutation of a set H of n nonvertical planes in $\mathbb{R}^3$. Let $H_i := \{h_1, \ldots, h_i\}$, for $i = 1, \ldots, n$, and let $LE(H_i)$ denote the lower envelope of H_i. We can preprocess H and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a query point $\kappa = (\xi, \eta) \in \mathbb{R}^2$, we can retrieve the sequence of all planes $h_i \in H$ that attain $LE(H_i)$ at κ, for some $i = 1, \ldots, n$, in $O(\log n)$ expected time.*

As a corollary, we obtain:

Corollary 2.2 *Let H be a set of n nonvertical planes in $\mathbb{R}^3$, and let π be a random permutation of these planes. We can preprocess H and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a query point $\kappa^* = (\xi, \eta, \zeta) \in \mathbb{R}^3$, we can find the plane $h \in H$ of minimum rank $\pi(p)$ that passes below κ^*, in $O(\log n)$ expected time.*

Using standard duality in $\mathbb{R}^3$, as in [15], we obtain the following corollary of Theorem 2.1.

Corollary 2.3 *Let $\pi = (p_1, \ldots, p_n)$ be a random permutation of a set P of n points in $\mathbb{R}^3$. Let $P_i := \{p_1, \ldots, p_i\}$, for $i = 1, \ldots, n$, and let $CH(P_i)$ denote the convex hull of P_i. We can preprocess P and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a direction ω in $\mathbb{R}^3$, we can retrieve the sequence of all vertices $p_i \in P$ that are touched by the planes with outward direction ω that support $CH(P_i)$, for some $i = 1, \ldots, n$, in $O(\log n)$ expected time.*

One can also prove Corollary 2.3 explicitly, employing a proof analogous to the proof of Theorem 2.1 that uses the *normal (or Gaussian) diagram* of the convex hull [19] rather than minimization diagrams of lower envelopes of planes. Corollary 2.3 can now be used to obtain the following result.

Corollary 2.4 *Let P, π, P_i and $CH(P_i)$ be as in Corollary 2.3. We can preprocess P and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a plane h in $\mathbb{R}^3$, we can find the point of minimum rank in P that lies above (or below) h, in $O(\log n)$ expected time.*

We next specialize Theorem 2.1 and Corollary 2.2 to the following setup.

Corollary 2.5 *Let $\pi = (p_1, \ldots, p_n)$ be a random permutation of a set P of n points in $\mathbb{R}^2$. Let $P_i = \{p_1, \ldots, p_i\}$, for $i = 1, \ldots, n$, and let $Vor(P_i)$ denote the Voronoi diagram of P_i. We can preprocess P and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a query point $\kappa = (\xi, \eta) \in \mathbb{R}^2$, we can retrieve the sequence of all points $p_i \in P$, for $i = 1, \ldots, n$, whose Voronoi cells in the corresponding partial diagrams $Vor(P_i)$ contain κ, in $O(\log n)$ expected time.*

Proof: Map each point $p_i = (a_i, b_i) \in P$ to the plane $h_i : z_i = -2a_i x - 2b_i y + a_i^2 + b_i^2$. Let $H_i := \{h_1, \ldots, h_i\}$, for $i = 1, \ldots, n$. It is well known that the xy-projection of $LE(H_i)$ is equal to $Vor(P_i)$ [7, 16]. We can therefore take our data structure to be the data structure of Theorem 2.1 for the set $H = \{h_1, \ldots, h_n\}$. $\Box$

Note that the algorithm of Corollary 2.5 effectively produces the sequence of the distinct nearest neighbors of κ in the prefix sets P_i.

Using the property that the lifting transformation in the proof of Corollary 2.5 maps disks in the plane to halfspaces in $\mathbb{R}^3$, we have the following additional corollary.

Corollary 2.6 *Let $\pi = (p_1, \ldots, p_n)$ be a random permutation of a set P of n points in $\mathbb{R}^2$. We can preprocess P and π in $O(n \log n)$ expected time and build a data structure of expected size $O(n \log n)$, such that, given a disk $D \subset \mathbb{R}^2$, we can compute the point of minimum rank in $P \cap D$, in $O(\log n)$ expected time.*

Remark. Note that Theorem 2.1, Corollary 2.3, and Corollary 2.5 implicitly imply that the expected size of the sequences that are being output is $O(\log n)$; this property is well known and follows from the standard analysis of randomized incremental constructions; see [7].

Here is an outline of the remainder of this paper. Section 3 reviews the randomized incremental construction of minimization diagrams (which is dual to the explicit construction given in [7]). Section 4 describes the algorithm that establishes Theorem 2.1. Section 5 establishes an upper bound of $O(n \log n)$ on the expected complexity of the overlay of minimization diagrams, and shows it to be worst-case tight. We conclude in Section 6 by applying the machinery to obtain efficient algorithms for the approximate range counting problems discussed in the introduction.

3 Randomized Incremental Construction of Minimization Diagrams

In this section we review the randomized incremental construction of minimization diagrams of planes in $\mathbb{R}^3$ (which is dual to the explicit construction presented in [7]). Let H be a set of n non-vertical planes in $\mathbb{R}^3$ in general position, and let $\pi = (h_1, h_2, \ldots, h_n)$ be a random permutation of H. The *minimization diagram* $M(H)$ of H is the xy-projection of the lower envelope $LE(H)$. We insert the planes one at a time, in their order in π, and update, after each insertion, the minimization diagram $M(H_i)$ for the prefix set H_i of planes inserted so far.

Suppose we have constructed $M(H_i)$. When the next plane h_{i+1} is added, we obtain $M(H_{i+1})$ from $M(H_i)$ as follows. If h_{i+1} lies fully above $LE(H_i)$, we do nothing. Otherwise, we find the intersection $\varphi_{h_{i+1}}$ of h_{i+1} with $LE(H_i)$. This is a convex polygon in $\mathbb{R}^3$, whose edges also lie on adjacent faces of $LE(H_i)$, where each such edge cuts the corresponding old face of $LE(H_i)$ into two portions, one of which appears on $LE(H_{i+1})$, and the other is hidden from this envelope by h_{i+1}. The same behavior shows up on the minimization diagram $M(H_i)$. We trim each of the affected old faces of $M(H_i)$, and "glue in" the xy-projection $\varphi^*_{h_{i+1}}$ of $\varphi_{h_{i+1}}$.

An illustration of this process, for the special case of Voronoi diagrams, is given in Figure 1. (The figure also illustrates a disk range query of the sort addressed in Corollary 2.6. Each time the center of the disk "moves" to a new Voronoi face, its nearest neighbor in the prefix set changes (and gets closer), until this neighbor enters the disk; this is point 7 in the figure.)

The actual implementation of this update step uses *conflict lists* that store, for each plane h_j, for $j > i$, the list of all vertices of $LE(H_i)$ that lie above h_j, with reverse pointers from the vertices to the future planes that will hide them from the envelope. Using this information, it is straightforward to construct $LE(H_{i+1})$ from $LE(H_i)$ (or, actually, $M(H_{i+1})$ from $M(H_i)$), in time proportional to the number of new vertices plus the number of removed vertices. Revising the conflict lists after insertion of a plane is also routine; see [7]. As is well known, the expected number of vertices generated by the algorithm is $O(n)$, and the expected running time, dominated by the cost of updating the conflict lists, is $O(n \log n)$. (This is dual to the construction of the lower convex hull of the points dual to the planes of H, as described, e.g., in [7].)

Using an approach that extends the one of Guibas et al. [18], one can link together the faces of the minimization diagram, as they are constructed by the algorithm, so that, roughly speaking, old modified faces point to the new faces that "step on them", and assemble these links into a point location data structure that locates the face of the final diagram that contains a query point. In fact, the faces are linked in such a way, that one obtains not just the final face containing the query, but the entire sequence of faces that contain it, which is exactly what is needed for establishing Theorem 2.1 or the analogous Corollaries 2.3 and 2.5.

Unfortunately, as mentioned in the introduction, the time it takes to answer a query, using this structure, is $O(\log^2 n)$; see Guibas et al. [18] for the special case of Voronoi diagrams, which also extends to more general minimization diagrams.[3] Using our technique we can reduce the expected query time to $O(\log n)$, at the cost of increasing the expected storage size to $O(n \log n)$.

4 Overlaying Incremental Minimization Diagrams

During the incremental construction, we collect the edges of the newly generated faces of all the versions $M(H_i)$ of the minimization diagram into a "global" set E. Once the incremental construction is over, we compute the arrangement $\mathcal{A}(E)$ in the xy-plane, and preprocess it for efficient point location [7]. Clearly, each face f of $\mathcal{A}(E)$ is contained in a single face of each of the minimization diagrams $M(H_i)$. See Figure 2 for an illustration. In Section 5 we prove (in Theorem 5.1) that the

[3]Informally, since faces of the Voronoi (or minimization) diagram need not have constant complexity, the algorithm has to maintain a triangulation of each face, to ensure the expected behavior of the point-location mechanism. However, this incurs a penalty of having to locate the new triangle that contains the query point whenever the point "moves" to a new face of the diagram. This takes $O(\log n)$ time per face change, for a total of $O(\log^2 n)$ expected cost; see [18].

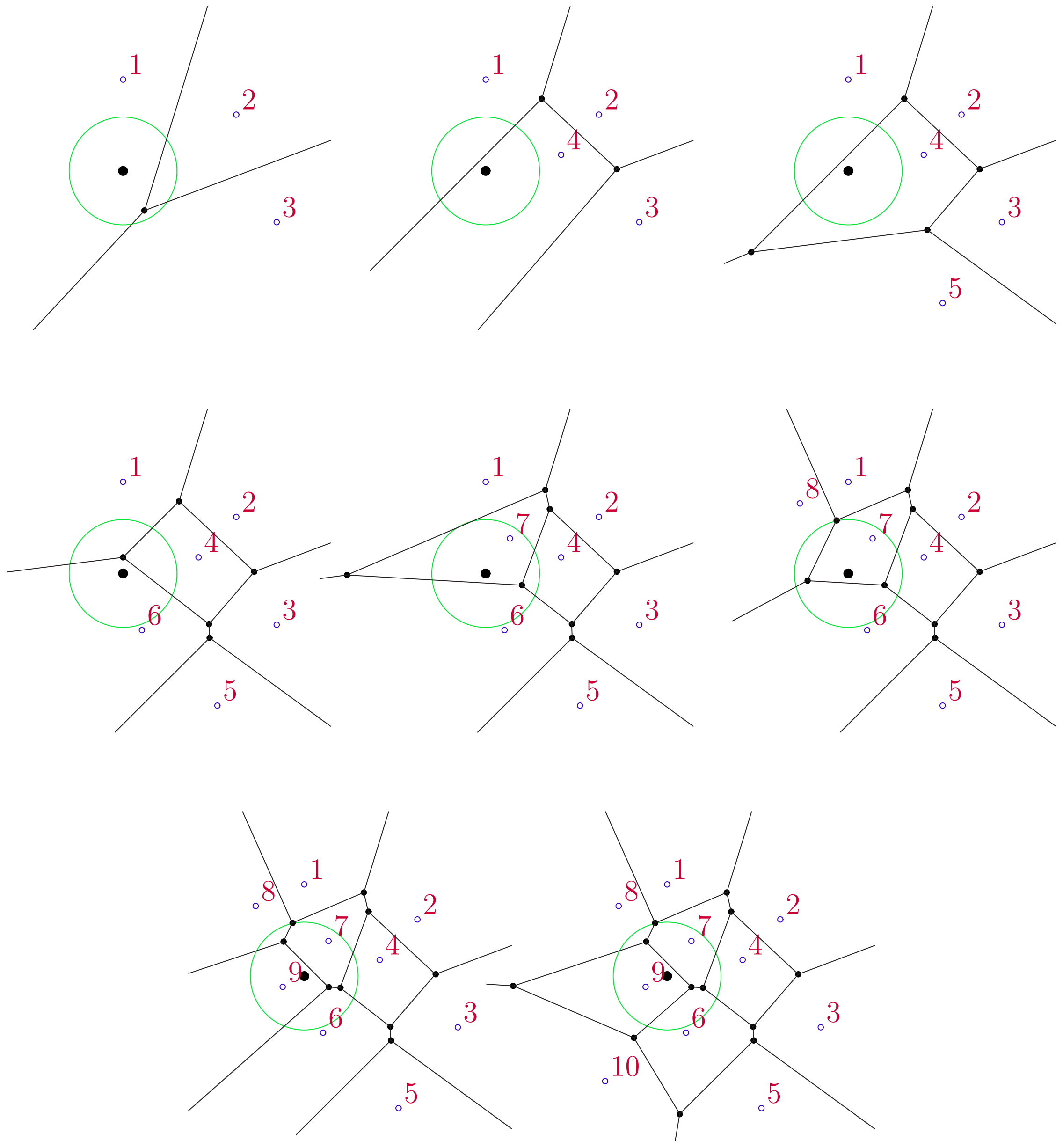

Figure 1: Randomized incremental construction of the Voronoi diagram of 10 points in the plane.

expected complexity of $\mathcal{A}(E)$ is $O(n \log n)$. Therefore if we preprocess $\mathcal{A}(E)$ to build either the point location data structure of Mulmuley [25], or the data structure of Chazelle and Edelsbrunner [10], we consume $O(n \log n)$ space in expectation, and can locate the face f of $\mathcal{A}(E)$ containing a query point in $O(\log n)$ expected time.

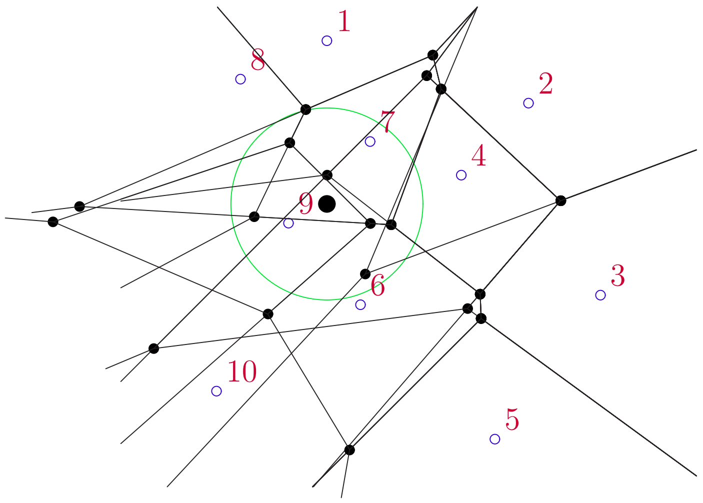

Figure 2: The overlay arrangement $\mathcal{A}(E)$ for the special case of the Voronoi diagram of the point set in Figure 1. As above, the sites are drawn as empty circles and are labeled by their ranks in π. The query disk from the preceding figure is also shown. For the face f of the overlay containing its center, we have $PM(f) = (1, 4, 6, 7, 9)$.

Let $PM(\mathbf{x})$ denote the sequence of *prefix minima* of the permutation π of H at the point $\mathbf{x} \in \mathbb{R}^2$. That is, we add a plane h_i to $PM(\mathbf{x})$ if h_i attains $LE(H_i)$ at $\mathbf{x}$; i.e., it is the lowest plane of H_i at $\mathbf{x}$. Note that this implies that h_i changes the lower envelope when it is inserted, and that $\mathbf{x} \in \varphi^*_{h_i}$. It now follows that, for each face f of $\mathcal{A}(E)$, all the sequences $PM(\mathbf{x})$, for $\mathbf{x} \in f$, are identical, and we denote this common sequence by $PM(f)$. Moreover, by construction, the sequence $PM(f)$ coincides with the sequence of faces that contain f in the dynamic minimization diagram.

For each face f of $\mathcal{A}(E)$, the algorithm constructs the sequence $PM(f)$, and stores it with f. The actual details of this construction will be provided in Section 4.1. This completes the description of the data structure.

We use this data structure in the proofs of Theorem 2.1 and Corollary 2.2. For Theorem 2.1, given a query point (a, b), we locate the face f of $\mathcal{A}(E)$ that contains it, and retrieve the sequence $PM(f)$. This is done in $O(\log n)$ time, using the point location data structure for $\mathcal{A}(E)$, and the fact that the expected size of $PM(f)$ is $O(\log n)$; see also Section 4.1. The list $PM(f)$ is the answer to the query as specified in Theorem 2.1.

To answer a query as specified in Corollary 2.2, where the query is now a point $(a, b, c) \in \mathbb{R}^3$ and the goal is to find the minimum-rank plane that lies below (a, b, c), we query the data structure with (a, b), locate the face f of $\mathcal{A}(E)$ that contains (a, b), and retrieve the sequence $PM(f)$. We then traverse $PM(f)$ to find the highest plane $h^* \in PM(f)$ that lies below (a, b, c). Note that the planes in $PM(f)$ do not intersect over f, and their order in $PM(f)$ coincides with the decreasing order of their heights over f.

Notice that since the expected size of $PM(f)$ is $O(\log n)$, we can use any simple, and even unordered, list representation of $PM(f)$, and still answer queries efficiently. To find h^* with such a representation, we scan all elements in $PM(f)$ sequentially, maintaining the highest plane that lies below the query point, in $O(\log n)$ expected time.

4.1 Constructing and maintaining the sequences $PM(f)$.

The construction is incremental, and proceeds as follows. We form the dual graph G of $\mathcal{A}(E)$, where the faces of $\mathcal{A}(E)$ are its nodes, and each pair of adjacent faces are connected by an edge in G. We pick an initial face f_0, pick any point $\mathbf{x} \in f_0$, and construct $PM(f_0) = PM(\mathbf{x})$, by scanning all the planes of H, in $O(n)$ time. We now apply a BFS to G, and, for each edge (f', f) that connects a processed face f' to an unprocessed face f, we construct $PM(f)$ from $PM(f')$, as follows. The faces f and f' are separated in $\mathcal{A}(E)$ by a single edge e that bounds the face $\varphi^*_{h_i}$ in $M(H_i)$, for some i (this is the new face of $M(H_i)$, of its last inserted plane). Suppose that f is contained in $\varphi^*_{h_i}$ while f' is not. Then, as noted above, all we have to do is to insert h_i into $PM(f')$ to obtain $PM(f)$. Similarly, if f' is contained in $\varphi^*_{h_i}$ while f is not, we obtain $PM(f)$ by deleting h_i from $PM(f')$.

The preprocessing time and space of our data structure depends on the size of G and on the time and additional storage it takes to obtain $PM(f)$ from $PM(f')$. If we represent $PM(f')$ as an unordered list, then we can produce $PM(f)$ by first copying the list, and then by inserting or deleting the appropriate plane from the new copy. This takes $O(\log n)$ expected time and space per update, and, combined with Theorem 5.1, leads to $O(n \log^2 n)$ overall expected preprocessing time and storage.

We can reduce the space required by the data structure by using fully persistent lists rather than regular lists [14, 28]. This allows us to delete or insert an element from/into $PM(f)$, without destroying the previous list, in $O(\log n)$ expected time and $O(1)$ additional space. This results in a data structure that requires $O(n \log n)$ expected space, but the expected preprocessing time remains $O(n \log^2 n)$.

We can reduce the preprocessing time as well by using an implicit representation of $PM(f)$. In this representation we use two persistent lists,[4] the first of which is called the *inserted* list and the second is called the *deleted* list. When we have to insert a plane in order to obtain $PM(f)$ from $PM(f')$, we insert it in the front of the *inserted* list. When we have to delete a plane in order to obtain $PM(f)$ from $PM(f')$, we insert it in the front of the *deleted* list. When the size of the *deleted* list becomes $\log n$, we rebuild the lists so that the *inserted* list contains all the planes in $PM(f)$ and the *deleted* list is empty. We do the rebuilding by creating a new *inserted* list that contains each plane whose number of occurrences in the *inserted* list is larger by one than its number of occurrences in the *deleted* list. We reset the *deleted* list of $PM(f)$ to the empty list.

Clearly the preprocessing time is constant per face of $\mathcal{A}(E)$, plus the total time required by the rebuildings. Since the expected size of $PM(f)$ is $O(\log n)$ and the size of the deleted list is $\log n$ at the time we do a rebuilding, it follows that the expected size of the inserted list at the time of the rebuilding is also $O(\log n)$. Therefore the expected time of a rebuilding is $O(\log n)$ which can be charged to the $\log n$ deleted items. Since each deleted item is charged a constant amount of time exactly once and the expected number of deleted items is $O(n \log n)$, we obtain that the expected preprocessing time is $O(n \log n)$. An analogous argument shows that the expected amount of space remains $O(n \log n)$.

We can still answer queries using the implicit representation of $PM(f)$ in $O(\log n)$ expected time. Indeed we can convert on the fly the implicit representation to an explicit one in $O(\log n)$

[4]In fact here we only need stacks, whose persistent implementation is easier, since we only insert at the front of each list.

time as we did above during rebuildings.

5 The Complexity of the Overlay of the Minimization Diagrams

The actual incremental construction of the minimization diagram [7] uses $O(n)$ expected storage, and takes $O(n \log n)$ expected time. This implies that the expected size of E is $O(n)$. However, apriori, the complexity of $\mathcal{A}(E)$ could be quadratic in $|E|$. Our main combinatorial result is that *the expected complexity of $\mathcal{A}(E)$ is only $O(n \log n)$.* This is asserted in the following main technical contribution of the paper.

Theorem 5.1 *Let H be a set of n planes in $\mathbb{R}^3$, and let $\pi = (h_1, \ldots, h_n)$ be a random permutation of H. Let E be the set of edges of the newly generated faces of all the minimization diagrams $M(H_i)$ that arise when the planes are inserted in their order in π. Then the expected complexity of the arrangement $\mathcal{A}(E)$ is $O(n \log n)$. There exist arbitrarily large sets of hyperplanes H for which this bound is tight.*

Clearly, to establish Theorem 5.1 it suffices to bound the expected number of *crossings* between the edges of E.

Let us analyze the geometric configuration encoded by such a crossing. Let e_1, e_2 be two edges in E that cross at some point $\mathbf{x}$. For $i = 1, 2$, let $a_i, b_i \in H$ be the planes such that e_i is the edge of $\varphi^*_{a_i}$, at the time a_i was inserted, that separates it from $\varphi^*_{b_i}$, which already existed in the diagram, and got shrunk when a_i was inserted. (Note the asymmetric roles of a_1, b_1 and of a_2, b_2 in this definition.) Suppose, without loss of generality, that a_2 has been inserted after a_1. We cannot have $a_1 = a_2$, because different edges of the same newly created face $\varphi^*_{a_1}$ do not cross each other. It is also impossible that $b_2 = a_1$, because then the edge e_2 must be fully contained in the interior of the original version of $\varphi^*_{a_1}$ (this face can only shrink during the construction), and thus cannot cross e_1, which is an edge of that original face. Finally, $b_1 = b_2$ is also impossible: When a_1 is added, it shrinks $\varphi^*_{b_1}$, and when a_2 is added later, the edge e_2 that is formed between $\varphi^*_{a_2}$ and $\varphi^*_{b_1}$ must be fully contained in the interior of that shrunk face. Note also that b_2 cannot be inserted before a_1: If this were the case, then $\varphi^*_{b_1}$ and $\varphi^*_{b_2}$ are already disjoint before any of a_1, a_2 are inserted, so the edges e_1, e_2 would also have to be disjoint. To summarize, the four planes a_1, b_1, a_2, b_2 are all distinct, and they are inserted in the order b_1, a_1, b_2, a_2.

By construction, for $i = 1, 2$, when a_i is inserted, the intersection line $a_i \cap b_i$ contains the edge e_i of the lower envelope, whose xy-projection contains $\mathbf{x}$. Let $\lambda_{\mathbf{x}}$ denote the vertical line at $\mathbf{x}$. Let k denote the number of planes of H that cross the portion of $\lambda_{\mathbf{x}}$ between e_1 and e_2, and let ℓ denote the number of planes of H that cross $\lambda_{\mathbf{x}}$ below e_2; see Figure 3. We say that the quadruple (a_1, b_1, a_2, b_2) has *weight* (k, ℓ). This definition applies to *any* quadruple (a_1, b_1, a_2, b_2) of distinct planes of H, for which the line $a_1 \cap b_1$ passes above the line $a_2 \cap b_2$. Indeed, assuming general position, the xy-projection of these two lines meet at a unique point $\mathbf{x}$. We then take k (resp., ℓ) to be the number of planes that cross $\lambda_{\mathbf{x}}$ between $a_1 \cap b_1 \cap \lambda_{\mathbf{x}}$ and $a_2 \cap b_2 \cap \lambda_{\mathbf{x}}$ (resp., below $a_2 \cap b_2 \cap \lambda_{\mathbf{x}}$), and (k, ℓ) is then the weight of the quadruple.

Let (a_1, b_1, a_2, b_2) be a quadruple with weight (k, ℓ). We next analyze the probability of the event that this quadruple gives rise to a crossing between two edges of E, in the manner discussed above. For this event to occur, the following conditions are necessary and sufficient, as is easily verified.

(i) The four planes a_1, a_2, b_1, b_2 are inserted in the order b_1, a_1, b_2, a_2.

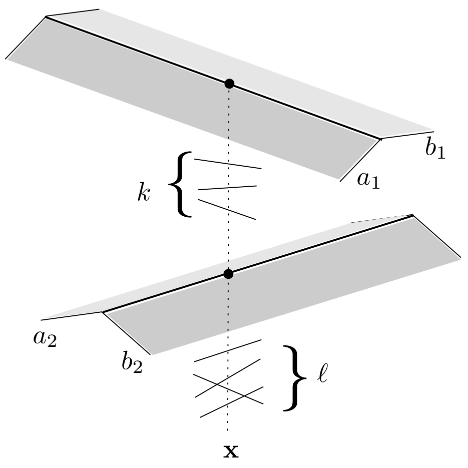

Figure 3: The weight of a quadruple (a_1, b_1, a_2, b_2).

(ii) The k planes that cross $\lambda_{\mathbf{x}}$ between $a_1 \cap b_1 \cap \lambda_{\mathbf{x}}$ and $a_2 \cap b_2 \cap \lambda_{\mathbf{x}}$ are all inserted after a_1.

(iii) The ℓ planes that cross $\lambda_{\mathbf{x}}$ below $a_2 \cap b_2 \cap \lambda_{\mathbf{x}}$ are all inserted after a_2.

Consider the random sub-permutation consisting of these $k+\ell+4$ planes. There are $(k+\ell+4)!$ such permutations. To obtain a permutation that satisfies (i)–(iii), we put b_1 in the first place and put a_1 second. We then choose the locations of the first k conflicting planes, in $\binom{k+\ell+2}{k}$ ways. Then b_2 has to be placed at the first free location, and a_2 at the second free location, and the remaining ℓ planes at the remaining free locations. Finally, we permute the first k planes in their locations, and similarly for the last ℓ planes. Hence, the number of permutations that satisfy (i)–(iii) is

$$\binom{k+\ell+2}{k} k!\ell!,$$

so the probability of our event is

$$q_{k,\ell} = \frac{\binom{k+\ell+2}{k} k!\ell!}{(k+\ell+4)!} = \frac{1}{(\ell+1)(\ell+2)(k+\ell+3)(k+\ell+4)}.$$

Let $N_{k,\ell} = N_{k,\ell}(H)$ denote the number of quadruples (a_1, b_1, a_2, b_2) with weight (k, ℓ). Then the expected number of crossings between the edges of E is

$$X := \sum_{k\geq 0,\ \ell\geq 0,\ k+\ell\leq n-4} q_{k,\ell} N_{k,\ell}.$$

Let $N_{\leq k,\leq \ell} = N_{\leq k,\leq \ell}(H)$ denote the number of quadruples (a_1, b_1, a_2, b_2) with weights (ξ, η) that satisfy $\xi \leq k$, $\eta \leq \ell$. We have

$$N_{k,\ell} = N_{\leq k,\leq \ell} - N_{\leq k-1,\leq \ell} - N_{\leq k,\leq \ell-1} + N_{\leq k-1,\leq \ell-1}.$$

Substituting in the expression for X, we obtain

$$X = \sum_{k\geq 0,\ \ell\geq 0,\ k+\ell\leq n-4} \Big(q_{k,\ell} - q_{k+1,\ell} - q_{k,\ell+1} + q_{k+1,\ell+1} \Big) N_{\leq k,\leq \ell},$$

where probabilities $q_{k,\ell}$ with $k+\ell > n-4$ are taken to be 0. Put

$$\Delta^2 q_{k,\ell} = q_{k,\ell} - q_{k+1,\ell} - q_{k,\ell+1} + q_{k+1,\ell+1}.$$

If $k+\ell \le n-6$ then

$$\Delta^2 q_{k,\ell} = \frac{1}{(\ell+1)(\ell+2)(k+\ell+3)(k+\ell+4)} - \frac{1}{(\ell+1)(\ell+2)(k+\ell+4)(k+\ell+5)}$$
$$-\frac{1}{(\ell+2)(\ell+3)(k+\ell+4)(k+\ell+5)} + \frac{1}{(\ell+2)(\ell+3)(k+\ell+5)(k+\ell+6)} =$$
$$\frac{2}{(\ell+1)(\ell+2)(k+\ell+3)(k+\ell+4)(k+\ell+5)} -$$
$$\frac{2}{(\ell+2)(\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)} =$$
$$\frac{2(\ell+3)(k+\ell+6) - 2(\ell+1)(k+\ell+3)}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)} =$$
$$\frac{4k+10\ell+30}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)}.$$

If $k+\ell = n-5$ then

$$\Delta^2 q_{k,\ell} = \frac{1}{(\ell+1)(\ell+2)(n-1)(n-2)} - \frac{1}{(\ell+1)(\ell+2)n(n-1)} - \frac{1}{(\ell+2)(\ell+3)n(n-1)} =$$
$$\frac{n(\ell+3) - (n-2)(\ell+3) - (n-2)(\ell+1)}{(\ell+1)(\ell+2)(\ell+3)n(n-1)(n-2)} = \frac{2(\ell+3) - (n-2)(\ell+1)}{(\ell+1)(\ell+2)(\ell+3)n(n-1)(n-2)}.$$

Finally, if $k+\ell = n-4$ then

$$\Delta^2 q_{k,\ell} = \frac{1}{(\ell+1)(\ell+2)n(n-1)}.$$

We thus have,

$$X = \sum_{k\ge0,\ \ell\ge0,\ k+\ell\le n-6} \frac{\Big(4k+10\ell+30\Big) N_{\le k,\le \ell}}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)}$$
$$+ \sum_{k\ge0,\ \ell\ge0,\ k+\ell=n-5} \frac{2(\ell+3)-(n-2)(\ell+1)}{(\ell+1)(\ell+2)(\ell+3)n(n-1)(n-2)} N_{\le k,\le \ell}$$
$$+ \sum_{k\ge0,\ \ell\ge0,\ k+\ell=n-4} \frac{N_{\le k,\le \ell}}{(\ell+1)(\ell+2)n(n-1)}.$$

By rearranging the last two terms we obtain that

$$X = \sum_{k\ge0,\ \ell\ge0,\ k+\ell\le n-6} \frac{\Big(4k+10\ell+30\Big) N_{\le k,\le \ell}}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)}$$

$$+\sum_{\ell=0}^{n-5} \frac{[2(\ell+3)-(n-2)(\ell+1)]N_{\le n-5-\ell,\le\ell}+(n-2)(\ell+3)N_{\le n-4-\ell,\le\ell}}{(\ell+1)(\ell+2)(\ell+3)n(n-1)(n-2)}$$

$$+\frac{N_{0,\le n-4}}{n(n-1)(n-2)(n-3)}.$$

By substituting $N_{\le n-4-\ell,\le\ell} = N_{n-4-\ell,\le\ell} + N_{\le n-5-\ell,\le\ell}$ in the middle summation, we get

$$X = \sum_{k\ge0,\ \ell\ge0,\ k+\ell\le n-6} \frac{\Big(4k+10\ell+30\Big)N_{\le k,\le\ell}}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)} \qquad (1)$$

$$+\sum_{\ell=0}^{n-5}\left(\frac{2N_{\le n-5-\ell,\le\ell}}{(\ell+1)(\ell+2)n(n-1)(n-2)}+\frac{2N_{\le n-5-\ell,\le\ell}}{(\ell+1)(\ell+2)(\ell+3)n(n-1)}\right)+\sum_{\ell=0}^{n-4}\frac{N_{n-4-\ell,\le\ell}}{(\ell+1)(\ell+2)n(n-1)}.$$

We next derive an upper bound for $N_{\le k,\le\ell}$ in two stages, as follows.

Estimating $N_{\le t,0}$.

Lemma 5.2 *Let H be a set of n planes in $\mathbb{R}^3$ in general position, and let $t \le n-2$ be a parameter. Then there are at most $O(n(t+1)^2)$ quadruples (a,b,c,d) of distinct planes of H that satisfy the following property: Let $\mathbf{x}$ be the intersection of the xy-projections of the lines $a\cap b$ and $c\cap d$. Then $c\cap d$ attains $LE(H)$ at $\mathbf{x}$, and at most t planes of H cross the vertical line $\lambda_{\mathbf{x}}$ between $a\cap b$ and $c\cap d$.* [5]

Proof: We first claim that the number of pairs (a,b) that participate in such a quadruple (a,b,c,d), over all possible planes $c,d\in H$, is $O(n(t+1))$. Indeed, let (a,b,c,d) be such a quadruple, and let $\mathbf{x}$ be the corresponding intersection point in the xy-plane. Then the portion of $\lambda_{\mathbf{x}}$ below $a\cap b$ is crossed by at most $t+2$ planes of H. Define the *shallowness* of the line $a\cap b$ to be the *minimum* number of planes of H that cross a downward directed vertical ray that emanates from $a\cap b$; see Figure 4. Clearly, the shallowness of $a\cap b$ is at most $t+2$. It is then a routine application of the Clarkson-Shor technique [11] to show that the number of such pairs (a,b) is $O((t+2)^2\cdot n/(t+2)) = O(n(t+1))$.

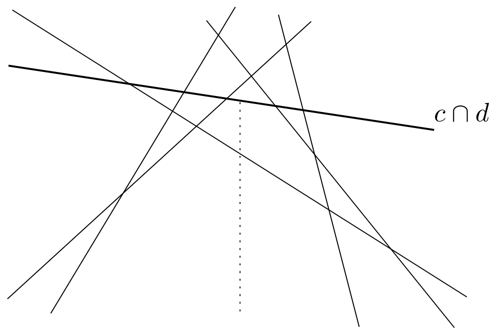

Figure 4: The shallowness of the line $c\cap d$ is 1, as indicated.

[5]When $t=\Theta(n)$ the argument is easy: There are $O(n^2)$ intersection lines $a\cap b$, and only $O(n)$ intersection lines $c\cap d$ (which contain edges of the lower envelope). So the number of quadruples (a,b,c,d) that satisfy the theorem is $O(n^3)$.

We next claim that, for a fixed pair (a,b), the set Q of pairs (c,d) that form with (a,b) a quadruple that satisfies the property in the lemma has size at most $O(t+1)$. The proof considers the vertical plane V through $a \cap b$ and the arrangement within V of the lines $c \cap V$, for $c \in H \setminus \{a,b\}$. Any pair (c,d) in Q defines a vertex $v_{c,d} = (c \cap V) \cap (d \cap V)$ in that arrangement, such that the downward vertical ray from $a \cap b$ through $v_{c,d}$ meets no other line below $v_{c,d}$, and meets at most t lines above $v_{c,d}$ (and below $a \cap b$). We call a line $c \cap V$ *positive* (resp., *negative*), if it lies below $a \cap b$ to the left (resp., right) of its intersection with $a \cap b$ (with respect to some appropriate coordinate frame in V).

We note that no line $c \cap V$ can be incident to more than two vertices $v_{c,d}$ with $(c,d) \in Q$. Indeed, Assume that it is incident to three such vertices, which are the intersections of $c \cap V$ with three respective lines $d_1 \cap V$, $d_2 \cap V$, and $d_3 \cap V$, and which appear in this order along $c \cap V$. Then the middle line $d_2 \cap V$ has to pass below one of the other two vertices, contrary to our assumption.

Set $M := |Q|$, and consider the middle $M/3$ vertices $v_{c,d}$, for $(c,d) \in Q$, when ordered according to the order of their vertical projections on $a \cap b$. The preceding argument implies that they are formed by at least $M/3$ distinct lines. Either at least half of these lines are positive, or at least half of them are negative. Assume, without loss of generality, that at least half of them are positive. Then, for any vertex $v_{c,d}$ among the $M/3$ leftmost vertices that are formed by members of Q, at least $M/6 - 2$ of these lines must pass above $v_{c,d}$ and below $a \cap b$; this is because, by construction, none of these lines can pass below v, at most two pass through v (by the general position assumption), and all the positive lines must pass below $a \cap b$ over $v_{c,d}$. This implies that $M/6 - 2 \le t$, or that $M \le 6(t+2)$, as asserted. $\Box$

Estimating $N_{\le k, \le \ell}$. Based on the preceding lemma, we obtain an upper bound for $N_{\le k,\le \ell}(n)$, which is the maximum possible value of $N_{\le k,\le \ell}$, over all sets H of n planes in $\mathbb{R}^3$ in general position. We do this by applying the Clarkson-Shor technique [11], in the following somewhat nonstandard manner.

Let H be a set of n planes in general position in $\mathbb{R}^3$, and let $k \ge 0$, $\ell \ge 0$, $k + \ell \le n - 4$, be given. If $\ell = 0$, we pass directly to the second stage of the analysis (which just applies Lemma 5.2), so we assume for now that $\ell > 0$. We draw a random sample $R \subset H$, by choosing each plane of H independently, with probability $p = 1/\ell$. The expected size of R is np. Put $t := \lceil k/\ell \rceil$. Let (a,b,c,d) be a quadruple with weight (ξ, η), with $\xi \le k$ and $\eta \le \ell$. The probability that (a,b,c,d) appears in R and that its R-weight (ξ^R, η^R) satisfies $\eta^R = 0$ and $\xi^R \le 2t$, is p^4 (which is the probability of choosing a,b,c,d) times $(1-p)^\eta$ (the probability of not choosing any of the η planes that contribute to the second component of the weight) times the probability $F(p, \xi, 2t)$ of choosing at most $2t$ planes of the ξ planes that contribute to the first component of the weight. The number of these planes in the sample is a binomial random variable whose expectation is $\xi/\ell \le t$. Hence, by Markov's inequality, the probability of choosing more than $2t$ of these planes is at most $1/2$, so $F(p, \xi, 2t) \ge 1/2$. In summary, the overall desired probability is at least

$$p^4(1-p)^\eta F(p,\xi,t) \ge p^4(1-p)^\ell F(p,\xi,t) \ge cp^4,$$

for an appropriate constant $c \approx \frac{1}{2e}$. Hence,

$$\mathbf{E}[N_{\le 2t,0}(R)] \ge cp^4 \sum_{\xi \le k,\, \eta \le \ell} N_{\xi,\eta}(H),$$

which is easily seen to imply that, for $\ell > 0$,

$$N_{\le k, \le \ell}(H) = O(\ell^4 N_{\le 2t,0}(n/\ell)).$$

Substituting the bound in Lemma 5.2, we get

$$N_{\le k,\le \ell}(n) = O\left(\ell^4 \left(\lceil k/\ell \rceil + 1\right)^2 (n/\ell)\right) = O(n(k+\ell+1)^2(\ell+1)),$$

where we have replaced the last factor ℓ by $\ell+1$, to cater also to the case $\ell = 0$, for which the above bound is simply what Lemma 5.2 asserts. That is, we have shown:

Lemma 5.3 *Let H be a set of n non-vertical planes in general position in $\mathbb{R}^3$, and let $k \ge 0$, $\ell \ge 0$, $k+\ell \le n-4$, be given. Then $N_{\le k,\le \ell}(H) = O(n(k+\ell+1)^2(\ell+1))$.*

Note that analogous versions of Lemma 5.3 exist for point sets in $\mathbb{R}^3$, in the context of normal diagrams, and in $\mathbb{R}^2$, in the context of Voronoi diagrams.

Wrapping up. Returning to Equation (1), we now obtain

$$X = O\left(\sum_{k\ge 0,\ \ell\ge 0,\ k+\ell\le n-6} \frac{\Big(4k+10\ell+30\Big) n(k+\ell+1)^2(\ell+1)}{(\ell+1)(\ell+2)(\ell+3)(k+\ell+3)(k+\ell+4)(k+\ell+5)(k+\ell+6)} \right. \tag{2}$$

$$\left. + \sum_{\ell=0}^{n-5} \frac{n(n-4-\ell)^2(\ell+1)}{(\ell+1)(\ell+2)(\ell+3)n(n-1)} + \sum_{\ell=0}^{n-4} \frac{N_{n-4-\ell,\le \ell}}{(\ell+1)(\ell+2)n(n-1)} \right).$$

The second sum is easily seen to be $O(n)$. As for the third sum, another routine application of the Clarkson-Shor technique [11] shows that $N_{n-4-\ell,\le\ell}$ is bounded by $O((\ell+1)^4)$ times the maximum number of configurations (a,b,c,d) in a sample R of expected size $n/(\ell+1)$, such that a segment of the line $a\cap b$ is on the upper envelope of R and a segment of the line $c \cap d$ is on the lower envelope of R. Since the expected complexities of the upper and of the lower envelope of R are both $O(n/(\ell+1))$, we obtain that $N_{n-4-\ell,\le\ell} = O\left((\ell+1)^4 \cdot \frac{n^2}{(\ell+1)^2}\right) = O(n^2(\ell+1)^2)$, Therefore the third sum in Equation (2) is $O(n)$. We thus have

$$X = O\left(\sum_{k\ge 0,\ \ell\ge 0,\ k+\ell\le n-6} \frac{n(k+\ell+1)^3(\ell+1)}{(k+\ell+1)^4(\ell+1)^3} \right) + O(n) =$$

$$O\left(n \cdot \sum_{k\ge 0,\ \ell\ge 0,\ k+\ell\le n-6} \frac{1}{(k+\ell+1)(\ell+1)^2} \right) + O(n) =$$

$$O(n\log n)\cdot \left(\sum_{0\le \ell\le n-6} \frac{1}{(\ell+1)^2} \right) + O(n) = O(n\log n).$$

We have thus shown that the expected number of crossings between the edges of E, and thus the expected complexity of $\mathcal{A}(E)$, is $O(n\log n)$. The derivation of this bound indicates that the significant contribution to this bound is from quadruples with weight (k,ℓ) where $\ell = O(\log n)$. Quadruples with higher values of ℓ contribure only $O(n)$ to X, and thus to the complexity of $\mathcal{A}(E)$, as is easily verified from the last expression.

As argued above, the bound on $\mathcal{A}(E)$ also bounds the expected overall size of the data struture that establishes Theorem 2.1. The expected preprocessing time is $O(n\log n)$, using, e.g., the randomized incremental algorithm of Mulmuley [25] to construct $\mathcal{A}(E)$ or the algorithm of Chazelle and Edelsbrunner [10].

A lower bound for the overlay complexity. To complete the proof of Theorem 5.1, we next provide a construction where the expected complexity of the overlay is $\Theta(n \log n)$.

Consider a set H of $2(n+1)$ planes, partitioned into two sets U and L. The set U consists of $n+1$ planes $\{u_1, \dots u_{n+1}\}$, all tangent from above to a sufficiently small cylinder around the x-axis, so that the directions of their normals span a sufficiently small angle. The set U has the property that, for any subset $U' \subseteq U$, all the planes in U' appear on the lower envelope of U', which consists of $|U'|$ strips parallel to the x-axis, separated by $|U'|-1$ lines parallel and very close to the x-axis.

The set L consists of $n+1$ planes $\{q_1, \dots q_{n+1}\}$, all parallel to the y-axis. It is simpler to present the construction of L by describing the cross-section of their arrangement within the xz-plane, which is depicted in Figure 5. The line representing q_1 emanates from the origin and has a slightly negative slope. Suppose that the lines representing $q_1, \dots, q_i$ have already been constructed. We then construct the line representing q_{i+1} so that it emanates from a point on the x-axis that lies to the right of the intersection point of q_{i-1} and q_i (or of the origin, for $i = 1$), and its slope is slightly smaller (more negative) than that of q_i.

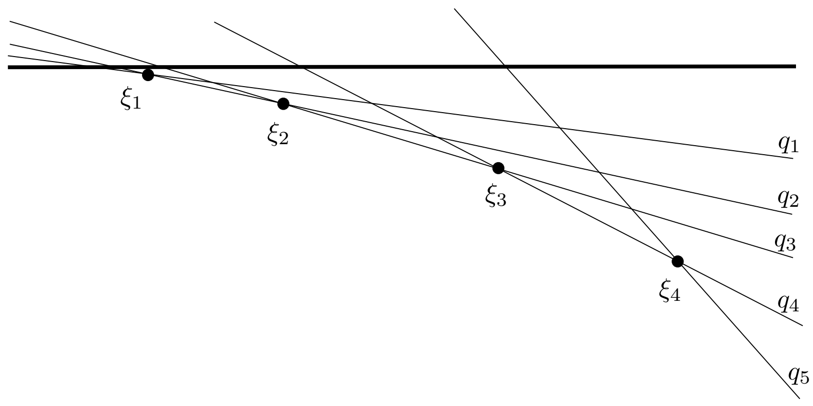

Figure 5: The lower bound construction.

Let ξ_k, for $k = 1, \dots, n$, denote the intersection line of q_k and q_{k+1}. Clearly, ξ_k appears on the lower envelope of any subset $U' \cup L'$, with $U' \subseteq U$, $L' \subseteq L$, such that L' contains both q_k and q_{k+1}. Let ξ_k^* denote the xy-projection of ξ_k.

Let X_k be a random variable that is equal to the number of intersections of ξ_k^* in the overlay of the minimization diagrams that are constructed during a randomized incremental construction of $LE(H)$, at the step when ξ_k first appears on the lower envelope. That is, let $(h_1, \dots, h_{2(n+1)})$ be a random permutation of H. Let i be the index such that either $h_i = q_k$ and $q_{k+1} = h_j$ for some $j < i$, or $h_i = q_{k+1}$ and $q_k = h_j$ for some $j < i$. Then when h_i is added, ξ_k first appears as an edge s_k of the lower envelope of $LE(H_i)$, and X_k counts the number of intersections of the projection s_k^* of s_k with edges of the minimization diagrams $M(H_j)$, for $j < i$.

Clearly s_k^* can intersect only projections of intersections between planes in U. Moreover, s_k^* intersects the projection of the intersection between u_i and u_j, for $i < j$, if and only if (i) u_i and u_j are inserted before any of the planes $u_{i+1}, \dots, u_{j-1}$, and (ii) the first inserted plane among $q_1, \dots, q_{k+1}$ is inserted after both u_i and u_j.

Let A_j, for $j = 1, \dots, n$, be the event that exactly $j+1$ of the planes in U have been inserted before the first plane among $q_1, \dots, q_{k+1}$ has been inserted. As just argued, if A_j happens then s_k^* intersects at least j lines in the overlay of the minimization diagrams.

The event A_j happens if and only if the planes of U and the planes $q_1, \dots, q_{k+1}$ appear in the permutation in the following order. First appear $j+1$ of the planes of U, then any single one of the planes $q_1, \ldots, q_{k+1}$, and then all other planes in $U \cup \{q_1, \ldots, q_{k+1}\}$ in an arbitrary order. Let p_j be the probability that A_j happens. We thus have

$$p_j = \frac{\binom{n+1}{j+1}(j+1)!(k+1)(n+k-j)!}{(n+k+2)!} ,$$

and we have the following lower bound on the expectation $\mathbf{E}(X_k)$:

$$\begin{aligned}
\mathbf{E}(X_k) &\geq \sum_{j=1}^{n} j p_j \\
&= \sum_{j=1}^{n} j \frac{\binom{n+1}{j+1}(j+1)!(k+1)(n+k-j)!}{(n+k+2)!} \\
&= \sum_{j=1}^{n} j \frac{(n+1)!(k+1)(n+k-j)!}{(n-j)!(n+k+2)!} \\
&= \frac{(n+1)!(k+1)}{(n+k+2)!} \sum_{j=1}^{n} j \frac{(n+k-j)!}{(n-j)!} \\
&= \frac{(n+1)!(k+1)!}{(n+k+2)!} \sum_{j=1}^{n} j \binom{n+k-j}{k} \\
&= \frac{1}{\binom{n+k+2}{k+1}} \sum_{j=1}^{n} j \binom{n+k-j}{k} \\
&= \frac{\binom{n+k+1}{k+2}}{\binom{n+k+2}{k+1}} \qquad (3) \\
&= \frac{(n+1)n}{(k+2)(n+k+2)} \\
&\geq \frac{n}{2(k+2)} .
\end{aligned}$$

The equality (3) follows as a special case of the binomial identity (5.26) in [17].[6] Now clearly

$$\sum_{k=1}^{n} \mathbf{E}(X_k) \geq \sum_{k=1}^{n} \frac{n}{2(k+2)} = \Omega(n \log n)$$

is a lower bound on the total expected number of intersections in the overlay of the minimization digrams. This completes the proof of Theorem 5.1. □

Discussion. Since, with high probability, the size of each of the prefix minima sequences $PM(f)$ is $O(\log n)$, it follows that, under the random insertion order, the overlay of the faces of the minimization diagram, normal diagram, or Voronoi diagram is *shallow*: Each point in $\mathbb{S}^2$ or in $\mathbb{R}^2$ lies in at most $k = O(\log n)$ expected number of faces. Using the Clarkson-Shor technique,

[6] One can see it directly by counting the number of ways to choose a subset of size $k+2$ from a set of size $n+k+1$, by enumerating over the possible choices of the second largest item in the sample.

it follows that the complexity of $\mathcal{A}(E)$ is $O(k^2)$ times the expected complexity of the *union* of a random sample of $O(n/k)$ faces. If we could prove a general linear bound on the complexity of the union of such faces, then we would have obtained an alternative proof that the complexity of $\mathcal{A}(E)$ is $O(n \log n)$ (this time with high probability). One possible line of attack would suggest itself if one could argue that the faces of the minimization diagram were *pseudodisks* (i.e., each pair of boundaries intersect at most twice) [21]. However, this is not the case in general, as is illustrated in Figure 6. The figure depicts two Voronoi faces whose boundaries intersect at four points. By creating arbitrarily many appropriate copies of these points, we obtain a construction where the overlay has quadratic complexity. Of course, this cannot be attained in expectation in a *random* insertion order, as we have just shown. Nevertheless, since there is a constant positive probability that the six points shown in Figure 6 are inserted in an order that makes the Voronoi faces of a and of b intersect at four points, it follows that the generated Voronoi faces need not be pseudodisks, even under a random insertion order.

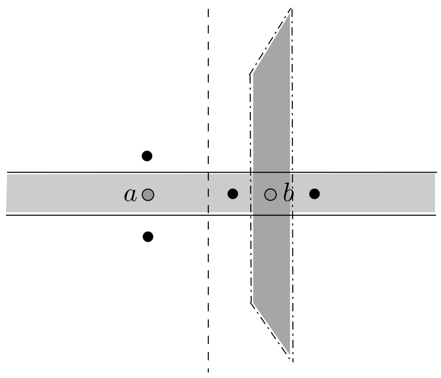

Figure 6: Two Voronoi faces whose boundaries intersect at four points. The point a and the two points near it are inserted first, and then b and the two points near it are inserted.

6 Approximate Range Counting

In this section we exploit the machinery developed in the preceding sections for our main application, stated in Corollary 2.4, to approximate half-space range counting in $\mathbb{R}^3$. Recall that in this application we are given a set P of n points in $\mathbb{R}^3$, which we want to preprocess into a data structure, such that, given a lower halfspace h^- bounded by a plane h, we can efficiently approximate the number of points in $P \cap h^-$ to within a relative error of ε, with high probability (i.e., the error probability should go to zero as $1/\text{poly}(n)$).

We present two solutions. The first is a straightforward consequence of the machinery developed above, and the other combines it with a simple trick that reduces the storage cost by a factor of $O(\log n)$.

The first solution proceeds as follows. As explained in the introduction, following the framework of Cohen [12], we construct $O\left(\frac{1}{\varepsilon^2} \log n\right)$ copies of the data structure provided in Corollary 2.4, each based on a different random permutation of the points, obtained by sorting the points according to the random weights that they are assigned (see the introduction and [12] for details). We now query each of the structures with the query plane h, retrieve in $O(\log n)$ time the point of minimum rank in $P \cap h^-$, and record its weight. We output the reciprocal of the average of these weights as an estimator for the desired count.

The same technique applies to the problem of approximate range counting of points in a query disk in the plane (which, as noted above, is a special case of the problem just discussed).

To reduce the storage cost we employ the following technique. Let P be the given set of n points in $\mathbb{R}^3$. Consider the process that draws one of the $O\left(\frac{1}{\varepsilon^2}\log n\right)$ random permutations π of P. In this process, each point $p \in P$ is independently assigned a random weight $w(p)$ from the exponential distribution, and π sorts the points in the order of increasing weight. Let R denote the set of the first $t := n/\log n$ points in π. Since the weights are i.i.d., R is a random sample of P of size t, where each t-element subset is equally likely to arise. Moreover, conditioned on R assuming a fixed value, the prefix π_t of the first t elements of π is a random permutation of R.

We now construct our data structure for R only, inserting the points of R in their order in π_t, and maintaining the overlay of the resulting normal diagrams, as above. Since π_t is a random permutation of R, the conditional expectation of the complexity of the overlay is $O(t\log t) = O(n)$, which is thus also the value of the *unconditional* expectation. Repeating this for $O\left(\frac{1}{\varepsilon^2}\log n\right)$ permutations, the total expected storage is $O\left(\frac{1}{\varepsilon^2}n\log n\right)$.

A query half-space h^- is processed as follows. For each permutation π and associated prefix R, we find the point of R of minimum rank that lies in h^-. If there exists such a point, it is also the minimum-rank point of $P \cap h^-$ and we proceed as above. Suppose however that $R \cap h^- = \emptyset$. In this case, since R is a random sample of P of size t, the ε-net theory [20] implies that, with high probability, $|P \cap h^-| = O\left(\frac{n}{t}\log t\right) = O(\log^2 n)$. In this case, we can afford to *report* the points in $P \cap h^-$ in time $O(\log^2 n)$, using the range reporting data structures mentioned in the introduction. For example, the algorithm of Chan [8] uses $O(n\log n)$ expected storage, and reports the k points of $P \cap h^-$ in time $O(\log n + k) = O(\log^2 n)$. We then count the number of reported points exactly, by brute force. We proceed in this way if $R \cap h^-$ is empty for at least one of the samples R. Otherwise, we correctly collect the minimum-rank elements in each of the permutations, and can obtain the approximate count as above.

In summary, we thus have:

Theorem 6.1 *Let P be a set of n points in $\mathbb{R}^3$, and let $\varepsilon > 0$ be given. Then we can preprocess P into a data structure of expected size $O\left(\frac{1}{\varepsilon^2}n\log n\right)$, in $O\left(\frac{1}{\varepsilon^2}n\log n\right)$ expected time, so that, given a query halfspace h^-, we can approximate, with high probability, the count $|P \cap h^-|$ to within relative error ε, in $O\left(\frac{1}{\varepsilon^2}\log^2 n\right)$ expected time. The same performance parameters hold for the case of a planar point set, where the queries are disks.*

Discussion. (i) As already discussed, compared with the technique of Aronov and Har-Peled [5], the storage and preprocessing bounds are asymptotically the same in both cases, but the query time is smaller by a factor of $O\left(\log\left(\frac{1}{\varepsilon}\log n\right)\right)$.
(ii) We remark that the general machinery of Cohen [12] can be applied to any range space. However, it requires a black-box procedure for efficiently finding the minimum rank of the data elements in a query range, according to a random permutation. It is an interesting open challenge to design data structures of this kind for other approximate geometric (and non-geometric) range counting problems. Some solutions to this problem are suggested by Aronov and Sharir [6].

References

[1] P.K. Agarwal and J. Erickson, Geometric range searching and its relatives, in *Advances in Discrete and Computational Geometry* (B. Chazelle, J. E. Goodman and R. Pollack, Eds.), AMS

Press, Providence, RI, 1998, pp. 1–56.

[2] P.K. Agarwal, J. Erickson, and L. Guibas, Kinetic binary space partitions for intersecting segments and disjoint triangles, *Proc. 9th Annu. ACM-SIAM Sympos. Discrete Algo.*, 1998, 107–116.

[3] P.K. Agarwal, L.J. Guibas, T.M. Murali, and J.S. Vitter, Cylindrical static and kinetic binary space partitions, *Comput. Geom. Theory Appl.* 16 (2000), 103–127.

[4] P.K. Agarwal and J. Matoušek, On range searching with semialgebraic sets, *Discrete Comput. Geom.* 11 (1994), 393–418.

[5] B. Aronov and S. Har-Peled, On approximating the depth and related problems, *Proc. 16th Annu. ACM-SIAM Sympos. Discrete Algo.*, 2005, 886–894.

[6] B. Aronov and M. Sharir, Approximate range counting, in preparation.

[7] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer verlag, Heidelberg, 2000.

[8] T. M. Chan, Random Sampling, Halfspace Range Reporting, and Construction of ($\leq k$)-Levels in Three Dimensions, *SIAM J. on Comput.* 30 (2000), 561–575.

[9] B. Chazelle, *The Discrepancy Method*, Cambridge University Press, Cambridge, UK, 2000.

[10] B. Chazelle and H. Edelsbrunner, An optimal algorithm for intersecting line segments in the plane, *J. Assoc. Comput. Mach.*, 39 (1992), 1–54.

[11] K. Clarkson and P. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* 4 (1989), 387–421.

[12] E. Cohen, Size-estimation framework with applications to transitive closure and reachability, *J. Comput. Syst. Sci.* 55 (1997), 441–453.

[13] E. Cohen and H. Kaplan, Spatially-decaying aggregation over a network: model and algorithms, *SIGMOD '04: Proc. 2004 ACM SIGMOD Internat. Conf. on Management of Data*, 2004, 707–718.

[14] J. R. Driscoll, N. Sarnak, D. D. Sleator and R E Tarjan, Making data structures persistent, *J. Comput. System Sci.* 38 (1989), 86–124.

[15] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[16] H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements, *Discrete Comput. Geom.* 1 (1986), 25–44.

[17] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*, Addison-Wesley, Reading, MA, 1994.

[18] L. Guibas, D. E. Knuth, and M. Sharir, Randomized incremental construction of Voronoi and Delaunay diagrams, *Algorithmica* 7 (1992), 381–413.

[19] M. E. Houle and G. T. Toussaint, Computing the width of a set, *IEEE Trans. Pattern Anal. Mach. Intell.* 10 (1988), 761–765.

[20] D. Haussler and E. Welzl, Epsilon-nets and simplex range queries, *Disc. Comput. Geom.* 2 (1987), 127-151.

[21] K. Kedem, R. Livne, J. Pach and M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, *Discrete Comput. Geom.* 1 (1986), 59–71.

[22] J. Matoušek, Reporting points in halfspaces, *Comput. Geom. Theory Appl.* 2 (1992), 169–186.

[23] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete Comput. Geom.* 10 (1993), 157–182.

[24] J. Matoušek, *Geometric Discrepancy*, Algorithms and Combinatorics, Vol. 18, Springer Verlag, Heidelberg, 1999.

[25] K. Mulmuley, A fast planar partition algorithm I, *J. of Symbolic Computation*, 10 (1990), 253-280.

[26] J. Pach and P.K. Agarwal, *Combinatorial Geometry*, Wiley Interscience, New York, 1995.

[27] E. A. Ramos, On range reporting, ray shooting and k-level construction, *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, (1999), 390–399.

[28] N. Sarnak and R.E. Tarjan, Planar point location using persistent search trees, *Comm. ACM* 29 (1986), 669–679.

[29] M. Sharir and H. Shaul, Ray shooting amid balls, farthest point from a line, and range emptiness searching, *Proc. 16th Annu. ACM-SIAM Sympos. Discrete Algo.*, 2005, 525–534.