Incremental k-Lowest Planes and Planar k-Nearest Neighbor with Optimal Query Time

John Iacono
Université libre de Bruxelles, Belgium

Yakov Nekrich
Michigan Technological University, Houghton, MI, USA

Martin P. Seybold
Faculty of Computer Science, Theory and Applications of Algorithms, University of Vienna, Austria
Université libre de Bruxelles, Belgium

Abstract

In a set of planes in $\mathbb{R}^3$, the *k-lowest planes query* asks for the k lowest planes pierced by a vertical line q. In this paper we describe a semi-dynamic insertion-only data structure that answers k-lowest planes queries in optimal $O(\log n + k)$ time. Our data structure uses $O(n)$ space, where n is the number of stored planes, and supports insertions in $O(\log^8 n)$ amortized time. This result provides the first query optimal data structures for several fundamental problems:

- An insertion-only structure that answers 3D halfspace range reporting queries on a set of n points in $O(\log n + k)$ time, where k is the number of reported points.
- An insertion-only structure that answers 3D vertical ray shooting queries on a set of n planes in $O(\log n + k)$ time, where k is the number of reported planes.
- An insertion-only structure that answers planar k-nearest neighbor queries in $O(\log n + k)$ time for any prescribed k (specified at query time).
- An insertion-only structure that answers planar circular range reporting queries in time $O(\log n + k)$, where k is the number of reported points.

For all of the above problems the query bound $O(\log n + k)$ is optimal, even in the static scenario. All of the above structures use linear $O(n)$ space and support insertions in $O(\log^8 n)$ amortized time.

We also obtain a query optimal static structure for a 4D problem. That is, one can compute in near-linear time a data structure that answers weighted halfspace range reporting queries in $O(\log n + k)$ time, where k is the number of reported points. The structure uses near-linear space.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Data Structures, Dynamic Data Structures, k Nearest-Neighbor Queries

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.113

Category Track A: Algorithms, Complexity and Games

Funding Work supported by the Fonds de la Recherche Scientifique – FNRS. Supported by the National Science Foundation under NSF grant 2203278. This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/PAT2190825.

Acknowledgements Many thanks to all contributors of Vorosketch and Ipe.

1 Introduction

Data structures for sets of 3D planes have been studied extensively. Some important examples of such problems are 3D halfspace range reporting [1] and dynamic 3D convex hull [5].

In this paper we study the following k-lowest planes problem: A set of three-dimensional planes is stored in a data structure. For any query tuple (q, k), where q is a vertical line and k is a positive integer, we must report the k-lowest planes pierced by q. We investigate

this problem in the semi-dynamic insertion-only scenario and describe a data structure with $O(\log^8 n)$ amortized insertion time that supports k-lowest planes queries in optimal $O(\log n + k)$ time. Using this, we obtain improved solutions for extensively studied problems.

Results. Our main result is the insertion-only data structure that answers k-lowest planes queries in optimal time $O(\log n + k)$. This data structure can be used to obtain optimal query time for several semi-dynamic and static problems. Henceforth we use n to denote the number of geometric objects (points, planes) stored in the data structure and we use k to denote the number of geometric objects that are reported when we answer a query.

1. **Vertical ray shooting in a set of 3D planes.** A vertical ray shooting query q, for a three-dimensional point q, asks for all planes that are pierced by the downward vertical ray from q. We obtain a semi-dynamic insertion-only data structure that answers vertical ray shooting queries in $O(\log n + k)$ time.
2. **Halfspace range reporting in a set of 3D points.** A halfspace range reporting query h asks for all points that are below a plane h. We obtain a semi-dynamic insertion-only data structure that answers halfspace range reporting queries in $O(\log n + k)$ time.
3. **Nearest neighbor in a set of planar points.** A nearest neighbor query q, for a two-dimensional point q, asks for the point that is closest to q. A k-nearest neighbor query (q, k), for a point q and integer $k \geq 1$, asks for k points that are closest to q. We describe a semi-dynamic insertion-only data structure that answers k-nearest neighbor queries in $O(\log n + k)$ time for any k; integer k can be specified at query time.
4. **Circular range reporting in a set of planar points.** A circular range reporting query (q, R) asks for all points in the circle with center q and radius R. We obtain a semi-dynamic insertion-only data structure that answers circular range reporting queries in $O(\log n + k)$ time.
5. **Weighted halfspace range reporting.** In this generalization of halfspace reporting, each 3D point has an additional coordinate called *weight*. A weighted halfspace reporting query (h, w_1, w_2), for a 3D plane h and reals w_1 and w_2, asks for all points that are below h and have weight within $[w_1, w_2]$. We describe a static data structure that answers weighted halfspace reporting queries in $O(\log n + k)$ time.
   (This result is somewhat unexpected because weighted halfspace reporting is a 4D problem. For comparison, the fastest currently known data structure for orthogonal range reporting in 4D answers queries in $O(\log n \log\log n + k)$ time in the pointer machine model [22].)

All semi-dynamic data structures use $O(n)$ space and support insertions in $O(\log^8 n)$ amortized time. Our static structure for weighted halfspace reporting uses $O(n \log^9 n)$ space.

Related Work. The complexity of the k-lowest planes problem, and almost all its applications that we consider in this paper, are well understood in static scenarios. If k is fixed at preprocessing time, we can work with the k-level of the set of planes. The k-level of a set of n planes is a polyhedral terrain of complexity $O(n \cdot k^{3/2})$ [23]. The fastest known deterministic construction takes $O(n \log n + nk^{3/2})$ time [8]. In order to answer the k-lowest planes query it is sufficient to find the face of the k-level that is pierced by q, which can be solved in $O(\log n)$ time. If k is prescribed at query time, state-of-the-art solutions are based on a hierarchy of k_i-shallow cuttings with values k_i that form a geometric series. Chan and Tsakalidis [9] showed that such a hierarchy can be computed in $O(n \log n)$ deterministic time, which yields a simple data structure with $O(\log n + k)$ query time. The space bound of this approach can be made $O(n)$, while retaining the query bound, using 3D halfspace range reporting queries from the (recursive) structure of Afshani and Chan [1], which was derandomized in [9].

Halfspace range reporting with $O(\log n + k)$ query time and polylog(n) updates is only known for 2D, which degrades to $O(\log^2 n/\log\log n + k)$ query time in 3D [6, Section 3], since interval trees and fractional cascading do not generalize [11].

For weighted halfspace reporting, one can use a range tree on the weights of points to obtain a data structure with $O(n \log n)$ space and $O(\log^2 n + k)$ query time. Since this variant of halfspace reporting is a 4D problem, this query time appears to be close to optimal. However, the very recent data structure of Afshani et al. [2] supports weighted halfspace reporting queries in $O(\log^{3/2} n + k)$ time, for a given query plane h and weight range $[w_1, w_2]$.

If insertions and deletions are required, one can answer 1-nearest neighbor queries in $O(\log^2 n)$ time with polylogarithmic updates [7]. In [13] de Berg and Staals extended this result to k-nearest neighbor and obtained a data structure with $O(\log^2 n/\log\log n + k)$ query time. Improving the query time of these works is challenging; most progress has been made on update time [5, 19, 7].

Recently, Iacono and Nekrich [17] described a novel, insertion-only structure that answers 1-nearest neighbor queries in $O(\log n)$ time with $O(\log^{1+\varepsilon} n)$ amortized insertion time, where $\varepsilon > 0$ can be made an arbitrary small constant.

Contribution and Overview. This work is inspired by the $O(\log n)$ query time result in [17] for incremental planar 1-nearest neighbor, that introduced a sampling technique that allows maintaining the optimal $O(\log n)$ query time, with $O(\log^{1+\varepsilon} n)$ amortized insertion cost. However, immediate extension to the incremental k-lowest planes problem is quite unclear, already for $k = 1$, since the algorithms and correctness arguments heavily rely on Euclidean distances and geometric observations about the Voronoi cells around points.

One contribution of our work is to simplify their approach into a more general framework, that applies to the incremental k-lowest planes problem. Specifically, we show in Section 3 that query correctness, and the $O(\log n)$ query time, can be established based on a suitable implicit representation of (sample augmented) sets of planes. This is somehow surprising, as a query may, for example, ask for the $\lceil \log^{3/2} n \rceil$-lowest planes in some q. In obtaining the explicit answer of the reporting query, we are tackling the difficulty that our technique, which gives the speedup, prevents direct access to the known Heap-Selection techniques from [13, 15, 18]. To solve this, we introduce a Deduplication Lemma in Subsection 3.1 for such sampling techniques, which allows us to obtain the optimal $O(\log n + k)$ query time.

Our Section 4 shows how to reduce space to $O(n)$, which we believe is of general interest for several dynamic problems (beside k-lowest planes). Section 5 shows how to obtain the first optimal bounds for aforementioned applications 1-5 from our proposed method.

We see the obtained generalization to the k-lowest problem and the Deduplication Lemma as the main technical contributions of this work. Furthermore, our proofs in Section 3 proceed in three abstract steps (i) query correctness, (ii) construction time, and (iii) query runtime, which makes our framework easily accessible for future work on optimal query time.

Though our $O(\log^8 n)$ insertion bound for the k-lowest planes problem is considerably larger than the almost logarithmic bound for planar 1-nearest neighbor from [17], we believe that it is very challenging to improve the insertion bound to a sub-cubic exponent (see proof of Lemma 5). While it is an interesting technical problem to accelerate our Skip-Decider construction, one barrier would remain. A central obstacle that comes with our approach is the apparent complexity of the $\lceil \log^2 n \rceil$-level of a set of n planes, where the best known bound $O(n \log^3 n)$ remained unchanged for decades [23].

The new *optimal* bounds improve the state-of-the-art for fundamental problems planar k-nearest-neighbor [13], 3D halfspace reporting [6, 7], and weighted halfspace reporting [2].

2 Preliminaries: The k-Level of Planes and Planar Triangulations

We rely on the standard, general position assumption: Any point $p \in \mathbb{R}^3$ is contained in $|\{h \in H : p \in h\}| \leq 3$ planes, and no plane in H contains a vertical line. For example, any vertical query line q intersects all $|H|$ planes. The at most k-level of a set of planes H, denoted by $L(k, H)$, is the set of points from $\mathbb{R}^3$ that is the closure of the region of points that have $< k$ planes below it. The k-level $L(k, H)$ is a polyhedral surface with $O(|H|k^{3/2})$ vertices [3, Sec. 3][23], and the fastest known algorithm computes it in deterministic $O(|H| \log |H| + |H|k^{3/2})$ time [8]. Recall that the k-level of planes, tangent to the (downward opened) lifting paraboloid [12, Sec. 11.5], is in one-to-one relation to the planar order-k Voronoi diagram.

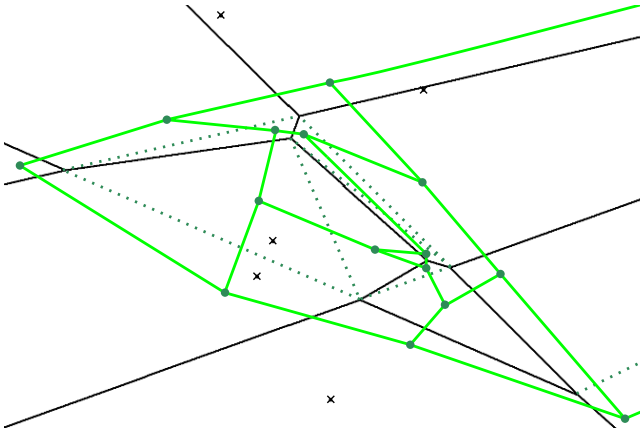

Figure 1 Order-2 Voronoi diagram of fives sites (black), its triangulation (dotted green lines), and triangle-adjacency graph (green).

We will frequently project (sets of) points in 3D with an orthogonal projection down into the xy-plane at $z = -\infty$, and work with 2D objects in that plane. This will simply be referred to as *projection* (into the plane) throughout the entire paper.

In our insertion algorithm, we will frequently compute the $\bar{k}$-level of certain subsets $H' \subseteq H$ where $\bar{k}$ is a fixed integer, project it into the plane, and triangulate the faces. The number of faces in the triangulation is near-linear $O(|H'|\bar{k}^{3/2})$ in $|H'|$ for small $\bar{k}$. Any face of the triangulation shares a common boundary with two or three other faces, and any unbounded face is bounded by two rays and possibly one segment. We call the unbounded faces degenerate triangles, as they may contain only one or two corner points. (See Figure 1.) For a vertical query line that intersects an edge of the k-level, we allow either set of k-lowest planes from the faces as query result. That is, we allow that ties are broken arbitrarily in reporting queries, and thus assume that queries intersect triangles in their *interior*.

We will also use algorithms for computing separators in a planar graph with n nodes [15, 16, 21], specifically in the triangle-adjacency graph described above. Using [21, Sec. 2.5], for given integer $r < n$ one can compute in $O(n)$ time a decomposition in $O(n/r)$ connected components, where each connected component contains $O(r)$ nodes, from which $O(\sqrt{r})$ are in the separator. Such a separator has $O(n/\sqrt{r})$ nodes.

Let us remark that points on the k-level with equal sets of planes below are connected; see Lemma 1. We denote the k-lowest planes of H in vertical line q by k-lowest(q, H).

▶ **Lemma 1** (Convexity). *Let q_1 and q_2 denote the vertical lines through some points q_1^* and q_2^* respectively. Let q^* be an arbitrary point on the segment $\overline{q_1^* q_2^*}$, spanned by q_1^* and q_2^* and let q denote the vertical line through the point q^*.*

If k-lowest(q_1, H) =k-lowest$(q_2, H) = H'$, then k-lowest$(q, H) = H'$.

(See Appendix A for a standard proof.)

3 Optimal k-lowest Queries with Fast Plane Insertion

Our insertion-only data structure is based on the Bentley-Saxe (BS) partial-rebuilding approach for decomposable search problems [4]. There, a list of subsets that *partition* the semi-dynamic set $H = H_f \cup \ldots \cup H_0$ is maintained, with one static structure for each non-empty H_i. Let integer $N = |H_f|$ denote the number of planes at time of the last *complete* rebuild of one static structure ($N = |H|$ and $H_i = \emptyset$ for $i < f$). Taking $d := \lceil \log_2 N \rceil$, and maintaining set sizes $|H_i| = O(d^{i+1})$ leads to $f = O(\frac{\log n}{\log \log n})$ sets in the partition, and amortized insertion cost $O(fg)$, where g bounds the ratios $P(|H_i|)/d^i$ of the construction time P, of static structures for H_i, after d^i insertions.

Static data structures with $O(\log n + k)$ query time can be obtained from the known deterministic $O(n \log n)$ time construction of (a hierarchy of) shallow cuttings in [9]. The $O(n \log n)$ space bound can be reduced to $O(n)$ by on-the-fly recomputation of conflict-lists with a known optimal half-space range-reporting structure [1, 9]. (This idea is credited to Afshani after Corollary 10 in [7].) Running the standard linear time k-Selection algorithm [14, Sec. 1.8] then yields the k planes that are lowest in vertical line q in $O(\log n + k)$ time. Using this optimal static structure, combined with the Heap-Selection method of [13, Thm. 1], immediately yields $O(\log^2 n + k)$ query time for k-lowest planes in the insertion-only setting. This solution is optimal for queries with *large* k.

Thus, the remainder of this section is concerned with the difficult cases of $k = o(\log^2 n)$, though we will combine our approach (Subsection 3.1) with the structure from [9] and Heap-Selection from [13], and in Section 4 with the known, optimal static structure above.

Let $\bar{k} := \lceil \log_2^2 N \rceil$ throughout the entire paper. Our data structure contains:

- $H_0, \ldots H_f$: A partition of the semi-dynamic set H. Set H_i has size $O(d^{i+1})$.
- $\hat{H}_0, \ldots \hat{H}_f$: Set $\hat{H}_i$ contains all planes of H_i as well as some from the sets H_j, $j > i$. The added planes serve a similar purpose as samples in classical fractional cascading.
- D_i: A triangulation of the projection of the $\bar{k}$-level of $\hat{H}_i$.
- For any $\Delta \in D_i$, $H(\Delta)$ is the set of $\bar{k}$-lowest planes from $\hat{H}_i$ for any q that intersects Δ.
- $\mathsf{SMPL}_i(\ell)$ is a subset of the triangles of D_i. The planes $\cup_{\Delta \in \mathsf{SMPL}_i(\ell)} H(\Delta)$ are in $\hat{H}_{i-\ell}$.

Description here omits additional search structures attached to each set $\hat{H}_i$, and each triangle $\Delta \in D_i$. The full details are described in Lemma 5, which shows that they can be computed in near-linear time. Our query method is a greedy trial-and-error approach, that relies on two functions FIX and SKIP, that we formalize in Lemma 2 and Lemma 4.

Intuitively, any greedy-step assumes that a "tentative result" for $[0, i]$ is "correct" for even larger prefixes $[0, i + \ell]$ of BS sets. For certain extension lengths ℓ, this assumption is checked "quickly" with an approximate decider[1] SKIP. On rejection, the FIX step then "quickly" computes a new tentative result, and index $i' > i$, such that the result is correct for the prefix $[0, i']$. Transition from i to i' is called a greedy-step, and it involves several SKIP and one FIX calls. This completes the abstract high-level description of our query method, which we formalize in detail below.

[1] Suitable approximate deciders have a one-sided error, i.e. have reliable True answers but may return False even if a tentative result is valid for the prefix.

Next, we specify what "quickly" means for a greedy-step in the context of BS sets, after which we will specify what "tentative result" and "correct for a prefix" means in the context of our search problem, $\bar{k}$-lowest planes in q.

Doing a greedy-step "quickly". As it turns out for the $\bar{k}$-lowest planes problem, the structures obtained from algorithm CONSTRUCT in Lemma 5 will allow performing one greedy-step, from i to i', in $(i'-i)O(\log d)$ time (cf. Lemmas 6 and 7). That is, the total query time then telescopes to optimal

$$\forall\ i_0 < i_1 < \ldots < i_m\ :\ \sum_{j<m} (i_{j+1} - i_j)O(\log d) = (i_m - i_0)O(\log d) \tag{1}$$

$$= O(f \log d) = O\left(\frac{\log N}{\log d} \log d\right) \tag{2}$$

time, regardless of the number m of greedy-steps. With respect to the $\bar{k}$-lowest planes problem, the proposed greedy trial-and-error approach for queries can be described concisely with the pseudo-code in Figure 2.

```
1  QueryLowest(k, q):
2     Let Δ ∈ D_i be intersected by q, i minimal
3     seq := (Δ)
4     WHILE i < f {
5        j:=0; WHILE SKIP(Δ, q, 2^j) { j++ }
6        (i', Δ') := FIX(Δ, q, 2^j)
7        i := i'; Δ := Δ'; append Δ to seq
8     }
9     RETURN PRUNE(k, q, seq)
```

Figure 2 Greedy algorithm for k-lowest queries in pseudo-code form. See below for SKIP and FIX.

A naïve implementation for the post-processing $\mathsf{PRUNE}(k, \ldots)$ runs in $O(\bar{k} \log_d N)$ time, since each of $O(\log_d N)$ triangles in the sequence has $\bar{k}$ elements in $H(\Delta)$.

In the following lemma, we define the semantics of $\mathsf{SKIP}(\Delta, q, \ell)$ and what an evaluation to True implies for $\bar{k}$-lowest$(q, \hat{H}_i \cup \ldots \cup \hat{H}_{i+\ell})$. After this, Lemma 4 will then discuss the semantic of FIX and what "tentative result" and "correct for a prefix" means. Discussion of how quickly SKIP and FIX can be computed with suitable data structures is carried out in Lemmas 5, 6, and 7.

▶ **Lemma 2** (SKIP-Correctness). *Let indices $i < \ell$, triangle $\Delta \in D_i$ intersect q in point $q^* \in \mathbb{R}^2$, planar region $S_j = \bigcup_{\Delta' \in \mathsf{SMPL}_j(j-i)} \Delta'$ for any $j > i$, and $H(\Delta') \subseteq \hat{H}_i \setminus H_i$ for all triangles $\Delta' \in \mathsf{SMPL}_j(j-i)$.*

Define $\mathsf{SKIP}(\Delta, q, \ell) \iff q^* \in \bigcap_{j=i+1}^{i+\ell} \mathsf{hull}(\Delta \cap S_j)$.

If $\mathsf{SKIP}(\Delta, q, \ell) = \mathsf{True}$, *then* $\bar{k}$*-lowest*$(q, \hat{H}_i) = \bar{k}$*-lowest*$(q, \hat{H}_i \cup \ldots \cup \hat{H}_{i+\ell})$.

Proof. We call degenerate triangles just triangles in the following, as the arguments will hold regardless of corner count. It suffices to show for arbitrary $j > i$, that

$$q^* \in \mathsf{hull}(\Delta \cap S_j) \implies \bar{k}\text{-lowest}(q, \hat{H}_i) = \bar{k}\text{-lowest}(q, \hat{H}_i \cup \hat{H}_j)\ . \tag{3}$$

Since $\Delta \in D_i$ is the triangle that intersects vertical line q, we have $\bar{k}$-lowest$(q, \hat{H}_i) = H(\Delta)$. See Figure 3.

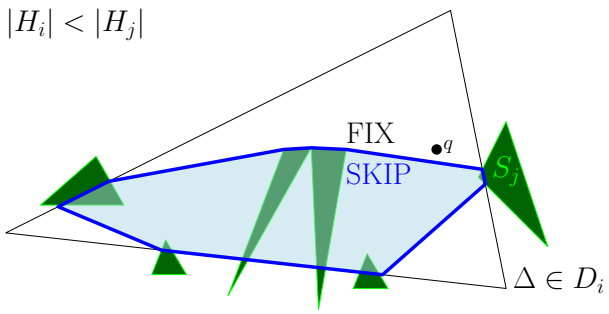

Figure 3 $\Delta \in D_i$ intersecting q (black), $S_j \subseteq \mathbb{R}^2$ of some $j > i$ (green), and $\mathsf{hull}(\Delta \cap S_j)$ (blue).

Consider the set of points on the polyhedral surface $A \subseteq L(\bar{k}, \hat{H}_i \cup \hat{H}_j)$ where the $\bar{k}$-lowest planes coincide with $H(\Delta)$. Let $A^* \subseteq \mathbb{R}^2$ be the orthogonal projection of A. (In other words, the region of queries that may be skipped as their set of $\bar{k}$-lowest planes is equal to $H(\Delta)$.) The implication (3) follows, if we can show that $\mathsf{hull}(\Delta \cap S_j) \subseteq A^*$ is true.

If $\Delta \cap S_j = \emptyset$, then $\mathsf{hull}(\Delta \cap S_j) = \emptyset$ and the implication is true[2]. So let $q' \in \Delta \cap S_j$ be arbitrary. Since $q' \in \Delta'$ of some $\Delta' \in \mathsf{SMPL}_j(j-i)$, we have $\bar{k}\text{-lowest}(q', \hat{H}_j) = H(\Delta')$ are in the augmented set, i.e. $H(\Delta') \subseteq \hat{H}_i \setminus H_i$. From $q' \in \Delta$, we have that $\bar{k}\text{-lowest}(q', \hat{H}_i) = H(\Delta)$. Thus, $q' \in A^*$ and $A^* \neq \emptyset \neq A$.

The lemma now follows from A^* being convex, due to A being convex (Lemma 1). ◀

In the following, we define the necessary semantic of FIX, "tentative result", and "correct for a prefix", which allows us to establish the correctness of our greedy trial-and-error approach for the $\bar{k}$-lowest planes problem. Establishing runtime efficiency, with suitable data structures, is carried out in Lemmas 5, 6, and 7.

▶ **Definition 3** (Tentative Result, Prefix-Validity). *A sequence of triangles $(\Delta_{i_0}, \Delta_{i_1}, \ldots, \Delta_{i_m})$ is called a tentative result for prefix $[0, i]$, if $\Delta_{i_j} \in D_{i_j}$ and $0 \leq i_0 < i_1 < \ldots < i_m = i$. Let $\bar{H} = \bigcup_{j=0}^{m} H(\Delta_{i_j})$ denote the planes appearing in the tentative result.*

A tentative result is valid for prefix $[0, i]$, if $\bar{k}\text{-lowest}(q, \bar{H}) = \bar{k}\text{-lowest}(q, \hat{H}_0 \cup \ldots \cup \hat{H}_i)$.

▶ **Lemma 4** (FIX-Correctness). *Let $\Delta \in D_i$ be the triangle intersecting q, and $j \geq 0$ the smallest integer with $\mathsf{SKIP}(\Delta, q, 2^j) = \mathsf{False}$.*

Define $\mathsf{FIX}(\Delta, q, 2^j)$ to return pair (i', Δ'), such that triangle $\Delta' \in D_{i'}$ intersects q and index $i' \in [i + 2^j, i + 2^{j+1})$ is the smallest index with $\mathsf{SKIP}(\Delta, q, i') = \mathsf{False}$.

If $(\Delta_{i_j})_{j \leq m}$ is valid for prefix $[0, i]$ and $(i', \Delta') = \mathsf{FIX}(\Delta_{i_m}, q, 2^j)$, then $(\Delta_{i_0}, \ldots, \Delta_{i_m}, \Delta')$ is valid for prefix $[0, i']$.

In particular, defined semantics for SKIP and FIX maintain the prefix-validity throughout any greedy-step of the query algorithm. That is, the final triangle sequence is valid for $[0, f]$.

Proof. The proof is by induction. The prefix invariant holds for $[0, i]$ for some $i \geq 0$, since the query algorithm always determines the intersected triangle in D_i with minimal index i in the BS partition as the first triangle in a tentative result. Let $\Delta \in D_i$ be the last triangle in the sequence that is valid for $[0, i]$. Apply Lemma 2, and use that i' is minimal, and $\Delta' \in D_{i'}$ with $q \cap \Delta' \neq \emptyset$, so true for $[0, i']$. ◀

[2] In other words, SKIP returns False in this case.

Next, we discuss the algorithm CONSTRUCT for computing $\hat{H}_i$ and its associated triangulation D_i, including how the triangles $\mathsf{SMPL}_i(j) \subseteq D_i$ are obtained for all possible integers $j \in [1, i]$, and the three kinds of planar point-location structures associated to D_i that we will need to establish that SKIP and FIX can be computed sufficiently fast during queries.

▶ **Lemma 5** (Construction). *There is an algorithm* $\mathsf{CONSTRUCT}(\{H_0, \dots H_i\}, \{D_{i+1}, \dots D_f\})$ *that computes a planar triangulation* D_i *of projected* $L(\bar{k}, \hat{H}_i)$*, where* $|\hat{H}_i| = O(|H_i|)$*.*

The triangulation has size $|D_i| = O(|H_i|\bar{k}^{3/2})$*, and is augmented with the structures:*

1. *Global-Search: A point-location structure for determining a triangle* $\Delta \in D_i$*.*
2. *Samples: For each integer* $\ell \in [1, i]$ *a subset* $\mathsf{SMPL}_i(\ell) \subseteq D_i$ *of triangles, such that every connected component of* $D_i \setminus \mathsf{SMPL}_i(\ell)$ *has* $O(d^{16+2\ell})$ *triangles, is adjacent to* $O(d^{8+\ell})$ *triangles from* $\mathsf{SMPL}_i(\ell)$*, and the separator contains* $|\mathsf{SMPL}_i(\ell)| = O(|D_i|/d^{8+\ell})$ *triangles.*
3. *Fix-Search: For each integer* $\ell \in [1, i]$ *and each connected component of* $D_i \setminus \mathsf{SMPL}_i(\ell)$*, a point-location structure for the triangles in that component.*
4. *Fix-Pointer: For every* $\Delta \in D_i$ *and integer* $\ell \in (i, f]$*,* $O(1)$ *pointers identifying the connected component of* $D_\ell \setminus \mathsf{SMPL}_\ell(\ell - i)$ *containing the corners of* Δ*.*
5. *Skip-Decider: For each* $\Delta \in D_i$ *and* $2^j \in [1, f - i]$ *one point-location structure, for the overlay of* $\{\mathsf{hull}(\Delta \cap S_{i'}) : i' \in (i, i + 2^j]\}$*, with worst-case query time* $O(2^j \log d)$*.*

Construction of these data structures takes $O(|D_i| \log^3 N)$ *time, they occupy* $O(f|D_i|)$ *space.*

Proof. We first assemble H_i by merging the old prefix sets $\bigcup_{\ell \le i} H_\ell$ from the partition, drop all structures related to the old prefix sets, and then add all planes from the sampled triangles from higher indices to obtain the augmented set

$$\hat{H}_i = H_i \cup \bigcup_{\ell > i} \bigcup_{\Delta \in \mathsf{SMPL}_\ell(\ell - i)} H(\Delta) \quad .$$

Then we compute $\bar{k}$-level $L(\bar{k}, \hat{H}_i)$, project its faces into the plane, and compute its triangulation D_i, which takes $O(|\hat{H}_i| \log^3 N)$ deterministic time, since $\bar{k} = O(\log^2 N)$.

We will show that the size of the augmented set remains of the same order $|\hat{H}_i| = O(|H_i|)$, by showing that $\bar{k}|\mathsf{SMPL}_\ell(\ell - i)|$ is sufficiently small in the following.

Samples. The number of triangles in the triangulation is at most $|D_i| = O(|\hat{H}_i|\bar{k}^{3/2})$, i.e. bounding the nodes in its planar face-adjacency graph (cf. Section 2). Next, we compute a triangle subset $\mathsf{SMPL}_i(\ell)$ from the face-adjacency graph for each integer $\ell \in [1, i]$. That is, we compute in linear time a planar-separator $\mathsf{SMPL}_i(\ell) \subseteq D_i$ such that each connected component of $D_i \setminus \mathsf{SMPL}_i(\ell)$ contains $\le r_\ell := d^{16+2\ell}$ triangles, is adjacent to $O(d^{8+\ell})$ triangles from the separator, and contains a number of triangles that is at most $|\mathsf{SMPL}_i(\ell)| \le O(|D_i|/d^{8+\ell})$. Note that invocations with $r_\ell \ge |D_i|$ are trivial, as this yields one connected component containing all nodes and empty separator $\mathsf{SMPL}_i(\ell) = \emptyset$. (E.g. any skip-range containing such an empty $\mathsf{SMPL}_i(\ell)$ evaluates to False.)

To show the claimed $|\hat{H}_i| = O(|H_i|)$, we first consider the right-most BS set ($i = f$), where $H_f = \hat{H}_f$ and $|H_f| = N = O(d^f)$ holds.

Since $|D_f| = O(|H_f|\bar{k}^{3/2}) = O(N \log^3 N)$, the number of planes in the down-samples at index distance ℓ is $\bar{k}|\mathsf{SMPL}_f(\ell)| = O(Nd^5/d^{8+\ell})$.

Thus, $|\hat{H}_{f-1}| \le |H_{f-1}|(1 + \frac{O(|H_f|d^5/d^{8+1})}{|H_{f-1}|}) \le |H_{f-1}|(1 + \frac{O(d^{f+5-9})}{d^{f-1}}) = O(|H_{f-1}|)$.

Using this, we have for $|\hat{H}_{f-2}| \leq |H_{f-2}|(1 + \frac{O(|\hat{H}_{f-1}|d^{5-(8+1)})+O(|H_f|d^{5-(8+2)})}{d^{f-2}})$, that $|\hat{H}_{f-2}| \leq |H_{f-2}|(1 + \frac{O(d^{f+5-9})+O(d^{f+5-10})}{d^{f-2}})$. Thus, $|\hat{H}_{f-2}| = O(|H_{f-2}|)$. Consequently,

$$|\hat{H}_i| \leq |H_i|\Big(1 + \frac{1}{d^i}\underbrace{\sum_{\ell=1}^{f-i} O(|\hat{H}_{i+\ell}|d^5/d^{8+\ell})}_{\leq f\cdot O(d^{i-2})=O(d^{i-1})}\Big) = O(|H_i|) \ .$$

That is, $|D_i| = O(|H_i|\bar{k}^{3/2})$, as claimed.

The planar separator algorithm takes linear $O(|D_i|)$ time for one execution, and is executed $\ell \leq i = O(f)$ times. Thus computing the samples takes $O(f|D_i|)$ total time, also bounding the space for storing the obtained triangle subsets $\{\mathsf{SMPL}_i(\ell) \subseteq D_i\}$.

Global-Search. Computing a planar point-location structure for triangulation D_i takes $O(|D_i| \log N)$ time and occupies $O(|D_i|)$ space.

Fix-Pointer. For each corner p of a $\Delta \in D_i$ and each integer $\ell \in (i, f]$, we perform a global search in D_ℓ to determine its containing triangle Δ, and store the obtained connected component of Δ in $D_\ell \setminus \mathsf{SMPL}_\ell(\ell - i)$ with corner p. In case Δ is a degenerate triangle, two of its boundaries are infinite rays in the plane. There we determine the component membership for the furthest points inside the ray. (Recall that any degenerate triangle is described by two points at infinity and one or two ordinary corner points, and that all standard planar point location structures naturally support queries for a point at infinity specified as ray.) As with a corner point, this also yields either a connected component or a triangle from separator $\mathsf{SMPL}_\ell(\ell - i)$. That is, every $\Delta \in D_i$ stores at most four pointers to connected components, and locating given point in D_ℓ takes $O(\log N)$ time, and determining if found triangle is in the separator $\mathsf{SMPL}_\ell(\ell - i) \subseteq D_\ell$, or the connected component, also takes $O(\log N)$ time. This takes $O(|D_i|f \log N)$ total time and occupies $O(|D_i|f)$ space.

Fix-Search. For each integer $\ell \in [1, i]$ and each connected component of $D_i \setminus \mathsf{SMPL}_i(\ell)$, we compute a point-location structure for the triangles in this connected component. This takes $O(|D_i|f \log N)$ total time and occupies $O(|D_i|f)$ space.

Skip-Decider. For every $\Delta \in D_i$ and skip-length $2^j \in [1, f - i]$, we compute the planar subdivision that is the overlay of the regions $\{\mathsf{hull}(\Delta \cap S_\ell) \ : \ \ell \in (i, i + 2^j]\}$. Note that $j = O(\log f)$. The size is bounded by the number of vertices, so bounded by the total number of vertices in the given convex regions plus the total number of intersection points of edges from the given convex regions. We first bound the total size for fixed j. For the first kind, the number of vertices, across all $\Delta \in D_i$, is already bounded by $O(|D_i|)$ due to the sampling bound above. For the second kind, we consider one $\Delta \in D_i$. There, any hull edge can intersect only $O(1)$ many edges from another convex hull, and there are $O(2^j)$ many hulls to consider. Let M_Δ be the complexity of the largest of these hulls within Δ. Any of these hulls is composed of at most four sections of boundaries of connected-components, referenced by the fix-pointers of Δ, note that M_Δ has a worst-case bound of $O(d^{8+2^j})$. (See Figure 4.) Thus, the overlay within Δ is a planar subdivision of size $O(2^j M_\Delta)$. Consequently, the total number of vertices of the second kind is bounded within an $O(2^j)$-factor of $\sum_{\Delta \in D_i} M_\Delta$, where the latter sum is bounded by the number of vertices of the first kind. That is, the total size of all overlays for fixed j is $O(2^j|D_i|)$; summing over $j = O(\log f)$ shows that the total space is $O(f|D_i|)$. With the space bound, it follows that the total construction time, for all Δ and j, is $O(f|D_i| \cdot \log N)$.

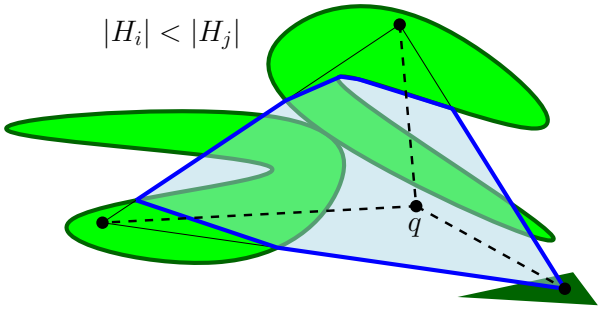

Figure 4 For $\Delta \in D_i$ and $j > i$, any connected component of $D_j \setminus \mathsf{SMPL}_j(j-i)$ containing a corner has $O(d^{16+2(j-i)})$ triangles, and is bounded by $O(d^{8+(j-i)})$ triangles, from $\mathsf{SMPL}_j(j-i)$.

Next, we label each face, in the overlay for given Δ and j. We label (the interior of) each face with the largest integer $i'' \leq i + 2^j$ so that its points are contained in *each* region $(\mathsf{hull}(\Delta \cap S_{i+1}), \ldots, \mathsf{hull}(\Delta \cap S_{i''}))$. Taking a point p within the face, we determine for $i'' = i+1, i+2, \ldots$ if p is contained in $\mathsf{hull}(\Delta \cap S_{i''})$. This takes $O(\log N)$ time for given point p, as discussed for corner point component membership in $D_{i''}$ above, and determining i'' for p thus takes $O(2^j \log N)$ time. Summing over $j = O(\log f)$, this is $O(f \log N)$ time. Thus, the total time to label each of the $O(f|D_i|)$ faces for all $j = O(\log f)$ is $O(f^2|D_i| \log N) = O(|D_i| \log^3 N)$[3].

Finally, we compute a planar point-location structure for each labeled overlay, which retains the space bound and takes $O(|D_i| f \log N)$ total time. For given j and Δ intersected by q, the query time to report the largest integer $i'' \leq i + 2^j$ that q is contained in each hull $(\mathsf{hull}(\Delta \cap S_{i+1}), \ldots, \mathsf{hull}(\Delta \cap S_{i''}))$ is $O(\log(2^j d^{8+2^j})) = O(j + (8+2^j)\log d)$, as required. ◀

It remains to show that the computed structures of D_i allow us to compute SKIP and FIX sufficiently fast for the outlined telescoping argument above.

▶ **Lemma 6** (SKIP-Runtime). *Let $j \geq 0$ be the index where $\mathsf{SKIP}(\Delta, q, 2^j)$ returns false. The structures stored with Δ allow to answer the $j+1$ calls of SKIP in $O(2^j \log d)$ total time.*

Proof. Recall the construction of D_i computed for each $\Delta \in D_i$ and integer $2^\ell = O(f)$ query structures, that allow to answer if $q^* \in \mathsf{hull}(\Delta \cap S_{i+1} \cap \ldots \cap S_{i+2^\ell})$, and return the smallest integer i' of a $S_{i'}$ where it is not contained. Answering a planar point-location query in the overlay for skip-length 2^ℓ takes $O(2^\ell \log d)$ time. Consequently, answering $j+1$ such queries takes $(2^0 + 2^1 + \ldots + 2^j)O(\log d)$ time using the skip-decider structures. ◀

▶ **Lemma 7** (FIX-Runtime). *Finding the smallest integer $i' \in [i + 2^j, i + 2^{j+1})$ where SKIP is false on prefix $[0, i']$ can be implemented in $(i'-i)O(\log d)$ time. Moreover, obtaining the triangle $\Delta' \in D_{i'}$ that intersects q takes $(i'-i)O(\log d)$ time.*

Proof. Recall that Lemma 5 precomputed a point-location structure for the planar subdivision that is the overlay of the subdivisions in $\{\mathsf{hull}(\Delta \cap S_{i'}) : i' \in [i+2^j, i+2^{j+1})\}$. As each region of that subdivision was labeled with the largest integer i'' where its points are contained in all $\mathsf{hull}(\Delta \cap S_{i''})$, we obtain the smallest i' in time $O(2^j \log d) = (i'-i)O(\log d)$, since $2^j = O(i'-i)$. Recall that corner-component membership information was precomputed in construction of D_i for all $i' > i$. For any given $i' > i$ and connected component of $D_{i'} \setminus \mathsf{SMPL}_{i'}(i'-i)$, a point-location query takes $O(\log d^{16+2(i'-i)}) = (i'-i)O(\log d)$ time, and there are $O(1)$ connected components to query. ◀

[3] This time bound here is the dominating factor in the construction time.

Note that both time bounds, of Lemmas 6 and 7, are $(i' - i)O(\log d)$, as desired.

▶ **Theorem 8.** *Maintaining a query structure for a semi-dynamic set H of planes under insertions such that for given vertical line q we can report a sequence of at most $f = O(\frac{\log |H|}{\log \log |H|})$ triangles such that $\bar{k}$-lowest$(q, H) = \bar{k}$-lowest$(q, \bar{H})$ is possible with $O(f \log^7 |H|) = O(\log^8 |H|)$ amortized insertion time. The structure has size $O(|H| \frac{\log^4 |H|}{\log \log |H|})$.*

The worst-case query time to obtain the triangle sequence is $O(\log |H|)$.

Proof. Lemmas 2 and 4 showed that the triangle sequence (Δ_{i_j}) has a set of planes $\bar{H} = H(\Delta_{i_1}) \cup \ldots \cup H(\Delta_{i_m})$ with $\bar{k}$-lowest$(q, \bar{H}) = \bar{k}$-lowest(q, H). The query time of $O(f \log d) = O(\log |H|)$ follows from Lemmas 6 and 7 in the telescoping argument Eq. (1). The amortized update time of $O(fg) = O(f \frac{|H_i| \log^6 N}{d^i}) = O(\log^8 |H|)$, follows from Lemma 5, showing a construction time of $O(|D_i| \log^3 |H|) = O(|H_i| \log^6 |H|)$, where $|H_i| = O(d^{i+1})$. From Lemma 5, space is dominated by the bound for D_f, i.e. $O(f|D_f|) = O(f|H| \log^3 |H|)$. ◀

In Subsection 3.1, we introduce a Deduplication Lemma that enables us to apply known techniques for our efficient post-processing PRUNE$(k, \ldots)$ of the triangle sequence.

3.1 PRUNE: Searching the $\bar{k}$-lists of $\leq f$ triangles in $O(\log N + k)$ time

This section capitalizes on the fact that the cardinality $\bar{k}$ sets $H(\Delta)$ in our proposed structure are fixed at construction time. That is, we will augment each triangle Δ, of any triangulation, with a static query structure that allows us to quickly retrieve the 2^x-lowest for desired granularity $2^x = O(k)$ during the post-processing phase of the $\leq f$ triangles.

Our main technical difficulty in obtaining the optimal query bound is avoiding duplicate entries in the sets $(H(\Delta_{i_1}), H(\Delta_{i_2}), \ldots, H(\Delta_{i_m}))$ of triangles in the sequence $(\Delta_{i_j} \in D_{i_j})$, as the sets $\hat{H}_i$ are, unlike the sets H_i, not necessarily disjoint, preventing us direct access to the Heap-Selection method from [13]. To this end, we define the pruned set $H^*(\Delta)$ of a triangle $\Delta \in D_i$ to be

$$H^*(\Delta) := H(\Delta) \setminus \bigcup_{\ell < i,\ \Delta' \in \mathsf{SMPL}_i(\ell)} H(\Delta') \ ,$$

i.e. those planes in $H(\Delta)$ that do not appear by sampling in some $\hat{H}_\ell$ with index $\ell < i$.

Note that this definition does not involve query locations, i.e. we can build one static 2^x-lowest query structure for $H(\Delta)$, and one for $H^*(\Delta)$, during the construction of $\Delta \in D_i$.

▶ **Lemma 9** (Deduplication). *Let $\bar{H} = \bigcup_{j=1}^{m} H(\Delta_{i_j})$ be the union over the triangle sequence obtained by the query algorithm, $H^* = \bigcup_{j=2}^{m} H^*(\Delta_{i_j})$, and $H_\perp = H(\Delta_{i_1})$ from the leftmost triangle in the sequence.*

Then, the sets $\{H_\perp, H^*(\Delta_{i_2}), \ldots, H^*(\Delta_{i_m})\}$ are disjoint and the deduplicated sets have $\bar{k}$-lowest$(q, \bar{H}) = \bar{k}$-lowest$(q, H_\perp \cup H^*)$.

Proof. Disjointness follows immediately from the definition. For the first triangle in the sequence $|H_\perp| = \bar{k}$. For $m > 1$ triangles, we argue as follows.

For a contradiction, assume there is a plane $h \in \bar{k}$-lowest$(q, \bar{H}) \setminus \bar{k}$-lowest$(q, H_\perp \cup H^*)$, i.e. h is contained in the $\bar{k}$-lowest on the original set $\bar{H}$ but not in the deduplicated set.

Let index $i \geq i_2$ be minimal with $h \in \hat{H}_i$ (leftmost BS set), and $\Delta \in D_i$ intersecting q.

There are two possible cases, Δ may be present/absent in the triangle sequence.

If Δ is present, we argue as follows. Since h is a $\bar{k}$-lowest plane in the result, the triangle $\Delta \in D_i$ intersecting q must contain $h \in H(\Delta)$. As $\hat{H}_i$ is the left-most set containing h, i.e. i is minimal, we have that the plane must also be contained post-deduplication $h \in H^*(\Delta)$. So $h \in H_\perp \cup H^*$, and consequently $h \in \bar{k}\text{-lowest}(q, H_\perp \cup H^*)$, a contradiction.

Otherwise Δ is absent in the triangle sequence generated by our query algorithm. For $\Delta \in D_i$ being absent, index i must have been skipped by the query algorithm. Consider largest $i' < i$ with $q^* \in \Delta' \in D_{i'}$ that appears in the sequence. Indeed, i' exists, since Δ cannot be the first triangle in the sequence. However, Δ absent in the sequence requires that q is in the region $q^* \in \mathsf{hull}(\Delta' \cap S_i)$. By Lemma 2, $\bar{k}\text{-lowest}(q, \hat{H}_{i'} \cup \ldots \cup \hat{H}_i) = \bar{k}\text{-lowest}(q, \hat{H}_{i'})$. Thus $\bar{k}\text{-lowest}(q, \hat{H}_{i'}) = H(\Delta')$ must contain h, contradicting i is minimal. ◀

Thus, our implementation of PRUNE uses static query structures obtained from a shallow-cutting hierarchy for the planes for $H(\Delta)$, and a separate data structure for $H^*(\Delta)$, for all D_i and $\Delta \in D_i$. Recall[4] that computing such a query structure takes near-linear $O(\bar{k} \log \bar{k}) = O(\log^2 N \log\log N)$ time and space. This increases the time, and space, bound for our construction algorithm by no more than one $O(\log\log N)$ factor (see proof of Lemma 5), thus giving a $O(|H| \log^4 |H|)$ space and $O(\log^8 |H|)$ amortized insertion bound from Theorem 8. With those, querying any of the $\leq f$ sets $(H_\perp, H^*(\Delta_{i_2}), \ldots, H^*(\Delta_{i_m}))$ for the k-lowest takes $O(\log \bar{k} + k)$ time.

With our Deduplication Lemma, the result follows immediately from the adaptation [13, Thm. 1] of the Heap-Selection algorithm [15], since we have $f = O(\frac{\log N}{\log\log N})$ structures over disjoint sets with $O(\log\log(N) + k)$ query time each.

▶ **Corollary 10.** *Increasing the space and amortized insertion bounds in Theorem 8 by one $O(\log\log|H|)$ factor allows reporting the k-lowest in $O(f \log \bar{k} + k) = O(\log |H| + k)$ time.*

We conclude this section by presenting an alternative path to obtain the result, combining some basic ideas from [13, Thm. 1] with the, remarkable powerful, Soft-Heaps [10, 20] that are known to offer an alternative to Heap-Selection [18, Sec. 3].

Note that, for very small $k = O(\log\log N)$, we can just query the k-lowest from each triangle and run the standard k-Selection algorithm, taking $O(kf) = O(\log N)$ time.

Simple $O(\log N + k \log\log N)$ queries with standard min-heaps

The following simplistic algorithm gives optimal query time for all $k = O(\frac{\log N}{\log\log N})$.

We query $k_0 = \lceil \log_2 \log_2 N \rceil$ lowest planes from each set, and insert all in a min-heap using the z-coordinate of $q \cap h$ as key, i.e. we start on a min-heap with $O(fk_0) = O(\log N)$ items. Then, we keep extracting the element with smallest key, until we extracted the k-lowest globally. Whenever the elements from one set (of a Δ in the sequence) were *all extracted*, we double that set size with a fresh query from Δ and insert the new items of its larger batch into the heap, which takes $O(\log \bar{k} + 2^x k_0) = O(2^x k_0)$ time. This completes the description of the basic algorithm.

From the disjointness, we have that the k planes from the final set are partitioned among the $m \leq f$ disjoint subsets of $\leq \bar{k}$ planes, that is there are integers $k_j^* \geq 0$ with $k = \sum_{j=1}^m k_j^*$. Consequently, the total time for the iterated queries from Δ_{i_j}, to obtain the replenishment items, remains $(1 + \frac{1}{2} + \frac{1}{4} + \ldots)O(k_j^*)$. That is $O(k)$ time overall to insert replenishment items.

[4] See start of Section 3 for discussion of related work [9].

Since there are $O(\log N + k)$ items in the heap throughout the processing, we have that each of the k extract-min operations takes $O(\log\log N)$ time. That is, the basic algorithm takes $O(\log N + k + k\log\log N)$ total time for initializing, inserting, and extracting.

The bottleneck for obtaining optimal bounds for larger $k = \omega(\frac{\log N}{\log\log N})$ is due to extractMin requiring $O(\log\log N)$ time with standard heaps. (The planes are reported in sorted order.)

Soft-Heaps: Allowing Key-Corruption accelerates operations to $O(\log 1/\varepsilon)$. As observed in [18, Sec. 3.1], one can use Soft-Heaps to simplify the Heap-Selection in [15]. Using a Soft-Heap with constant error parameter, $\varepsilon = 1/4$, one can obtain the k-smallest keys, in not necessarily sorted order, with $O(1)$ amortized cost per operation, as ε is constant.

4 Reducing Space to Linear

▶ **Lemma 11.** *Space of the insertion-only k-lowest structure can be reduced to linear $O(|H|)$, without increasing the query and update bounds.*

Proof. Since our approach uses $d = \lceil \log_2 N \rceil$, we can reduce our $O(|H|\log^4|H|)$ space bound to $O(|H|)$ as follows.

We apply our proposed data structure (Theorem 8 and Corollary 10) only to the set $H^- = \bigcup_{i<f-8} H_i$ of planes from the first $f-8$ sets of the partition $H = \bigcup_{i\le f} H_i$. Since $\sum_{i<f-8}|H_i| = O(d^{f-8})$, and $|H| = \Theta(d^f)$, this set has $|H^-| = O(|H|/\log^4 N)$ planes; thus occupies $O(|H|)$ space. For the last $O(1)$ sets H_i of the partition with $i \in [f-8, f]$, we use an optimal, static $O(|H_i|)$ space structure from [9, 1] that allows reporting the k-lowest planes in $O(\log|H_i| + k)$ time. The expected construction time was made deterministic in [9].

To answer a query, we simply compute the k-lowest(q, H^-) in $O(\log|H| + k)$ time, and the k-lowest(q, H_i) for each in the $O(1)$ sets with $i \in [f-8, f]$. The resulting set contains $O(k)$ planes, and we run the standard k-Selection algorithm (see also Subsection 3.1) to obtain the k-lowest planes in vertical query line q, which takes $O(k)$ time. ◀

▶ **Corollary 12.** *There exists a deterministic $O(n)$ space data structure that answers k-lowest planes queries in $O(\log n + k)$ time and supports insertions in $O(\log^8 n)$ amortized time.*

5 Applications

The applications stated in Section 1 follow immediately by the standard paraboloid lifting of 2D points (k-Nearest neighbor and circular range), or by the standard point-plane duality (vertical ray shooting and halfspace range reporting) and using iterated queries, with $k_i = 2^i\lceil\log_2 N\rceil$, for the k_i-lowest planes from our proposed structure. (Applications 1-4.)

The only exception is the application 5, weighted halfspace range reporting. There we reduce the problem to two, one-sided queries. Our solution for the one-sided queries then observes that our proposed structure from Corollary 10 can be turned into a partially-persistent data structure with standard techniques (see Subsection 5.1).

Weighted halfspace range reporting. The two-sided problem reduces to answering two one-sided queries, i.e. queries of the form $[-\infty, W]$. By inserting the elements with ascending weights into our proposed structure, the one-sided queries reduce to halfspace reporting among the t-oldest elements of the insertion sequence, where t is the number of elements with weight $\le W$. The symmetric approach applies for one-sided queries of the form $[W, +\infty]$.

To reduce a two-sided query $[w_1, w_2]$ to two one-sided queries, we build a balanced, binary tree over the n leaves, sorted by their weight p_w. There we determine the successor of w_1, called p_1, and predecessor of w_2, called p_2. Let v be the lowest common ancestor of p_1 and p_2 in the tree, and u_ℓ and u_r be the left and right child of v, respectively. Having precomputed a static structure for all elements within a subtree of a node, we can answer one one-sided reporting query in the structure of u_ℓ in time $O(\log n + k_\ell)$. Answering another one-sided reporting query in the structure of u_r takes time $O(\log n + k_r)$. Since $k = k_\ell + k_r$, the query time is $O(\log n + k)$.

It remains to obtain an $O(\log n + k)$ query time for the k-lowest, among the t-oldest inserted planes, from our proposed structure.

5.1 Partial Persistence: Finding the k-lowest among the t-oldest planes

During the entire sequence of insertions $h_1, h_2, \dots$, we keep track of the time, that is the number of insertions to the semi-dynamic set $|H| = t$. Whenever the t-th insertion, h_t, triggers a rebuild of a prefix of Bentley-Saxe sets, we tag all structures D_i that are dropped with timestamp $t^\dagger := t$ and the newly created structure with timestamp $t^* := t$.

Our extension to partial persistence is now to simply keep the obsolete data structures in memory, without deleting them. There are $O(n)$ structures in total, and we store their $[t^*, t^\dagger)$ intervals in a tree. To answer a query among the t-oldest planes $\{h_1, \dots, h_t\}$, we first determine those $\leq f = O(\frac{\log n}{\log \log n})$ data-structures $\{D_i\}$ that are alive at time t, i.e. have $t \in [t^*, t^\dagger)$. Since the fix-pointers of our proposed data structure only refer to structures with higher indices (see proof of Lemma 5), we can simply run the greedy query algorithm on this set of data structures.

Since our structure has $O(\log^8 n)$ amortized insertion time, total space remains $O(n \log^8 n)$. Consequently, we can answer k-lowest queries among any t-oldest planes in $O(\log n + k)$ time.

References

1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 180–186, 2009. doi:10.1137/1.9781611973068.21.

2 Peyman Afshani, Yakov Nekrich, and Frank Staals. Convexity helps iterated search in 3d. In *41st International Symposium on Computational Geometry (SoCG)*, pages 3:1–3:14, 2025. doi:10.4230/LIPIcs.SOCG.2025.3.

3 Pankaj K. Agarwal, Boris Aronov, Timothy M. Chan, and Micha Sharir. On Levels in Arrangements of Lines, Segments, Planes, and Triangles. *Discret. Comput. Geom.*, 19(3):315–331, 1998. doi:10.1007/PL00009348.

4 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

5 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

6 Timothy M. Chan. Three problems about dynamic convex hulls. *Int. J. Comput. Geom. Appl.*, 22(4):341–364, 2012. doi:10.1142/S0218195912600096.

7 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. In *35th International Symposium on Computational Geometry (SoCG)*, pages 24:1–24:13, 2019. doi:10.4230/LIPIcs.SOCG.2019.24.

8 Timothy M. Chan, Pingan Cheng, and Da Wei Zheng. An Optimal Algorithm for Higher-Order Voronoi Diagrams in the Plane: The Usefulness of Nondeterminism. In *35th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4451–4463, 2024. doi:10.1137/1.9781611977912.156.

9 Timothy M. Chan and Konstantinos Tsakalidis. Optimal Deterministic Algorithms for 2-d and 3-d Shallow Cuttings. *Discret. Comput. Geom.*, 56(4):866–881, 2016. doi:10.1007/S00454-016-9784-4.

10 Bernard Chazelle. The soft heap: an approximate priority queue with optimal error rate. *J. ACM*, 47(6):1012–1027, 2000. doi:10.1145/355541.355554.

11 Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *J. Comput. Syst. Sci.*, 68(2):269–284, 2004. doi:10.1016/J.JCSS.2003.07.003.

12 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. doi:10.1007/978-3-540-77974-2.

13 Sarita de Berg and Frank Staals. Dynamic data structures for k-nearest neighbor queries. *Comput. Geom.*, 111:101976, 2023. doi:10.1016/J.COMGEO.2022.101976.

14 Jeff Erickson. Algorithms, 2019. URL: http://archive.org/details/Algorithms-Jeff-Erickson.

15 Greg N. Frederickson. An Optimal Algorithm for Selection in a Min-Heap. *Inf. Comput.*, 104(2):197–214, 1993. doi:10.1006/INCO.1993.1030.

16 Michael T. Goodrich. Planar Separators and Parallel Polygon Triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995. doi:10.1006/JCSS.1995.1076.

17 John Iacono and Yakov Nekrich. Incremental Planar Nearest Neighbor Queries with Optimal Query Time. In *41st Symposium on Computational Geometry (SoCG)*, pages 59:1–59:15, 2025. doi:10.4230/LIPIcs.SOCG.2025.59.

18 Haim Kaplan, László Kozma, Or Zamir, and Uri Zwick. Selection from Heaps, Row-Sorted Matrices, and X+Y Using Soft Heaps. In *2nd Symposium on Simplicity in Algorithms (SOSA)*, pages 5:1–5:21, 2019. doi:10.4230/OASIcs.SOSA.2019.5.

19 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. In *28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2495–2504, 2017. doi:10.1137/1.9781611974782.165.

20 Haim Kaplan and Uri Zwick. A simpler implementation and analysis of Chazelle's soft heaps. In *20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 477–485, 2009. doi:10.1137/1.9781611973068.53.

21 Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Symposium on Theory of Computing Conference (STOC)*, pages 505–514, 2013. doi:10.1145/2488608.2488672.

22 Yakov Nekrich and Saladi Rahul. 4d range reporting in the pointer machine model in almost-optimal time. In *34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1862–1876, 2023. doi:10.1137/1.9781611977554.CH71.

23 Micha Sharir, Shakhar Smorodinsky, and Gábor Tardos. An improved bound for k-sets in three dimensions. In *16th Symposium on Computational Geometry (SoCG)*, pages 43–49, 2000. doi:10.1145/336154.336173.

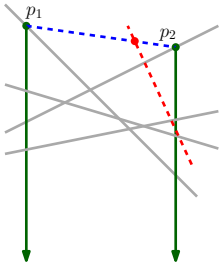

Figure 5 Illustration of the proof of Lemma 1.

A Proof of Lemma 1

Proof. We consider the problem in the plane G spanned by the segment $\overline{q_1^* q_2^*}$ and vertical line q_1, thus containing q_2 as well. Any intersection of $h \in H$ with G is a line, as any h does not contain vertical lines. Let p_1 be the intersection point of q_1 with its k-lowest plane, and p_2 the intersection point of q_2 with its k-lowest plane. That is, the orthogonal projection of $\overline{p_1 p_2}$ is $\overline{q_1^* q_2^*}$. (See Figure 5.)

The proof is by contradiction. Assume there is a vertical line q with $q^* \in \overline{q_1^* q_2^*}$ that has a different set k-lowest$(q, H) \neq H'$ planes.

Let h be one such plane not in H'. The projection of its intersection with $\overline{p_1 p_2}$ is q^*. Now, if h intersects the downward vertical ray in p_1 we have a contradiction to $h \notin H'$, and the same argument applies to the ray in p_2. So h would need to contain a vertical line, which is not possible for any $h \in H$, or intersect $\overline{p_1 p_2}$ more than once, which is also not possible for any line and segment pair. ◀