Optimal Deterministic Algorithms for 2-d and 3-d Shallow Cuttings

Timothy M. Chan*[1] and Konstantinos Tsakalidis[2]

1 **Cheriton School of Computer Science, University of Waterloo, Canada**
tmchan@uwaterloo.ca

2 **Department of Computer Science and Engineering, Hong Kong University of Science and Technology, China**
tsakalid@cse.ust.hk

Abstract

We present optimal deterministic algorithms for constructing shallow cuttings in an arrangement of lines in two dimensions or planes in three dimensions. Our results improve the deterministic polynomial-time algorithm of Matoušek (1992) and the optimal but randomized algorithm of Ramos (1999). This leads to efficient derandomization of previous algorithms for numerous well-studied problems in computational geometry, including halfspace range reporting in 2-d and 3-d, k nearest neighbors search in 2-d, ($\leq k$)-levels in 3-d, order-k Voronoi diagrams in 2-d, linear programming with k violations in 2-d, dynamic convex hulls in 3-d, dynamic nearest neighbor search in 2-d, convex layers (onion peeling) in 3-d, ε-nets for halfspace ranges in 3-d, and more. As a side product we also describe an optimal deterministic algorithm for constructing standard (non-shallow) cuttings in two dimensions, which is arguably simpler than the known optimal algorithms by Matoušek (1991) and Chazelle (1993).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases shallow cuttings, derandomization, halfspace range reporting, geometric data structures

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.719

1 Introduction

Shallow cuttings were introduced by Matoušek [25] as a tool for range searching, specifically, *halfspace range reporting*. They have since found applications to numerous other central problems in computational geometry, including ($\leq k$)-levels in arrangements of hyperplanes, order-k Voronoi diagrams, linear programming with k violations, dynamic convex hulls, and dynamic nearest neighbor search (see Section 1.4 for more information). At SoCG'99, Ramos [29] presented an optimal randomized algorithm for constructing shallow cuttings in two and three dimensions. A nagging question that has remained open is whether there is an equally efficient deterministic algorithm. The main result of this paper is a positive resolution to this question. Although the question is mainly about theoretical understanding, and derandomization isn't the most "fashionable" topic in computational geometry, we believe that in this case the fundamental nature of the problem and its wide-ranging consequences make the problem important to study.

* Part of this work was done during the author's visit to the Hong Kong University of Science and Technology.

31st International Symposium on Computational Geometry (SoCG'15).
Editors: Lars Arge and János Pach; pp. 719–732

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Standard Cuttings

▶ **Definition 1.** Let H be a set of n hyperplanes in $\mathbb{R}^d$. Given a parameter $r \in [1, n]$ and a region $L \subseteq \mathbb{R}^d$, a $\frac{1}{r}$-*cutting for* H *covering* L is a set of interior-disjoint simplices (cells) such that

(i) the interior of every cell intersects at most $\frac{n}{r}$ hyperplanes of H, and

(ii) the union of the cells covers L.

The *conflict list* H_Δ of a cell Δ is the set of (at most $\frac{n}{r}$) hyperplanes of H that intersect Δ. The *size* of the cutting is the number of its cells.

Cuttings are a fundamental tool in geometric divide-and-conquer. In the default "standard" setting, a cutting covers all of $\mathbb{R}^d$, i.e., $L = \mathbb{R}^d$.

Random sampling techniques by Clarkson [16] and Haussler and Welzl [21] imply the existence of (standard) $\frac{1}{r}$-cuttings of size $O((r \log r)^d)$. Chazelle and Friedman [15] refined the bound to $O(r^d)$, which is optimal. (In the 2-d case, there is a simple alternative proof based on *levels* by Matoušek [23].)

Considerable effort was spent in finding efficient deterministic algorithms to construct such an optimal-size cutting. Even the 2-d case turned out to be a challenge. At SoCG'89, Matoušek [23] presented an $O(nr^2 \log r)$-time algorithm for $d = 2$. At the same conference, Agarwal [4] (see also his PhD thesis [5]) presented an $O(nr \log n \log^{3.33} r)$-time algorithm for $d = 2$. In a subsequent paper, Matoušek [24] improved the deterministic time bound to $O(nr)$ for $d = 2$, which is optimal if the algorithm is required to output the conflict lists of all the cells (since the worst-case total size of the conflict lists is $\Theta(r^2 \cdot \frac{n}{r}) = \Theta(nr)$). Matoušek's later paper also described a deterministic $O(nr^{d-1})$-time algorithm for any constant dimension d, which is again optimal if we need to output all conflict lists, but this result holds under the restriction that r is not too big, i.e., $r < n^{1-\delta}$ for some constant $\delta > 0$. Finally, Chazelle [14] obtained a deterministic $O(nr^{d-1})$-time algorithm without any restriction on r for all constant dimensions d. All of these deterministic algorithms are complicated and/or make use of advanced derandomization techniques such as ε-*approximations* [21].

1.2 Shallow Cuttings

Given a point p, the *level* of p in H is the number of hyperplanes of H that are below p. We define $L_{\le k}(H)$ to be the $(\le k)$-*level*, i.e., the region of all points with level in H at most k. A *shallow cutting* is a variant of the standard cutting that is required to cover only points that are "shallow", i.e., have small levels.

▶ **Definition 2.** Given parameters $k, r \in [1, n]$, a k-*shallow* $\frac{1}{r}$-*cutting* is a $\frac{1}{r}$-cutting for $L_{\le k}(H)$.

We concentrate on the most important case of $k = \Theta(\frac{n}{r})$, which is sufficient for all of the applications encountered; in fact, shallow cuttings for any value of k can be reduced to this case—see the remarks in Section 5. Matoušek [25] proved the existence of a $\Theta(\frac{n}{r})$-shallow $\frac{1}{r}$-cutting of size $O(r^{\lfloor d/2 \rfloor})$, which is smaller than the $O(r^d)$ bound for standard $\frac{1}{r}$-cuttings and is optimal in the worst case. In particular, for $d \in \{2, 3\}$, the size is $O(r)$.

In the same paper, Matoušek presented a deterministic algorithm that can construct such a shallow cutting in polynomial time; the running time improves to $O(n \log r)$ but only when r is small, i.e., $r < n^\delta$ for a sufficiently small constant δ. Later, Ramos [29] presented a complicated randomized algorithm for $d = 3$ (and hence $d = 2$ as well) with $O(n \log n)$ expected running time to construct not just a single shallow cutting, but a hierarchy of $O(\log n)$ such shallow cuttings for all r's forming a geometric sequence from 1 to n. (Such a

hierarchy is useful in certain applications.) Recently, at SODA'14, Afshani and Tsakalidis [3] managed to achieve the same bound deterministically, albeit only for an orthogonal variant of the problem where the input objects are orthants in $\mathbb{R}^3$ (which nonetheless has applications to *dominance range reporting*); subsequently, Afshani et al. [2] improved the time bound for a single shallow cutting to $O(n \log\log n)$ in the word RAM model. The case of orthants is indeed a special case, as orthants can be mapped to halfspaces via a certain transformation [12].

1.3 Our Contributions

We present deterministic algorithms to construct a $\Theta(\frac{n}{r})$-shallow $\frac{1}{r}$-cutting of size $O(r)$ for $d \in \{2, 3\}$ in $O(n \log r)$ time, which is optimal in a comparison-based model (the default model in this paper). Like Ramos' randomized algorithm [29], our algorithms can in fact construct a hierarchy of such shallow cuttings for all r's in a geometric sequence, along with the conflict lists of all cells, in $O(n \log n)$ total time. (Note that for our 3-d algorithm, we do not insist the cutting in one layer of the hierarchy be nested inside the cutting in the next layer.)

Considering how involved known deterministic algorithms for standard cuttings are, we are happy to report that the new results are not complicated to derive. All the needed background is provided in Section 2; no advanced derandomization techniques are used. The main algorithms are describable in a few lines, as seen in Sections 3 and 4, although their analyses are not trivial.

Like in Chazelle's cutting algorithm [14], we will construct the hierarchy layer by layer, refining the shallow cutting in the previous layer to obtain the shallow cutting in the next layer. A naive implementation would cause an amplification of the constant factor in the cutting size bound, which may "blow up" after logarithmically many iterations. Chazelle used ε-approximations and *sparse ε-nets* to refine the cutting in each cell, and controlled the blow-up by charging cost to some easily summable quantity (namely, the number of vertices inside the cell). We replace ε-approximations and sparse ε-nets with the more elementary techniques by Megiddo and Dyer [28, 17]. We use a brute-force search to find the best way to refine the cutting in each cell, and control the blow-up by bounding cost in terms of the cost of an optimal cutting—this strategy is reminiscent of the analysis of approximation algorithms or PTASes (although we do not explicitly design an approximation algorithm to find the minimum-size cutting).

The strategy works beautifully in 2-d, but the constant-factor blow-up becomes tougher to deal with in 3-d, because cost of substructures along the cell boundaries becomes non-negligible. To tackle this issue, we borrow an idea from a different paper by Ramos [30], of using planar graph separators to group cells into regions, which we call "supercells", so that the total size of the boundaries of the supercells is reduced. (Ramos originally applied this idea to obtain an optimal deterministic algorithm for the 3-d diameter problem and for computing lower envelopes of certain bivariate surfaces in 3-d, but did not consider shallow cuttings in that paper. Also, the details of his algorithms appear more complicated, using ε-nets and supercells of size n^δ, whereas we use only supercells of constant size.)

In the appendix, we show that our ideas can also lead to a new presentation of a deterministic $O(nr)$-time algorithm for constructing standard $\frac{1}{r}$-cuttings in 2-d. This may be of independent pedagogical interest, considering the long line of previous complicated algorithms.

1.4 Applications

As mentioned, shallow cuttings are important because of their numerous applications. Below we list some of the specific implications of our new 2-d and 3-d deterministic algorithms.

1. The first optimal deterministic $O(n \log n)$-time algorithm to preprocess a set of n points in $\mathbb{R}^3$ into an $O(n)$-space data structure, so that we can answer a *halfspace range reporting* query (i.e., report all k points that lie within any given halfspace) in $O(\log n + k)$ time. This result follows from the work of Afshani and Chan [1], which was almost deterministic except for the invocation of Ramos' algorithm to construct a shallow cutting during preprocessing.
   By a standard lifting transformation, the same result holds for *circular range reporting* in $\mathbb{R}^2$ (reporting all k points that lie inside any given circle) and k *nearest neighbors search* in $\mathbb{R}^2$ (reporting all k nearest neighbors to a given point, in arbitrary order, under the Euclidean metric).
2. The first optimal deterministic $O(n \log n + nk^2)$-time algorithm to construct the $(\leq k)$-*level* of an arrangement of n planes in $\mathbb{R}^3$. This result follows from the work of Chan [9], which previously required randomization.
3. The currently fastest deterministic $O(n \log n + nk \cdot t(k))$-time algorithm for constructing the k-*th order Voronoi diagram* of n points in $\mathbb{R}^2$. Here, $t(\cdot)$ denotes the (amortized) update and query time complexity for the 2-d dynamic convex hull problem (under gift-wrapping queries). We have $t(k) = O(\log k \log \log k)$ [7], or better still, $t(k) = O(\log k)$ [8] if one has confidence in the over-100-page proof in the latter paper. This result again follows from the work of Chan [9]. Compare the result with Ramos' randomized $O(n \log n + nk2^{O(\log^* k)})$-time algorithm [29].
4. A deterministic $O((n + k^2) \log k)$-time algorithm for *2-d linear programming with at most k violations* (i.e., given a set of n halfspaces, find the point that lies inside all but k of the halfspaces and is extreme along a given direction). This result follows from another work of Chan [10], which was almost deterministic except for the construction of a 2-d shallow cutting in one step.
5. The first deterministic data structure for *dynamic 3-d convex hull* with polylogarithmic amortized update and query time, namely, $O(\log^3 n)$ amortized insertion time, $O(\log^6 n)$ amortized deletion time, and $O(\log^2 n)$ time for a gift-wrapping query. This result follows from another work of Chan [11], which was almost deterministic except for the construction of a hierarchy of 3-d shallow cuttings during certain update operations.
   This result itself spawns countless additional consequences, for example, to *dynamic 2-d smallest enclosing circle*, *dynamic 2-d bichromatic closest pair*, *dynamic 2-d diameter*, *dynamic 2-d Euclidean minimum spanning tree*, *3-d convex layers (onion peeling)*, output-sensitive construction of 3-d k-*levels*, and so on.
6. A deterministic data structure for *dynamic 2-d halfspace range reporting* with $O(\log^{6+\varepsilon} n)$ amortized update time and $O(\log n + k)$ query time for any fixed $\varepsilon > 0$. In 3-d, the query time increases to $O(\log^2 n / \log \log n + k)$. This result follows from yet another work of Chan [11], which was almost deterministic except for the construction of a hierarchy of shallow cuttings during certain update operations.
7. A deterministic $O(n \log r)$-time algorithm to construct a $\frac{1}{r}$-*net* of size $O(r)$ for n points in $\mathbb{R}^3$ with respect to halfspace ranges. This application actually appeared in Matoušek's original paper on shallow cuttings [25]. There, he was interested in proving existence of $O(r)$-size nets, but with our shallow cutting algorithm, the deterministic time bound follows. (Roughly speaking, in the dual, we construct a $\frac{n}{r}$-shallow $O(\frac{1}{r})$-cutting, construct

an ε-cutting within each cell for a sufficiently small constant ε, and output an arbitrary plane passing below each subcell.) Of course, ε-nets are well known and central to combinatorial and computational geometry. Previously, there were deterministic $nr^{O(1)}$-time algorithms (e.g., see a recent note [20]), and an $O(n \log r)$-time algorithm but only when r is small, i.e., $r < n^{\delta}$ for some constant δ [25].

By a standard lifting transformation, the same result holds for ε-nets for points in $\mathbb{R}^2$ with respect to circular disk ranges.

2 Preliminaries

It will be more convenient to work with the parameter $K := \frac{n}{r}$ instead of r. For brevity, a k-shallow $\frac{K}{n}$-cutting will be referred to as a (k, K)-*shallow cutting*. It satisfies the properties that (i) each cell intersects at most K hyperplanes, and (ii) the cells cover $L_{\leq k}(H)$. Our goal is to compute a $(k, \Theta(k))$-shallow cutting of size $O(\frac{n}{k})$.

For a set V of points in $\mathbb{R}^d$, we denote by $\mathrm{UH}(V)$ the region underneath the *upper hull* of V. We define the *vertical decomposition* $\mathrm{VD}(V)$ to be the set of interior-disjoint cells covering $\mathrm{UH}(V)$, such that each cell is bounded from above by a different face of $\mathrm{UH}(V)$, is bounded from the sides by vertical *walls*, and is unbounded from below. For example, in 2-d, the boundary of $\mathrm{UH}(V)$ is a concave chain; a cell in $\mathrm{VD}(V)$ is bounded by an edge of $\mathrm{UH}(V)$ and two walls (downward vertical rays). In 3-d, the boundary of $\mathrm{UH}(V)$ is a concave polygonal surface with triangular faces; a cell in $\mathrm{VD}(V)$ is bounded by a triangle and three walls (trapezoids that are unbounded from below).

In the studied dimensions $d \in \{2, 3\}$, we find it simpler to work with the following equivalent form of shallow cuttings:

▶ **Definition 3.** Given parameters $k, K \in [1, n]$, a (k, K)-*shallow cutting* for H *in vertex form* is a set V of points such that
(i) every point in V has level at most K, and
(ii) $\mathrm{UH}(V)$ covers $L_{\leq k}(H)$.
The *conflict list* of a point $v \in V$ is the set of (at most K) hyperplanes in H that are below v.

A (k, K)-shallow cutting under the original definition can be transformed into a $(k, k+K)$-shallow cutting in vertex form simply by letting V be the set of vertices of the cells (after pruning cells that do not intersect $L_{\leq k}(H)$). In the reverse direction, a (k, K)-shallow cutting V in vertex form can be transformed to a (k, dK)-shallow cutting under the original definition simply by taking $\mathrm{VD}(V)$, since the conflict list H_Δ of a cell Δ is contained in the union of the conflict lists of the d vertices of Δ, and thus $|H_\Delta| \leq dK$. In 2-d and 3-d, the size of $\mathrm{VD}(V)$ is $O(|V|)$, and computing $\mathrm{VD}(V)$ takes $O(|V| \log |V|)$ time by an optimal convex hull algorithm.

From now on, all shallow cuttings will be in vertex form by default.

Our algorithms do not require any advanced derandomization techniques at all. Only three facts are needed (the third is used only for the 3-d case):

▶ **Fact 4** (Constant-Size Cuttings). *Given a set of n lines in $\mathbb{R}^2$ or planes in $\mathbb{R}^3$ and any constant $\varepsilon > 0$, a (standard) ε-cutting of constant size can be computed in $O(n)$ worst-case time.*

▶ **Fact 5** (Existence of $O(\frac{n}{k})$-Size Shallow Cuttings). *Given a set of n lines in $\mathbb{R}^2$ or planes in $\mathbb{R}^3$ and a parameter $k \in [1, n]$, there exists a $(k, c_0 k)$-shallow cutting (in vertex form) of maximum size $c_0' \frac{n}{k}$, for some universal constants c_0, c_0'.*

▶ **Fact 6** (Planar Graph Separators). *Given a triangulated planar graph with n vertices and a parameter $t \in [1, n]$, we can group the triangles into at most $a_0 \frac{n}{t}$ connected* regions *where each region contains at most t triangles, and the total number of edges along the boundaries of the regions is at most $a'_0 \frac{n}{\sqrt{t}}$ for some universal constants a_0, a'_0. Such regions can be computed in $O(n \log n)$ time.*

Fact 4 was known in the 1980s even before the term "cutting" was coined. In deriving their linear-time algorithm for 3-d linear programming, Megiddo [27] and Dyer [17] implicitly gave a linear-time construction of a $\frac{7}{8}$-cutting of size 4 in 2-d. Megiddo [28] subsequently generalized the construction to d dimensions, yielding a $(1 - 1/2^{2^d-1})$-cutting of size $2^{2^{d-1}}$ in linear time. (The cells may not be simplices, but we can triangulate them and the size remains bounded by a constant.) Although these constructions give ε-cuttings for one specific constant $\varepsilon > 0$, iterating a constant number of times automatically yields ε-cuttings for any given constant $\varepsilon > 0$ in linear time. The size of such a cutting may be suboptimal, but for our purposes, any constant size bound will be sufficient. More powerful techniques based on ε-approximations and ε-nets [21, 15] can yield better bounds, but a virtue of Megiddo and Dyer's constructions is that they are completely elementary, relying on linear-time median finding as the only subroutine.

Fact 5 was proved by Matoušek [25] by using Chazelle and Friedman's random sampling techniques [15]. (In the 2-d case, there is a simpler alternative proof using levels, similar to [23] and implicit in one of the proofs in [6].) For our purposes, we do not actually need to know how Fact 5 is proved and do not care about the construction time—we just need the existence of $O(\frac{n}{k})$-size shallow cuttings, not for our algorithms themselves but for their analyses.

Fact 6 is a multiple-regions version [18] of the well-known planar graph separator theorem [22], as applied to the dual of the given graph. The multiple-regions version follows from the standard version by recursion. The running time $O(n \log n)$ can actually be reduced to $O(n)$ [19], although we do not need this improvement. A version by Frederickson [18] can further guarantee that each region has $O(\sqrt{r})$ boundary edges (Fact 6 guarantees the same bound but on average only); again, we do not need such an improvement.

3 A 2-d Shallow Cutting Algorithm

We begin in 2-d and prove the following theorem, from which our main result will follow as a corollary:

▶ **Theorem 7.** *For a set H of n lines in $\mathbb{R}^2$, a parameter $k \in [1, n]$, and some suitable constants B, C, C', given a (Bk, CBk)-shallow cutting V_{IN} (in vertex form) for H of size at most $C' \frac{n}{Bk}$ along with its conflict lists, we can compute a (k, Ck)-shallow cutting V_{OUT} (in vertex form) for H of size at most $C' \frac{n}{k}$ along with its conflict lists in $O(n + \frac{n}{k} \log \frac{n}{k})$ deterministic time.*

Proof.

Algorithm. Let ε be a constant to be set later. Our algorithm is conceptually simple:

1. For each cell $\Delta \in \text{VD}(V_{\text{IN}})$:
   1.1. Compute by Fact 4 an ε-cutting Γ_Δ for H_Δ of $O(1)$ size, where the cells are clipped (and re-triangulated) to lie within Δ. Let Λ_Δ be the set of vertices that define the cells of Γ_Δ.

1.2. Compute by brute force the *smallest subset* $V_\Delta \subseteq \Lambda_\Delta$ such that
  (i) every vertex in V_Δ has level in H_Δ at most Ck, and
  (ii) $\mathrm{UH}(V_\Delta)$ covers all vertices in Λ_Δ that are in $L_{\le 2k}(H_\Delta)$.

3. Return $V_{\text{OUT}} := \bigcup_{\Delta \in \mathrm{VD}(V_{\text{IN}})} V_\Delta$ and all its conflict lists.

Complexity. In Line 1, computing $\mathrm{VD}(V_{\text{IN}})$ takes $O(\frac{n}{k}\log\frac{n}{k})$ time by an optimal convex hull algorithm, since $|V_{\text{IN}}| \le C'\frac{n}{Bk} = O(\frac{n}{k})$. Line 1.1 takes time linear in $|H_\Delta|$ by Fact 4, for a total of $\sum_{\Delta\in\mathrm{VD}(V_{\text{IN}})} O(|H_\Delta|) = O(C'\frac{n}{Bk}\cdot 2CBk) = O(n)$ time. For Line 1.2, first we determine the level in H_Δ of every vertex in Λ_Δ by a linear scan over H_Δ, and then we probe all possible subsets of Λ_Δ. Since Γ_Δ and Λ_Δ have $O(1)$ size, there are "only" $O(1)$ subsets to test (although the constant is exponentially bigger) and each subset can be tested for the two stated conditions in $O(1)$ time. Thus, the whole step takes time linear in $|H_\Delta|$, which again totals to $O(n)$. In Line 3, computing V_{OUT} takes time linear in the output size. The conflict list of every output vertex in V_Δ can be computed by a linear scan over H_Δ, again in $O(n)$ total time.

Correctness. To show that V_{OUT} is a correct (k, Ck)-shallow cutting for H, we just check that $\mathrm{UH}(V_{\text{OUT}})$ covers $L_{\le k}(H)$. This follows since for any point inside a cell of Γ_Δ with level at most k, the three vertices of the cell in Γ_Δ have levels at most $k+\varepsilon|H_\Delta| \le k+\varepsilon(2CBk) = 2k$ by setting the constant $\varepsilon := \frac{1}{2CB}$, and are thus covered by $\mathrm{UH}(V_\Delta)$.

To bound the size of V_{OUT}, we compare it against a $(2k, 2c_0k)$-shallow cutting V^* of size $c_0'\frac{n}{2k}$ provided by Fact 5. Note that V^* is covered by $\mathrm{VD}(V_{\text{IN}})$ by picking a constant $B \ge 2c_0$, since every vertex in V^* has level at most $2c_0k$, and $\mathrm{VD}(V_{\text{IN}})$ covers $L_{\le Bk}(H)$.

We render V^* comparable to V_{OUT} by modifying V^* in two steps (we emphasize that these steps are not part of the algorithm but are for the correctness proof only):

- First, we *chop* $\mathrm{UH}(V^*)$ at the walls of the cells of $\mathrm{VD}(V_{\text{IN}})$. A new vertex is formed at each wall; we create two copies of each such vertex (one assigned to each of the two incident cells of $\mathrm{VD}(V_{\text{IN}})$) and add them to V^*. (See Figure 1.) For each cell $\Delta \in \mathrm{VD}(V_{\text{IN}})$, let $V^*_\Delta := V^* \cap \Delta$. Then (i) every vertex in V^*_Δ (including the extra vertices added) has level at most $4c_0k$, and (ii) $\mathrm{UH}(V^*_\Delta)$ is exactly $\mathrm{UH}(V^*)\cap\Delta$ and thus covers $L_{\le 2k}(H)\cap\Delta$. The number of extra vertices added is at most $2C'\frac{n}{Bk}$, so the size of V^* is now at most $(\frac{c_0'}{2} + 2\frac{C'}{B})\frac{n}{k}$.
- Next, for every cell $\Delta \in \mathrm{VD}(V_{\text{IN}})$, we *snap* the vertices in V^*_Δ to the vertices of Γ_Δ, i.e., we replace every vertex $v \in V^*_\Delta$ with the three vertices of the cell in Γ_Δ containing v. (See Figure 1.) This makes $V^*_\Delta \subseteq \Lambda_\Delta$. Then (i) every vertex in V^*_Δ now has level at most $4c_0k + \varepsilon|H_\Delta| \le 4c_0k + \varepsilon(2CBk) = (4c_0+1)k$, and (ii) $\mathrm{UH}(V^*_\Delta)$ can only increase in its coverage. The size of V^* triples to at most $(\frac{3}{2}c_0' + 6\frac{C'}{B})\frac{n}{k}$.

Then Line 1.2 guarantees that $|V_\Delta| \le |V^*_\Delta|$ by setting the constant $C := 4c_0 + 1$, since the subset $V^*_\Delta \subseteq \Lambda_\Delta$ satisfies the two stated conditions and V_Δ is the smallest such subset. Therefore, totalling over all cells in $\mathrm{VD}(V_{\text{IN}})$, we have $|V_{\text{OUT}}| \le |V^*| \le (\frac{3}{2}c_0' + 6\frac{C'}{B})\frac{n}{k} \le C'\frac{n}{k}$ as desired, by setting the constant $C' := \frac{\frac{3}{2}c_0'}{1-\frac{6}{B}}$ and picking a constant $B > 6$. ◀

▶ **Corollary 8.** *For a set H of n lines in $\mathbb{R}^2$, a parameter $k \in [1, n]$, and some suitable constants B, C, C', we can compute a (B^ik, CB^ik)-shallow cutting of size at most $C'\frac{n}{B^ik}$, along with its conflict lists, for all $i = 0, 1, \ldots, \log_B\frac{n}{k}$ in $O(n\log\frac{n}{k})$ total deterministic time. In particular, we can compute a (k, Ck)-shallow cutting of size $O(\frac{n}{k})$ in the stated time.*

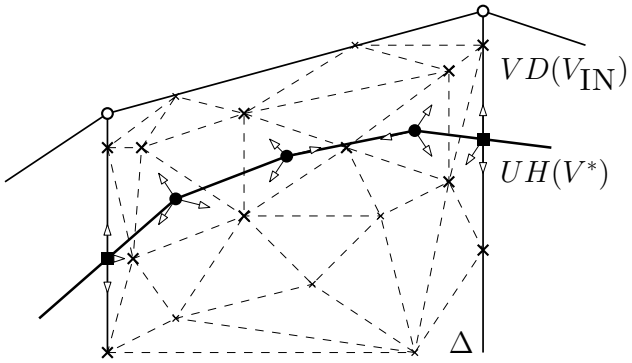

Figure 1 Modifying V^* by *chopping* (adding points marked by black squares) and *snapping* (replacing a point with three points indicated by white arrows). The cutting Γ_Δ is shown in dashed lines, and its vertices Λ_Δ are marked by crosses.

Proof. By Theorem 7, the running time $T(n,k)$ satisfies the recurrence

$$T(n,k) = T(n,Bk) + O\left(n + \frac{n}{k}\log\frac{n}{k}\right),$$

with the trivial base case $T(n,n) = O(n)$. The recurrence solves to $T(n,k) = O(n\log_B\frac{n}{k}) + O\left(\frac{n}{k}\log\frac{n}{k}\right)\sum_{i=0}^{\log_B\frac{n}{k}}\frac{1}{B^i} = O(n\log\frac{n}{k})$. ◀

4 A 3-d Shallow Cutting Algorithm

We now extend the approach from the previous section to 3-d. We need to incorporate planar separators in the algorithm and further new ideas in the analysis.

▶ **Theorem 9.** *For a set H of n planes in $\mathbb{R}^3$, a parameter $k \in [1,n]$ and some suitable constants B, C, C', given a (Bk, CBk)-shallow cutting V_{IN} for H of size at most $C'\frac{n}{Bk}$ along with its conflict lists, we can compute a (k,Ck)-shallow cutting V_{OUT} for H of size at most $C'\frac{n}{k}$ along with its conflict lists in $O(n + \frac{n}{k}\log\frac{n}{k})$ deterministic time.*

Proof.

Algorithm. Let ε and t be constants to be set later.

0. Group the faces of $\text{UH}(V_{\text{IN}})$ into regions by applying Fact 6 with parameter t. The union of the cells of $\text{VD}(V_{\text{IN}})$ defined by the triangles in a region will be called a *supercell* of $\text{VD}(V_{\text{IN}})$.

1. For each *supercell* Δ of $\text{VD}(V_{\text{IN}})$:

  1.1. Do as in Line 1.1 of the algorithm in Section 3.

  1.2. Do as in Line 1.2 of the algorithm in Section 3.

2. Do as in Line 3 of the algorithm in Section 3.

Complexity. Line 0 takes $O(\frac{n}{k}\log\frac{n}{k})$ time by Fact 6, since $|V_{\text{IN}}| \le C'\frac{n}{Bk} = O(\frac{n}{k})$. Lines 1.1, 1.2, and 2 take $O(n + \frac{n}{k}\log\frac{n}{k})$ time by an analysis similar to Section 3, since each supercell still has $O(1)$ complexity for t constant.

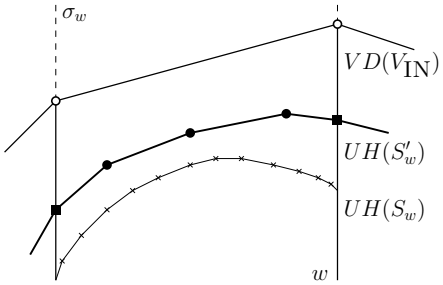

Figure 2 *Replacing* the concave chain $\text{UH}(S_w)$ with a sparse concave chain $\text{UH}(S'_w)$ at a wall w of a supercell of $\text{VD}(V_{\text{IN}})$. The set S'_w is a planar shallow cutting.

Correctness. By the same argument as in Section 3, we see that V_{OUT} is a correct (k, Ck)-shallow cutting for H, this time by setting the constant $\varepsilon := \frac{1}{3tCB}$, since $|H_\Delta| \le 3tCBk$ for each supercell Δ of $\text{VD}(V_{\text{IN}})$.

As in Section 3, we bound the size of V_{OUT} by comparing it against a $(2k, 2c_0k)$-shallow cutting V^* of size $c'_0\frac{n}{2k}$ provided by Fact 5. As before, V^* is covered by $\text{VD}(V_{\text{IN}})$, this time by picking a constant $B \ge 6c_0$.

We render V^* comparable to V_{OUT} by modifying V^* in three steps, the second of which is new (again these steps are not part of the algorithm but are for the correctness proof only):

- First, we *chop* $\text{UH}(V^*)$ at the walls of the supercells of $\text{VD}(V_{\text{IN}})$. A new planar concave chain of vertices is formed at each wall; we create two copies of the chain (one assigned to each of the two incident cells of $\text{VD}(V_{\text{IN}})$) and add their vertices to V^*. For each supercell Δ of $\text{VD}(V_{\text{IN}})$, let $V^*_\Delta := V^* \cap \Delta$. Then (i) every vertex in V^*_Δ has level at most $6c_0k$, and (ii) $\text{UH}(V^*_\Delta) \cap \Delta$ is exactly $\text{UH}(V^*) \cap \Delta$ and thus covers $L_{\le 2k}(H) \cap \Delta$. Unfortunately we do not have good enough bounds on the number of extra vertices added to V^*.
- To reduce the size of V^*, we *replace* the chain S_w of vertices at every wall w of a supercell with a *sparser* set S'_w of vertices defined as follows. (See Figure 2.) Let H_w be the set of planes in H intersecting w, and let S'_w be a planar $(6c_0k, 6c_0^2k)$-shallow cutting provided by Fact 5 for the intersection of H_w with the vertical plane through w (a set of lines). Let σ_w be the slab delimited by the two vertical lines through the two subwalls of w. We clip $\text{UH}(S'_w)$ to σ_w, add the two new vertices to S'_w, and remove any vertices outside σ_w. Observe that S'_w is covered by w (and thus by $\text{VD}(V_{\text{IN}})$) by picking a constant $B \ge 12c_0^2$, because every vertex in S'_w (including the two extra vertices added) has level in H_w at most $12c_0^2k$, and w covers $L_{\le Bk}(H) \cap \sigma_w = L_{\le Bk}(H_w) \cap \sigma_w$.
  Then (i) every vertex in V^*_Δ now has level (in H) at most $12c_0^2k$, and (ii) $\text{UH}(V^*_\Delta)$ can only increase in its coverage, because each old set S_w is contained in $L_{\le 6c_0k}(H) \cap \sigma_w$ and the new concave chain $\text{UH}(S'_w)$ covers $L_{\le 6c_0k}(H) \cap \sigma_w$.
  For each wall w, the size of S'_w is at most $c'_0\frac{|H_w|}{6c_0k} \le c'_0\frac{2CBk}{6c_0k} = \frac{c'_0CB}{3c_0}$. The number of walls of the supercells is at most $a'_0\frac{|V_{\text{IN}}|}{\sqrt{t}} \le \frac{a'_0C'}{B\sqrt{t}}\frac{n}{k}$. Thus, the total number of extra vertices added to V^* (two copies included) is at most $\frac{2a'_0c'_0CC'}{3c_0\sqrt{t}}\frac{n}{k}$, and the size of V^* is now at most $(\frac{c'_0}{2} + \frac{2a'_0c'_0CC'}{3c_0\sqrt{t}})\frac{n}{k}$.
- For every supercell Δ of $\text{VD}(V_{\text{IN}})$, we *snap* the vertices in V^*_Δ to vertices of Γ_Δ i.e., we replace every vertex $v \in V^*_\Delta$ with the four vertices of the cell in Γ_Δ containing v. This makes $V^*_\Delta \subseteq \Lambda_\Delta$. Then (i) every vertex in V^*_Δ now has level at most $12c_0^2k + \varepsilon|H_\Delta| \le 12c_0^2k + \varepsilon(3tCBk) = (12c_0^2 + 1)k$, and (ii) $\text{UH}(V^*_\Delta)$ can only increase in its coverage. The size of V^* quadruples to at most $(2c'_0 + \frac{8a'_0c'_0CC'}{3c_0\sqrt{t}})\frac{n}{k}$.

Then Line 1.2 guarantees that $|V_\Delta| \le |V^*_\Delta|$ by setting the constant $C := 12c_0^2+1$. Therefore, totalling over all cells in $\mathrm{VD}(V_{\text{IN}})$, we have $|V_{\text{OUT}}| \le |V^*| \le (2c'_0 + \frac{8a'_0c'_0CC'}{3c_0\sqrt{t}})\frac{n}{k} \le C'\frac{n}{k}$ as desired, by setting the constant $C' := \frac{2c'_0}{1-\frac{8a'_0c'_0C}{3c_0\sqrt{t}}}$ and picking any constant $t > (\frac{8a'_0c'_0C}{3c_0})^2$. ◀

As in Section 3, it follows that:

▶ **Corollary 10.** *For a set H of n planes in $\mathbb{R}^3$, a parameter $k \in [1,n]$, and some suitable constants B, C, C', we can compute a (B^ik, CB^ik)-shallow cutting of size at most $C'\frac{n}{B^ik}$, along with its conflict lists, for all $i = 0, 1, \ldots, \log_B \frac{n}{k}$ in $O(n \log \frac{n}{k})$ total deterministic time. In particular, we can compute a (k, Ck)-shallow cutting of size $O(\frac{n}{k})$ in the stated time.*

5 Final Remarks

We remark that concentrating on the $k = \Theta(\frac{n}{r})$ case is indeed without loss of generality—our algorithms can be easily applied to construct k-shallow $\frac{1}{r}$-cuttings for any k and r. Matoušek [25] proved the existence of such cuttings of size $O(r^{\lfloor d/2 \rfloor}(\frac{kr}{n}+1)^{\lceil d/2 \rceil})$. We can construct cuttings of this size with the following time bounds for $d \in \{2,3\}$, which are optimal if we are required to output all conflict lists (since the worst-case total size is $\Theta(r^{\lfloor d/2 \rfloor}(\frac{kr}{n}+1)^{\lceil d/2 \rceil}\frac{n}{r})$):

▶ **Corollary 11.** *For a set H of n lines in $\mathbb{R}^2$ and parameters $k, r \in [1,n]$, we can compute a k-shallow $\frac{1}{r}$-shallow cutting of size $O(r(\frac{kr}{n}+1))$, along with its conflict lists, in $O(n \log r + r(\frac{kr}{n}+1)\frac{n}{r})$ deterministic time.*

For a set H of n planes in $\mathbb{R}^3$ and parameters $k, r \in [1,n]$, we can compute a k-shallow $\frac{1}{r}$-shallow cutting of size $O(r(\frac{kr}{n}+1)^2)$, along with its conflict lists, in $O(n \log r + r(\frac{kr}{n}+1)^2\frac{n}{r})$ deterministic time.

Proof. If $k \le \frac{n}{cr}$ for a suitable constant c, then we can just apply our algorithm to compute a $\frac{n}{cr}$-shallow $\frac{1}{r}$-cutting of size $O(r)$ in $O(n \log r)$ deterministic time.

So assume $k > \frac{n}{cr}$. We first apply our algorithm to compute a k-shallow $\frac{ck}{n}$-cutting of size $O(\frac{n}{k})$ in $O(n \log \frac{n}{k}) = O(n \log r)$ deterministic time. Inside each cell Δ of this cutting, the conflict list H_Δ has size at most ck and we compute a standard $\frac{n/r}{ck}$-cutting of H_Δ of size $O((\frac{k}{n/r})^d)$ in deterministic time $O(k(\frac{k}{n/r})^{d-1})$ by known results (e.g., Chazelle's algorithm [14], or in the $d = 2$ case, our algorithm from the appendix). This yields a k-shallow $\frac{1}{r}$-cutting of H of total size $O(\frac{n}{k} \cdot (\frac{k}{n/r})^d)$ in total time $O(n \log r + \frac{n}{k} \cdot k(\frac{k}{n/r})^{d-1})$. The size and time bounds are exactly as stated. ◀

We should mention that despite their conceptual simplicity, our algorithms are not likely to be practical in their present form, because of the huge hidden constant factors.

Our approach of incorporating brute-force search and comparing the cost of our solution to that of an optimal solution was inspired by approximation algorithms. An interesting problem is to actually find PTASes to compute the minimum-size (shallow or standard) cutting, or compute cuttings with constant factors approaching the worst-case optimum [26], with comparable running time.

The optimality of the $O(n \log r)$ time bound assumes a comparison-based model, but it remains to be seen if there are faster algorithms to compute a single shallow cutting in the word RAM model for integer input [13].

Generalization of our shallow cutting algorithms to higher dimensions is also open; odd dimensions appear particularly challenging.

References

1. Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 180–186. SIAM, 2009.
2. Peyman Afshani, Timothy M. Chan, and Konstantinos Tsakalidis. Deterministic rectangle enclosure and offline dominance reporting on the RAM. In *Proceedings of the Forty-First International Colloquium on Automata, Languages, and Programming, Part I*, ICALP '14, pages 77–88, 2014.
3. Peyman Afshani and Konstantinos Tsakalidis. Optimal deterministic shallow cuttings for 3d dominance ranges. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1389–1398. SIAM, 2014.
4. Pankaj K. Agarwal. Partitioning arrangements of lines I: An efficient deterministic algorithm. *Discrete & Computational Geometry*, 5(1):449–483, 1990.
5. Pankaj K. Agarwal. *Intersection and Decomposition Algorithms for Planar Arrangements*. Cambridge University Press, New York, NY, USA, 1991.
6. Pankaj K. Agarwal, Boris Aronov, Timothy M. Chan, and Micha Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete & Computational Geometry*, 19(3):315–331, 1998.
7. Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull with optimal query time. In *Proceedings of the Seventh Scandinavian Workshop on Algorithm Theory*, SWAT '00, pages 57–70, 2000.
8. Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the Forty-Third Symposium on Foundations of Computer Science*, FOCS '02, pages 617–626. IEEE, 2002. Current draft of full paper at https://pwgrp1.inf.ethz.ch/Current/DPCH/Journal/topdown.pdf.
9. Timothy M. Chan. Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions. *SIAM Journal on Computing*, 30(2):561–575, 2000.
10. Timothy M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, April 2005.
11. Timothy M. Chan. Three problems about dynamic convex hulls. *International Journal of Computational Geometry & Applications*, 22(04):341–364, 2012.
12. Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proceedings of the Twenty-Seventh Symposium on Computational Geometry*, SOCG '11, pages 1–10. ACM, 2011.
13. Timothy M. Chan and Mihai Pătraşcu. Transdichotomous results in computational geometry, I: point location in sublogarithmic time. *SIAM J. Comput.*, 39(2):703–729, 2009.
14. Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(1):145–158, 1993.
15. Bernard Chazelle and Joel Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.
16. Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.
17. Martin E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM Journal on Computing*, 13(1):31–45, 1984.
18. Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987.
19. Michael T. Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and System Sciences*, 51(3):374–389, 1995.
20. Sariel Har-Peled, Haim Kaplan, Micha Sharir, and Shakhar Smorodinsky. Epsilon-nets for halfspaces revisited. *CoRR*, abs/1410.3154, 2014.

21 David Haussler and Emo Welzl. ϵ-nets and simplex range queries. *Discrete & Computational Geometry*, 2(1):127–151, 1987.

22 Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

23 Jiří Matoušek. Construction of ε-nets. *Discrete & Computational Geometry*, 5(1):427–448, 1990.

24 Jiří Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6(1):385–406, 1991.

25 Jiří Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2(3):169–186, 1992.

26 Jiří Matoušek. On constants for cuttings in the plane. *Discrete & Computational Geometry*, 20(4):427–448, 1998.

27 Nimrod Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

28 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, 1984.

29 Edgar A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry*, SoCG '99, pages 390–399. ACM, 1999.

30 Edgar A. Ramos. Deterministic algorithms for 3-d diameter and some 2-d lower envelopes. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*, SoCG '00, pages 290–299. ACM, 2000.

A Appendix: A 2-d Standard Cutting Algorithm

In this appendix, we describe how our ideas can be used to rederive known results by Matoušek [24] and Chazelle [14] for standard cuttings in 2-d.

As before, it will be more convenient to work with the parameter $K := \frac{n}{r}$ instead of r. The target $O(nr)$ time bound becomes $O(\frac{n^2}{K})$. Our cuttings will be the vertical decompositions of noncrossing line segments. Given a set S of noncrossing line segments inside a cell Δ, we define the *vertical decomposition* $\mathrm{VD}(S)$ to be the subdivision into trapezoids, obtained by drawing a vertical upward/downward ray at each vertex till the ray hits another segment. We define $\mathrm{VD}_\Delta(S)$ to be $\mathrm{VD}(S)$ clipped inside a given cell Δ.

▶ **Theorem 12.** *For a set H of n lines in $\mathbb{R}^2$, a parameter $K \in [1, n]$ and suitable constants B, C, given a $\frac{BK}{n}$-cutting T_{IN} for H of size at most $C(\frac{n}{BK})^2$ along with its conflict lists, we can compute a $\frac{K}{n}$-cutting T_{OUT} for H of size at most $C(\frac{n}{K})^2$ along with its conflict lists in $O(\frac{n^2}{K})$ deterministic time.*

Proof.

Algorithm. Let ε be a constant to be set later.

1. For each cell $\Delta \in T_{\text{IN}}$:
   1.1. Compute by Fact 4 an ε-cutting Γ_Δ for H_Δ of $O(1)$ size, where the cells are clipped (and re-triangulated) to lie within Δ. Further refine the cells of Γ_Δ by drawing a vertical line at every vertex of Γ_Δ. Let Λ_Δ be the set of vertices that define the cells of (the refined) Γ_Δ.

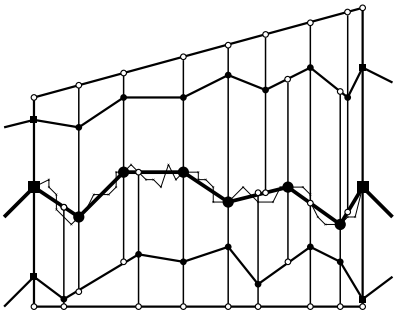

Figure 3 *Simplifying* a level.

1.2. Compute by brute force the *smallest set* of noncrossing line segments S_Δ, whose endpoints are from Λ_Δ, such that each trapezoid in $\mathrm{VD}_\Delta(S_\Delta)$ intersects at most K lines of H_Δ.

2. Return $T_{\text{OUT}} := \bigcup_{\Delta \in T_{\text{IN}}} \mathrm{VD}_\Delta(S_\Delta)$ and all its conflict lists.

Complexity. Line 1.1 takes time linear in $|H_\Delta|$ by Fact 4, for a total of $\sum_{\Delta \in T_{\text{IN}}} O(|H_\Delta|) = O(C(\frac{n}{BK})^2 \cdot BK) = O(\frac{n^2}{K})$ time. For Line 1.2, we probe all possible sets S_Δ of line segments with endpoints from Λ_Δ. Since Γ_Δ and Λ_Δ have $O(1)$ size, there are "only" $O(1)$ sets to test and each set can be tested in $O(|H_\Delta|)$ time. Thus, the whole step takes time linear in $|H_\Delta|$, which again totals to $O(\frac{n^2}{K})$.

Correctness. Clearly T_{OUT} is a $\frac{K}{n}$-cutting for H. To bound the size of T_{OUT}, we compare it against some optimal $\frac{K}{n}$-cutting for H of size $O((\frac{n}{K})^2)$, specifically, the cutting produced by Matoušek's construction [23] using levels. (We would have preferred a cleaner proof that compares T_{OUT} against an arbitrary optimal-size cutting, like in our earlier proofs, but were unable to make the details work.) We adapt his construction to incorporate our earlier ideas of *chopping* and *snapping*.

- We first pick a random index $j_0 \in [1, 0.5K]$. For each $j \equiv j_0 \pmod{0.5K}$, consider the j-level (the set of points on the arrangement with level j), which is an x-monotone chain. Since the arrangement has $O(n^2)$ vertices in total, the expected total number of vertices in these chains is $O(\frac{n^2}{K})$.
- We *chop* these chains into subchains at the boundaries of the cells of T_{IN}. Since the total number of vertices along cell boundaries is $O(C(\frac{n}{BK})^2 \cdot BK) = O(\frac{C}{B})\frac{n^2}{K}$, the expected total number of subchains created is at most $O(\frac{C}{B})\frac{n^2}{K^2}$.
- We *simplify* each subchain by selecting every $0.1k$-th vertex from the subchain and forming a shorter x-monotone chain through these vertices, while keeping the start and end vertex. (See Figure 3.) Note that the levels of points on a simplified subchain can deviate from the original level by at most $\pm 0.1K$. Let S^* be the set of the edges of the simplified subchains. Since the size of a simplified subchain is at most one plus $\frac{1}{0.1K}$-th the original size of the subchain, the expected size of S^* is at most $O(1 + \frac{C}{B})(\frac{n}{K})^2$. We pick a j_0 so that the size of S^* is at most its expectation.
  For each boundary edge of the cells of T_{IN}, we subdivide it by selecting every $0.1K$-th vertex of the arrangement lying on the edge. We add two copies of the resulting edges to S^* (one assigned to each of the two incident cells). Since the number of extra edges added is $O(C(\frac{n}{BK})^2 \cdot \frac{BK}{0.1K}) = O(\frac{C}{B}(\frac{n}{K})^2)$, the size of S^* remains at most $O(1 + \frac{C}{B})(\frac{n}{K})^2$.

Let $S^*_\Delta := S^* \cap \Delta$. We claim that each trapezoid in $\mathrm{VD}_\Delta(S^*_\Delta)$ intersects at most $0.9K$ lines. This follows because the left side of the trapezoid intersects at most $0.5K + 0.1K + 0.1K$ lines, and the top or bottom side intersects at most $0.1K$ lines.

- For every cell $\Delta \in T_{\text{IN}}$, we *snap* the endpoints of the segments in S^*_Δ to the vertices of Γ_Δ, i.e., we replace each such endpoint v with the rightmost vertex of the cell in Γ_Δ containing v. For each endpoint v that lie on a boundary edge of Δ, we snap it to a vertex of Γ_Δ on that edge.
  Note that the x-order of the vertices in S^*_Δ is preserved after snapping, because we have refined Γ_Δ with extra vertical lines. Thus, the simplified subchains inside Δ of a common chain remain x-monotone and noncrossing. Furthermore, two simplified subchains of two different chains remain noncrossing for a sufficiently small ε, since the two chains have levels at least $0.5K$ apart, simplification changes levels by at most $0.1K$, and snapping changes levels by at most $O(\varepsilon BK)$. Thus, S^*_Δ remains noncrossing.
  By modifying the previous argument, we see that each trapezoid in $\mathrm{VD}_\Delta(S^*_\Delta)$ intersects at most $0.9K + O(\varepsilon BK)$ lines; the number can be made at most K for a sufficiently small constant ε.

Then Line 1.2 guarantees that $|S_\Delta| \le |S^*_\Delta|$. Therefore, $|T_{\text{OUT}}| \le O(|S^*|) \le O(1 + \frac{C}{B})(\frac{n}{K})^2$, which can be made at most $C(\frac{n}{K})^2$ as desired, by choosing a sufficiently large constant B. (Note that in the entire correctness proof, constants hidden in the O notation are universal constants.) ◀

▶ **Corollary 13.** *For a set H of n lines in $\mathbb{R}^2$, a parameter $K \in [1, n]$, and some suitable constants B, C, we can compute a $\frac{B^i K}{n}$-cutting of size at most $C(\frac{n}{B^i K})^2$ for all $i = 0, 1, \ldots, \log_B \frac{n}{K}$, along with its conflict lists, in $O(\frac{n^2}{K})$ total deterministic time. In particular, we can compute a $\frac{1}{r}$-cutting of size $O(r^2)$ in $O(nr)$ deterministic time.*

Proof. The recurrence $T(n, K) = T(n, BK) + O((\frac{n}{K})^2)$, with the trivial base case $T(n, n) = O(n)$, solves to $T(n, K) = O(\sum_{i=0}^{\log_B \frac{n}{K}} (\frac{n}{B^i K})^2) = O((\frac{n}{K})^2)$. ◀

Our above algorithm can be viewed as a reinterpretation of Chazelle's algorithm [14], where ε-approximations and sparse ε-nets are replaced by a brute-force component that is more self-contained to describe. Our analysis only works in 2-d, however; Chazelle's approach is still more powerful.