Unions of Onions: Preprocessing Imprecise Points for Fast Onion Layer Decomposition

Maarten Löffler[1] and Wolfgang Mulzer[2]

[1] Department of Information and Computing Sciences, Universiteit Utrecht, The Netherlands, m.loffler@uu.nl
[2] Institut für Informatik, Freie Universität Berlin, Germany, mulzer@inf.fu-berlin.de

Abstract. Let $\mathcal{D}$ be a set of n pairwise disjoint unit disks in the plane. We describe how to build a data structure for $\mathcal{D}$ so that for any point set P containing exactly one point from each disk, we can quickly find the onion decomposition (convex layers) of P.
Our data structure can be built in $O(n \log n)$ time and has linear size. Given P, we can find its onion decomposition in $O(n \log k)$ time, where k is the number of layers. We also provide a matching lower bound.
Our solution is based on a recursive space decomposition, combined with a fast algorithm to compute the union of two disjoint onion decompositions.

1 Introduction

Let P be a planar n-point set. Take the convex hull of P and remove it; repeat until P becomes empty. This process is called *onion peeling*, and the resulting decomposition of P into convex polygons is the *onion decomposition*, or *onion* for short, of P. It can be computed in $O(n \log n)$ time [6]. Onions provide a natural, more robust, generalization of the convex hull, and they have applications in pattern recognition, statistics, and planar halfspace range searching [7, 14, 22]

Recently, a new paradigm has emerged for modeling data imprecision. Suppose we need to compute some interesting property of a planar point set. Suppose further that we have some advance knowledge about the possible locations of the points, e.g., from an imprecise sensor measurement. We would like to preprocess this information, so that once the precise inputs are available, we can obtain our structure faster. We will study the complexity of computing onions in this framework.

1.1 Related Work

The notion of onion layer decompositions first appears in the computational statistics literature [14], and several rather brute-force algorithms to compute it have been suggested (see [9] and the references therein). In the computational geometry community, Overmars and van Leeuwen [21] presented the first near-linear time algorithm, requiring $O(n \log^2 n)$ time. Chazelle [6] improved this

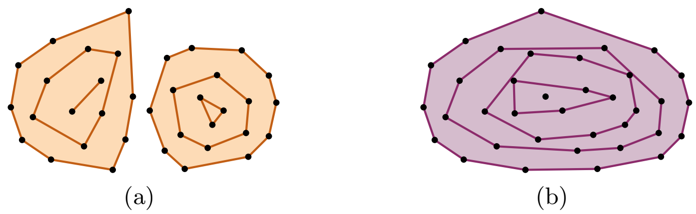


Fig. 1. (a) Two disjoint onions. (b) Their union.

to an optimal $O(n \log n)$ time algorithm. Nielsen [20] gave an output-sensitive algorithm to compute only the outermost k layers in $O(n \log h_k)$ time, where h_k is the number of vertices participating on the outermost k layers. In $\mathbb{R}^3$, Chan [5] described an $O(n \log^6 n)$ expected time algorithm.

The framework for preprocessing regions that represent points was first introduced by Held and Mitchell [12], who show how to store a set of disjoint unit disks in a data structure such that any point set containing one point from each disk can be triangulated in linear time. This result was later extended to arbitrary disjoint regions in the plane by van Kreveld *et al.* [16]. Löffler and Snoeyink first showed that the Delaunay triangulation (or its dual, the Voronoi diagram) can also be computed in linear time after preprocessing a set of disjoint unit disks [17]. This result was later extended by Buchin *et al.* [4], and Devillers gives a practical alternative [8]. Ezra and Mulzer [10] show how to preprocess a set of lines in the plane such that the convex hull of a set of points with one point on each line can be computed faster than $n \log n$ time.

These results also relate to the *update complexity* model. In this paradigm, the input values or points come with some uncertainty, but it is assumed that during the execution of the algorithm, the values or locations can be obtained exactly, or with increased precision, at a certain cost. The goal is then to compute a certain combinatorial property or structure of the precise set of points, while minimising the cost of the updates made by the algorithm [3, 11, 13, 23].

1.2 Results

We begin by showing that the union of two disjoint onions can be computed in $O(n + k^2 \log n)$ time, where k is the number of layers in the resulting onion.

We apply this algorithm to obtain an efficient solution to the onion preprocessing problem mentioned in the introduction. Given n pairwise disjoint unit disks that model an imprecise point set, we build a data structure of size $O(n)$ such that the onion decomposition of an instance can be retrieved in $O(n \log k)$ time, where k is the number of layers in the resulting onion. We present several preprocessing algorithms. The first is very simple and achieves $O(n \log n)$ expected time. The second and third algorithm make this guarantee deterministic, at the cost of worse constants and/or a more involved algorithm.

We also show that the dependence on k is necessary: in the worst case, any comparison-based algorithm can be forced to take $\Omega(n \log k)$ time on some instances.

2 Preliminaries and Definitions

Let P be a set of n points in $\mathbb{R}^2$. The *onion decomposition*, or *onion*, of P, is the sequence $\circledcirc(P)$ of nested convex polygons with vertices from P, constructed recursively as follows: if $P \neq \emptyset$, we set $\circledcirc(P) := \{\mathrm{ch}(P)\} \cup \circledcirc(P \setminus \mathrm{ch}(P))$, where $\mathrm{ch}(P)$ is the convex hull of P; if $P = \emptyset$, then $\circledcirc(P) := \emptyset$ [6]. An element of $\circledcirc(P)$ is called a *layer* of P. We represent the layers of $\circledcirc(P)$ as dynamic balanced binary search trees, so that operations *split* and *join* can be performed in $O(\log n)$ time.

Let $\mathcal{D}$ be a set of disjoint unit disks in $\mathbb{R}^2$. We say a point set P is a *sample* from $\mathcal{D}$ if every disk in $\mathcal{D}$ contains exactly one point from P. We write $\log$ for the logarithm with base 2.

3 The Algorithm

Our algorithm requires several pieces, to be described in the following sections.

3.1 Unions of Onions

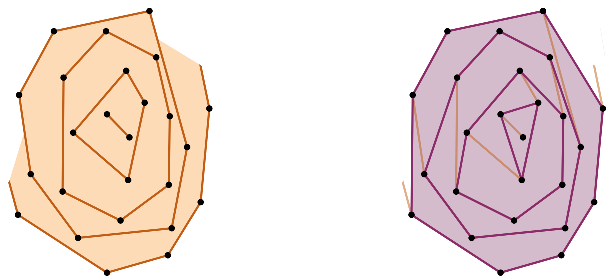
Suppose we have two point sets P and Q, together with their onions. We show how to find $\circledcirc(P \cup Q)$ quickly, given that $\circledcirc(P)$ and $\circledcirc(Q)$ are disjoint. Deleting points can only decrease the number of layers, so:

Observation 3.1 *Let $P, Q \subseteq \mathbb{R}^2$. Then $\circledcirc(P)$ and $\circledcirc(Q)$ cannot have more layers than $\circledcirc(P \cup Q)$.* □

The following lemma constitutes the main ingredient of our onion-union algorithm. A *convex chain* is any connected subset of a convex closed curve.

Lemma 3.2. *Let A and B be two non-crossing convex chains. We can find $\mathrm{ch}(A \cup B)$ in $O(\log n)$ time.*

Proof. Since A and B do not cross, the pieces of A and B that appear on $\mathrm{ch}(A \cup B)$ are both connected: otherwise, $\mathrm{ch}(A \cup B)$ would contain four points belonging to A, B, A, and B, in that order. However, the points on A must be connected inside $\mathrm{ch}(A \cup B)$; as do the points on B. Thus, the chains A and B cross, which is impossible. Since A and B are convex chains, we can compute $\mathrm{ch}(A), \mathrm{ch}(B)$ in $O(\log n)$ time. Furthermore, since A and B are disjoint, we can also, in $O(\log n)$ time, make sure that $\mathrm{ch}(A) \cap \mathrm{ch}(B) = \emptyset$, by removing parts from A or B, if necessary. Now we can find the bitangents of $\mathrm{ch}(A)$ and $\mathrm{ch}(B)$ in logarithmic time [15]. □



(a) (b)

Fig. 2. (a) A half-eaten onion; (b) the restored onion.

Lemma 3.3. *Suppose* 🧅(P) *has* k *layers. Let* A *be the outer layer of* 🧅(P)*, and* p, q *be two vertices of* A*. Let* A_1 *be the points on* A *between* p *and* q*, going counter-clockwise. We can find* 🧅$(P \setminus A_1)$ *in* $O(k \log n)$ *time.*

Proof. The points p and q partition A into two pieces, A_1 and A_2. Let B be the second layer of 🧅(P). The outer layer of 🧅$(P \setminus A_1)$ is the convex hull of $P \setminus A_1$, i.e., the convex hull of A_2 and B. By Lemma 3.2, we can find it in $O(\log n)$ time. Let $p', q' \in P$ be the points on B where the outer layer of 🧅$(P \setminus A_1)$ connects. We remove the part between p' and q' from B, and use recursion to compute the remaining layers of 🧅$(P \setminus A_1)$ in $O((k-1)\log n)$ time; see Figure 2. □

We conclude with the main theorem of this section:

Theorem 3.4. *Let* P *and* Q *be two planar point sets of total size* n*. Suppose that* 🧅(P) *and* 🧅(Q) *are disjoint. We can find the onion* 🧅$(P \cup Q)$ *in* $O(k^2 \log n)$ *time, where* k *is the resulting number of layers.*

Proof. By Observation 3.1, 🧅(P) and 🧅(Q) each have at most k layers. We use Lemma 3.2 to find $\mathrm{ch}(P \cup Q)$ in $O(\log n)$ time. By Lemma 3.3, the remainders of 🧅(P) and 🧅(Q) can be restored to proper onions in $O(k \log n)$ time. The result follows by induction. □

3.2 Space Decomposition Trees

We now describe how to preprocess the disks in $\mathcal{D}$ for fast divide-and-conquer. A *space decomposition tree* (SDT) T is a rooted binary tree where each node v is associated with a planar region R_v. The root corresponds to all of $\mathbb{R}^2$; for each leaf v of T, the region R_v intersects only a constant number of disks in $\mathcal{D}$. Furthermore, each inner node v in T is associated with a directed line ℓ_v, so that if u is the left child and w the right child of v, then $R_u := R_v \cap \ell_v^+$ and $R_w := R_v \cap \ell_v^-$. Here, ℓ_v^+ is the halfplane to the left of ℓ_v and ℓ_v^- the halfplane to the right of ℓ_v; see Figure 3

Let $\alpha, \beta \in (0, 1)$, and let T be an SDT. For a node v of T, let d_v denote the number of disks in $\mathcal{D}$ that intersect R_v. We call T an (α, β)-SDT for $\mathcal{D}$ if for every inner node v we have that (i) the line ℓ_v intersects at most d_v^β disks in R_v; and (ii) $d_u, d_w \leq \alpha d_v$, where u and w are the children of v.

Lemma 3.5. *Let T be an (α, β)-SDT. The tree T has height $O(\log n)$ and $O(n)$ nodes. Furthermore, $\sum_{v \in T} d_v = O(n \log n)$.*

Proof. The fact that T has height $O(\log n)$ is immediate from property (ii) of an (α, β)-SDT. For $i = 0, \dots, \log n$, let $V_i := \{v \in T \mid d_v \in [2^i, 2^{i+1})\}$, the set of nodes whose regions intersect between 2^i and 2^{i+1} disks. Note that the sets V_i constitute a partition of the nodes. Let $\widetilde{V}_i \subseteq V_i$ be the nodes in V_i whose parent is not in V_i. By property (ii) again, the d_v along any root-leaf path in T are monotonically decreasing, so the nodes in $\widetilde{V}_i$ are unrelated (i.e., no node in $\widetilde{V}_i$ is an ancestor or descendant of another node in $\widetilde{V}_i$). Furthermore, the nodes in V_i induce in T a forest F_i such that each tree in F_i has a root from $\widetilde{V}_i$ and constant height (depending on α).

Let $D_i := \sum_{v \in \widetilde{V}_i} d_v$. We claim that for $i = 0, \dots, \log n$, we have

$$D_i \leq n \prod_{j=i}^{\log n} \left(1 + c2^{j(\beta-1)}\right), \tag{1}$$

for some large enough constant c. Indeed, consider a node $v \in \widetilde{V}_j$. As noted above, v is the root of a tree F_v of constant height in the forest induced by V_j. By property (i), any node u in this subtree adds at most $d_u^\beta < 2^{(j+1)\beta}$ additional disk intersections (i.e., $d_a + d_b \leq d_u + 2^{(j+1)\beta}$, where a, b are the children of u). Since F_v has constant size, the total increase in disk intersections in F_v is thus at most $c'2^{(j+1)\beta}$, for some constant c'. Since $d_v \geq 2^j$, it follows that the number of disk intersections increases multiplicatively by a factor of at most $1 + c'2^{(j+1)\beta}/2^j \leq 1 + c2^{j(\beta-1)}$, for some constant c. The trees F_v partition T and the root intersects n disks, so for the nodes in $\widetilde{V}_i$, the total number of disk intersections has increased by a factor of at most $\prod_{j=i}^{\log n}\left(1 + c2^{j(\beta-1)}\right)$, giving (1). The product in (1) is easily estimated:

$$D_i \leq n \prod_{j=i}^{\log n} (1 + c2^{j(\beta-1)}) \leq ne^{\sum_{j=i}^{\log n} c2^{j(\beta-1)}} = ne^{O(1)} = O(n),$$

since $\beta < 1$. Hence, each set $\widetilde{V}_i$ has at most $O(n/2^i)$ nodes for $i = 1, \dots, \log n$. The total size of all $\widetilde{V}_i$ is $O(n)$. Since each $v \in V_i$ lies in a constant size subtree rooted at a $w \in \widetilde{V}_i$, it follows that T has $O(n)$ nodes. For the same reason, we get that $\sum_{v \in T} d_v = O(n \log n)$. □

Now there are several ways to obtain an (α, β)-SDT for $\mathcal{D}$. A very simple construction is based on the following lemma, which is an algorithmic version of a result by Alon *et al.* [2, Theorem 1.2]. See Section 4 for alternative approaches.

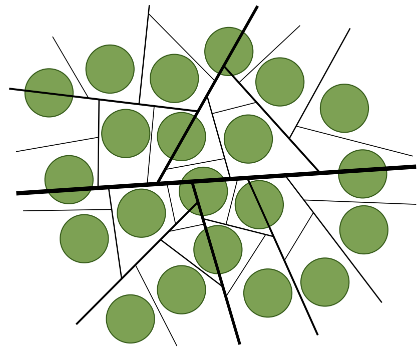


Fig. 3. A space decomposition tree for 21 unit disks.

Lemma 3.6. *There exists a constant $c \geq 0$, so that for any set $\mathcal{D}$ of m congruent nonoverlapping disks in the plane, there is a line ℓ with at least $m/2 - c\sqrt{m \log m}$ disks completely to each side of it. We can find ℓ in $O(m)$ expected time.*

Proof. Our proof closely follows Alon *et al.* [2, Section 2]. Set $r := \lfloor \sqrt{m/\log m} \rfloor$, and pick a random integer z between 1 and $r/2$. Find a line ℓ whose angle with the x-axis is $(z/r)\pi$ and that has $\lfloor m/2 \rfloor$ disk centers on each side. Given z, we can find ℓ in $O(m)$ time by a median computation. The proof by Alon *et al.* implies that with probability at least $1/2$ over the choice of z, the line ℓ intersects at most $c\sqrt{m \log m}$ disks in $\mathcal{D}$, for some constant $c \geq 0$. Thus, we need two tries in expectation to find a good line ℓ. The expected running time is $O(m)$. □

To obtain a $(1/2+\varepsilon, 1/2+\varepsilon)$-SDT T for $\mathcal{D}$, we apply Lemma 3.6 recursively until the region for each node intersects only a constant number of disks. Since the expected running time per node is linear in the number of intersected disks, Lemma 3.5 shows that the total expected running time is $O(n \log n)$.

By Lemma 3.5, the leaves of T induce a planar subdivision G_T with $O(n)$ faces. We add a large enough bounding box to G_T and triangulate the resulting graph. Since G_T is planar, the triangulation has complexity $O(n)$ and can be computed in the same time (no need for heavy machinery—all faces of G_T are convex). With each disk in $\mathcal{D}$, we store the list of triangles that intersect it (recall that each triangle intersects a constant number of disks). This again takes $O(n)$ time and space. We conclude with the main theorem of this section:

Theorem 3.7. *Let $\mathcal{D}$ be a set of n disjoint unit disks in $\mathbb{R}^2$. In $O(n \log n)$ expected time, we can construct an $(1/2+\varepsilon, 1/2+\varepsilon)$ space partition tree T for $\mathcal{D}$. Furthermore, for each disk $D \in \mathcal{D}$, we have a list of triangles T_D that cover the leaf regions of T that intersect D.* □

3.3 Processing a Precise Input

Suppose we have an (α, β)-SDT together with a point location structure as in Theorem 3.7. Let P be a sample from $\mathcal{D}$. Suppose first that we know k, the number of layers in 🧅(P). For each input point p_i, let $D_i \in \mathcal{D}$ be the corresponding disk. We check all triangles in T_{D_i}, until we find the one that contains p_i. Since there are $O(n)$ triangles, this takes $O(n)$ time. Afterwards, we know for each point in P the leaf of T that contains it.

For each node v of T, let n_v be the number of points in the subtree rooted at v. We can compute the n_v's in total time $O(n)$ by a postorder traversal of T. The *upper tree* T_u of T consists of all nodes v with $n_v \geq k^2$. Each leaf of T_u corresponds to a subset of P with $O(k^2)$ points. For each such subset, we use Chazelle's algorithm [6] to find its onion decomposition in $O(k^2 \log k)$ time. Since the subsets are disjoint, this takes $O(n \log k)$ total time. Now, in order to obtain 🧅(P), we perform a postorder traversal of T_u, using Theorem 3.4 in each node to unite the onions of its children. This gives 🧅(P) at the root.

The time for the onion union at a node v is $O(k^2 \log n_v)$. We claim that for $i = 2\log k, \dots, \log n$, the upper tree T_u contains at most $O(n/2^i)$ nodes v with $n_v \in [2^i, 2^{i+1})$. Given the claim, the total work is proportional to

$$\sum_{v \in T_u} k^2 \log n_v \leq \sum_{i=2\log k}^{\log n} \frac{n}{2^i} k^2 (i+1) = nk^2 \sum_{i=2\log k}^{\log n} \frac{i+1}{2^i} = O(n \log k),$$

since the series $\sum_{i=2\log k}^{\log n} (i+1)/2^i$ is dominated by the first term $(\log k)/k^2$. It remains to prove the claim. Fix $i \in \{2\log k, \dots, \log n\}$ and let V_i be the nodes in T_u with $n_v \in [2_i, 2^{i+1})$, whose parents have $n_v \geq 2^{i+1}$. Since the nodes in V_i represent disjoint subsets of P, we have $|V_i| \leq n/2^i$. Furthermore, by property (i) of an (α, β)-SDT , both children w_1, w_2 for every node $v \in T_u$ have $n_{w_1}, n_{w_2} \leq \alpha n_v$, so that after $O(1)$ levels, all descendants w of $v \in V$ have $n_w < 2^i$. The claim follows.

So far, we have assumed that k is given. Using standard exponential search, this requirement can be removed. More precisely, for $i = 1, \dots, \log\log n$, set $k_i = 2^{2^i}$. Run the above algorithm for $k = k_0, k_1, \dots$. If the algorithm succeeds, report the result. If not, abort as soon as it turns out that an intermediate onion has more than k_i layers and try k_{i+1}. The total time is

$$\sum_{i=0}^{\log\log k} O(n2^i) = O(n \log k),$$

as desired. This finally proves our main result.

Theorem 3.8. *Let $\mathcal{D}$ be a set of n disjoint unit disks in $\mathbb{R}^2$. We can build a data structure that stores $\mathcal{D}$, of size $O(n)$, in $O(n \log n)$ expected time, such that given a sample P of $\mathcal{D}$, we can compute* 🧅(P) *in $O(n \log k)$ time, where k is the number of layers in* 🧅(P). □

Remark. Using the same approach, without the exponential search, we can also compute the outermost k layers of an onion with arbitrarily many layers in $O(n \log k)$ time, for any k. In order to achieve this, we simply abort the union algorithm whenever k layers have been found, and note that by Observation 3.1, the points in P not on the outermost k layers of 🧅(P) will never be part of the outermost k layers of 🧅(Q) for any $Q \supset P$.

4 Deterministic Preprocessing

We now present alternatives to Lemma 3.6. First, we describe a very simple construction that gives a deterministic way to build an $(9/10+\varepsilon, 1/2+\varepsilon)$-SDT in $O(n\log n)$ time.

Lemma 4.1. *Let $\mathcal{D}$ be a set of m non-overlapping unit disks. Suppose that the centers of $\mathcal{D}$ have been sorted in horizontal and vertical direction. Then we can find in $O(m)$ time a (vertical or horizontal) line ℓ, such that ℓ intersects $O(\sqrt{m})$ disks and such that ℓ has at least $m/10$ disks from $\mathcal{D}$ completely to each side.*

Proof. Let $\mathcal{D}_l$, $\mathcal{D}_r$, $\mathcal{D}_t$, $\mathcal{D}_b$ be the $m/10$ left-, right-, top-, and bottommost disks in $\mathcal{D}$, respectively. We can find these disks in $O(m)$ time, since we know the horizontal and vertical order of their centers. We call $\mathcal{D}_o := \mathcal{D}_l \cup \mathcal{D}_r \cup \mathcal{D}_t \cup \mathcal{D}_b$ the *outer disks*, and $\mathcal{D}_i := \mathcal{D} \setminus \mathcal{D}_o$ the *inner disks*.

Let R be the smallest axis-aligned rectangle that contains all inner disks. Again, R can be found in linear time. There are $\Omega(m)$ inner disks, and all disks are disjoint, so the area of R must be $\Omega(m)$. Thus, R has width or height $\Omega(\sqrt{m})$; assume wlog that it has width $\Omega(\sqrt{m})$. Let $R' \subseteq R$ be the rectangle obtained by moving the left boundary of R to the right by two units, and the right boundary of R to the left by two units. The rectangle R' still has width $\Omega(\sqrt{m})$, and it intersects no disks from $\mathcal{D}_l \cup \mathcal{D}_r$. There are $\Omega(\sqrt{m})$ vertical lines that intersect R' and that are spaced at least one unit apart. Each such line has at least $m/10$ disks completely to each side, and each disk is intersected by at most one line. Hence, there must be a line that intersects $O(\sqrt{m})$ disks, as claimed. We can find such a line in $O(m)$ time by sweeping the disks from left to right. ☐

The next lemma improves the constants of the previous construction. It allows us to compute an $(1/2+\varepsilon, 5/6+\varepsilon)$-SDT tree in deterministic time $O(n\log^2 n)$, but it requires comparatively heavy machinery.

Lemma 4.2. *Let $\mathcal{D}$ be a set of m congruent non-overlapping disks. In deterministic time $O(m\log m)$, we can find a line ℓ such that there are at least $m/2 - cm^{5/6}$ disks completely to each side of ℓ.*

Proof. Let X be a planar n-point set, and let $1 \le r \le n$ be a parameter. A *simplicial r-partition of X* is a sequence $\Delta_1, \dots, \Delta_a$ of $a = \Theta(r)$ triangles and a partition $X = X_1 \dot\cup \cdots \dot\cup X_a$ of X into a pieces such that (i) for $i = 1, \dots, a$, we have $X_i \subseteq \Delta_i$ and $|X_i| \in \{n/r, \dots, 2n/r\}$; and (ii) every line ℓ intersects $O(\sqrt{r})$ triangles Δ_i. Matoušek showed that a simplicial r-partition exists for every planar n-point set and for every r. Furthermore, this partition can be found in $O(n\log r)$ time (provided that $r \le n^{1-\delta}$, for some $\delta > 0$) [18, Theorem 4.7].

Let $\gamma, \delta \in (0,1)$ be two constants to be determined later. Set $r := m^\gamma$. Let Q be the set of centers of the disks in $\mathcal{D}$. We compute a simplicial r-partition for Q in $O(m\log m)$ time. Let $\Delta_1, \dots, \Delta_a$ be the resulting triangles and $Q = Q_1 \dot\cup \cdots \dot\cup Q_a$ the partition of Q. Set $s := m^\delta$, and for $i = 1, \dots, s$, let ℓ_i' be the line through the origin that forms an angle $(i/2s)\pi$ with the positive x-axis.

Let Y_i be the projection of the triangles $\Delta_1, \dots, \Delta_a$ onto ℓ'_i. We interpret Y_i as a set of weighted intervals, where the weight of an interval is the size $|Q_j|$ of the associated point set for the corresponding triangle. By the properties of the simplicial partition, the interval set Y_i has *depth* $O(\sqrt{r})$, i.e., every point on ℓ'_i is covered by at most $O(\sqrt{r})$ intervals of Y_i.

Note that the sets Y_i can be determined in $O(sr \log r) = O(m^{\gamma+\delta} \log m) = O(m)$ total time, for γ, δ small enough. Now, for each Y_i, we find a point c_i on ℓ'_i that has intervals of total weight $m/2 - O(\sqrt{r}(m/r)) = m/2 - O(m^{1-\gamma/2})$ completely to each side. Since the depth of Y_i is $O(\sqrt{r})$, we can find such a point in time $O(\log r)$ with binary search, for a total of $O(s \log r) = O(m)$ time (it would even be permissible to spend time $O(r)$ on each Y_i). Let ℓ_i be the line perpendicular to ℓ'_i through c_i.

The analysis of Alon *et al.* shows that for each ℓ_i, there are at most $O(s \log s)$ disks that intersect ℓ_i and at least one other line ℓ_j [2, Section 2]. Thus, it suffices to focus on the disks in $\mathcal{D}$ that intersect at most one line ℓ_i. By simple counting, there is a line ℓ_i that exclusively intersects at most $m/s = m^{1-\delta}$ disks. It remains to find such a line in $O(m)$ time. For this, we compute the arrangement $\mathcal{A}$ of the strips with width 2 centered around each ℓ_i, together with an efficient point location structure. For each cell in the arrangement, we store whether it is covered by 0, 1, or more strips. Using standard techniques, the construction takes $O(s^2) = O(m^{2\delta})$ time. We locate for each triangle Δ_i the cells of $\mathcal{A}$ that contain the vertices of Δ_i. This needs $O(r \log s) = O(m^\gamma \log m)$ steps. Since every line intersects at most $O(\sqrt{r}) = O(m^{\gamma/2})$ triangles, we know that there are at most $O(sm^{\gamma/2}) = O(m^{\delta+\gamma/2})$ triangles that intersect a cell boundary of $\mathcal{A}$. We call these triangles the *bad* triangles.

For all other triangles Δ_i, we know that the associated point set Q_i lies completely in one cell of $\mathcal{A}$. Let $\mathcal{D}_i$ be the corresponding disks. By using the information stored with the cells, we can now determine for each disk $D \in \mathcal{D}_i$ in $O(1)$ time whether D intersects exactly one line ℓ_i. Thus, we can determine in total time $O(m)$ for each line ℓ_i the total number of disks that intersect only ℓ_i and whose center is not associated with a bad triangle. Let ℓ be the line for which this number is minimum.

In total, it has taken us $O(m \log m)$ steps to find ℓ. Let us bound the number of disks that intersect ℓ. First, we know that there are at most $O(m^{\delta+\gamma/2} \cdot m^{1-\gamma}) = O(m^{1+\delta-\gamma/2})$ disks whose centers lie in bad triangles. Then, there are at most $O(m^\delta \log m)$ disks that intersect ℓ and at least one other line. Finally, there are at most $m^{1-\delta}$ disks with a center in a good triangle that intersect only ℓ. Thus, if we choose, say, $\delta = 1/6$ and $\gamma = 2/3$, then ℓ crosses at most $O(m^{5/6})$ disks in $\mathcal{D}$. Furthermore, by construction, ℓ has at least $m/2 - O(m^{2/3})$ disk centers on each side. The result follows. $\square$

Remark. Actually, we can use the approach from Lemma 4.2 to compute an $(1/2+\varepsilon, 5/6+\varepsilon)$-SDT in total deterministic time $O(m \log m)$. The bottleneck lies in finding the simplicial partition for Q. All other steps take $O(m)$ time. However, when applying Lemma 4.2 recursively, we do not need to compute a simplicial partition from scratch. Instead, as in Matoušek's paper, we can recursively refine

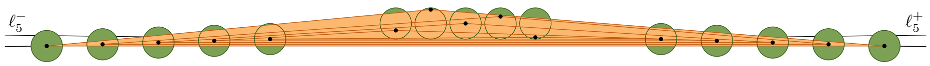


Fig. 4. The lower bound construction consists of $n/3$ unit disks centered on a horizontal line (5 in the figure), and two groups of $n/3$ points sufficiently far to the left and to the right of the disks. Distances not to scale.

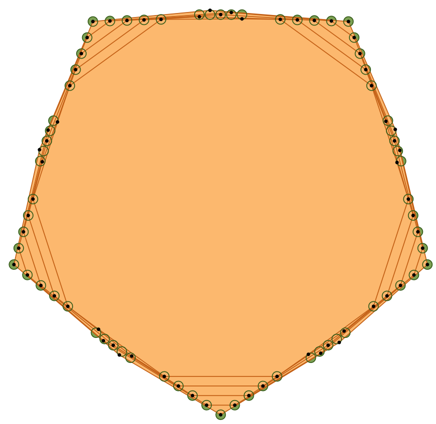


Fig. 5. n/k copies of the construction on a regular n/k-gon.

the existing partitions in linear time [18, Corollary 3.5] (while duplicating the triangles for the disks that are intersected by ℓ). Thus, after spending $O(m \log m)$ time on the simplicial partition for the root, we need only linear time per node to find the dividing lines, for a total of $O(m \log m)$, by Lemma 3.5.

5 Lower Bounds

We now show that our algorithm is optimal in the decision tree model. We begin with a lower bound of $\Omega(n \log n)$ for $k = \Omega(n)$. Let n be a multiple of 3, and consider the lines

$$\ell_n^- : y = -1/2 - 6/n - x/n^2; \quad \ell_n^+ : y = -1/2 - 6/n + x/n^2.$$

Let $\mathcal{D}_n$ consist of $n/3$ disks centered on the x-axis at x-coordinates between $-n/6$ and $n/6$; a group of $n/3$ disks centered on ℓ_n^- at x-coordinates between n^2 and $n^2 + n/3$; and a symmetric group of $n/3$ disks centered on ℓ_n^+ at x-coordinates between $-n^2 - n/3$ and $-n^2$. Figure 4 shows $\mathcal{D}_{15}$.

Lemma 5.1. *Let π be a permutation on $n/3$ elements. There is a sample P of $\mathcal{D}_n$ such that p_i (the point for the ith disk from the left in the main group) lies on layer $\pi(i)$ of $\circledcirc(P)$.*

Proof. Take P as the $n/3$ centers of the disks in $\mathcal{D}$ on ℓ_n^-, the $n/3$ centers of the disks in $\mathcal{D}$ on ℓ_n^+, and for each disk $D_i \in \mathcal{D}$ on the x-axis the point $p_i = (i - n/6, \pi(i) \cdot 3/n - 1/2)$. By construction, the outermost layer of $\circledcirc(P)$ contains at least the leftmost point on ℓ_n^+, the rightmost point on ℓ_n^-, and the highest point (with y-coordinate $1/2$). However, it does not contain any more points: the line segments connecting these three points have slope at most $2/n^2$. The second highest point lies $3/n$ lower, and at most $n/3$ further to the left or the right. The lemma follows by induction. □

There are $(n/3)! = 2^{\Theta(n \log n)}$ permutations π; so any corresponding decision tree has height $\Omega(n \log n)$. We can strengthen the lower bound to $\Omega(n \log k)$ by taking n/k copies of $\mathcal{D}_k$ and placing them on the sides of a regular (n/k)-gon, see Figure 5. By Lemma 5.1, we can choose independently for each side of the (n/k)-gon one of $(k/3)!$ permutations. The onion depth will be $k/3$, and the number of permutations is $((k/3)!)^{n/k} = 2^{\Theta(n \log k)}$.

Theorem 5.2. *Let $k \in \mathbb{N}$ and $n \geq k$. There is a set $\mathcal{D}$ of n disjoint unit disks in $\mathbb{R}^2$, such that any decision-based algorithm to compute $\circledcirc(P)$ for a sample P of $\mathcal{D}$, based only on prior knowledge of $\mathcal{D}$, takes $\Omega(n \log k)$ time in the worst case.*

The lower bound still applies if the input points come from an appropriate probability distribution (e.g., [1, Claim 2.2]). Thus, Yao's minimax principle [19, Chapter 2.2] yields a corresponding lower bound for any randomized algorithm.

6 Conclusion and Further Work

It would be interesting how much the parameter k can vary for a set of imprecise bounds and how to estimate k efficiently. Further work includes considering more general regions, such as overlapping disks, disks of different sizes, or fat regions. It would also be interesting to consider the problem in 3D. Three-dimensional onions are not well understood. The best general algorithm is due to Chan and needs $O(n \log^6 n)$ expected time [5], giving more room for improvement.

Acknowledgments. The authors would like to thank an anonymous reviewer for comments that improved the paper. M.L. supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123. W.M. supported in part by DFG project MU/3501/1.

References

[1] N. Ailon, B. Chazelle, K. L. Clarkson, D. Liu, W. Mulzer, and C. Seshadhri. Self-improving algorithms. *SIAM J. Comput.*, 40(2):350–375, 2011.

[2] N. Alon, M. Katchalski, and W. R. Pulleyblank. Cutting disjoint disks by straight lines. *Discrete Comput. Geom.*, 4(3):239–243, 1989.

[3] R. Bruce, M. Hoffmann, D. Krizanc, and R. Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005.

[4] K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Preprocessing imprecise points for Delaunay triangulation: simplified and extended. *Algorithmica*, 61(3):675–693, 2011.

[5] T. M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *J. ACM*, 57(3):Art. 16, 15 pp., 2010.

[6] B. Chazelle. On the convex layers of a planar set. *IEEE Trans. Inform. Theory*, 31(4):509–517, 1985.

[7] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76–90, 1985.

[8] O. Devillers. Delaunay triangulation of imprecise points: preprocess and actually get a fast query time. *J. Comput. Geom.*, 2(1):30–45, 2011.

[9] W. F. Eddy. Convex hull peeling. In *Proc. 5th Symp. Comp. Statistics (COMPSTAT)*, pages 42–47, 1982.

[10] E. Ezra and W. Mulzer. Convex hull of points lying on lines in $o(n \log n)$ time after preprocessing. *Comput. Geom.*, 46(4):417–434, 2013.

[11] P. G. Franciosa, C. Gaibisso, G. Gambosi, and M. Talamo. A convex hull algorithm for points with approximately known positions. *Internat. J. Comput. Geom. Appl.*, 4(2):153–163, 1994.

[12] M. Held and J. S. B. Mitchell. Triangulating input-constrained planar point sets. *Inform. Process. Lett.*, 109(1):54–56, 2008.

[13] M. Hoffmann, T. Erlebach, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *Proc. 25th Sympos. Theoret. Aspects Comput. Sci. (STACS)*, pages 277–288, 2008.

[14] P. J. Huber. Robust statistics: A review. *Ann. Math. Statist.*, 43:1041–1067, 1972.

[15] D. Kirkpatrick and J. Snoeyink. Computing common tangents without a separating line. In *Proc. 4th Workshop on Algorithms and Data Structures (WADS)*, pages 183–193, 1995.

[16] M. van Kreveld, M. Löffler, and J. S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. *SIAM J. Comput.*, 39(7):2990–3000, 2010.

[17] M. Löffler and J. Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Comput. Geom.*, 43(3):234–242, 2010.

[18] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8(3):315–334, 1992.

[19] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[20] F. Nielsen. Output-sensitive peeling of convex and maximal layers. *Inform. Process. Lett.*, 59:255–259, 1996.

[21] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23(2):166–204, 1981.

[22] T. Suk and J. Flusser. Convex layers: A new tool for recognition of projectively deformed point sets. In *Proc. 8th Int. Conf. Computer Analysis of Images and Patterns (CAIP)*, pages 454–461, 1999.

[23] K.-C. R. Tseng and D. G. Kirkpatrick. Input-thrifty extrema testing. In *Proc. 22nd Annu. Internat. Sympos. Algorithms Comput. (ISAAC)*, pages 554–563, 2011.