Preprocessing Disks for Convex Hulls, Revisited

Maarten Löffler ✉ 🆔
Department of Information and Computing Sciences; Utrecht University, the Netherlands

Benjamin Raichel ✉ 🏠 🆔
Department of Computer Science; University of Texas at Dallas, USA

—— **Abstract** ——————————————————————————————————

In the preprocessing framework one is given a set of regions that one is allowed to preprocess to create some auxiliary structure such that when a realization of these regions is given, consisting of one point per region, this auxiliary structure can be used to reconstruct some desired output geometric structure more efficiently than would have been possible without preprocessing. Prior work showed that a set of n unit disks of constant ply can be preprocessed in $O(n \log n)$ time such that the convex hull of any realization can be reconstructed in $O(n)$ time. (This prior work focused on triangulations and the convex hull was a byproduct.) In this work we show for the first time that we can reconstruct the convex hull in time proportional to the number of *unstable* disks, which may be sublinear, and that such a running time is the best possible. Here a disk is called *stable* if the combinatorial structure of the convex hull does not depend on the location of its realized point.

The main tool by which we achieve our results is by using a supersequence as the auxiliary structure constructed in the preprocessing phase, that is we output a supersequence of the disks such that the convex hull of any realization is a subsequence. One advantage of using a supersequence as the auxiliary structure is that it allows us to decouple the preprocessing phase from the reconstruction phase in a stronger sense than was possible in previous work, resulting in two separate algorithmic problems which may be independent interest. Finally, in the process of obtaining our results for convex hulls, we solve the corresponding problem of creating such supersequences for intervals in one dimension, yielding corresponding results for that case.
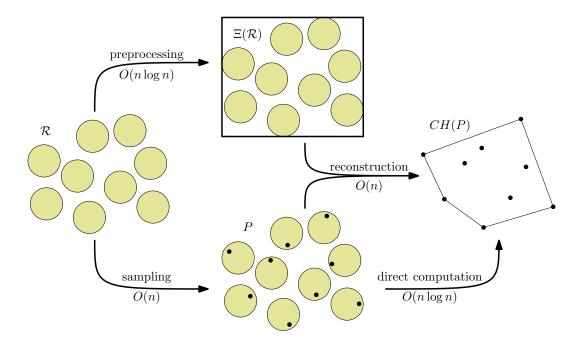
Contents

Figure 1 A set of disjoint unit disks $\mathcal{R}$ can be preprocessed into an auxiliery structure $\Xi(\mathcal{R})$ in $O(n \log n)$ time, such that the convex hull of a set of points P that respects $\mathcal{R}$ can be computed in linear time using $\Xi(\mathcal{R})$ (compared to $\Theta(n \log n)$ time without preprocessing) [21, 41, 6].

1 Introduction

Preprocessing framework.   The *preprocessing framework* for dealing with data uncertainty in geometric algorithms was initially proposed by Held and Mitchell [21]. In this framework, we have a set $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ of *regions*, often in $\mathbb{R}^2$, and a point set $P = \{p_1, p_2, \ldots, p_n\}$ with $p_i \in R_i$ (we also write $P \Subset \mathcal{R}$). This model has 2 consecutive phases: a preprocessing phase, followed by a reconstruction phase. In the preprocessing phase we have access only to $\mathcal{R}$ and we typically want to preprocess $\mathcal{R}$ in $O(n \log n)$ time to create some linear-size auxiliary data structure which we will denote by Ξ. In the reconstruction phase, we have access to P and we want to construct a desired output structure $S(P)$ on P using Ξ faster than would be possible otherwise (see Figure 1). Löffler and Snoeyink [27] were the first to use this model as a way to deal with data uncertainty: one may interpret the regions $\mathcal{R}$ as *imprecise* points, and the points in P as their true (initially unknown) locations. This interpretation of the preprocessing framework has since been successfully applied to various problems in computational geometry [7, 10, 17, 26, 41, 40].

Convex hull.   The *convex hull* of a set of points in $\mathbb{R}^2$ is the smallest convex set that contains all points. The algorithmic problem of computing the convex hull is arguably one of the most fundamental problems in computational geometry [2], and has been considered in an impressive range of different models for geometric uncertainty [19, 20, 30, 13, 22, 28, 16, 17, 24, 11, 3, 23]. The first result for convex hulls within the preprocessing framework can in fact be derived from the work by Held and Mitchell [21], who show that a set of disjoint unit disks can be preprocessed in $O(n \log n)$ time such that a triangulation can be reconstructed in linear time: since any triangulation contains at least the edges of the convex hull the
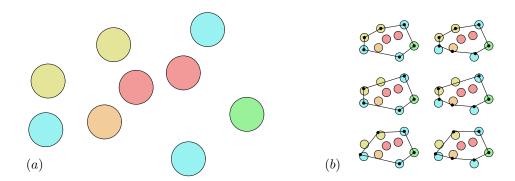
Figure 2 (a) A set of disjoint unit disks $\mathcal{R}$. (b) All possible combinatorial convex hulls. We identify five types of disks: (i) *stable impossible interior* disks that never contribute to the convex hull (red); (ii) *unstable potential interior* disks that may or may not contribute (orange); (ii) *unstable potential boundary* disks that may or may not contribute (yellow); (iii) *unstable guaranteed boundary* disks that are guaranteed to contribute, but for which the location may influence the structure of the hull (blue); and (iv) *stable guaranteed boundary* disks that are guaranteed to contribute, and the structure is independent of their location (green). (Refer to Section 2.1 for precise definitions).

convex hull can be extracted in the same amount of time (see Figure 1). Similarly, the work by van Kreveld *et al.* [41] implies that a set of disjoint disks of arbitrary radii (or a set of disjoint polygons of constant complexity) can also be preprocessed in $O(n \log n)$ time for the same purpose, and the work by Buchin *et al.* [6] implies that the same is true for a set of moderately overlapping unit disks. While these results focus on different problems (triangulation or Delaunay triangulation) and the convex hull is "only a by-product", Ezra and Mulzer [17] explicitly study the convex hull and show that any set of *lines* in the plane can also be preprocessed to speed up the reconstruction of the convex hull, although not to linear time: they achieve a reconstruction time of $O(n\alpha(n) \log^* n)$ in expectation.

Sublinear reconstruction. Traditionally, the aim in the preprocessing model has been to achieve reconstruction times that are faster than computing a solution from scratch, with the understanding that there is a natural lower bound of $\Omega(n)$ time to reconstruct $S(P)$, since even replacing each region of $\mathcal{R}$ with the corresponding point in P will take linear time. However, for problems in which the output is an ordered sequence of points, this reasoning is not entirely satisfactory, for two reasons:

1. it is possible that not all elements of P appear in $S(P)$; in this case it is not a priori clear that we need to spend time retrieving points that do not need to be output;

2. even for those points that do appear in $S(P)$, in some applications one might be happy knowing just the order and not the exact points; if this order is already clear from $\mathcal{R}$ we also do not necessarily need to spend even $|S(P)|$ time in the reconstruction phase.

Motivated by these observations, van der Hoog *et al.* [39, 40] have recently explored how the preprocessing model may be modified to allow for such sublinear reconstruction; for instance, for sorting they achieve reconstruction time proportional to the entropy of the interval graph, which can be less than n if many intervals are isolated. In order to support sublinear reconstruction, van der Hoog *et al.* introduce an additional phase to the preprocessing framework, see Section 4.2 for more details.
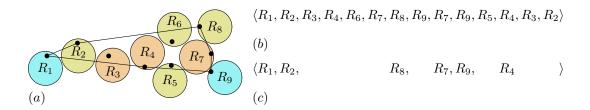
Figure 3 (a) A set of disjoint unit disks $\mathcal{R}$ and (b) a sequence of disks that is guaranteed to contain the vertices of the convex hull in the correct order. (c) A possible true convex hull.

1.1 Our contribution

Preprocessing disks for convex hulls. In this work, we investigate the problem of preprocessing a set of disks in the plane to reconstruct the convex hull directly for the first time. Doing so allows us to achieve features that are not possible when viewing the convex hull as a by-product of a triangulation, such as potential sub-linear reconstruction times. For some sets of disks we can already identify a subset of disks that is guaranteed to not take part in the convex hull, and a subset of disks that is guaranteed to take part in the convex hull (see Figure 2), and thus we need not spend time on them during the reconstruction phase.

Supersequences. As the main tool in our solution, we explore a new generic auxiliary structure: the *supersequence*. In principle, it is applicable to any computational problem where the output structure $S(P)$ is an ordered subset of P; that is, $S(P) = \langle p_{i_1}, p_{i_2}, \ldots, p_{i_s} \rangle$ for some indices $\{i_1, i_2, \ldots, i_s\} \subseteq [n]$. For such problems, an attractive option to use for the auxiliary structure Ξ is a *supersequence* of the regions corresponding to S; that is, Ξ is a sequence of (possibly reccuring) elements of $\mathcal{R}$ with the guarantee that, no matter where the true points P lie in their regions, the sequence of elements of P which we would obtain by replacing the regions in $\mathcal{R}$ by their points will always contain $S(P)$ as a subsequence. In the case of the convex hull $S(P)$ is the vertices of the convex hull in counter-clockwise order (the fact that this is a cyclic order slightly complicates things; in our solution we instead compute four *quarter hulls* separately) and $\Xi(P)$ is a sequence of disks (see Figure 3).

An advantage of using a supersequence as auxiliary structure is that it allows us to decouple the preprocessing phase from the reconstruction phase into two separate algorithmic problems of independent value, in a stronger sense than was possible in previous work. In *heriditary* algorithms, introduced by Chazelle and Mulzer [8], one is given a structure on a superset, and is interested in computing the structure on a subset. Heriditary algorithms have been used in the context of the preprocessing framework, for instance, van Kreveld *et al.* [41] in their reconstruction phase first produce a triangulation of red and blue points, where the red points are the points of interest, and then describe a linear-time hereditary algorithm for triangulations to obtain the final result. Our approach of using supersequences is different in that we do not require explicit construction of a structure on a superset; instead, in the reconstruction phase we can trivially replace the disks by their points and are left with a pure computational problem (Open Problem 14, in the case of convex hulls).

Preprocessing intervals for sorting. As a subroutine in our solution, we require to preprocess certain sets of moderately overlapping unit intervals into a supersequence such that when we get a single value per interval, the sorted order can be reconstructed quickly. We believe this result is also of independent interest; refer to Section 3 for more background on this
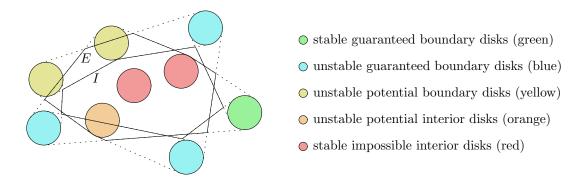
stable guaranteed boundary disks (green)

unstable guaranteed boundary disks (blue)

unstable potential boundary disks (yellow)

unstable potential interior disks (orange)

stable impossible interior disks (red)

Figure 4 The same example as Figure 2 with the regions I and E drawn in.

problem.

1.2 Organization

The remainder of this paper is organized as follows.

- In Section 2, we first establish the necessary vocabulary to be able to state our main results, which includes a classification of the input disks that is interesting in its own right. We then state our results for preprocessing unit disks for the convex hull, and discuss some of their limitations.

- Before proceeding with the proofs of our main results, in Section 3, we first switch to the one-dimensional problem of preprocessing a set of intervals for sorting, which is required as a subroutine later, but is also of independent interest. Our main result here is an algorithm to produce a supersequence of intervals that is guaranteed to contain any instance in sorted order.

- In Section 4, we then consider the corresponding problem of reconstructing the sorted order from such a sequence. In particular, we show that it is always possible to recover the sorted order from a *smooth* supersequence in linear time, but for general sequences this depends on the model of computation.

- Then, in Section 5 we return to the two-dimensional problem and discuss how to preprocess a set of disjoint unit disks as a supersequence of the convex hull. In Section 6 we extend this to overlapping unit disks using the one-dimensional result.

- In Section 7, we consider the corresponding reconstruction problem. Again, we show that it is always possible to recover the convex hull from a *smooth* supersequence. Whether it is possible for general supersequences is left as an open problem.

- Finally, in Section 8, we summarize our findings and discuss directions for future research.

2 Preliminaries, Results, and Limitations

In this section we formally define the problem and state our result for preprocessing unit disks of bounded ply in $\mathbb{R}^2$ for reconstructing the *convex hull*. Before we can state our results, we need to first discuss a *classification* of the input disks into five types, a notion of *quarter hull* to decompose the problem into smaller parts, and a notion of *smoothness*.

2.1   Disk Characterization

We may classify disks[1] in the plane in multiple ways with respect to the convex hull in the preprocessing framework. First, we classify the disks on whether their corresponding points will appear on the convex hull or not. We will define *impossible*, *potential*, and *guaranteed* disks.

▶ **Definition 1.** *For a set of disks $\mathcal{R}$ in the plane, we classify each disk $D \in \mathcal{R}$ as one of the following three types: D is either*

- impossible *when its realization is never a vertex of the convex hull of any realization of $\mathcal{R}$*
- guaranteed *when its realization is always a vertex of the convex hull of any realization of $\mathcal{R}$*
- potential *when it is neither impossible nor guaranteed.*

Prior work has considered the union or intersection of all possible realizations of $\mathcal{R}$ under different names (see for example [29, 33]), and these structures are closely related to our above classification of disks. In particular, the above definitions can be equivalently phrased in terms of inclusion or exclusion in certain regions; the following observations have been made before. Let I be the intersection of all halfplanes that intersect all disks. Let E be the intersection of all halfplanes that fully contain at least $n-1$ disks. See Figure 4.

▶ **Observation 2.** *A disk is impossible if and only if it is contained in I, and a disk is guaranteed if and only if it lies outside E.*

Second, we may also classify the disks according to whether the location of their corresponding point influences the combinatorial structure of the convex hull or not.

▶ **Definition 3.** *A disk $D \in \mathcal{R}$ is called* stable*, if for any fixed realization P' of $\mathcal{R}' = \mathcal{R} \setminus \{D\}$, the combinatorial structure[2] of the convex hull of $P' \cup \{p\}$ is the same for any realization p of D. A disk that is not stable is* unstable*.*

Note that stable disks are candidates for saving on reconstruction time, as they need not necessarily be inspected during the reconstruction phase to recover the convex hull. So, the best we may hope for is a reconstruction time proportional to the number of unstable disks.

Third, we classify disks depending on whether they appear on the convex hull of the disks themselves.

▶ **Definition 4.** *A disk $D \in \mathcal{R}$ is called a* boundary *disk if the boundary of $\mathrm{CH}(\mathcal{R})$ intersects D; otherwise D is called an* interior *disk.*

These 3 different classifiers for each disk, in principle give $3 \times 2 \times 2 = 12$ different types of disks. However, due to certain dependencies many of these types are not possible.

▶ **Lemma 5.** *We observe the following implications:*

- *every guaranteed disk is a boundary disk;*
- *every potential disk is unstable; and*
- *every impossible disk is both interior and stable.*

---

[1] Although we mostly consider unit disks in this work, the classification in this section applies to (potentially overlapping) disks of any radii.
[2] The ordered list of disks corresponding to the clockwise sorted order of the vertices of the convex hull.

Proof. First, observe that a guaranteed disk $D \in \mathcal{R}$ must be a boundary disk, as by Observation 2, D must lie outside E, or equivalently outside the convex hull of $\mathcal{R} \setminus \{D\}$.

Second, a potential disk $D \in \mathcal{R}$ must be unstable, as D being potential means there is both a realization of $\mathcal{R}$ where D is realized as a vertex of the convex hull and a realization of $\mathcal{R}$ where it is not a vertex of the convex hull, and thus the combinatorial structure depends on the realization of D.[3]

Finally, an impossible disk must be both interior and stable. Clearly it is stable as its realization is never a vertex of the convex hull and thus does not affect its combinatorial structure. It is interior as by Observation 2 it lies inside I which lies strictly inside the conex hull of $\mathcal{R}$.   ◄

The above observation implies there are only 5 possible types of disks are (see Figure 4).

▶ **Lemma 6.** *Given a set $\mathcal{R}$ of n disks, we can classify them in $O(n \log n)$ time.*

Proof of Lemma 6. First, the boundary disks can be easily detected in $O(n \log n)$ time by computing the convex hull of the disks [12]. Second, Observation 2 allows us to classify the disks into guaranteed, potential, and impossible disks using a simple $O(n \log n)$ algorithm [29]. By Lemma 5 all impossible interior disks are stable. It remains to distinguish the stable from the unstable guaranteed boundary disks.

Let $D \in \mathcal{R}$ be a stable guaranteed boundary disk. Let $B(\mathcal{R}) = \{B_1, \ldots, B_k\}$ be the ordered set of boundary disks. As D is a boundary disks, $D = B_i$ for some i, where B_i has two adjacent strips, namely $\mathrm{CH}(B_{i-1}, B_i)$ and $\mathrm{CH}(B_i, B_{i+1})$.

First, observe that no disk in $\mathcal{R} \setminus \{B_{i-1}, B_i\}$ can properly intersect the strip $\mathrm{CH}(B_{i-1}, B_i)$ (and the same holds for $\mathrm{CH}(B_i, B_{i+1})$). To see this, suppose there is at least one such disk, and that its realization in inside this strip. Now if B_{i-1} and B_i are realized at the points in the disks that are extreme in the direction orthogonal to the spine of this strip and exterior to the hull, then any realization of any other disk intersecting this strip cannot be a vertex of the convex hull. Conversely, if B_{i-1} and B_i are realized at the opposite extremes, then at least one of the other disks whose realization is in this strip will be a vertex of the convex hull.

By the same reasoning, B_{i-1} must be a guaranteed disk (and similarly for B_{i+1}). Specifically, if B_{i-1} is not guaranteed it intersects the convex hull of $\mathcal{R} \setminus \{B_{i-1}\}$, and because B_{i-1} is a boundary disk, it in fact intersects a strip of $\mathcal{R} \setminus \{B_{i-1}\}$. Moreover, this strip must have B_i as one of its defining disks as the strip $\mathrm{CH}(B_{i-1}, B_i)$ had no intersecting disks. Thus by the same reasoning used to argue the strip $\mathrm{CH}(B_{i-1}, B_i)$ had no intersecting disks, B_{i-1} cannot intersect this strip, and so must be guaranteed.

So in summary, in order for a guaranteed boundary disk to be stable, its adjacent strips must be empty and its neighbors on the strip hull must be guaranteed. Furthermore, it is clear that any guaranteed boundary disk with these properties is stable as then the realizations of B_{i-1}, B_i, B_{i+1} all appear on the convex hull as they are all guaranteed, and no vertex can occur between the realization of B_{i-1} and B_i (nor B_{i-1} and B_{i+1}), effectively blocking $D = B_i$ from affecting the combinatorial structure of the rest of the convex hull.

As for the running time for finding the stable guaranteed boundary disk, as discussed above we can already determine all boundary disks, and for each such disks whether it and its

---

[3] The definition of stable does not directly prohibit a scenario where for some fixed realization of $\mathcal{R} \setminus \{D\}$, any realization of D is always a vertex of the convex hull, while for some other fixed realization of $\mathcal{R} \setminus \{D\}$, any realization of D is never a vertex of the convex hull. However, such a scenario is impossible due to the continuity of our disk based uncertain regions.

neighbors are guaranteed, in $O(n \log n)$ time. Thus we only determine whether the adjacent strips are empty, though this can be done for all stips in $O(n \log n)$ time using Lemma 37. ◄

2.2 Quarter Hulls

Note that the convex hull is cyclic in nature: it has no well-defined start or end point, and in the presence of uncertainty, it may be difficult even to designate an arbitrary disk as start point, as there may not be any guaranteed disks.

To make our lives easier, we opt to focus on only one quarter of the convex hull: the part from the rightmost point on the hull to the topmost point on the hull (i.e. the positive quadrant). Indeed, the idea of splitting the computation of the convex hull into separate pieces is well established [5], and choosing quarters will have the added benefit that within such a quarter, no vertex can have an internal angle smaller than 90°, which will aid our geometric arguments later.

Formally, the *quarter hull* of a point set P is the sequence of points on the convex hull starting from the rightmost point in P and following the hull in counterclockwise order until ending with the topmost point in P.

We will define a supersequence for one quadrant; clearly, to obtain a supersequence for the entire hull, one can simply repeat this process four times for appropriately rotated copies of $\mathcal{R}$. Note that in subsequent sections, depending on the context, the term *convex hull* may refer to either the quarter hull or the full convex hull.

In the remainder of this work, we also adapt the notation established in Section 2.1 to the notion of quarter hulls. That is, a disk will be called *guaranteed* or *impossible* when it is guaranteed or impossible to contribute to the quarter hull, and it will be *stable* when it does not influence the combinatorial structure of the quarter hull.

Notably, observe that the total number of unstable disks for the entire convex hull is at most a constant number more than the sum of the number of unstable disks in the four quarter hull problems; hence, dividing the problem into quarters does not jeopardize our aim to achieve sublinear reconstruction times.

2.3 Supersequences

We are now ready to formally define supersequences.

▶ **Definition 7.** *Let $\mathcal{R}$ be a set of n regions in the plane. A* quarter-hull-supersequence *of $\mathcal{R}$ is a sequence Ξ of (possibly reoccurring) disks of $\mathcal{R}$ such that, for any point set $P \Subset \mathcal{R}$, the quarter hull of P is a subsequence of Ξ.*

We will focus on the case where $\mathcal{R}$ is a set of unit disks of bounded ply Δ in this work, where the *ply* of a set of regions is the maximum number of regions that any point in the plane is contained in. Replacing the disks in a quarter-hull-supersequence by their points, we now obtain a sequence of points that contains all points of the quarter hull of P in the correct order. Since it is not clear if the convex hull of any such sequence can be computed in linear time (Open Problem 14), we additionally introduce *smooth* sequences.

For two points $p, q \in \mathbb{R}^2$, let $\searrow (p, q)$ be the signed distance from p to q projected on a line with slope -1; that is,

$$\searrow (p, q) = (p.x - p.y) - (q.x - q.y).$$

▶ **Definition 8.** *A sequence X of points in $\mathbb{R}^2$ is (α, β)-smooth if:*

- *for any two elements p_i and p_j in X with $i < j$ we have that if p_i is on the quarter hull of X, then $\nwarrow (p_j, p_i) \leq \alpha$, and symmetrically if p_j is on the quarter hull of X, then $\nwarrow (p_i, p_j) \geq -\alpha$.*   *(distance property)*
- *for any disk D in the plane of diameter $d > 1$, there are only βd distinct points from the sequence in D.*   *(packing property)*

Essentially, a smooth supersequence may have points that are out of order, but the extent in which they can be out of order is limited by the parametesr α and β, which makes the reconstruction problem significantly easier.

We will also say a quarter-hull-supersequence Ξ is (α, β)-*smooth* if the corresponding sequence of points is (α, β)-*smooth* for any $P \Subset \mathcal{R}$.

2.4   Statement of results

We now state our main results for preprocessing unit disks in the plane for the convex hull. Our first three theorems imply that the classical result of $O(n \log n)$ preprocessing for $O(n)$ reconstruction (as in Figure 1) can indeed be replicated for unit disks of bounded ply using supersequences.

▶ **Theorem 9.** *Let $\mathcal{R}$ be a set of n unit disks in the plane of ply Δ. There exists a $(3\sqrt{2}, 12\Delta)$-smooth quarter-hull-supersequence Ξ of $\mathcal{R}$ of size $O(\Delta^2 n)$.*

▶ **Theorem 10.** *Let $\mathcal{R}$ be a set of n unit disks in the plane of ply Δ. A $(3\sqrt{2}, 12\Delta)$-smooth quarter-hull-supersequence Ξ of $\mathcal{R}$ of size $O(\Delta^2 n)$ can be computed in $O(\Delta n(\Delta + \log n))$ time.*

▶ **Theorem 11.** *Let $\mathcal{R}$ be a set of n unit disks. Given an (α, β)-smooth quarter-hull-supersequence Ξ of $\mathcal{R}$, and given a point set $P \Subset \mathcal{R}$, the quarter hull of P can be computed in $O(\alpha\beta|\Xi|)$ time.*

The next theorem implies that we can also obtain sublinear reconstruction using this method.

▶ **Theorem 12.** *Let $\mathcal{R}$ be a set of n unit disks. We are given an (α, β)-smooth quarter-hull-supersequence Ξ of $\mathcal{R}$ in which all its stable disks have been marked, and let μ denote the number of unstable disks in Ξ. Then given a point set $P \Subset \mathcal{R}$, the quarter hull of P can be computed in $O(\alpha\beta\mu)$ time.*

When $\mathcal{R}$ has ply Δ, Theorem 10 gives a $(3\sqrt{2}, 12\Delta)$-smooth quarter-hull-supersequence Ξ of size $O(\Delta^2 n)$. Thus we immediately have the following corollary of the theorems above.

▶ **Corollary 13.** *If $\mathcal{R}$ has ply Δ, then for the $(3\sqrt{2}, 12\Delta)$-smooth quarter-hull-supersequence of Theorem 10 with its stable disks marked, then the quarter hull of P can be computed in $O(\Delta^3 m)$ time, where m is the number of unstable disks in $\mathcal{R}$.*

For ease of exposition, in Section 5 we first prove Theorems 9 and 10 for the case $\Delta = 1$; i.e., disjoint unit disks.

In Section 3 we solve the problem of preprocessing a set of unit intervals of bounded ply in one dimension, which is needed to prove Theorems 9 and 10 in full generality for disks of bounded ply (i.e. $\Delta > 1$) in Section 6.

In Section 7 we prove Theorems 11 and 12, which are decoupled from the preprocessing phase and hold for any set of disks.

2.5   Limitations & open problems

Restrictions on the supersequences.   Our reconstruction algorithms require not only that the auxiliary structure is a supersequence of the desired output, but we also require that these sequences are *smooth*. This property is naturally fulfilled by our preprocessing algorithms and simplifies the reconstruction task. However, it is not clear whether this restriction is necessary for the reconstruction problem, and being able to reconstruct the convex hull from *any* supersequence would even further decouple the preprocessing and reconstruction phases.

▶ **Open Problem 14.** *We are given a sequence of points P in $\mathbb{R}^2$ (possibly with duplicates), and the guarantee that there exists a subsequence Q of P such that the convex hull of P equals the convex hull of Q, and the points in Q are sorted in counterclockwise order. Is it possible to compute the convex hull of P in $o(n \log n)$ time?*

Note that, although we require handling duplicates in our application, Problem 14 is also open even when all points are unique. We believe this question is of independent interest.

Model of computation.   We assume we are working in the real RAM model of computation, which is consistent with earlier work in our framework. (For a formal discussion of the Real RAM, refer to [15].) In order to achieve linear reconstruction time, we require our supersequences to be smooth, however, in Section 4 we do show that reconstruction in $\mathbb{R}^1$ can be done in linear time, without this smooth restriction, in the word RAM model, where universal hashing is allowed.

General position.   For ease of exposition, throughout we assume *general position* of the disk centers in the preprocessing phase and general position of the true points in the reconstruction phase; in both cases for us general position means no three points (or centers) are collinear. However, because we actively duplicate regions and points in our sequences, we will not be able to assume full general position in all subroutines (e.g. the discussion in Section 4 hinges critically on the potential existence of duplicates). Our general position assumptions may be completely lifted by using symbolic perturbation [14].

3   Preprocessing Intervals for Sorting in One Dimension

In this section we consider the problem of preprocessing unit intervals of bounded ply in $\mathbb{R}^1$ such that the *sorted order* of a corresponding set of points (values) can be recovered quickly. This will be later used as a subroutine in Section 6. We remark that the problem of optimally preprocessing intervals for sorting was considered in prior work (see for example [39] and reference therein). Here though we require that our output structure is a supersequence, as this is what will be required to use it as a subroutine for our results in $\mathbb{R}^2$.

▶ **Definition 15.** *Let $\mathcal{R}$ be a set of n intervals. A* sorting-supersequence *of $\mathcal{R}$ is a sequence Ξ of (possibly reoccurring) intervals of $\mathcal{R}$ such that, for any point set $P \Subset \mathcal{R}$, the sorted order of P is a subsequence of Ξ.*

While this definition works for any set of intervals, we will focus on unit intervals of ply Δ in this work. Furthermore, we will consider an additional restriction on the sequences, and we will show that such a restricted sequence can both be computed on a set of unit intervals of ply Δ, and used to recover the sorted order; whether this restriction is necessary is left as an open problem (and may depend on the model of computation; refer to Section 4).

▶ **Definition 16.** *A sequence of values $X = x_1, \ldots, x_n$ is (α, β)-smooth if:*

- *for any two elements $x_i, x_j \in X$ such that $i < j$ but $x_i > x_j$, we have $x_i - x_j \leq \alpha$, and (distance property)*
- *for any interval $J \subset \mathbb{R}$ with $|J| > 1$, there are $\leq \beta|J|$ distinct elements from X in J. (packing property)*

We will also say a sorting-supersequence Ξ is (α, β)-*smooth* if the corresponding sequence of points (values) is (α, β)-*smooth* for any $P \Subset \mathcal{R}$.

The goal of this section is to prove the following theorems.

▶ **Theorem 17.** *Let $\mathcal{R}$ be a set of n unit intervals of ply Δ. There exists a $(3, 2\Delta)$-smooth sorting-supersequence Ξ of $\mathcal{R}$ of size $O(n\Delta)$. Furthermore, such a sequence can be computed in $O(n(\Delta + \log n))$ time.*

▶ **Theorem 18.** *Let $\mathcal{R}$ be a set of n unit intervals. Given an (α, β)-smooth sorting-supersequence Ξ of $\mathcal{R}$, and given a point set $P \Subset \mathcal{R}$, the sorted order of P can be computed in $O(|\Xi| \log(\alpha\beta))$ time.*

When $\mathcal{R}$ has ply Δ, Theorem 17 gives a $(3, 2\Delta)$-smooth sorting-supersequence Ξ of size $O(n\Delta)$. Thus we immediately have the following corollary of Theorem 18.

▶ **Corollary 19.** *If $\mathcal{R}$ has ply Δ, then for the $(3, 2\Delta)$-smooth sorting-supersequence of Theorem 17, the sorted order of P can be computed in $O(n\Delta \log(\Delta))$ time.*

3.1 Preliminaries

Supersequences and sorting.

A *universal word* is a string that contains as subsequences all permutations of its distinct characters. The question of constructing the shortest universal word was first posed by Knuth and attributed to Karp [9]. The length of known shortest universal words are in [1]. The best known upper bound is due to Tan [37]. An earlier bound of $n^2 - 7/3n + 19/3$ was given by Radomirovic [31]. His proof is constructive and also gives an $O(n^2)$ algorithm to construct such a word. Uznański shows that testing whether a given string is a universal word is coNP-complete [38].

The problem of finding a sorted sequence in a supersequence is closely related to the *longest increasing subsequence* problem. Fredman shows that the problem has a $\Omega(n \log n)$ lower bound [18]. There is also a matching $O(n \log n)$ dynamic programming algorithm, which is commonly taught in advanced algorithms courses.

3.2 Preprocessing algorithm

We will use the following result.

▶ **Lemma 20** ([32]). *The length of the shortest sequence on m symbols which contains all permutations of $[1, m]$ as a subsequence is upper-bounded by m^2. Moreover, such a sequence of length m^2 can be computed in $O(m^2)$ time.*

While Radomirocić [32] does not discuss the running time, it is implicit in their construction. Ultimately we will apply it when m is the ply, which we assume is bounded. Using this result, we prove the following technical lemma, which we will also reuse later in Section 2.
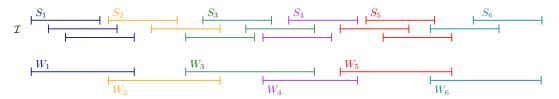
Figure 5 An example of a sequence of unit intervals of ply $\Delta = 3$.

▶ **Lemma 21.** *Let $\mathcal{I}$ be a set of n unit intervals in $\mathbb{R}$ of ply Δ. There is a $(3, 2\Delta)$-smooth sorting-supersequence Ξ of $\mathcal{I}$ with at most $4\Delta n$ intervals.*

Note, to get a sequence of length $n\Delta^2$, we could simply apply Lemma 20 to every cell of the subdivision induced by I. However, Lemma 21 gives a more refined bound.

Proof. Assume the intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ are ordered and indexed from left to right (since they are unit intervals, this order is well-defined). Now, let $S_1 = \{I_i \mid I_i \cap I_1 \neq \emptyset\}$ be the set of all intervals that intersect the leftmost interval (including the leftmost interval itself) and let $\eta_1 = \min\{i \mid I_i \cap I_1 = \emptyset\}$ be the index of the first interval not in S_1. We recursively define S_{i+1} to be the set of all intervals that intersect I_{η_i} and are not in S_i. Then, let $W_i = \bigcup_{I \in S_i} I$ be the window from the left endpoint of the leftmost interval in S_i to the right endpoint of the rightmost interval in S_i.

Now, define S_i' to be the set of all intervals from S_i unioned with all intervals from S_{i+1} which intersect W_i. We claim that $|S_i'| \leq 2|S_i| - 1 \leq 2\Delta - 1$. This is because any interval from S_{i+1} intersecting W_i, that is not already in S_i, must contain the right endpoint of W_i, and as Δ is the ply, there can therefore be at most $\Delta - 1$ such intervals. We further claim that every pair of intervals that intersect each other is present in some set S_i': indeed, an interval from S_i cannot intersect an interval from S_{i+2} since then the interval in S_{i+2} would also intersect the leftmost interval of S_{i+1} (but then it would be part of S_{i+1}).

Let k be the resulting number of S_i' sets. We process these sets of intervals from left to right, and apply Lemma 20 to S_i' for $i = 1, \ldots, k$. Let the resulting ith sequence be Ξ_i. Then we simply concatenate the resulting sequences, to get a sequence $\Xi = \Xi_1 \cup \Xi_2 \cup \cdots \cup \Xi_k$.

As argued above, $|S_i'| \leq 2\Delta$ and moreover $\sum_{i=1}^k |S_i'| \leq 2\sum_{i=1}^k |S_i| = 2n$. By Lemma 20, $|\Xi_i| = |S_i'|^2$. Thus $|\Xi|$ is bounded by a sum of squares of values, i.e. the $|S_i'|$, which are each bounded by 2Δ and sum to at most $2n$. It is not hard to see that such a sum of squares is maximized when $S_i' = 2\Delta$ for all i and $k = n/\Delta$. Thus the length of Ξ is at most $n/\Delta \cdot 4\Delta^2 = 4n\Delta$.

Now, let $P \Subset \mathcal{I}$ be any realization, listed in sorted left to right order on the real line. We divide the points in P into k groups: let $P_i \subset P$ be the set of all points that are contained in W_i but not in W_{i-1}. Let $\Pi = \Pi(P)$ be the ordered string of intervals corresponding to the ordered points of P, and let Π_i be the part of Π corresponding to P_i. The string Π_i is a substring of Ξ_i (by Lemma 20). Therefore, the points in P_i can be charged to the subsequence Ξ_i, thus proving Ξ is a sorting-supersequence, as the Ξ_i are concatenated in order.

Finally, we need to argue that the resulting sequence Ξ is $(3, 2\Delta)$-smooth. To argue that $\alpha = 3$, note that all intervals in S_i intersect each other, and thus all intervals in S_i' either intersect each other or intersect a common interval. In the first case, the distance between any two points in them is at most 2; in the latter case they could have points at distance 3 from each other. Moreover, the S_i' are sorted on the real line (say by the left ends of their leftmost interval), and since we concatenate the Ξ_i in this order, this bound for α on each Ξ_i

holds for Ξ as well. To argue that $\beta = 2\Delta$, note that the largest number of unit intervals that can intersect an interval of length J is $(|J| + 1) \cdot \Delta$, where $(|J| + 1) \cdot \Delta \leq 2\Delta|J|$ when $|J| > 1$. ◀

We are now ready to finish the preprocessing algorithm.

Proof of Theorem 17. Observe that Lemma 21 establishes the existence of the desired sorting-supersequence. Moreover, the proof of Lemma 21 is constructive. First, sort the n unit intervals in $O(n \log n)$ time, yielding the S_i sets. S_i' can then be computed (using this sorted ordering) from S_i in $O(\Delta)$ time, and thus $O(\Delta n)$ time over all i. Finally, Lemma 20 states that the time to compute Ξ_i from S_i' is proportional to $|\Xi_i|$ (which itself is $|S_i'|^2$). Thus Ξ is computed from the S_i' in $O(\Delta n)$ time overall, as it is the concatenation of the Ξ_i, and Lemma 21 already argued $|\Xi| \leq 4\Delta n$. ◀

4 Reconstruction in One Dimension

We now consider the problem of recovering the sorted order of a set of values from a supersequence as computed in Section 3. We will first consider the problem in the classical preprocessing framework, where the goal is to achieve linear-time reconstruction, in Section 4.1, and show that for smooth sequences this is achievable in the Real RAM, while on the Word RAM we do not require the sequences to be smooth. Then, in Section 4.2, we discuss how to combine the idea of supersequences with the aim of achieving sublinear reconstruction.

4.1 Standard Reconstruction

For the reconstruction phase, we are now given a sorting-supersequence Ξ of $\mathcal{R}$, and additionally a set of points (values) $P \subseteq \mathcal{R}$, and the goal is to efficiently compute the sorted order of P using Ξ. (In particular, in linear time when Ξ is $(O(1), O(1))$-smooth.)

We will assume that the regions in Ξ come as pointers to the original regions in $\mathcal{R}$, and that our sets $\mathcal{R}$ and P are equipped with bidirectional pointers.

The natural approach to the reconstruction problem is to replace each region in Ξ by its corresponding point in P. This results now in a sequence X of numbers, which comes with a guarantee that the sorted order is a subsequence of X.

▶ **Definition 22.** *Let $X = x_1, \ldots, x_n$ be a sequence of n numbers (possibly with duplicate numbers), which contains $k \leq n$ unique numbers. Then we call X a* hidden sorted *sequence if it contains a subsequence of length k with all k distinct numbers in sorted order.*

First, we show that given such a sequence X, we can find the sorted subsequence in linear time when universal hashing is allowed. Subsequently, we show even without universal hashing linear time is possible when the sequence is appropriately smooth. It is an open problem whether linear time is possible without universal hashing nor a smoothness bound.

Sorting hidden sorted sequences in linear time in the word RAM model.

We first show that there is a natural greedy algorithm to sort hidden sorted sequences, which runs in expected linear time assuming universal hashing.

▶ **Lemma 23.** *Given a hidden sorted sequence $X = x_1, \ldots, x_n$ with k unique numbers, there is a greedy algorithm which outputs a sorted subsequence of length k containing all k distinct numbers. This algorithm runs in $O(n)$ expected time if universal hashing is allowed, and otherwise it runs in $O(n \log k)$ time.*

Proof. Create a stack which initially contains just x_1. Now process the items in the sequence in order (starting at x_2). Let x_i be the current item being processed. If x_i exists in the stack already then discard it. Otherwise, pop items off the stack until the top item is less than x_i, at which point we push x_i onto the stack.[4]

Let $x_{i_1}, \ldots, x_{i_k}$ be any subsequence of X containing all k unique items in sorted order. To argue correctness, we prove by induction that after x_{i_j} has been processed by our algorithm, the bottom of the stack contains the j smallest (unique) items in sorted order. (For $j < k$ there may be additional items above on the stack, but for $j = k$ this is not possible as by construction the stack contains sorted unique items.) For the base case when $j = 1$, the claim trivially holds as then the smallest item has been processed (either when x_{i_j} is processed or earlier), at which point the algorithm will place it at the bottom of the stack, and the algorithm will never remove it. Now consider some $j > 1$. By induction, after $x_{i_{j-1}}$ was processed, the stack contains the $j - 1$ smallest (unique) items in sorted order at the bottom, and these can never be removed by the algorithm. Thus as x_{i_j} is the next smallest item, the same logic as in the base case implies that after x_{i_j} is processed, it will be on the stack immediately after the first $j - 1$ items, and will never be removed.

Assuming we store the current stack items in a hash table, it takes expected constant time to check if x_i is already in the stack. Thus we get a linear time algorithm, since in each time step we are either inserting or deleting from the stack, and each item is inserted at most once. If hashing is not allowed, we can instead use a binary tree for the items in the stack (whose size is at most k), and thus the running time is $O(n \log k)$.   ◀

Note that the algorithm runs in linear time when hashing is available, but as we are interested in working in the Real RAM model, hashing is not available to us. We note that the problem of finding the sorted subsequence in a hidden sorted sequence is a special case of the longest increasing subsequnce problem, which has a $\Omega(n \log n)$ lower bound in the decision tree model; however, in our problem if their are no duplicates the sequence is already sorted, whereas this is not the case for LIS, and thus it is unclear whether a lower bound for LIS implies a lowerbound for our problem. Whether a hidden sorted sequence can be recovered in linear time in the Real RAM model is left as an open problem.

Sorting smooth sequences

For our purposes, it is sufficient to recover the sorted subsequence from a hidden sorted sequence that additionally is (α, β)-smooth. We assume α and β are constants, so the goal is to sort the sequence in time of the form $O(f(\alpha, \beta) \cdot n)$.

Note that if we convert the supersequence Ξ as constructed in Section 3 to their corresponding values in P, we get a sequence that is $(3, 2\Delta)$-smooth.

▶ **Lemma 24.** *Given a (α, β)-smooth hidden sorted sequence $X = x_1, \ldots, x_n$ with k unique numbers we can output a sorted subsequence of length k containing all k distinct numbers in $O(n \log(\alpha\beta))$ time in the Real RAM model.*

Proof. Recall the algorithm of Lemma 23, which greedily processes the items in X in order, discarding x_i if it has already occurred, and otherwise popping items off the stack until the top item is less than x_i, at which point we push x_i onto the stack. We already argued this algorithm is correct, though its running time depends on how long it takes to determine whether x_i has already occurred.

---

[4] Note that the above algorithm is oblivious of the value k.

The claim is that if x_i has already occurred, then is must be one of the largest $\alpha\beta$ items seen so far. Assuming this claim is true, we can implement this algorithm in $O(n \log(\alpha\beta))$ time, as maintaining a balanced binary tree with the largest $\alpha\beta$ items seen, takes $O(\log(\alpha\beta))$ time per round (i.e. $O(n \log(\alpha\beta))$ overall), and the remaining stack operations are linear time overall.

To prove the claim, fix any x_i, and let x_j be the largest value item such that $j < i$. If $x_i > x_j$, the x_i has not been seen before, and the claim trivially holds. So suppose that $x_i \leq x_j$. Since X is (α, β)-smooth, we have that $x_j - x_i \leq \alpha$. Thus if x_i has occurred already, it must be within an interval of length α (ending at x_j), and there can be at most $\alpha\beta$ distinct points in such an interval, by the packing property of (α, β)-smooth sequences, thus implying the claim. ◀

Proof of Theorem 18. If Ξ is an (α, β)-*smooth* sorting-supersequence of $\mathcal{R}$, then by definition for any $P \Subset \mathcal{R}$, if we replace the regions from Ξ by their corresponding points in P, then it results in an (α, β)-smooth hidden sorted sequence. Thus Lemma 24, directly states we can recover the sorted order of P in $O(|\Xi| \log(\alpha\beta))$ time. ◀

4.2 Sublinear reconstruction

The preprocessing framework was originally developed with linear-time reconstruction as the ultimate goal: since, during the reconstruction phase, as we need to spend linear time to replace each region in $\mathcal{R}$ by the corresponding point in P, we cannot hope to be faster. Van der Hoog *et al.* [39, 40] argue that this viewpoint may be limiting, and that in some applications we may be content with an output structure that still contains some regions, as long as we are guaranteed that the true points in those regions are combinatorially in the correct location. They argue that the preprocessing framework may be adapted to allow for this behaviour, at the cost of introducing another phase: the reconstruction phase is formally separated into a first subphase (that can take sublinear time), in which the auxiliary structure Ξ is transformed into another structure Ξ' which is combinatorially equivalent to the desired output $S(P)$, and a second subphase in which Ξ' is actually transformed into $S(P)$ in linear time, if so desired.

We show that the same behavior can be quite naturally obtained when the auxiliary structure Ξ is a supersequence.

▶ **Definition 25.** *A* marked *sorting-supersequence Ξ of $\mathcal{R}$ is a sequence of (possibly recurring) intervals of $\mathcal{R}$, some of which may be marked, and such that, for any point set $P \Subset \mathcal{R}$, the sorted order of P is a subsequence of Ξ, that contains all marked items of Ξ.*

A normal sorting-supersequence is a marked supersequence with no items marked. To achieve maximum benefit, we want to mark as many items as possible, since these may be skipped during reconstruction.

In the case of sorting, intervals are marked if the position in the sorted order does not depend on the location of the point in the interval. This happens exactly when the interval is disjoint from all other intervals.

We adapt the preprocessing algorithm as follows. In $O(n \log n)$ time, we sort the intervals (by their left endpoint) as before. Now, in linear time, we scan through the intervals and detect any intervals that do not overlap with any other interval. We mark these intervals, and collect clusters of remaining intervals that are separated by the marked intervals. We apply Lemma 21 to each cluster separately, and concatenate the resulting sequences.
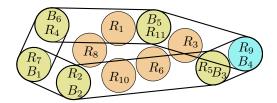
Figure 6 The convex hull of $\mathcal{R} = \{R_1, \ldots, R_{11}\}$, with *yellow* and *blue* disks on the boundary (depending on whether potential or guaranteed, see Figure 2), *orange* internal disks, and *strips* consisting of two consecutive boundary disks. In this case $\mathcal{B}(\mathcal{R}) = \{B_1, \ldots, B_6\} = \{R_7, R_2, R_5, R_9, R_{11}, R_4\}$.

In order to be able to quickly skip over marked intervals during the reconstruction phase, we can equip the resulting sorting-supersequence with pointers from each item to the next non-marked item. During the reconstruction phase we then also apply Lemma 24 separately to each subsequence of consecutive non-marked items, in time propertional to the total number of non-marked items in the list. This results in a mixed sequence of sorted points and marked intervals, which is guaranteed to be in the correct sorted order (this list would be called Ξ' in [39]).

5 Preprocessing Disjoint Unit Disks for the convex hull

Our goal in this section is to prove Theorems 9 and 10 for the case of disjoint unit disks.

5.1 Definitions and notation

Throughout this section, let $\mathcal{R} = \{R_1, \ldots, R_n\}$ be a set of n disjoint unit disks in the plane, arbitrarily indexed. Let $\mathcal{B}(\mathcal{R}) = \{B_1, \ldots, B_k\}$ be the subset of disks from $\mathcal{R}$ on the boundary of the convex hull of $\mathcal{R}$, indexed in counterclockwise order of appearance on the convex hull. We refer to $\mathcal{B}(\mathcal{R})$ as the boundary disks, and disks in the set $\mathcal{I}(\mathcal{R}) = \mathcal{R} \setminus \mathcal{B}(\mathcal{R})$ as interior disks. Refer to Figure 6. When $\mathcal{R}$ is clear from the context, we may write simply $\mathcal{B}$ and $\mathcal{I}$ instead of $\mathcal{B}(\mathcal{R})$ and $\mathcal{I}(\mathcal{R})$.

For any $i \in [1, k]$, define the ith *strip* as $s_i = \mathrm{CH}(B_i, B_{i+1})$, where CH denotes the convex hull (and $s_k = \mathrm{CH}(B_k, B_1)$). We refer to the chain of strips, ordered by increasing i, as the *strip hull* of $\mathcal{R}$. (Indeed the convex hull of all strips is equal to that of all regions, where straight segments on the boundary follow individual strips and turns occur at the intersection of the common disk with the next strip.) Furthermore, for any i, we call the line segment connecting the centers of B_i and B_{i+1} the *spine* of the corresponding strip. Observe that the spines themselves define an inner convex polygon, which we call the *spine hull*.

For a point set $P \Subset \mathcal{R}$, we refer to the vertices $Q = \{q_1, \ldots q_k\}$ of P's convex hull, as the *Q-hull*.

▶ **Observation 26.** *Every vertex of the Q-hull is contained in at least one strip.*

Since our goal is to recover the *quarter hull*, we extend the above hull definitions to quarter hulls. The *quarter Q-hull* is simply the quarter hull of Q. The *quarter spine hull* is the quarter hull of the centers of the disks in $\mathcal{B}(\mathcal{R})$. Finally, the *quarter strip hull* is the subchain of the strip hull defined by the disks whose centers are on the quarter spine hull, with the addition of two new unbounded *half-strips*, one which goes downward from the
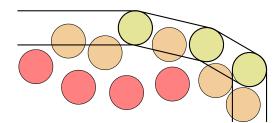
■ **Figure 7** The quarter strip hull of $\mathcal{R}$, with *yellow* disks on the boundary, *orange* and *red* internal disks (depending on whether unstable potential or stable impossible, see Figure 2), and *strips* consisting of two consecutive boundary disks; there are two infinite *half-strips* from the rightmost disk and the topmost disk.

rightmost disk, and the other which goes leftward from the topmost disk.[5] See Figure 7.

5.2 Geometric preliminaries

In this section we prove various geometric properties that will be required for the proof of Theorem 9.

▶ **Lemma 27.** *Let P be a convex polygon, with edges of length at least $x > 0$. Let D be a unit radius disk. Then D can intersect at most $\lfloor 2\pi/x \rfloor + 1$ edges of P.*

Proof of Lemma 27. Let D' be a disk of radius $1 + x$ concentric with D. Observe that for every edge e of P that intersects D, the intersection of e with D' must have length at least x.

We now argue that the total length of P that can be contained in D' is at most $2\pi(1+x)$. More generally, let γ be any convex closed curve, bounding a closed convex body Γ. As D' is convex, the intersection $\Gamma \cap D'$ is also convex, and thus the maximum length of the boundary of $\Gamma \cap D'$ is the length of the circumference of D', namely $2\pi(1+x)$. Thus as $\gamma \cap D'$ is a subset of the boundary of $\Gamma \cap D'$, its maximum possible length is $2\pi(1+x)$.

Combining our upper bound on the length of P contained in D' with our lower bound on the length of edges in D' that intersect D, we can conclude that the number of edges of P that D intersects is at most $\lfloor 2\pi(1+x)/x \rfloor \leq \lfloor 2\pi/x \rfloor + 1$. ◀

▶ **Corollary 28.** *Every unit disk intersects at most 7 strips.*

Proof of Corollary 28. A unit radius disk D intersects a strip if and only if a radius 2 disk D' with the same center point intersects the spine of the strip. Now consider the convex polygon P which is the spine hull of the strips. Observe that the edges of P have length at least 2, as the defining disks of a strip are disjoint. Now scale both P and D' by a factor $1/2$, and now the edges of P have length at least 1 and D' is a unit radius disk. So now we can apply Lemma 27 where $x = 1$, implying that at most 7 edges of P are intersected by D', which by the above implies that D intersected at most 7 strips. ◀

▶ **Corollary 29.** *The arrangement of strips has ply at most 4.*

Proof of Corollary 29. Consider any point z in the plane. z is contained in a strip a strip if and only if a unit disk D centered at z intersects the spine of that strip. So consider the

---

[5] Note that the rightmost and topmost extremal disks do not necessarily contribute the rightmost and topmost points in P.
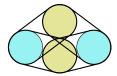
Figure 8 An example of four disjoint unit disks where the arrangement of strips has ply 4.

convex polygon P which is the spine hull of the strips. Observe that the edges of P have length at least 2, as the defining disks of a strip are disjoint. Thus applying Lemma 27 with $x = 2$, we know that D intersects at most 4 edges of P, and therefore z intersects at most 4 strips. The ply of the arrangement is the maximum of all points in the plane, and since z was an arbitrary point, the ply is at most 4.   ◀

We observe that Corollary 29 is in fact tight, as seen by Figure 8.

5.2.1 Fréchet and Hausdorff

Here we define and discuss basic properties of the Fréchet and Hausdorff distance measures.

Roughly speaking, the Hausdorff distance between two point sets is the distance of the furthest point from its nearest neighbor in the other set.

▶ **Definition 30.** *Given two point sets $P, Q \subseteq \mathbb{R}^2$, their Hausdorff distance is*

$$\delta_H(P, Q) = \max\{\sup_{p \in P} d(p, Q), \sup_{q \in Q} d(q, P)\}$$

We use the following definition of Fréchet distance from [25, Definition 3.10], which applies both to the unit interval $\mathbb{S}^0$, and well as the unit circle $\mathbb{S}^1$.

▶ **Definition 31** (Fréchet distance). *Let $P : \mathbb{S}^k \to \mathbb{R}^2$ and $Q : \mathbb{S}^k \to \mathbb{R}^2$ with $k \in \{0, 1\}$ be (closed) curves. Then $\delta_F(P, Q)$ denotes the Fréchet distance between P and Q, defined as*

$$\delta_F(P, Q) := \inf_{\alpha:\mathbb{S}^k \to \mathbb{S}^k, \beta:\mathbb{S}^k \to \mathbb{S}^k} \max_{t \in \mathbb{S}^k} ||P(\alpha(t)) - Q(\beta(t))||,$$

where α and β range over all continuous and increasing bijections on $\mathbb{S}^k$.

[25, Theorem 5.1] discusses the equivalence between Hausdorff distance and Fréchet distance for closed convex curves, originally argued in [4].

▶ **Theorem 32** ([4]). *For two convex closed curves in the plane, the Hausdorff distance is the same as the Fréchet distance.*

We will require the following lemma, which is a standard observation, though for completeness we include a short proof. Note that in the following lemma, a convex polygon refers only to the boundary and not the interior of the enclosed region.

▶ **Lemma 33.** *Given two convex polygons A and B, where every vertex of A is within distance ε of B and every vertex of B is within distance ε of A, then the Hausdorff distance between A and B is at most ε.*

Proof. Let p be any point on A. We wish to argue that p is within distance ε of some point on B. (A symmetric argument can then be applied for any point on B). If p is a vertex of A then the property holds by the lemma statement. Otherwise p is an interior point of some segment with endpoints s and t, and let ℓ be the supporting line of this segment.
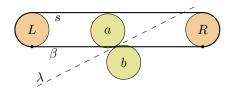
Figure 9 a and b can only both appear on the upper hull in one order.

Let D be the ε radius ball centered at p. Assume for contradiction that D does not intersect B. Then there are two cases. In the first case B contains D in its interior. This implies B has at least one vertex farther than ε from ℓ on both sides of ℓ. By convexity A lies entirely on one side of ℓ, and thus this in turn implies one of these vertices from B is farther than ε from A, which is a contradiction with the lemma statement.

In the second case B is separated from D by some line m. Thus p is farther than ε from m, and as p lies on the segment st, at least one of the vertices s or t is also separated by m from B and lies farther than ε from m. However, in this case this vertex is also farther than ε away from B, which again is a contradiction with the lemma statement. ◄

5.3 Proof of Theorem 9 (Existence) for Disjoint Disks

In this section, we argue that a sequence Ξ, as described in Theorem 9, exists. We first argue that we can collect the disks that intersect a single strip.

Consider any strip s spanned by two disks L and R, and assume w.l.o.g. that the strip is horizontal and is at the top of the strip hull; let β be the bottom edge of the strip (the segment connecting the bottommost point of L to the bottommost point of R). (See Figure 9.) Now consider the set $\mathcal{R}_s \subset \mathcal{R} \setminus \{L, R\}$ of other disks that intersect s. Clearly, all disks in $\mathcal{R}_s$ must intersect β. But since they are disjoint, they intersect β in a well-defined order.

▶ **Lemma 34.** *A portion of the convex hull that intersects L and R may only contain vertices from $\mathcal{R}_s$ in the order in which they intersect β.*

Proof. Consider any two disks a and b in $\mathcal{R}_s$ with a left of b in their intersection order with β. Let λ be the bisector of a and b. Now assume that a is higher than b (the other case is symmetric). Then λ has positive slope, and it intersects β (since it passes between a and b), therefore L lies completely above λ. But then we can never have a convex curve that passes through L hit first b and then a. So, if a and b both appear on the hull, a comes before b. ◄

Lemma 34 implies that for a single strip s, we can simply put the disks in $\mathcal{R}_s$ in this order to form a subsequence Ξ_s.

Next, we argue that we can simply concatenate the sequences per strip to obtain a single global sequence.

▶ **Lemma 35.** *Let $P \subseteq \mathcal{R}$ be a point set, and let $Q = \{q_1, \dots, q_m\}$ be its convex hull. Let k be the number of vertices on the spine hull of $\mathcal{R}$. There exist a mapping $f : [1 : m] \to [1 : k]$ such that:*

- *q_i is contained in $s_{f(i)}$; and*
- *if a, b, c are increasing $(\mathrm{mod}\, m)$ then $f(a), f(b), f(c)$ are non-decreasing $(\mathrm{mod}\, k)$*

Proof of Lemma 35. We start by applying Lemma 33 in order to conclude that the Hausdorff distance between the spine hull and the Q-hull is at most 1. Specifically, Lemma 33 can be applied since:

(a) every vertex of the spine hull is in distance 1 from the Q-hull (if not, the corresponding disk would be either completely inside or completely outside the Q-hull, both of which are impossible), and

(b) every vertex of the Q-hull is in distance 1 from the spine hull, or, in other words, every vertex of the Q-hull is contained in one of the strips, which follows from Observation 26.

Now given this bound on the Hausdorrf distance, by Theorem 32, the Fréchet distance between the Q-hull and the spine hull is at most 1.

Let this Fréchet mapping be denoted f. As it maps the Q-hull to the spine-hull, with Fréchet distance ≤ 1, f trivially satisfies the first condition of the lemma. As for the second condition, we consider two cases. First suppose that of three values $f(a)$, $f(b)$, and $f(c)$, that at least two are equal. Then since we are working $\bmod k$ this implies the $f(a)$, $f(b)$, $f(c)$ are non-decreasing, regardless of the third value, and so the second condition in the lemma trivially holds. On the other hand, if $f(a)$, $f(b)$, and $f(c)$ are all distinct then the second condition holds due to the monotonicity of the Fréchet distance in Definition 31. ◀

We will now prove Theorem 9 for the case of disjoint unit disks; that is, there exists a $(3\sqrt{2}, 12)$-smooth quarter-hull-supersequence Ξ for $\mathcal{R}$. In fact, the sequence in the case of disjoint disks is even $(1, 3)$-smooth; we will argue the precise smoothness constants only for the general bounded ply case in Section 6.

Proof of Theorem 9 for disjoint disks.   Consider the quarter strip hull, as shown in Figure 7. For each strip (including the two unbounded half-strips), let $\mathcal{R}_i \subseteq \mathcal{R}$ be the set of disks that intersect the strip s_i. By Corollary 28, $\sum |\mathcal{R}_i| \in O(n)$.

For each strip s_i separately, we define a sequence Ξ_i consisting of the disks of $\mathcal{R}_i$, ordered according to Lemma 34. As the disks in $\mathcal{R}_i$ are disjoint and all intersect the bottom edge of the strip in order, the corresponding points can never be our of order by more than 1, and no more than 3 points can be within a region of radius 1, so Ξ_i is $(1, 3)$-smooth.

Finally, to produce the required sequence of the theorem statement, we concatenate the Ξ_i producing $\Xi = \langle \Xi_1, \Xi_2, \ldots, \Xi_k \rangle$. To see that Ξ has the desired size, by the above discussion we have $|\Xi| = \sum_i |\Xi_i| = O(\sum_i |\mathcal{R}_i|) = O(n)$.

Let $P \Subset \mathcal{R}$ be an arbitrary realization. What remains is to argue that the sequence of points on the quarter hull of P (i.e. the quarter Q-hull) is a subsequence of Ξ. By Lemma 35, there exists a mapping f that maps the points in the Q-hull to the strips in the strip hull, such that their cyclic orderings match.

Recall that when defining the quarter strip hull, we included the two unbounded half strips extending to the left of the topmost disk and below the rightmost disk. Let the *bounded quarter strip hull* refer to the portion of the quarter strip hull excluding these two half strips. Now consider the image of the quarter Q-hull under f; this is a sequence of strips of the strip hull. Note that not all of these strips need to be strips of the bounded quarter strip hull; however, all points of the quarter Q-hull that are mapped to strips not on the bounded quarter strip hull must necessarily lie in the two half strips in the (unbounded) quarter strip hull.

Thus, as Ξ appends the Ξ_i in the ordering of the strips of the quarter strip hull (including the unbounded ones), it suffices to argue that for all i, the order of the subset of Q-hull vertices mapped to s_i under f appears as a subsequence of Ξ_i; however this is guaranteed by Lemma 34. ◀

5.4   Proof of Theorem 10 (Preprocessing) for Disjoint Disks

In this section, we argue that we can also compute such a sequence Ξ in $O(n \log n)$ time. First, we want to argue that we can compute the arrangement of all disks and strips in $O(n \log n)$ time.

▶ **Lemma 36.** *The arrangement of all strips has compleixty $O(n)$*

Proof. By Corollary 29, the arrangements of strips has constant ply. Since the strips are in convex position, they form a system of pseudodisks. Finally, any system of pseudodisks of bounded ply has complexity $O(n)$ [35]. ◀

By the above lemma and Corollary 28, we have that the arrangement of all strips and all disks has complexity $O(n)$. Furthermore, such an arrangement can be computed in time proportional to its complexity multiplied by an additional log factor [34].

▶ **Lemma 37.** *We can compute the sets $\mathcal{R}_i$, for all i simultaneously, in $O(n \log n)$ time.*

Proof. First, we compute the arrangement of all disks and all strips, which can be done in $O(n \log n)$ time as the arrangement has complexity $O(n)$. Then, for each strip, we identify the set of disks $\mathcal{R}_i$ that intersect the strip s_i. Since we constructed the arrangement explicitly, we can just walk through it to collect the disks intersecting each strip. ◀

Proof of Theorem 10 for disjoint disks. We follow the proof of Theorem 9, which is constructive. First, we compute the $\mathcal{R}_i$ sets for all i using Lemma 37. Once we have the sets $\mathcal{R}_i$, we need to construct the Ξ_i, after which we simply concatenate the Ξ_i in order. Each Ξ_i is constructed using Lemma 34, which in turn projects $\mathcal{R}_i$ onto a line and then applies Lemma 21, whose run time is proportional to the output sequence length. As Theorem 9 states that $|\Xi| = O(n)$, the overall running time is dominated by the $O(n \log n)$ time to construct all the $\mathcal{R}_i$ sets. ◀

6   Preprocessing overlapping unit disks for convex hulls

We will now prove Theorems 9 and 10 in full generality for overlapping disks, by utilizing our one dimensional results, namely Lemma 21. Generally, the main ideas and structure of the arguments remain the same; hence we only describe the main differences.

Most importantly, Lemma 34 from Section 5 fails for overlapping disks, which implies that we need to construct our sequences differently. Instead we will reuse Lemma 21 from Section 3.2 as a subroutine.

▶ **Lemma 38.** *Let ℓ be a line, and $\mathcal{R}$ a set of unit disks of ply Δ, which all have their centers within distance d to the line ℓ. There is a sequence Ξ of at most $(8\Delta\lceil d\rceil + 8\Delta)|\mathcal{R}|$ disks such that: for any point set $P \Subset \mathcal{R}$, the order of the points in P projected on ℓ is a subsequence of Ξ. Moreover, when considering the intervals resulting from the projection of $\mathcal{R}$ onto ℓ, Ξ is a $(3, (4\Delta\lceil d\rceil + 4\Delta))$-smooth sorting-supersequence.*

To prove Lemma 38 we will require a bound on the number of disks that can locally project onto a line.

▶ **Lemma 39.** *Let ℓ be a line, and $\mathcal{R}$ a set of unit disks of ply Δ, which all have their centers within distance d to the line ℓ. The ply of the projected intervals on ℓ is at most $2\Delta\lceil d\rceil + 2\Delta$.*
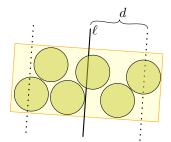
■ **Figure 10** The maximum ply of a set of projected disjoint unit disks is given by exactly the number of disjoint unit disks that fit inside a rectangle of dimensions $4 \times (2d + 2)$.

Proof. We need to find a bound on the ply of the projected disks in $\mathcal{I}$. The maximum ply is given by Δ times the number of disjoint disks that fit inside a rectangle of dimensions $4 \times (2d + 2)$. Refer to Figure 10. Specht *et al.* [36] show that this number is at most $2\lceil d \rceil + 2$.   ◄

Proof of Lemma 38. W.l.o.g. assume ℓ is horizontal. We will project all disks to ℓ and consider the resulting set of unit intervals $\mathcal{I}$. By Lemma 39, we can bound the ply of the projected disks in $\mathcal{I}$ by $\Delta' = 2\lceil d \rceil + 2$. Then, we apply Lemma 21 to obtain a sequence Ξ of at most $(8\Delta\lceil d \rceil + 8\Delta)|\mathcal{R}|$ intervals, which translates directly to the required sequence of disks. Moreover, Lemma 21 also tells us that Ξ is $(3, (4\Delta\lceil d \rceil + 4\Delta))$-smooth.   ◄

Using Lemma 38 in place of Lemma 34, all our algorithms extend straightforwardly to overlapping disks of bounded ply. In particular, the definitions of strips, inner and outer disks, smooth sequences all remain the same.

However, we can no longer derive Corollary 28 and Corollary 29 from Lemma 27. Instead, we have the following more general bounds:

▶ **Lemma 40.** *Consider the strip hull of a set $\mathcal{R}$ of unit disks of ply Δ. Every unit disk intersects at most $14\Delta - 7$ strips.*

Proof. Let $\mathcal{B}$ be the set of boundary disks of $\mathcal{R}$, and let $\mathcal{I}$ be a maximum independent set (in the intersection graph) of $\mathcal{B}$. By Corollary 28, every unit disk intersects at most 7 strips of the strip hull of $\mathcal{I}$. Clearly every vertex of the strip hull of $\mathcal{I}$ is also a vertex of the strip hull of $\mathcal{B}$; furthermore, since every disk in $\mathcal{B}$ intersects at least one disk in $\mathcal{I}$, there can be at most $2\Delta - 2$ additional vertices between any pair of adjacent vertices on the strip hull of $\mathcal{B}$; that is, each strip in of the strip hull of $\mathcal{I}$ corresponds to a sequence of at most $2\Delta - 1$ strips in the strip hull of $\mathcal{B}$. The lemma follows.   ◄

▶ **Lemma 41.** *Consider the strip hull of a set $\mathcal{R}$ of unit disks of ply Δ. The arrangement of strips has ply at most $8\Delta - 4$.*

Proof. Let $\mathcal{B}$ and $\mathcal{I}$ be as in the proof of Lemma 40. By Corollary 29, Any point in the plane is contained in at most 4 strips of the strip hull of $\mathcal{I}$, and thus in at most $8\Delta - 4$ strips of the strip hull of $\mathcal{B}$.   ◄

Proof of Theorem 9.   The proof remains the same as the disjoint disk case, except that our bounds on $\sum |\mathcal{R}_i|$ and $|\Xi_i|$ increase by Δ, and the smoothness constants change. Specifically, using Lemma 40 in place of Corollary 28 in the proof of Theorem 9, we now obtain the bound $\sum |\mathcal{R}_i| \in O(\Delta n)$.

For each strip s_i separately, we define a sequence Ξ_i consisting of possibly reoccurring disks of $\mathcal{R}_i$, according to Lemma 38. As the disks of $\mathcal{R}_i$ have their centers within distance 2 of the line supporting the strip s_i, by Lemma 38 we have $|\Xi_i| \leq (8 \cdot 2 + 8)\Delta|\mathcal{R}_i| = O(\Delta|\mathcal{R}_i|)$. Since we concatenate the Ξ_i to obtain Ξ, we have $|\Xi| \in O(\Delta^2 n)$ as required.

Furthermore, Lemma 38 states that when considering the intervals resulting from the projection of $\mathcal{R}$ on ℓ, Ξ_i is a $(3, (4 \cdot 2 + 4)\Delta) = (3, 12\Delta)$-smooth sorting-supersequence. Note that the angle between ℓ and a line of slope -1 is at most $45°$; therefore, the distance α increases by at most a factor $\sqrt{2}$. Moreover, β will not increase when going from the projection on the line back to the plane. We conclude that each Ξ_i is $(3\sqrt{2}, 12\Delta)$-smooth.

Proof of Theorem 10.   Using Lemma 41 in place of Corollary 29 in the proof of Theorem 10, we now have an arrangement of pseudodisks of ply $O(\Delta)$, which, using again the bound by Sharir [35], has complexity at most $O(\Delta n)$. The rest of the proof remains the same.

7   Reconstruction in Two Dimensions

In this section we present our reconstruction algorithm and then discuss how to extend it to achieve potential sublinear reconstruction by marking disks. Note the results in the section work directly for disks of bounded ply.

7.1   Standard Reconstruction

In this section, we consider the problem of recovering the true quarter hull from Ξ, given a realization $P \Subset \mathcal{R}$. We again start by replacing the disks in Ξ by their corresponding points; now we have a sequence X of points that contains the points on the quarter hull in counterclockwise order as a subsequence. Whether we can recover the quarter hull of such a sequence in linear time in general is equivalent to Open Problem 14. Here, we additionally assume the sequence X is (α, β)-smooth for constants α and β.

7.1.1   Algorithm for a Single Quadrant

We start with the following observation.

▶ **Observation 42.** *We can assume that X starts with the rightmost point and ends with the topmost point.*

Proof. We can find the rightmost and topmost points in X in linear time by scanning X once. Note that there might be multiple copies of these points. We delete the prefix of X before the first occurrence of the rightmost point, and we delete the suffix of X after the last occurrence of the topmost point. The resulting sequence must still contain the quarter hull of X as a subsequence; moreover, deleting points cannot violate that X is (α, β)-smooth.   ◀

We now describe our algorithm for extracting the quarter hull. It is an adaptation of the algorithm of Andrew [5] (which is in turn an adaptation of Graham's scan), but we don't have a guarantee that the points come completely sorted. However, whenever we encounter a point that is not in the correct sorted order, we argue that we can determine whether it should still be inserted or can be safely discarded in $O(\alpha \cdot \beta)$ time be checking only the last $\alpha \cdot \beta$ edges of the hull so far, similar to the algorithm in Section 4.

▶ **Lemma 43.** *Let X be a (α, β)-smooth sequence where the first point is the rightmost, the last is the topmost, and the quarter hull is a subsequence of X. We can compute the quarter hull of X in $O(\alpha\beta|X|)$ time.*

Proof. We process the points in order, maintaining a convex chain, C, of a subset the points processed so far, where initially C is just the rightmost point. So let q be the next point to be processed, and let p be the last (i.e. topmost) point on C. If $q = p$, then we skip q and move on to processing the next point. Otherwise, If the y-coordinate of q is greater than or equal to that of p, then we apply a standard Graham scan insert (i.e. pop from C until it is a left turn to q, and then insert q at the end of the chain). Finally, if the y-coordinate of q is less than that of p then there are two cases. Let ℓ be the line of slope -1 through p, and let $proj(q)$ be the orthogonal projection of q onto ℓ. If $||proj(q) - p|| > \alpha$ then we pop p from C, and we restart the process of trying to insert q. Otherwise, we check the last $\alpha \cdot \beta$ edges of C in order from left to right, where if for any edge q lies above the supporting line of that edge then we delete all vertices of C starting from the left endpoint of that edge and going to the left, after which we then perform a Graham scan insert of q into the remaining chain. If q does not lie above any such edge, then we skip q and move on to processing the next point.

For the running time, recall that in the last case when processing a point we might perform a constant time check on each of the last $\alpha \cdot \beta$ edges of C. Thus over all points this cost is $O(\alpha\beta|X|)$. This dominates the running time, as by the same analysis as in standard Graham scan, the remaining operations when processing a point take time proportional the number of deletions, and points are deleted at most once.

As for correctness, first observe that in the case that $||proj(q) - p|| > \alpha$, then as X is an (α, β)-smooth sequence, by definition p cannot be a vertex of the quarter hull. Thus the algorithm correctly deletes it, and so we will ignore this case in the remainder of the proof.

Let $C = c_1, \ldots c_m$ be the current convex chain at some point during the algorithm. Suppose that the prefix $c_1, \ldots, c_k$ is a prefix of the actual quarter hull. Then observe that this prefix can never be removed from C. Specifically, there are two cases when points are removed from C. The first case is by a Graham scan insertion, which by the correctness of Graham scan will not remove a vertex of the actual quarter hull. The second case is when if checking the last $\alpha \cdot \beta$ edges of C, the algorithm finds an edge where q lies above its supporting line, however, such an edge cannot be found in the $c_1, \ldots, c_k$ portion of C as it contains edges of the actual quarter hull.

Again let $C = c_1, \ldots, c_m$ be the convex chain at some iteration of the algorithm. Suppose that the $i - 1$ prefix of C, namely $C_{i-1} = c_1, \ldots, c_{i-1}$, is the $i - 1$ prefix of the quarter hull, and that the ith vertex of the quarter hull, call it q, is not in $c_i, \ldots, c_m$. In this case, we claim that if q is processed next by the algorithm then afterwards C_i with be the ith prefix of the quarter hull. As prefixes of the quarter hull are never removed as argued above, proving this claim inductively implies we correctly compute the quarter hull, as we know the quarter hull is a subsequence of X and thus we must eventually process the next vertex of the quarter hull. (The base case holds as initially C is the first point in X, which by the lemma statement is the first vertex of the quarter hull.)

To prove the claim we consider the different cases handled by the algorithm, in order. First, $q = c_m$ (c_m is called p above) is not possible since q by assumption of the claim statement is not in C. If the y-coordinate of q is greater or equal to c_m then we perform a Graham scan insert of q, which is guaranteed to place q in C after c_{i-1} (since the fact that q is above c_m implies that $c_1, \ldots, c_{i-1}$ are the only vertices of the quarter hull in C). So suppose that the y-coordinate of q is less than that of c_m. Next, in the case that $||proj(q) - c_m|| > \alpha$, we already remarked that c_m is correctly deleted. So suppose that $||proj(q) - c_m|| \leq \alpha$.

Now because q is on the actual quarter hull, though it lies below c_m, it must lie to the right of c_m. In particular, it must lie above the chain C, i.e. in the region right of and below c_m, above and left of c_1, and above C. Thus q lies above some edge of C, and since

$||proj(q) - c_m|| \leq \alpha$, as X is an (α, β)-smooth sequence, the packing property implies it must be one of the last $\alpha \cdot \beta$ edges of C. Finally, after identifying and removing up to the right end of this edge, q will make a right turn with the remainder of C until c_{i-1}. Thus after applying a Graham scan update, q will be placed after c_{i-1} as claimed. ◀

Note in the above algorithm it is possible that q is a duplicate of one of the vertices on the current chain (other than the last point p). This case is correctly handled, assuming in the later step we only check if q is strictly above the supporting line of the edges, as this will discard q which is the desired behavior.

7.1.2 Combining the Pieces

Proof of Theorem 11. If Ξ is an (α, β)-smooth quarter-hull-supersequence of $\mathcal{R}$, then by definition for any $P \Subset \mathcal{R}$, if we replace the regions from Ξ by their corresponding points in P, then it results in an (α, β)-smooth sequence such that the quarter hull of P is a subsequence. Moreover, by Observation 42 we can assume the first point in the sequence is the rightmost and the last is the topmost. Thus Lemma 43 directly states we can recover the quarter-hull of P in $O(\alpha\beta|\Xi|)$ time. ◀

7.2 Sublinear Reconstruction

If the reconstruction phase requires replacing each region in $\mathcal{R}$ by the corresponding point in P, then this phase has an immediate linear time lower bound. However, if the location of the realization of a certain region does not affect the combinatorial structure of the output, then it may not be necessary to replace it with its point. Following the approach of Van der Hoog *et al.* [39, 40], we thus modify the preprocessing framework by introducing another phase: the reconstruction phase is formally separated into a first subphase (that can take sublinear time), in which the auxiliary structure Ξ is transformed into another structure Ξ' which is combinatorially equivalent to the desired output $S(P)$, and a second subphase in which Ξ' is actually transformed into $S(P)$ in linear time, if so desired.

An advantage of using supersequences as an auxiliary structures is that Ξ' for our problem is simply a sequence of points and disks, which is the desired output convex hull when replacing the disks in the sequence with their corresponding realizations. As this second subphase is now trivial, we focus on the first subphase in the remainder of this section. This requires identifying the disks whose realizations do not affect the combinatorial structure of the convex hull, which are precisely the *stable* disks defined in Section 2.1.

As argued in Section 2.1, there are two types of stable disks, namely stable guaranteed boundary disks and stable impossible interior disks. First observe that stable impossible interior disks do not properly intersect any strip on the boundary (that is I from Observations 2 lies inside the chain of strips). Thus our constructed supersequence Ξ already omits such disks, as it is constructed only from disks which intersect strips.

The remaining stable disks are all guaranteed boundary disks, which Lemma 6 allows us to efficiently identify, and can be utilized as follows.

▶ **Definition 44.** *A* marked *convex-hull-supersequence Ξ of $\mathcal{R}$ is a sequence of (possibly recurring) disks of $\mathcal{R}$, some of which may be marked, and such that, for any point set $P \Subset \mathcal{R}$, the points on the convex hull of P appear as a subsequence of Ξ, that contains all marked items of Ξ.*

A normal convex-hull-supersequence is a marked supersequence with no items marked. Our goal is to mark as many items as possible, since these may be skipped during reconstruction.
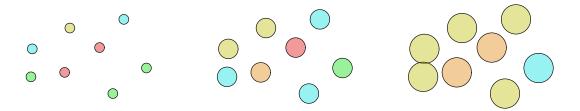
Figure 11 When we grow the uncertainty radii from 0 to ∞, all disks eventually turn from stable (red or green) to unstable potential (orange or yellow).

By the discussion above, the largest set possible to mark is precisely the stable guaranteed boundary disks, which we will mark using Lemma 6.

We now adapt the preprocessing algorithm to build a marked supersequence Ξ as follows. Let the counterclockwise sorted order of boundary disks be $\mathcal{B}(\mathcal{R}) = \{B_1, \ldots, B_k\}$. The marked disks in this ordering intuitively partition the disks into maximal length substrings of unmarked disks, which in turn partitions the strip hull into subchains. In particular, if B_i is a marked disk then both of its adjacent strips ($B_{i-1}B_i$ and B_iB_{i+1}) are no longer considered, which is the desired behavior as recall the proof of Lemma 6 argued no other disks intersected these strips, and because B_i is stable the location of its realization does not affect the result on the substring ending at B_{i-1} (and analogously for B_{i+1}). Thus we can safely apply Theorem 10 independently to each resulting substring and concatenate the results in order, with the respective marked disks in between.

In order to be able to quickly skip over marked disks during the reconstruction phase, we can equip the resulting supersequence with pointers from each item to the next non-marked item. During the reconstruction phase we then also apply Lemma 43 separately to each subsequence of consecutive non-marked items, in time proportional to the total number of non-marked items in the list. This results in a mixed sequence of points and marked disks, which is guaranteed to be the convex hull of P.

8 Conclusion & Future Work

We introduced the idea of using supersequences as auxiliary structures in the preprocessing model for dealing with uncertain geometric data. We were able to reproduce several known results with an arguably simpler auxiliary structure; moreover, we obtained at least one new result.

To follow up these first results using supersequences, we see three main lines of future work.

First, our work exposed an open problems that is of independent interest (Open Problem 14); settling whether this problem allows solutions in $o(n \log n)$ time or has a $\Omega(n \log n)$ lower bound would be very interesting.

Second, our results are restricted to unit intervals or disks of constant ply. Previous work in the preprocessing model has focused on allowing more general sets of regions, that may overlap more, be less fat, or have different sizes - which combinations of such generalizations are possible and for which problems is the topic of a whole line of papers. It would be very interesting to see whether our results can be generalized to less restricted regions as well; likely, such generalizations would require a relaxation of the smooth property, thus linking this line of future work to solving the open problems as well. Relatedly, when the uncertainty

radius is not fixed in advance, understanding the relation between the radius, the resulting ply, and the efficiency of reconstruction would be of interest (see Figure 11).

Finally, we only looked at two problems that produce an output that is a subsequence of the input: sorting and the convex hull. There are other natural problems that might fit in this framework, such as the Pareto Front, the traveling salesperson problem, shortest paths in polygons, and more. It would be interesting to study such problems in the same model.

Acknowledgements

──── **References** ────

1 Online encyclopedia of integer sequences, sequence a062714. Accessed: 2024-07-08. URL: `http://oeis.org/A062714`.

2 Suneet a Ramaswami. Convex hulls: Complexity and applications (a survey). Technical report, University of Pennsylvania, 1993.

3 Pankaj K. Agarwal, Sariel Har-Peled, Subhash Suri, Hakan Yildiz, and Wuzhou Zhang. Convex hulls under uncertainty. *Algorithmica*, 79(2):340–367, 2017. URL: `https://doi.org/10.1007/s00453-016-0195-y`, `doi:10.1007/S00453-016-0195-Y`.

4 Helmut Alt, Johannes Blömer, and Hubert Wagener. Approximation of convex polygons. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 703–716, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

5 A.M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216 – 219, 1979. URL: `http://www.sciencedirect.com/science/article/pii/0020019079900723`, `doi:https://doi.org/10.1016/0020-0190(79)90072-3`.

6 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*, 2011.

7 Kevin Buchin and Wolfgang Mulzer. Delaunay Triangulations in $O(sort(n))$ Time and More. *Journal of the ACM (JACM)*, 2011.

8 Bernard Chazelle and Wolfgang Mulzer. Computing hereditary convex structures. In *Proc. 24th Symposium on Computational Geometry*, 2009.

9 Vaclav Chvatal, David A. Klarner, and Donald E. Knuth. Selected combinatorial research problems. Technical report, Stanford, CA, USA, 1972.

10 Olivier Devillers. Delaunay Triangulation of Imprecise Points: Preprocess and Actually get a Fast Query Time. *Journal of Computational Geometry (JoCG)*, 2011.

11 Olivier Devillers, Marc Glisse, Xavier Goaoc, and Rémy Thomasse. Smoothed complexity of convex hulls by witnesses and collectors. *Journal of Computational Geometry*, 2016.

12 Olivier Devillers and Mordecai J. Golin. Incremental algorithms for finding the convex hulls of circles and the lower envelopes of parabolas. In *Information Processing Letters*, 1995. URL: `https://api.semanticscholar.org/CorpusID:10397073`.

13 A. Edalat, A. Lieutier, and E. Kashefi. The convex hull in a new model of computation. *Canadian Conference on Computational Geometry (CCCG)*, pages 93–96, 2001.

14 Herbert Edelsbrunner and Ernst P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990. `doi:10.1145/77635.77639`.

15 Jeff Erickson, Ivor Hoog, and Tillmann Miltzow. Smoothing the gap between np and er. *SIAM Journal on Computing*, pages FOCS20–102, 04 2022. `doi:10.1137/20M1385287`.

16 William Evans and Jeff Sember. The Possible Hull of Imprecise Points. *Canadian Conference on Computational Geometry (CCCG)*, 2011.

17   Esther Ezra and Wolfgang Mulzer. Convex Hull of Points Lying on Lines in $o(n \log n)$ Time after Preprocessing. *Computational Geometry*, 2013.

18   Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975. URL: `https://www.sciencedirect.com/science/article/pii/0012365X7590103X, doi:https://doi.org/10.1016/0012-365X(75)90103-X`.

19   Michael T. Goodrich and Jack Snoeyink. Stabbing parallel segments with a convex polygon. *Comput. Vis. Graph. Image Process.*, 49(2):152–170, 1990. `doi:10.1016/0734-189X(90)90135-I`.

20   Leonidas Guibas, David Salesin, and Jorge Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica*, 1993.

21   Martin Held and Joseph Mitchell. Triangulating Input-constrained Planar Point Sets. *Information Processing Letters (IPL)*, 2008.

22   Gyula Hermann. Robust Convex Hull-based Algorithm for Straightness and Flatness Determination in Coordinate Measuring. *Acta Polytechnica Hungarica*, 2007.

23   Hongyao Huang and Benjamin Raichel. Convex hull complexity of uncertain points. In J. Mark Keil and Debajyoti Mondal, editors, *Proceedings of the 32nd Canadian Conference on Computational Geometry, CCCG 2020, August 5-7, 2020, University of Saskatchewan, Saskatoon, Saskatchewan, Canada*, pages 6–14, 2020.

24   Leo Joskowicz and Yonatan Myers. Topological Stability and Convex Hull with Dependent Uncertainties. *European Workshop on Computational Geometry (EuroCG)*, 2014.

25   Christian Knauer. *Algorithms for Comparing Geometric Patterns*. PhD thesis, FU Berlin, 2003. URL: `http://webdoc.sub.gwdg.de/ebook/diss/2003/fu-berlin/2002/110/part2.pdfandhttp://webdoc.sub.gwdg.de/ebook/diss/2003/fu-berlin/2002/110/part2chap5.pdf`.

26   Maarten Löffler and Wolfgang Mulzer. Unions of Onions: Preprocessing Imprecise Points for Fast Onion Decomposition. *Journal of Computational Geometry (JoCG)*, 2014.

27   Maarten Löffler and Jack Snoeyink. Delaunay Triangulation of Imprecise Points in Linear Time after Preprocessing. *Computational Geometry*, 2010.

28   Maarten Löffler and Marc van Kreveld. Largest and Smallest Convex Hulls for Imprecise Points. *Algorithmica*, 2010.

29   Takayuki Nagai, Seigo Yasutome, and Nobuki Tokura. Convex hull problem with imprecise input. In *Discrete and Computational Geometry, Japanese Conference (JCDCG)*, volume 1763 of *Lecture Notes in Computer Science*, pages 207–219. Springer, 1998. URL: `https://doi.org/10.1007/978-3-540-46515-7_18`.

30   Takayuki Nagai, Seigo Yasutome, and Nobuki Tokura. Convex Hull Problem with Imprecise Input and its Solution. *Systems and Computers in Japan*, 1999.

31   Sasa Radomirovic. A construction of short sequences containing all permutations of a set as subsequences. *Electron. J. Comb.*, 19:31, 2012. URL: `https://api.semanticscholar.org/CorpusID:8872002`.

32   Saša Radomirović. A construction of short sequences containing all permutations of a set as subsequences. *The Electronic Journal of Combinatorics*, 19, 2012.

33   Jeffery Sember. *Guarantees concerning geometric objects with imprecise points*. PhD thesis, University of British Columbia, 2011. URL: `https://open.library.ubc.ca/collections/ubctheses/24/items/1.0052098`.

34   Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 208–215, 1976. `doi:10.1109/SFCS.1976.16`.

35   Micha Sharir. On k-sets in arrangement of curves and surfaces. *Discret. Comput. Geom.*, 6:593–613, 1991. `doi:10.1007/BF02574706`.

36   E. Specht. High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40(1):58–69, 2013. URL: `https://www.`

sciencedirect.com/science/article/pii/S0305054812001141, doi:https://doi.org/10.1016/j.cor.2012.05.011.

37   Oliver Tan. Skip letters for short supersequence of all permutations, 2022. URL: https://arxiv.org/abs/2201.06306, arXiv:2201.06306.

38   Przemysław Uznański. All permutations supersequence is conp-complete, 2015. URL: https://arxiv.org/abs/1506.05079, arXiv:1506.05079.

39   Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann. Preprocessing ambiguous imprecise points. In *Proc. 35th Symposium on Computational Geometry*, 2019.

40   Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann. Preprocessing imprecise points for the pareto front. In *Proc. 32nd Symposium on Discrete Algorithms*, 2022.

41   Marc van Kreveld, Maarten Löffler, and Joseph Mitchell. Preprocessing Imprecise Points and Splitting Triangulations. *SIAM Journal on Computing (SICOMP)*, 2010.