

Delaunay Triangulation of Imprecise Points Simplified and Extended

Kevin Buchin¹, Maarten Löffler², Pat Morin³, and Wolfgang Mulzer⁴

¹ Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands.
kbuchin@win.tue.nl

² Dep. of Information and Computing Sciences, Utrecht University, The Netherlands.
loffler@cs.uu.nl

³ School of Computer Science, Carleton University, Canada
morin@scs.carleton.ca

⁴ Department of Computer Science, Princeton University, USA,
wmulzer@cs.princeton.edu

Abstract. Suppose we want to compute the Delaunay triangulation of a set P whose points are restricted to a collection \mathcal{R} of input regions known in advance. Building on recent work by Löffler and Snoeyink [21], we show how to leverage our knowledge of \mathcal{R} for faster Delaunay computation. Our approach needs no fancy machinery and optimally handles a wide variety of inputs, eg, overlapping disks of different sizes and fat regions.

1 Introduction

Data imprecision is a fact of life that is often ignored in the design of geometric algorithms. The input for a typical computational geometry problem is a finite point set P in \mathbb{R}^2 , or more generally \mathbb{R}^d . Traditionally, one assumes that P is known exactly, and indeed, in the 1980s and 1990s this was often justified, as much of the input data was hand-constructed for computer graphics or simulations. Nowadays, however, the input is often sensed from the real world, and thus inherently imprecise. This leads to a growing need to deal with imprecision.

An early model for imprecise geometric data, motivated by finite precision of coordinates, is *ε -geometry* [17]. Here, the input is a traditional point set P and a parameter ε . The true point set is unknown, but each point is guaranteed to lie in a disk of radius ε . Even though this model has proven fruitful and remains popular due to its simplicity [2, 18], it may often be too restrictive: imprecision regions could be more complicated than disks, and their shapes and sizes may even differ from point to point to model, eg, imprecision from different sources or independent imprecision in different input dimensions. The extra freedom in modeling leads to more involved algorithms, but still many results are available.

1.1 Preprocessing

The above papers assume that the imprecise input is given once and simply has to be dealt with. While this holds in many applications, it is also often possible to

get (more) precise estimates of the points, but they will only become available later, or they come at a higher cost. For example, in the *update complexity* model [16, 6], each data point is given imprecisely at the beginning but can always be found precisely at a certain price.

One model that has received attention lately is that of *preprocessing* an imprecise point set so that some structure can be computed faster when the exact points become available later. Here, we consider triangulations: let \mathcal{R} be a collection of n planar regions, and suppose we know that the input has exactly one point from each region. The question is whether we can exploit our knowledge of \mathcal{R} to quickly triangulate the exact input, once it is known. More precisely, we want to preprocess \mathcal{R} into a data structure for *imprecise triangulation queries*: given a point p_i from each region $R_i \in \mathcal{R}$, compute a triangulation of $\{p_1, \dots, p_n\}$. There are many parameters to consider; not only do we want preprocessing time, space usage, and query time to be small, but we would also like to support general classes of input regions and obtain “nice” (ie, Delaunay) triangulations. In the latter case, we speak of *imprecise Delaunay queries*.

Held and Mitchell [19] show that if \mathcal{R} consists of n disjoint unit disks, it can be preprocessed in $O(n \log n)$ time into a linear-space data structure that can answer imprecise triangulation queries in linear time. This is improved by Löffler and Snoeyink [21] who can handle imprecise *Delaunay* queries with the same parameters. Both results generalize to regions with limited overlap and limited difference in shape and size—as long as these parameters are bounded, the same results hold. However, no attempt is made to optimize the dependence on the parameters.

Contrarily, van Kreveld, Löffler and Mitchell [23] study imprecise triangulation queries when \mathcal{R} consists of n disjoint polygons with a total of m vertices, and they obtain an $O(m)$ -space data structure with $O(m)$ query and $O(m \log m)$ preprocessing time. There is no restriction on the shapes and sizes of the individual regions (they do not even strictly have to be polygonal), only on the overlap. As these works already mention, a similar result for imprecise *Delaunay* queries is impossible. Djidjev and Lingas [15] show that if the points are sorted in any one direction, it still takes $\Omega(n \log n)$ time to compute their Delaunay triangulation. If \mathcal{R} consists of n vertical lines, the only information we could precompute is exactly this order (and the distances, but they can be found from the order in linear time anyway). All the algorithms above are deterministic.

1.2 Contribution

Our main concern will be imprecise Delaunay queries. First, we show that the algorithm by Löffler and Snoeyink [21] can be simplified considerably if we are happy with randomization and expected running time guarantees. In particular, we avoid the need for linear-time polygon triangulation [7], which was the main tool in the previous algorithm.

Second, though fast Delaunay queries for arbitrary regions are out of reach, we show that for realistic input we can get a better dependence on the realism parameters than in [21]. In particular, we consider k , the largest depth in the

arrangement of \mathcal{R} , and β_f , the smallest fatness of any region in \mathcal{R} (defined for a region R as the largest β such that for any disk D with center in R and intersecting ∂R , $\text{area}(R \cap D) \geq \beta \cdot \text{area}(D)$). We can preprocess \mathcal{R} in $O(n \log n)$ time into a data structure of $O(n)$ size that handles imprecise Delaunay queries in $O(n \log(k/\beta_f))$ time. We also consider β_t , the smallest thickness (defined as the fraction of the outcircle of a region occupied by it) of any region in \mathcal{R} , and r , the ratio between the diameters of the largest and the smallest region in \mathcal{R} . With the same preprocessing time and space, we can answer imprecise Delaunay queries in $O(n(\log(k/\beta_t) + \log \log r))$ time. For comparison, the previous bound is $O(nkr^2/\beta_t^2)$ [21]. Finally, we achieve similar results in various other realistic input models.

We describe two different approaches. The first, which gives the same result as [21], is extremely simple and illustrates the general idea. The second approach relies on quadtrees [12, Chapter 14] and is a bit more complicated, but generalizes easily. We extensively use a technique that has emerged just recently in the literature [9, 10, 23] and which we call *scaffolding*: in order to compute many related structures quickly, we first compute a “typical” structure—the *scaffold* Q —in a preprocessing phase. To answer a query, we insert the input points into Q and use a *hereditary* algorithm [9] to remove the scaffold efficiently. For removing the scaffold we use an algorithm for hereditary Delaunay triangulations:

Theorem 1 (Chazelle *et al* [8], see also [9]). *Let $Q \subseteq \mathbb{R}^2$ be a planar m -point set, let $P \subseteq Q$, and suppose that the Delaunay triangulation of Q is available. Then the Delaunay triangulation of P can be computed in expected time $O(m)$.* \square

2 Unit Disks: Simplified Algorithm

We begin with a very simple randomized algorithm for the original setting: given a sequence $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ of n disjoint unit disks, we show how to preprocess \mathcal{R} in $O(n \log n)$ time into a linear-space data structure that can handle imprecise Delaunay queries in $O(n)$ *expected* time.

Let c_i denote the center of R_i and for $r > 0$ let R_i^r be the disk centered at c_i with radius r . The preprocessing algorithm creates a point set Q that, for each R_i , contains c_i and 7 points equally spaced on ∂R_i^2 , the boundary of R_i^2 . Then it computes $\text{DT } Q$, the Delaunay triangulation of Q , and stores it. Since Q has $8n$ points, this takes $O(n \log n)$ time (eg, [12, Section 9]). We will need the following useful lemma about Q .

Lemma 1. *Let X be a point set with at most one point from each R_i . Any disk D of radius r contains at most $9(r+3)^2$ points of $Q \cup X$.*

Proof. Let c be the center of D . Any point of $Q \cup X$ in D comes from some R_i with $\|c - c_i\| \leq r + 2$. The number of such R_i is at most the number of disjoint unit disks that fit into a disk of radius $r + 3$. A simple volume argument bounds this by $(r+3)^2$. As each R_i contributes up to 9 points, the claim follows. \square

Given the input sequence $P = \langle p_1, \dots, p_n \rangle$, we construct $\text{DT} P$ by first inserting P into $\text{DT} Q$ to obtain $\text{DT}(Q \cup P)$ and then applying Theorem 1 to remove Q . To compute $\text{DT}(Q \cup P)$, we proceed as follows: for each point p_i we perform a (breadth-first or depth-first) search among the triangles of $\text{DT}(Q \cup \{p_1, \dots, p_{i-1}\})$ that starts at some triangle incident to c_i and never leaves R_i , until we find the triangle t_i that contains p_i . Then we insert p_i into $\text{DT}(Q \cup \{p_1, \dots, p_{i-1}\})$ by making it adjacent to the three vertices of t_i and performing *Delaunay flipping* [12, Section 9.3]. This takes time proportional to the number of triangles visited plus the degree of p_i in $\text{DT}(Q \cup \{p_1, \dots, p_i\})$. The next lemma allows us to bound these quantities.

Lemma 2. *Let $Y = Q \cup X$ where X is any point set and consider $\text{DT} Y$. Then, for any point $p \in Y \cap R_i$, all neighbors of p in $\text{DT} Y$ lie inside R_i^3 .*

Proof. Suppose there is an edge pq with $p \in R_i^1$ and $q \notin R_i^3$, see Figure 1. Then there is a disk C with p and q on its boundary and having no point of Y in its interior. The disk C contains a (generally smaller) disk C' tangent to R_i^1 and to ∂R_i^3 . The intersection of C' with ∂R_i^2 is a circular arc of length $8 \arcsin(1/4) > 4\pi/7$. But this is a contradiction since one of the 7 points on ∂R_i^2 must lie in $C' \subseteq C$, so C cannot be empty. \square

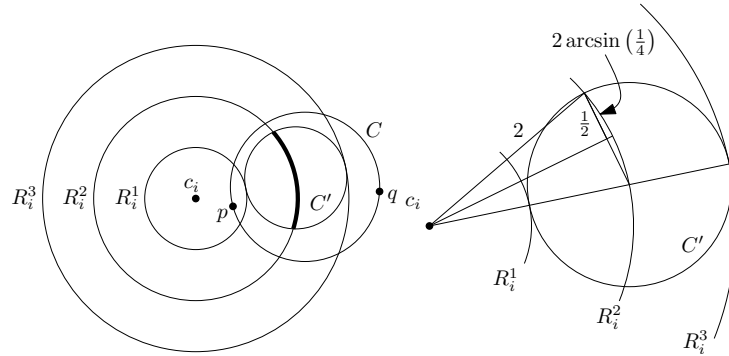


Fig. 1. C' covers a constant fraction of the boundary of R_i^2 and hence meets Q .

The next two lemmas bound the number of triangles visited while inserting p_i .

Lemma 3. *Any triangle of $\text{DT}(Q \cup \{p_1, \dots, p_i\})$ that intersects R_i has all three vertices in R_i^3 .*

Proof. Let t be a triangle with a vertex q outside of R_i^3 that intersects R_i . The outcircle C of t intersects R_i , has q on its boundary, and contains no point of $Q \cup \{p_1, \dots, p_i\}$ in its interior. As in Lemma 2, we see that C contains one of the 7 points on the boundary of R_i^2 , so t cannot be Delaunay. \square

Lemma 4. *At most 644 triangles of $\text{DT}(Q \cup \{p_1, \dots, p_i\})$ intersect R_i .*

Proof. The triangles that intersect R_i form the faces of a planar graph G . By Lemma 3 every vertex of G lies inside R_i^3 , so by Lemma 1, there are at most $v = 9(3 + 3)^2 = 324$ vertices, and thus at most $2v - 4 = 644$ faces. \square

The final lemma bounds the degree of p_i at the time it is inserted.

Lemma 5. *The degree of p_i in $\text{DT}(Q \cup \{p_1, \dots, p_i\})$ is at most 324.*

Proof. By Lemma 2, all the neighbours of p_i are inside R_i^3 and by Lemma 1 there are at most $9(3 + 3)^2 = 324$ points of $Q \cup \{p_1, \dots, p_i\}$ in R_i^3 . \square

Thus, by Lemmas 4 and 5 each point of P can be inserted in constant time, so we require $O(n)$ time to construct $\text{DT}(Q \cup P)$. A further $O(n)$ time is then needed to obtain $\text{DT} P$ using Theorem 1. This yields the desired result.

Theorem 2. *Let $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ be a sequence of disjoint planar unit disks. In $O(n \log n)$ time and using $O(n)$ space we can preprocess \mathcal{R} into a data structure that can answer imprecise Delaunay queries in $O(n)$ expected time.*

3 Disks of Different Size: Quadtree-Approach

We now extend Theorem 2 to differently-sized disks using a somewhat more involved approach. The main idea is the same: in the preprocessing phase, we derive a point set Q from \mathcal{R} , and to answer a Delaunay query P we construct $\text{DT}(P \cup Q)$ and split it using Theorem 1. The difference is that now we do not immediately precompute $\text{DT} Q$, but rather a *quadtree* for Q that can be used to construct $\text{DT} Q$ efficiently later on. With the additional structure of the quadtree we can handle disks of different sizes. The following lemma follows from the empty circle property of Delaunay triangulations.

Lemma 6 (see Rajan [22]). *Let $P \subseteq \mathbb{R}^2$ be a planar n -point set, and let \mathcal{T} be a triangulation of P with no obtuse angle. Then \mathcal{T} is a Delaunay triangulation.* \square

A *quadtree* T is an ordered rooted tree in which each node has degree at most four. It corresponds to a hierarchical decomposition of the plane into axis-aligned square *boxes*. The *size* of a box is the length of its sides. Each node v of T represents a box B_v , and v 's children usually correspond to the boxes obtained by subdividing B_v into four equal squares, in clockwise order. However, a child of v may also be a *cluster node*, representing a box that is contained in the square that the child would usually represent and whose size is smaller than its parent by at least a large constant factor 2^C (see Figure 2). Cluster nodes ensure that the complexity of T is linear [4, Section 3.2], and they are treated like a point in the surrounding box. Since cluster nodes may be somewhat non-standard, the reader may think of our quadtrees as a hierarchical collection of quadtrees where

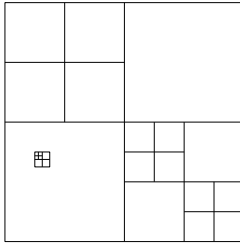


Fig. 2. A quadtree. The lower left box contains a cluster.

the cluster nodes correspond to tiny quadtrees contained in the leaves of a larger quadtree. Given a planar point set P , we say that T is a quadtree for P if (i) each leaf of T contains at most one point of P ; (ii) the root box of T contains all of P and its edges are sufficiently far apart from P ; and (iii) the complexity of T is $O(|P|)$. The next lemma is a variant of a theorem by Bern *et al* [4], to which we refer the reader for further details (see also [5]).

Lemma 7. *Let $P \subseteq \mathbb{R}^2$ be a planar n -point set, and let T be a quadtree for P . Then, given P and T , we can find the DT P in expected time $O(n)$.*

Proof. First, we extend T into a quadtree T' that is (i) *balanced* and (ii) *separated*, ie, (i) no leaf in T' shares an edge with a leaf whose size differs by more than a factor of two and (ii) each non-empty leaf of T' is surrounded by two layers of empty boxes of the same size. This can be done by a top-down traversal of T , adding additional boxes for the balance condition and by subdividing the non-empty leaves of T to ensure separation. If after C subdivision steps a non-empty leaf B still does not satisfy separation, we place a small box around the point in B and treat it as a cluster, in which separation obviously holds.

Given T' , we obtain a non-obtuse Steiner triangulation \mathcal{T} for P with $O(n)$ additional vertices through a sequence of local manipulations, as described by Bern *et al* [4, Section 3]. Since all these operations involve constantly many adjacent boxes, the total time for this step is linear. By Lemma 6, \mathcal{T} is Delaunay, and we can use Theorem 1 to extract DT P in $O(n)$ expected time. \square

To apply Lemma 7 we need to preprocess \mathcal{R} into a quadtree T such that any leaf of T intersects only constantly many disks. While we could consider the disks directly, we will instead use a quadtree T for a point set Q representing the disks. For each disk we include its center and top-, bottom-, left- and rightmost points in Q . Then, T can be constructed in $O(n \log n)$ time [4].

Lemma 8. *Every box B of T is intersected by $O(1)$ disks in \mathcal{R} .*

Proof. First, suppose that B does not contain a cluster. If a disk D intersects B and does not contain a corner of B , then B must either contain D 's center or one of its four extreme points (cf [13]). Thus, B intersects at most 5 disks, one for each corner and one for a point of Q it contains.

Now suppose B contains a cluster C . We must count the disks that intersect B , do not cover a corner of B , and have an extreme point or their center in C . For this, consider the at most four orthogonal neighbors of C in B (ie, copies of B directly to the left, to the right, above and below C). As we just argued, each of these neighbors meets $O(1)$ disks, and every disk D with an extreme point or center in C that intersects $B \setminus C$ also meets one of the orthogonal neighbors (if D has no extreme point or center in an orthogonal neighbor and does not cover any of its corners, it has to cover its center), which implies the claim.⁵ \square

Theorem 3. *Let $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ be a sequence of disjoint planar disks (of possibly different size). In $O(n \log n)$ time and using $O(n)$ space we can preprocess \mathcal{R} into a data structure that can answer imprecise Delaunay queries in $O(n)$ expected time.*

Proof. We construct Q and the quadtree T for Q as described above. For each R_i we store a list with the leaves in T that intersect it. By Lemma 8, the total size of these lists, and hence the complexity of the data structure, is linear. Now we describe how queries are handled: let $P = \langle p_1, \dots, p_n \rangle$ be the input sequence. For each p_i , we find the box that contains it by traversing the list for R_i . This takes linear time. Since each leaf of T contains constantly many input points, we can turn T into a quadtree for $P \cup Q$ in linear time. We now compute $\text{DT}(P \cup Q)$ via Lemma 7, and finally $\text{DT} P$ by using Theorem 1 again. \square

4 Overlapping Disks: Deflated Quadtrees

We extend the approach to disks with limited overlap. Now \mathcal{R} contains n planar disks such that no point is covered by more than k disks. Aronov and Har-Peled [1] show that k can be approximated up to a constant factor in $O(n \log n)$ time. It is easily seen that imprecise Delaunay queries take $\Omega(n \log k)$ time in the worst case, and we show that this bound can be achieved.

The general strategy is the same as in Section 3. Let Q be the $5n$ representative points for \mathcal{R} , and let T be a quadtree for Q . As before, T can be found in time $O(n \log n)$ and has complexity $O(n)$. Now we use T to build a k -deflated quadtree T' . For an integer $\lambda > 0$, a λ -deflated quadtree T' for a point set Q has the same general structure as a regular quadtree, but it has lower complexity: each leaf of T' can contain up to 3λ points of Q there are $O(n/\lambda)$ nodes. We distinguish four different types of nodes: (i) *leaves* are nodes without children, with up to 3λ points; (ii) *regular internal nodes* have four children that cover their parent, and there are no points associated with them directly; (iii) *cluster nodes* are, as before, nodes with a single—much smaller—child, and with no

⁵ There is a slight subtlety concerning clusters which are close to the boundary of B : we require that either C shares an edge with B or that B 's orthogonal neighbors are fully contained in B . This is ensured by positioning the clusters appropriately. The additional points on B 's boundary can easily be handled, eg, by building a corresponding cluster in the adjacent box.

Algorithm `DeflateTree`(v)

1. If $n_v \leq 3\lambda$, return the tree consisting of v .
2. Let T_v be the subtree rooted in v , and let Z be the path of nodes z in T_v with $n_z > n_v - \lambda$. Let v' be the lowest node in Z .
3. For all children w of v' , let $T'_w = \text{DeflateTree}(w)$.
4. Build a tree T'_v by picking v as the root, v' as the only child of v , and linking the trees T'_w to v' . The points in $B_v \setminus B_{v'}$ are assigned to v . Return T'_v as the result.

Algorithm 1: Turn a quadtree into a λ -deflated quadtree.

associated points; (iv) finally, a *deflated node* v has only one child v' —possibly much smaller than its parent—and additionally may contain up to λ points in its *proper interior* $B_v \setminus B_{v'}$.

Given a quadtree T for Q , a λ -deflated quadtree T' can be found in linear time. For every node v in T , compute $n_v = |B_v \cap Q|$. This takes $O(n)$ time. Then, T' is obtained by applying `DeflateTree` (Algorithm 1) to the root of T . Since `DeflateTree` performs a simple top-down traversal of T , it takes $O(n)$ time.

Lemma 9. *A λ -deflated quadtree T' produced by Algorithm 1 has $O(n/\lambda)$ nodes.*

Proof. Let $S(n)$ be count the nodes of a deflated quadtree for n points. Clearly,

$$S(n) \leq \begin{cases} 1 & \text{for } n \leq 3\lambda, \\ 2 + \sum_{i=1}^4 S(n_i) & \text{for } n > 3\lambda, \end{cases}$$

where n_i are the number of points contained in the children of v' . We claim that this solves to $S(n) = 1$ for $n \leq 3\lambda$ and $S(n) \leq c(n/\lambda) - 2$ for $n > 3\lambda$, for some constant c . The statement holds for $n \leq 13\lambda$. For $n > 13\lambda$, suppose that n_1 is the largest of the n_i . Since $n_1 \geq (n - \lambda)/4 > 3\lambda$ and $n_1 \leq n - \lambda$, the claim follows by induction. \square

Now let T' be a k -deflated quadtree for Q . By treating deflated nodes like clusters and noting that the center and corners of each box of T' can be contained in at most k disks, the same arguments as in Lemma 8 lead to the next lemma:

Lemma 10. *Any box b of T' is intersected by $O(k)$ disks of \mathcal{R} .* \square

Theorem 4. *Let $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ be a sequence of planar disks such that no point is covered by more than k disks. In $O(n \log n)$ time and using $O(n)$ space we can preprocess \mathcal{R} into a data structure that can answer imprecise Delaunay queries in $O(n \log k)$ expected time.*

Proof. It remains to show how to preprocess T' to handle the imprecise Delaunay queries in time $O(n \log k)$. By Lemmas 9 and 10, the total number of disk-box incidences in T' is $O(n)$. Thus, in $O(n)$ total time we can find for each $R \in \mathcal{R}$ the list of boxes in T' it intersects. Next, we determine for each deflated node v in T' the portion X_v of the original quadtree T in the proper interior of v and

build a point location data structure for X_v . Since X_v is a quadtree for at most k points, it has complexity $O(k)$, and since the X_v are disjoint, the total space requirement and construction time are linear. This finishes the preprocessing.

To handle an imprecise Delaunay query, we first locate the input points P in the boxes of T' just as in Theorem 3. This takes $O(n)$ time. Then we use the point location structures for the X_v to locate P in T in total time $O(n \log k)$. Now we turn T into a quadtree for $P \cup Q$ in time $O(n \log k)$, and find the Delaunay triangulation in time $O(n)$, as before. \square

5 Realistic Input Models

In this section we show that the results of the previous sections readily generalize to many realistic input models [14]. This is because a point set representing the regions—like the point set Q for the disks—exists in such models, eg, for fat regions. Thus, we directly get an algorithm for fat regions for which we provide a matching lower bound. We then demonstrate how to handle situations where a set like Q cannot be easily constructed by the example of *thick regions*.

Let β be a constant with $0 < \beta \leq 1$. A planar region R is β -fat if for any disk D with center in R and intersecting ∂R , $\text{area}(R \cap D) \geq \beta \cdot \text{area}(D)$. A planar region R is β -thick if $\text{area}(R) \geq \beta \cdot \text{area}(D_{\min}(R))$, where D_{\min} denotes the smallest disk enclosing R . Let κ be a positive integer. A set Q of points is called a κ -guarding set (against axis-aligned squares) for a set of planar regions \mathcal{R} , if any axis-aligned square not containing a point from Q intersects at most κ regions from \mathcal{R} . For instance, the point set Q considered in the previous sections is a 4-guarding set for disjoint disks [13]. It is also a $4k$ -guarding set for disks which do not cover any point more than k times.

The definition of a κ -guarding set Q does not explicitly state how many regions a square containing points of Q might intersect, but a square containing m points of Q can intersect only $4\kappa m$ regions [13, Theorem 2.8]. Now, assume each point in Q is assigned to a region in \mathcal{R} . We call Q a κ -strong-guarding set of \mathcal{R} if any square containing m points of Q intersects at most κ regions plus the regions assigned to the m points. This definition is motivated by the following relation between fatness and guardability. A set of planar regions \mathcal{R} is κ -cluttered if the corners of the bounding rectangles for \mathcal{R} constitute a κ -guarding set. De Berg *et al* prove that a set of disjoint β -fat regions is $8/\beta$ -cluttered [14, Theorem 3.1, Theorem 3.2]. Their argument actually shows strong guardability (cf [11, Lemma 2.8]) and easily extends to overlapping regions.

Lemma 11. *For a set of β -fat regions \mathcal{R} that cover no point more than k times, the corners of the bounding rectangles of \mathcal{R} constitute a $(8k/\beta)$ -strong-guarding set for \mathcal{R} (with corners assigned to the corresponding region). \square*

Since the argument in the proof of Lemma 8 (and of Lemma 10) is based on axis-aligned squares, it directly generalizes to the quadtree for a guarding set.

Lemma 12. *Let \mathcal{R} be a set of regions and Q a κ -strong-guarding set for \mathcal{R} . Let T' be a κ -deflated quadtree of Q . Then any box of T intersects $O(\kappa)$ regions. \square*

We say that \mathcal{R} is *traceable* if we can find the m incidences between the n regions in \mathcal{R} and the l leaves of a deflated quadtree T in $O(l + m + n)$ time and the bounding rectangles of the regions in \mathcal{R} in $O(n)$ time. For example, this holds for polygonal regions of total complexity $O(|\mathcal{R}|)$.

Theorem 5. *Let $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ be a sequence of traceable planar regions with linear-size κ -strong-guarding set Q , where κ is not necessarily known, but Q is. In $O(n \log n)$ time and using $O(n)$ space we can preprocess \mathcal{R} into a data structure that can answer imprecise Delaunay queries in $O(n \log \kappa)$ expected time.*

Proof. Lemma 12 would directly imply the theorem if κ was known. We can find a suitable λ -deflated tree with $\lambda \in O(\kappa)$ by an exponential search on λ , i.e., for a given λ we build a λ -deflated tree and check whether any box intersects more than $c\lambda$ regions for a constant $c \geq 6$. Recall that a deflated quadtree can be computed from a quadtree in linear time. Thus, finding a suitable λ takes $O(n \log n)$ time. \square

For overlapping fat regions Lemma 11 and Theorem 5 imply the following result.

Corollary 1. *Let $\mathcal{R} = \langle R_1, \dots, R_n \rangle$ be a sequence of planar traceable β -fat regions such that no point is covered by more than k of the regions. In $O(n \log n)$ time and using $O(n)$ space we can preprocess \mathcal{R} into a data structure that can answer imprecise Delaunay queries in $O(n \log(k/\beta))$ expected time.*

We next show that the $O(n \log(1/\beta))$ bound for disjoint fat regions is optimal.

Theorem 6. *For any n and $1/\beta \in [1, n]$, there exists a set \mathcal{R} of $O(n)$ planar β -fat rectangles such that imprecise Delaunay queries for \mathcal{R} take $\Omega(n \log(1/\beta))$ time in the algebraic computation tree model.*

Proof. We adapt a lower bound by Djidjev and Lingas [15, Section 4]. Wlog, β^{-1} is an integer. Consider the problem β -1-CLOSENESS: we are given $k = \beta n$ -sequences $\mathbf{x}_1, \dots, \mathbf{x}_k$, each containing β^{-1} real numbers in $[0, \beta^{-2}]$, and we need to decide whether any \mathbf{x}_i contains two numbers with difference at most 1. Any algebraic decision tree for β -1-CLOSENESS has cost $\Omega(n \log(1/\beta))$: let $W' \subseteq \mathbb{R}^n$ be defined as $W' = \{(\mathbf{x}_1, \dots, \mathbf{x}_k) \mid |x_{ij} - x_{il}| > 1 \text{ for } 1 \leq i \leq k; 1 \leq j \neq l \leq \beta^{-1}\}$, where x_{ij} is the j th coordinate of \mathbf{x}_i . Let $W = W' \cap [0, \beta^{-2}]^n$. Since W has at least $(\beta^{-1})^{\beta n}$ connected components, Ben-Or's lower bound [3, Theorem 5] implies the claim. Now, we construct \mathcal{R} . Let $\varepsilon = \beta/10$ and consider the $\beta^{-1} + 2$ intervals on the x -axis $B_\varepsilon[0], B_\varepsilon[\beta/3], B_\varepsilon[2\beta/3], \dots, B_\varepsilon[1/3], B_\varepsilon[1/2]$, where $B_\varepsilon[x]$ denotes the one-dimensional closed ε -ball around x . Extend the intervals into β^3 -fat rectangles with side lengths 2ε and β^{-2} . These rectangles constitute a *group*. Now, \mathcal{R} consists of k congruent groups G_1, \dots, G_k ; sufficiently far away from each other. Let $\mathbf{x}_1, \dots, \mathbf{x}_k$ be an instance of β -1-CLOSENESS. The input P consists of k sets P_1, \dots, P_k , one for each \mathbf{x}_i . Each P_i contains $\beta^{-1} + 2$ points, one from every rectangle in G_i : $P_i = \langle (0, 0), (1/(3\alpha), x_{i1}), \dots, (1/3, x_{i\beta^{-1}}), (1/2, 0) \rangle + v_i$, where v_i denotes the displacement vector for G_i . Clearly, P can be computed in

$O(n)$ time. Djidjev and Lingas [15] argue that either \mathbf{x}_i contains two numbers with difference at most $1/3$, or the Voronoi cells for P_i intersect the line through the left-most rectangle in G_i according to the sorted order of \mathbf{x}_i . In either case, we can decide β -1-CLOSENESS in linear time from DT P , as desired. \square

Finally, we consider β -thick regions. Although thickness does not give us a guarding set, we can still preprocess \mathcal{R} for efficient imprecise Delaunay queries if the ratio between the largest and smallest region in \mathcal{R} is bounded (we measure the size by the radius of the smallest enclosing circle).

Theorem 7. *Let \mathcal{R} be a sequence of n β -thick k -overlapping regions such that the ratio of the largest and the smallest region in \mathcal{R} is r . In $O(n \log n)$ time we can preprocess \mathcal{R} into a linear-space data structure that can answer imprecise Delaunay queries in time $O(n(\log(k/\beta) + \log \log r))$.*

Proof. Subdivide the regions into $\log r$ groups such that in each group the radii of the minimum enclosing circles differ by at most a factor of 2. For each group \mathcal{R}_i , let ρ_i be the largest radius of a minimum enclosing circle for a region in \mathcal{R}_i . We replace every region in \mathcal{R}_i by a disk of radius ρ_i that contains it. This set of disks is at most $(2k/\beta)$ -overlapping, so we can build a data structure for \mathcal{R}_i in $O(n_i \log n_i)$ time by Theorem 4. To answer an imprecise Delaunay query, we handle each group in $O(n_i \log(k/\beta))$ time and then use Kirkpatrick's algorithm [20] to combine the triangulations in time $O(n \log \log r)$. \square

6 Conclusions

We give an alternative proof of the result by Löffler and Snoeyink [21] with a much simpler, albeit randomized, algorithm that avoids heavy machinery. Our approach yields optimal results for overlapping disks of different sizes and fat regions. Furthermore, it enables us to leverage known facts about guarding sets to handle other realistic input models. We need randomization only when we apply Theorem 1 to remove the scaffold, and finding a deterministic algorithm for hereditary Delaunay triangulations remains an intriguing open problem.

Acknowledgments

The results in Section 2 were obtained at the NICTA Workshop on Computational Geometry for Imprecise Data, December 18–22, 2008 at the NICTA University of Sydney Campus. We would like to thank the other workshop participants, namely Hee-Kap Ahn, Sang Won Bae, Dan Chen, Otfried Cheong, Joachim Gudmundsson, Allan Jørgensen, Stefan Langerman, Marc Scherfenberg, Michiel Smid, Tasos Viglas and Thomas Wolle, for helpful discussions and for providing a stimulating working environment. We would also like to thank David Eppstein for answering our questions about [4] and pointing us to [5]. This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) through the project GOGO and the BRICKS/FOCUS project no. 642.065.503. W. Mulzer was supported in part by NSF grant CCF-0634958 and NSF CCF 0832797. Pat Morin was supported by NSERC, NICTA and the University of Sydney.

References

- [1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.
- [2] D. Bandyopadhyay and J. Snoeyink. Almost-Delaunay simplices: Nearest neighbor relations for imprecise points. In *SODA*, pages 403–412, 2004.
- [3] M. Ben-Or. Lower bounds for algebraic computation trees. In *STOC*, pages 80–86, 1983.
- [4] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comput. System Sci.*, 48(3):384–409, 1994.
- [5] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *Internat. J. Comput. Geom. Appl.*, 9(6):517–532, 1999.
- [6] R. Bruce, M. Hoffmann, D. Krizanc, and R. Raman. Efficient update strategies for geometric computing with uncertainty. *Theory Comput. Syst.*, 38(4):411–423, 2005.
- [7] B. Chazelle. Triangulating a simple polygon in linear time. *Disc. and Comp. Geometry*, 6:485–524, 1991.
- [8] B. Chazelle, O. Devillers, F. Hurtado, M. Mora, V. Sacristán, and M. Teillaud. Splitting a Delaunay triangulation in linear time. *Algorithmica*, 34(1):39–46, 2002.
- [9] B. Chazelle and W. Mulzer. Computing hereditary convex structures. *to appear in SoCG*, 2009.
- [10] K. L. Clarkson and C. Seshadhri. Self-improving algorithms for Delaunay triangulations. In *SoCG*, pages 148–155, 2008.
- [11] M. de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica*, 28(3):353–366, 2000.
- [12] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry*. Springer-Verlag, Berlin, third edition, 2000. Algorithms and applications.
- [13] M. de Berg, H. David, M. J. Katz, M. H. Overmars, A. F. van der Stappen, and J. Vleugels. Guarding scenes against invasive hypercubes. *Computational Geometry: Theory and Applications.*, 26(2):99–117, 2003.
- [14] M. de Berg, A. F. van der Stappen, J. Vleugels, and M. J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002.
- [15] H. N. Djidjev and A. Lingas. On computing Voronoi diagrams for sorted point sets. *Internat. J. Comput. Geom. Appl.*, 5(3):327–337, 1995.
- [16] P. G. Franciosa, C. Gaibisso, G. Gambosi, and M. Talamo. A convex hull algorithm for points with approximately known positions. *Internat. J. Comput. Geom. Appl.*, 4(2):153–163, 1994.
- [17] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *SoCG*, pages 208–217, 1989.
- [18] L. J. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica*, 9:534–560, 1993.
- [19] M. Held and J. S. B. Mitchell. Triangulating input-constrained planar point sets. *Inf. Process. Lett.*, 109(1):54–56, 2008.
- [20] D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *FOCS*, pages 18–27, 1979.
- [21] M. Löffler and J. Snoeyink. Delaunay triangulations of imprecise points in linear time after preprocessing. In *SoCG*, pages 298–304, 2008.
- [22] V. T. Rajan. Optimality of the Delaunay triangulation in \mathbb{R}^d . *Disc. and Comp. Geometry*, 12:189–202, 1994.
- [23] M. J. van Kreveld, M. Löffler, and J. S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. In *ISAAC*, pages 544–555, 2008.