Voronoi Diagrams for Moving Disks and Applications

Menelaos I. Karavelas *

Graphics Laboratory, Computer Science Department, Stanford University,
Stanford, CA 94305-9035, USA,
karavelas@cs.stanford.edu

Abstract. In this paper we discuss the kinetic maintenance of the Euclidean Voronoi diagram and its dual, the Delaunay triangulation, for a set of moving disks. The most important aspect in our approach is that we can maintain the Voronoi diagram even in the case of intersecting disks. We achieve that by augmenting the Delaunay triangulation with some edges associated with the disks that do not contribute to the Voronoi diagram. Using the augmented Delaunay triangulation of the set of disks as the underlying structure, we discuss how to maintain, as the disks move, (1) the closest pair, (2) the connectivity of the set of disks and (3) in the case of non-intersecting disks, the near neighbors of a disk.

1 Introduction

Geometric objects that move with time appear in many problems in motion planning, geometric modeling, computer simulations of physical systems and virtual environments, robotics, computer graphics and animation, mobile communications, *ad hoc* networks or group communication in military operations. The aim is to answer questions concerning proximity information among the geometric objects, such as find the closest/farthest pair, report all near neighbors, predict collisions or report reachability between a pair of geometric objects. In many cases we can approximate the geometric objects by disks. The problem then reduces to answering proximity questions for a set of disks.

The Voronoi diagram is a data structure that can be used to produce answers to many of these questions. Voronoi diagrams have been successful in robotics applications such as collision detection [11] and retraction motion planning [10]. Dynamic or kinetic Voronoi diagrams for moving objects in the plane have also appeared in the literature. There are papers discussing the maintenance of the Voronoi diagram for a set of points [4], the maintenance of the Voronoi diagram for a set of moving convex polygons [5], as well as for the maintenance of near neighbors of points in sets of moving points [9].

Algorithms for maintaining the Voronoi diagram for sets of non-intersecting moving disks have also appeared. In [2] the Voronoi diagram for disks with respect to the Euclidean metric is considered. The maintenance of the Voronoi diagram with respect to the power distance, also called the *Power diagram*, is discussed in [6]. The most appealing feature of the Power diagram is that it consists of straight arcs, in contrast to the Euclidean Voronoi diagram that consists of straight or hyperbolic arcs. The main

* Supported by NSF grant CCR-9910633.

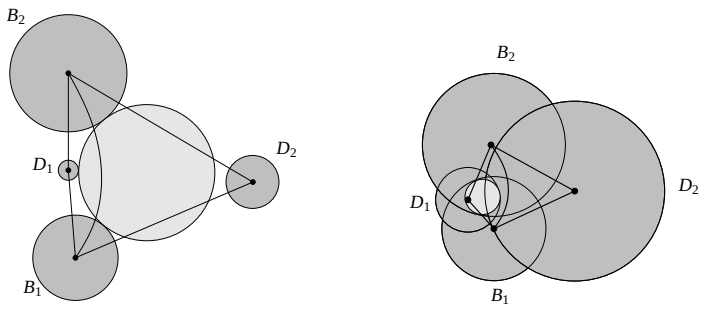

Fig. 1. Left: the edge connecting B_1 and B_2 is locally Delaunay because the exterior tangent ball of B_1, B_2 and D_1 does not intersect D_2. Right: the edge connecting B_1 and B_2 is locally Delaunay because the interior tangent ball of B_1, B_2 and D_1 is not contained in D_2. The exterior/interior tangent ball of B_1, B_2 and D_1 is shown in light gray.

drawback of the Power diagram is that the intersection between a disk and its Voronoi cell may be empty, even if the Voronoi cell is not empty [8]. In the Euclidean Voronoi diagram, however, a disk with non-empty Voronoi cell always intersects its cell [12].

In this paper we tackle the problem of maintaining the Voronoi diagram, or its dual the Delaunay Triangulation (DT) for a set of disks moving in the plane. The major contribution of this paper is that the disks are allowed to intersect. This enables us to not only report collisions between disks, but also to report when the penetration depth between two disks achieves a certain value, or when a disk is wholly contained inside another disk. Moreover, our data structure can be used for maintaining the connectivity of the set of disks as the disks move.

The Voronoi diagram is maintained using the Kinetic Data Structure (KDS) framework introduced in [1]. In the KDS setting one maintains a geometric structure under continuous motion through a set of certificates proving its correctness. An *event queue* is maintained for the failure times of these certificates and at each event the structure of interest and its kinetic proof are appropriately updated.

The kinetization process relies heavily on the fact that the local Delaunay property for the edges in the DT ensures that the triangulation is globally Delaunay. Let e be an edge connecting the disks B_1, B_2 and having the disks D_1, D_2 as its neighbors in the triangulation. The local Delaunay property states that the edge e is an edge of the DT if the exterior tangent ball of B_1, B_2 and D_1 does not intersect the disk D_2, or if the interior tangent ball of B_1, B_2 and D_1 is not contained in D_2 (see Fig. 1). The global Delaunay property states that there exists an edge in the DT between two disks B_1 and B_2 if there exists an exterior tangent ball to B_1 and B_2 that does not intersect any other disk or if there exists an interior tangent ball to B_1 and B_2 that is not contained in any other disk. In this work we prove the relationship between the global and local Delaunay properties.

Using the local Delaunay property, we can maintain the Voronoi diagram for the set of disks using two types of events, one of which appears only in the case of intersecting disks. The data structure that we use is called the *Augmented Delaunay Triangulation* (ADT) of the set of disks. It consists of the Delaunay triangulation of the disks aug-

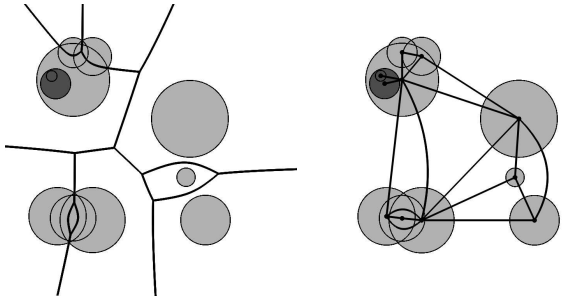

Fig. 2. The Voronoi diagram for a set of disks (left) and the corresponding Augmented Delaunay Triangulation (right). The disks in dark gray are trivial.

mented with some additional linear size data structure associated with the disks that do not contribute to the Voronoi diagram (see Fig. 2). We call these disks *trivial*. Since trivial disks exist only when we allow disk intersections, the ADT differs from the DT only when we have disk intersections.

An interesting property of the ADT is that the closest pair of the set of disks is realized between two disks that share an edge in the ADT. Thus, knowing how to maintain the ADT enables us to maintain the closest pair of the set of disks using a tournament tree on the edges of the ADT. The distance between two disks B_1 and B_2 is defined as :

$$\delta(B_1,B_2) = \begin{cases} d(b_1,b_2) - r_1 - r_2, & B_1 \not\subseteq B_2 \text{ and } B_2 \not\subseteq B_1 \\ -2\min\{r_1,r_2\}, & B_1 \subseteq B_2 \text{ or } B_2 \subseteq B_1 \end{cases}, \tag{1}$$

where b_i are the centers of the disks, r_i their radii and $d(\cdot,\cdot)$ denotes the Euclidean metric. If the set of disks does not have any intersecting disks the distance function (1) gives us the closest pair in the usual sense. If there are intersecting disks, then the closet pair with respect to (1) is either the pair of non-trivial disks with *maximum penetration depth* among all intersecting pairs of disks, or the largest trivial disk and its container.

Another important property of the ADT is that a subgraph of the ADT is a spanning subgraph of the connectivity graph of the set of disks. Knowing how to maintain the ADT enables us to maintain the connectivity of the set of disks by maintaining the afore-mentioned spanning subgraph.

Finally, the DT of a set of non-intersecting disks has the property that if we want to find the near neighbors of a disk we only need to look at its neighborhood in the DT. Therefore, in order to maintain the near neighbors of a disk we simply need to look at its neighborhood in the DT and update this neighborhood as the DT changes. We make this statement more precise in Section 7.

The rest of the paper is structured as follows. In Section 2 we introduce the Voronoi diagram for disks, and discuss some of its properties. Section 3 is devoted to proving the relationship between the global and local Delaunay properties. In Section 4 we show how to kinetize the Voronoi diagram. In Section 5 we describe how to maintain the closest pair. In Section 6 we show how to maintain a spanning subgraph of the

connectivity graph of the set of disks. In Section 7 we discuss the maintenance of near neighbors of disks. Finally, Section 8 is devoted to conclusions and further work.

2 The Voronoi Diagram for Disks and its Properties

Let S be a set of n disks B_j, with centers b_j and radii r_j. Let $d(\cdot,\cdot)$ be the Euclidean distance. We define the distance $\delta(p,B)$ between a point $p \in \mathbb{E}^2$ and a disk $B = \{b,r\}$, as $\delta(p,B) = d(p,b) - r$. We define the Voronoi diagram for the set S as follows. For each $i \neq j$, let $H_{ij} = \{y \in \mathbb{E}^2 : \delta(y,B_i) \leq \delta(y,B_j)\}$. Then we define the (closed) Voronoi cell of B_i to be the cell $V_i = \bigcap_{j \neq i} H_{ij}$. The *Voronoi diagram* $\mathrm{VD}(S)$ of S is defined to be the set of points which belong to more that one Voronoi cell. The Voronoi diagram just defined is a subdivision of the plane. It consists of straight or hyperbolic arcs and each Voronoi cell is star-shaped with respect to the center of the corresponding disk. In contrast to the Voronoi diagram for points, there may be disks whose corresponding Voronoi cell is empty. In particular, the Voronoi cell V_i of a disk B_i is empty if and only if B_i is wholly contained in another disk (see [12, Property 2]). A disk whose Voronoi cell has empty interior is called *trivial*, otherwise is called *non-trivial*.

We define the dual of $\mathrm{VD}(S)$ as follows. The vertices are the centers of the non-trivial disks. If $V_i \cap V_j \neq \emptyset$, we add an edge $[b_i,b_j]$ for every open arc α of $V_i \cap V_j$. It turns out that the dual graph is a planar graph and the size of both the Voronoi diagram and its dual graph is $O(n)$ [12, Properties 6 and 7]. If the Voronoi diagram consists of a single connected component and the disks are in general position, the dual graph is a generalized triangulation of the plane. By generalized we mean that the edges of the triangles may be curved arcs or polygonal lines instead of straight line segments. We assume throughout the rest of the paper that the Voronoi diagram consists of only one connected component. We shall refer to the dual graph of $\mathrm{VD}(S)$ as the *Delaunay Graph* $\mathrm{DG}(S)$ of S. If the disks are in general position we refer to the dual graph as the *Delaunay Triangulation* $\mathrm{DT}(S)$ of S.

Let B_i and B_j be two disks such that no disk is contained inside the other. A ball tangent to B_i and B_j that does not contain either of the two is an *exterior tangent ball*. A ball tangent to B_i and B_j that lies in $B_i \cap B_j$ is an *interior tangent ball*. The following theorem couples the existence of edges in $\mathrm{DG}(S)$ with exterior and interior tangent balls of disks in S.

Theorem 1 (Global Property). *There exists an edge $[b_i,b_j]$ in $DG(S)$ between B_i and B_j if and only if one of the following holds: (1) there exists an exterior tangent ball to B_i and B_j which does not intersect any disk $B_k \in S$, $k \neq i,j$; (2) there exists an interior tangent ball to B_i and B_j, which is not contained in any disk $B_k \in S$, $k \neq i,j$.*

Proof. (Sketch) Let $[b_i,b_j]$ be an edge of $\mathrm{DG}(S)$. Then $V_i \cap V_j$ consists of at least one arc α with non-empty interior. Let y be an interior point of α. Consider the ball C centered at y with radius $|\delta(y,B_i)| = |\delta(y,B_j)|$. Then C is tangent to both B_i, B_j and does not intersect any disk B_k, $k \neq i,j$, if $y \notin B_i \cap B_j$, and is not contained in any disk B_k, $k \neq i,j$, if $y \in B_i \cap B_j$. Conversely, let C be a common tangent ball of B_i, B_j, and let y be its center. If either assumption (1) or assumption (2) of the theorem holds, we have that $\delta(y,B_i) = \delta(y,B_j) < \delta(y,B_k)$, for all $k \neq i,j$. Hence y is an interior point of some arc α of $V_i \cap V_j$, and thus there exists at least one edge $[b_i,b_j]$ in $\mathrm{DG}(S)$. $\square$

In order to account for the trivial disks, we augment the Delaunay triangulation with some additional edges. For a trivial disk D we add an edge between D and its container disk. If D has more than one container we need to add an edge to only one of its containers, chosen arbitrarily. We call this structure the *Augmented Delaunay Triangulation* $\mathrm{ADT}(S)$ of S. The set of additional edges forms a forest, and the root of each tree in the forest is a non-trivial disk. Clearly, the forest has linear size. Hence the size of $\mathrm{ADT}(S)$ is still $O(n)$.

3 The Local Property of the Delaunay Triangulation

In this section we present the local Delaunay property for a set of possibly intersecting disks and we show that the local Delaunay property is a sufficient and necessary condition for a (generalized) triangulation of the set of disks to be globally Delaunay. We only consider non-trivial disks, since trivial disks do not contribute to the Voronoi diagram.

Let π_{ij} be the bisector of B_i and B_j. The bisectors are lines or hyperbolas which can be oriented. We define the orientation to be such that b_i is to the left of π_{ij}. Let $\prec$ be the linear ordering on the points of π_{ij}. Let o_{ij} be the midpoint of the subsegment of b_ib_j that lies either in free space or in $B_i \cap B_j$. We can parameterize π_{ij} as follows: if $p \prec o_{ij}$ then $\zeta_{ij}(p) = -(\delta(p,B_i) - \delta(o_{ij},B_i))$; otherwise $\zeta_{ij}(p) = \delta(p,B_i) - \delta(o_{ij},B_i)$. The function ζ_{ij} is a 1–1 and onto mapping from π_{ij} to $\mathbb{R}$.

Let B_i, B_j and B_k be three disks such that no disk is contained inside another. The three disks may have up to eight common tangent balls. Among those we are interested in only two kinds: those balls that do not contain any of the three disks, which we call *exterior tangent balls* and those that are contained entirely in the intersection of the three disks, which we call *interior tangent balls*. Let P_i, P_j, P_k be the points of tangency of the disks B_i, B_j, B_k with their common tangent ball. If $\mathsf{CCW}(P_i,P_j,P_k) > 0$ we call the common tangent ball the *left tangent ball* of the triple B_i, B_j, B_k. If $\mathsf{CCW}(P_i,P_j,P_k) < 0$, we call the common tangent ball the *right tangent ball* of the triple B_i, B_j, B_k. Note that three disks have at most one left/right exterior/interior tangent ball. Finally, we define $\zeta^L_{ij}(B_k)$ to be the parameter value of the center $c \in \pi_{ij}$ of the left tangent ball of B_i, B_j and B_k. Correspondingly, $\zeta^R_{ij}(B_k)$ is the parameter value of the center of the right tangent ball of B_i, B_j and B_k.

Let $\mathcal{T}(S)$ be a (generalized) triangulation of S that is constructed as follows. The vertices of $\mathcal{T}(S)$ are the centers of the disks in S. An oriented edge e^{kl}_{ij} in $\mathcal{T}(S)$ is an edge that connects the disks B_i and B_j and has as neighbors the disks B_k and B_l to its left and right, respectively. It is possible that the disks B_k and B_l are the same. The disk B_k is called the *left neighbor* of e^{kl}_{ij} and the disk B_l is called the *right neighbor* of e^{kl}_{ij}. Note that the quadruple (i,j,k,l) uniquely defines edges in $\mathcal{T}(S)$, i.e., there can be at most one oriented edge in the triangulation starting from B_i, ending at B_j and having B_k and B_l to its left and right, respectively. The left tangent ball of the triple B_i, B_j, B_k is called the *left (tangent) ball* of e^{kl}_{ij}, and similarly, the right tangent ball of the triple B_i, B_j, B_l is called the *right (tangent) ball* of e^{kl}_{ij}. We assume that for every edge in $\mathcal{T}(S)$ its left and right tangent balls exist. Then we can embed e^{kl}_{ij} with a two-leg polygonal line b_ixb_j, where x is a point on π_{ij} with parameter value $\zeta_{ij}(x)$ in between $\zeta^L_{ij}(B_k)$ and $\zeta^R_{ij}(B_l)$. For

every triangle $\Delta_{ijk} \in \mathcal{T}(S)$ that connects the disks B_i, B_j and B_k, in counterclockwise order, we associate the left tangent ball $\tilde{\Delta}_{ijk}$ of the triple B_i, B_j, B_k. This is called the *Delaunay ball* of Δ_{ijk}. Note that there is an 1–1 correspondance between triangles Δ in $\mathcal{T}(S)$ and their Delaunay balls $\tilde{\Delta}$.

An edge e_{ij}^{kl} in $\mathcal{T}(S)$ is called *locally Delaunay* if the predicate $\mathsf{InCircle}(B_i, B_j, B_k, B_l)$ is false. A triangle Δ in $\mathcal{T}(S)$ is called locally Delaunay if all its edges are locally Delaunay. The InCircle predicate is defined below.

Definition 1. *Let B_i, B_j, B_k, B_l be four disks. The predicate $\mathsf{InCircle}(B_i, B_j, B_k, B_l)$ is true if $k \neq l$ and either B_l intersects the exterior left tangent ball of B_i, B_j and B_k, or B_l contains the interior left tangent ball of B_i, B_j and B_k.*

Note that if an edge e_{ij}^{kl} is locally Delaunay then $\zeta_{ij}^{L}(B_k) > \zeta_{ij}^{R}(B_l)$. This imples that if a triangle Δ is locally Delaunay, then the center $c_{\tilde{\Delta}}$ of its Delaunay ball $\tilde{\Delta}$ lies in the interior of Δ. We are now ready to prove the main result of this section.

Theorem 2 (Local Property). *A (generalized) triangulation $\mathcal{T}(S)$ is the Delaunay triangulation of S if and only if all the triangles in $\mathcal{T}(S)$ are locally Delaunay.*

Proof. It is straightforward to verify that if a triangulation is globally Delaunay then it is locally Delaunay as well.

Suppose now that we have a triangulation $\mathcal{T}(S)$ that is locally Delaunay but not globally. We assume without loss of generality that for all triangles the corresponding Delaunay balls are interior. If this is not the case we can increase the radii of all the disks by a sufficiently large quantity. The triangulation $\mathcal{T}(S)$ is not affected by this change, other than that all the Delaunay balls become interior.

Since $\mathcal{T}(S)$ is not globally Delaunay there exists a triangle Δ that is locally Delaunay but its Delaunay ball $\tilde{\Delta}$ is contained inside some disk $B = \{b, r\}$. Consider the distance

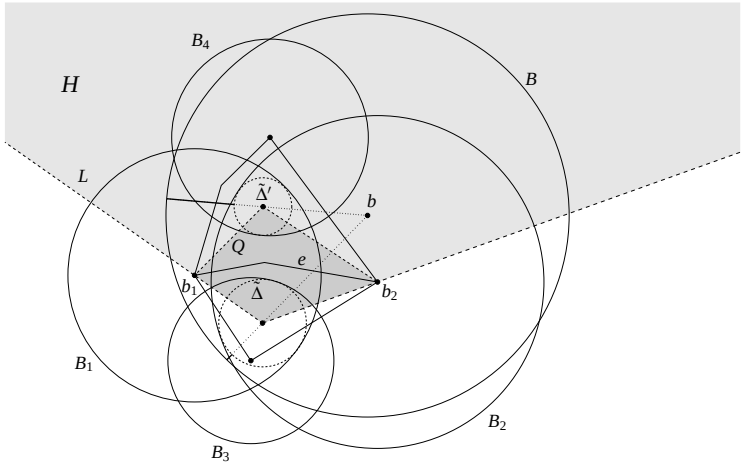

Fig. 3. Proof of the local property.

between the disk B and the Delaunay ball $\tilde{\Delta}$. This distance is $\delta(\tilde{\Delta}, B) = d(b, c_{\tilde{\Delta}}) + r_{\tilde{\Delta}} - r$, where $c_{\tilde{\Delta}}$ and $-r_{\tilde{\Delta}}$ are the center and radius of $\tilde{\Delta}$ (interior Delaunay balls are considered to have negative radius). Among all triangles Δ for which $\tilde{\Delta} \subset B$, choose Δ to be the one for which $\delta(\tilde{\Delta}, B)$ is minimized.

Let $e = [b_1, b_2]$ be the (oriented) edge of Δ that the segment $c_{\tilde{\Delta}} b$ intersects (see Fig. 3). Let L be the two-leg polygonal line $b_1 c_{\tilde{\Delta}} b_2$. Since $c_{\tilde{\Delta}} b$ intersects e, b must lie in the half-plane H bounded by L that contains e. Let Δ' be the left neighboring triangle of e. Since both Δ and Δ' are locally Delaunay the quad $Q = b_1 c_{\tilde{\Delta}} b_2 c_{\tilde{\Delta}'}$ is contained inside $\Delta \cup \Delta'$, and clearly b cannot lie inside Q. But then we have $\tilde{\Delta}' \subset B$, and moreover $\delta(\tilde{\Delta}', B) < \delta(\tilde{\Delta}, B)$, which contradicts the fact that $\delta(\tilde{\Delta}, B)$ is minimal. □

4 Kinetizing the Delaunay Triangulation

The framework that we use for maintaining the Voronoi diagram or equivalently the Augmented Delaunay triangulation is the Kinetic Data Structure (KDS) framework. The geometric attribute that we want to maintain as the objects move is called the *configuration function*, e.g., the Voronoi diagram of the set of disks. In the KDS setting we maintain a set of *certificates* that are conditions which ensure the correctness of the configuration function, e.g., that an edge in $\mathrm{DT}(S)$ passes the InCircle test. As the objects move the certificates fail. These are the critical events for the KDS and moreover the corresponding times are the only times that the configuration function can possibly change. When a critical event happens we need to change the set of certificates and possibly the configuration function itself. In order to efficiently update the configuration function we maintain an event queue of the certificates w.r.t. their failure times. When a critical event takes place we remove some certificates from the queue and add some new ones. For more details on KDSs see [1].

Maintaining the Voronoi diagram or its dual, the Delaunay triangulation, for a set of points moving on the plane is straightforward [4]. This is due to the local property of the Delaunay triangulation, which states that if the triangles of the Delaunay triangulation are locally Delaunay, then the triangulation is the Delaunay triangulation. When one of the conditions fails we simply have to do an *edge-flip* operation to restore the correctness of the Delaunay triangulation. The same principle is exploited to maintain the power diagram of non-intersecting moving disks [6] and the Voronoi diagram for rigidly moving polygons [5].

In the case of non-intersecting disks the very same ideas can be used. The local Delaunay property is also true for the Delaunay triangulation of disks, as we showed in the preceding section, and thus the critical events happen at times when four disks are cocircular or when three disks lying on the convex hull of S have a common tangent line. In fact if we add a disk at infinity and connect every disk lying on the convex hull of S with the disk at infinity, the compactified version of the Delaunay triangulation of S consists of triangles only, and every triangle has exactly three neighboring triangles. In this setting, the case of three disks having a common tangent reduces to a cocircularity event with one of the disks being a disk at infinity. When such a cocircularity event happens we only need to perform an edge-flip operation to restore the correctness of the Delaunay triangulation, and its dual the Voronoi diagram.

However, when we allow disk intersections the situation changes considerably. Unlike the points' case, there are disks that are not associated with a particular Voronoi cell, namely the trivial disks. We need to account for these disks, since as the disks move some of the trivial disks may become non-trivial and vice versa. This is done by considering the Augmented Delaunay triangulation instead of the Delaunay triangulation. There are two types of events that change the combinatorial structure of $\mathrm{ADT}(S)$: the *cocircularity* and the *appearance/disappearance* event. Both events are associated with edges of the $\mathrm{ADT}(S)$. In particular, an edge in $\mathrm{DT}(S)$ is associated with a cocircularity and a disappearance event. An edge in $\mathrm{ADT}(S) \setminus \mathrm{DT}(S)$ is associated with an appearance event. We now discuss each type of event separately.

The **cocircularity event** happens when four distinct disks have a common exterior or interior tangent ball. Let B_i, $i = 1,2,3,4$ be the four disks associated with the cocircularity event and let $[b_1,b_3]$ be the edge to be flipped. We need to delete that edge and add the edge $[b_2,b_4]$ (see Fig. 4(left)).

An **appearance event** occurs when a disk B_1 contained inside a disk B_2 is no longer wholly contained inside B_2. There are two possibilities when this happens: (1) B_2 is a trivial disk and (2) B_2 is a non-trivial disk. If B_2 is a trivial disk we delete the edge $[b_1,b_2]$ and add the edge $[b_1,b_3]$, where B_3 is the container disk of B_2. If B_2 is a non-trivial disk we need to first check its neighbors in the DT to see if B_1 is contained in any one of them. If such a neighbor B_3 exists we delete the edge $[b_1,b_2]$ and add the edge $[b_1,b_3]$. If B_1 is not contained in any of the neighbors of B_2, we need to identify the edge $[b_2,b_3]$ that corresponds to the edge of the Voronoi cell of B_2 that the half-line b_2b_1 intersects. Then duplicate this edge and add the edge $[b_1,b_3]$, thus creating two new triangles in $\mathrm{DT}(S)$ (see Fig. 4(right), from right to left).

A **disappearance event** takes place between two disks B_1 and B_2 when, e.g., B_1 becomes wholly contained in B_2. The edge $[b_1,b_2]$ belongs to two triangles with a common third point b_3 corresponding to a disk B_3. When the disappearance event happens

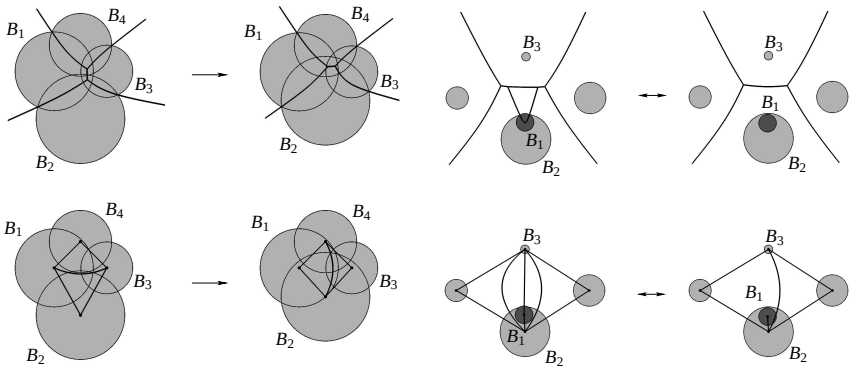

Fig. 4. The cocircularity (left) and appearance/disappearance events (right). Top row: the Voronoi diagram. Bottom row: the Augmented Delaunay triangulation.

we need to delete the edge $[b_1, b_3]$, and identify the two edges $[b_2, b_3]$, thus deleting two triangles from $\mathrm{DT}(S)$ (see Fig. 4(right), from left to right).

In any case, when an edge disappears we deschedule all the events associated with that edge. When an edge appears we schedule all the events corresponding to that edge and reschedule all the cocircularity events, if any, in which the new edge participates.

The number of certificates that we maintain is $O(n)$, since we have a constant number of certificates per edge in $\mathrm{ADT}(S)$, and the number of edges in $\mathrm{ADT}(S)$ is $O(n)$. The time to process the cocircularity and disappearance events is $O(\log n)$. The time to process the appearance event is $O(n)$. If the disks move along pseudo-algebraic trajectories, the total number of cocircularity events that our KDS has to process is $O(n^3\beta(n))$, where $\beta(n) = \lambda_s(n)/n$ and $\lambda_s(n)$ is the maximum length of a (n,s) Davenport-Schinzel sequence for some constant s. The total number of appearance/disappearance events that our KDS processes is obviously $O(n^2)$. Hence the total number of events that have to be processed is $O(n^3\beta(n))$. A lower bound of $\Omega(n^2)$ on the number of events can also be shown.

5 Closest Pair Maintenance

In this section we discuss how to maintain the closest pair of a set S of disks. The distance function that we use is given by relation (1). The trivial way to do the maintenance is to consider all $\binom{n}{2}$ pairs of disks and maintain the one of minimum distance with respect to the distance function (1). However, the Augmented Delaunay triangulation of S has the following property, the proof of which is omitted from this version of the paper.

Theorem 3. *Let B_1, B_2 be the closest pair in S. Then there exists an edge $[b_1, b_2]$ in $ADT(S)$.*

The theorem above suggests that we only need to look at $O(n)$ edges in order to determine the closest pair, namely the edges of $\mathrm{ADT}(S)$. In particular, we need to maintain a *tournament tree* T on the edges of $\mathrm{ADT}(S)$. Before describing how to actually maintain T we need some definitions. Let t_1 and t_2 be two nodes of T. We say that $t_1 \prec t_2$ if the depth of t_1 is smaller than the depth of t_2 in T, or if t_1 and t_2 are of the same depth and t_1 is to the left of t_2 in T. A node t_1 is *adjacent* to a node t_2 in T if they have the same parent. A node t is a *loser* if its parent is its adjacent node. Finally, a node t is a *winner* if its parent is itself.

The certificates associated with T are the winner-loser relationships. The tree T changes due to changes in the winner-loser relationships or due to changes of the $\mathrm{ADT}(S)$, because of cocircularity and appearance/disappearance events. When a winner-loser relationship changes we simply propagate the new winner up the tree, deschedule the old winner-loser relationships and schedule the new ones. During this propagation we visit $O(\log n)$ nodes of the tree and schedule/deschedule $O(\log n)$ certificates in total. Hence the cost per winner-loser relationship change is $O(\log^2 n)$. When an edge disappears we replace the corresponding leaf node with the last loser leaf node and delete the last winner and loser leaf nodes. Then we propagate the last loser leaf node up the tree as in the case of a winner-loser relationship change. Again this takes $O(\log^2 n)$ time. Finally, when an edge appears we create two new leaf nodes in the tree:

one for the new edge and one for the first leaf node. We attach the two new nodes under the current first leaf node and propagate the winner between these two nodes up the tree. Once again this takes $O(\log^2 n)$ time.

The number of changes in a single winner-loser relationship is constant for disks moving along pseudo-algebraic trajectories of constant degree. Hence the number of events that we have to process is dominated by the number of combinatorial changes of $\mathrm{ADT}(S)$, which is $O(n^3\beta(n))$. All, but the appearance event, are processed in $O(\log^2 n)$ time; the appearance event is processed in $O(n)$ time.

6 Kinetic Connectivity of Disks

In this section we discuss how to kinetically maintain the connectivity for a set of disks of different radii. The problem for unit disks has already been studied in [3].

The connectivity graph K of a set of disks S is defined as follows. The vertices of K are the centers of the disks in S. Two disks share an edge in K if they intersect. Let G be the subgraph of $\mathrm{ADT}(S)$ defined as follows. An edge e in $\mathrm{ADT}(S)$ belongs to G if and only if it is an edge between two non-trivial intersecting disks or between a trivial disk and its container disk. Clearly, G is a subgraph of the connectivity graph K of S (modulo multiple edges between two disks in $\mathrm{DT}(S)$). The important property of G is captured by the following theorem, the proof of which is omitted from this version of the paper.

Theorem 4. *If $B_1, B_2 \in S$ belong to the same connected component in K, then there exists a path in G that connects B_1 and B_2.*

In other words G is a spanning subgraph of K. This is really important since the size of K is $\Omega(n^2)$ in the worst case, whereas the size of G is $O(n)$. We can now maintain the connectivity of the disks using the dynamic graph data structure of Holm, de Lichtenberg and Thorup [7]. This data structure supports edge insertions and deletions in $O(\log^2 n)$ amortized time, and connectivity queries in $O(\log n/\log\log n)$ time. The graph that we maintain is the graph G defined above. Once we have $\mathrm{ADT}(S)$, maintaining G is really simple. First we color the edges of $\mathrm{ADT}(S)$ as follows: edges between non-intersecting non-trivial disks are *green*, edges between intersecting non-trivial disks are *orange* and edges between trivial disks and their containers are *red*. Clearly, G is the union of orange and red edges. The color of an edge changes if the corresponding disks become tangent. In particular, whenever a green edge becomes an orange edge or whenever an orange or a red edge appears we simply add it to G. Whenever an orange edge becomes a green edge or whenever an orange or a red edge disappears we simply delete it from G. Since the cost per insertion/deletion of edge in G is $O(\log^2 n)$, in the amortized sense, the cost per update of G is $O(\log^2 n)$ (amortized), except when we have an appearance event, in which case the update cost is $O(n)$. The number of times that two disks, moving along pseudo-algebraic trajectories of constant degree, can become tangent is constant. This implies that the number of events due to disk tangencies is $O(n^2)$. The total number of events for maintaining G is thus dominated by $O(n^3\beta(n))$, which is the number of times that the Delaunay triangulation of the set of disks can change combinatorially.

7 Near Neighbor Maintenance

Suppose that we have a set S of *non-intersecting* moving disks. Let P be a disk in S for which we want to know the disks in S that are within a certain, possibly time varying, distance R_P from P. Let C_P be the disk centered at P with radius R_P. The obvious approach is to maintain the distance from P to every other disk in S and keep those that intersect C_P. In fact, we can do better than that. If we are maintaining the DT of S, the only disks that can enter or exit C_P are those that are end points of edges of the DT crossing C_P exactly once. This is the essence of the following theorem, the proof of which we omit from this paper.

Theorem 5. *Let $\mathcal{T}(S)$ be the DT(S) and let $P \in S$. If a disk $Q \in S$ enters/exists the disk C_P at some time t_0, then there exists an edge in $\mathcal{T}(S)$ between Q and some disk that intersects C_P.*

Maintaining the near neighbors of P then reduces to maintaining the DT of S and updating the set E_P of DT edges, one end disk of which intersects C_P and the other does not. The set E_P changes when disks enter or exit C_P. Edge flips due to the maintenance of DT(S) may also change E_P. In case we want to maintain the k-nearest neighbors of P the same idea applies with two slight modifications: (1) the distance R_P is defined to be the distance of the center of P from P_k, where P_k is the k-th nearest neighbor of P and (2) the edges of DT(S) adjacent to P_k are all included in E_P.

We omit the details of the algorithms since they are essentially the same as the corresponding algorithms for points in [9].

8 Conclusion

In this paper we presented how to kinetically maintain the Voronoi diagram for a set of disks moving in the plane. The key steps in the kinetization process were the introduction of the Augmented Delaunay triangulation and the establishment of the relationship between the local and global Delaunay properties. We showed how to maintain the closest pair of the set of disks and how to maintain a spanning subgraph of the connectivity graph of the set of disks using the Augmented Delaunay triangulation as the underlying structure. Finally, if the disks do not intersect, we discussed how to maintain the disks that are within a prescribed distance from a reference disk or how to maintain the k-nearest neighbors of a reference disk.

We strongly believe that the results presented in this paper can be generalized to more general *additively weighted Voronoi diagrams*, in which the weights can be positive as well as non-positive. We would also like to extend the results presented here to general smooth convex objects or to environments where obstacles are present. Finally, the best known lower bound on the number of combinatorial changes of the DT is $\Omega(n^2)$, whereas our upper bound is $O(n^3\beta(n))$. Given this upper bound, the algorithms presented here for maintaining the DT, the closest pair and disk connectivity are not efficient; it would be of interest to find kinetic data structures that solve these problems efficiently, or prove a tighter lower or upper bound on the number of combinatorial changes of the DT.

Acknowledgments

The author wishes to thank Leonidas J. Guibas, Olaf Hall-Holt, Aristides Gionis and Natasha Gelfand for useful discussions.

References

1. J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 1:1–28, 1999.
2. M. Gavrilova and J. Rokne. Swap conditions for dynamic Voronoi diagrams for circles and line segments. *CAGD*, 16:89–106, 1999.
3. L. J. Guibas, J. Hershberger, S. Suri, and L. Zhang. Kinetic connectivity for unit disks. In *Proc. 16th ACM Symp. on Computat. Geom.*, pages 331–339, 2000.
4. L. J. Guibas, J. S. B. Mitchell, and T. Roos. Voronoi diagrams of moving points in the plane. In G. Schmidt and R. Berghammer, editors, *Proc. 17th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 570 of *LNCS*, pages 113–125. Springer, 1991.
5. L. J. Guibas, J. Snoeyink, and L. Zhang. Compact Voronoi diagrams for moving convex polygons. In Magnús M. Halldórsson, editor, *Proc. 7th SWAT*, volume 1851 of *LNCS*, pages 339–352. Springer, 2000.
6. L. J. Guibas and L. Zhang. Euclidean proximity and power diagram. In *Proc. 10th CCCG*, 1998.
7. J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. In *Proc. 30th ACM STOC*, pages 79–89, 1998.
8. H. Imai, M. Iri, and K. Murota. Voronoi diagram in the Laguerre geometry and its applications. *SIAM J. Comput.*, 14(1):93–105, 1985.
9. M. I. Karavelas and L. J. Guibas. Static and kinetic geometric spanners with applications. In *Proc. 12th ACM-SIAM SODA*, pages 168–176, 2001.
10. J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
11. M. C. Lin and J. F. Canny. Effi cient algorithms for incremental distance computation. In *Proc. IEEE Intern. Conf. Robot. Autom.*, volume 2, pages 1008–1014, 1991.
12. M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM J. of Comput.*, 14(2):448–468, May 1985.