Conic Nearest Neighbor Queries and Approximate Voronoi Diagrams*

Stefan Funke† Theocharis Malamatos‡ Domagoj Matijevic§ Nicola Wolpert¶

April 4, 2014

Abstract

Given a cone C and a set S of n points in $\mathbb{R}^d$, we want to preprocess S into a data structure so that we can find fast an approximate nearest neighbor to a query point q with respect to the points of S contained in the translation of C with apex at q.

We develop an approximate conic Voronoi diagram of $\widetilde{O}(n/\varepsilon^d)$ size that supports conic nearest neighbor queries in $O(\log(n/\varepsilon))$ time. Our preprocessing uses only the well-separated pair decomposition and a compressed quadtree. Previous results were restricted to simplicial cones and achieved polylogarithmic or higher query times.

By increasing space to $\widetilde{O}(n/\varepsilon^{2d})$ our data structure further supports queries for any cone direction and angle.

1 Introduction

Answering nearest neighbor queries for a given set S of n points in an Euclidean space of fixed dimension is a classic problem in computational geometry, and many methods have been proposed to solve it. A natural structure to solve the nearest neighbor problem is the *Voronoi diagram* $\mathcal{VD}(S)$. $\mathcal{VD}(S)$ is a decomposition of $\mathbb{R}^d$ into *Voronoi cells* such that for the cell $V(p,S)$ of a point $p \in S$ we have $V(p,S) = \{x \in \mathbb{R}^d \mid d(x,p) \leq d(x,p') \text{ for all } p' \in S\}$. Unfortunately, the Voronoi diagram has worst-case size $\Theta(n^{\lceil d/2 \rceil})$, so its use to answer nearest neighbor queries in dimensions $d > 2$ is costly. Clarkson [11] used the Voronoi diagram and showed how to answer nearest neighbor queries in $O(\log n)$ time and $O(n^{O(d)})$ space. Later Meiser [20] managed to reduce the dependence on dimension in the query time from exponential to polynomial. Despite much research on the problem no other structures of size $O(n^{1+\delta})$ for a small $\delta > 0$ are known that answer queries in polylogarithmic time.

The lack of methods that guarantee efficient query times with near linear space has led to the introduction of the notion of *approximate nearest neighbors*. For a query point q and an arbitrary small constant $\varepsilon > 0$, a point $p \in S$ is an ε-approximate nearest neighbor to q if $d(q,p) \leq (1+\varepsilon) \cdot d(q,p')$ for all $p' \in S$. Not insisting on the *exact* nearest neighbor has allowed for data structures of linear size and logarithmic query time [7, 15]. Similarly, the notion of an *approximate Voronoi diagram* or $\mathcal{AVD}$ has allowed for space decompositions of near-linear size

*Preliminary results of the paper appeared in *Proc. 18th Canad. Conf. Comput. Geom.*, 2006.

†Institute for Formal Methods of Informatics, Stuttgart University, Universitatsstr. 38, 70569 Stuttgart, Germany. Email: Stefan.Funke@informatik.uni-stuttgart.de

‡Department of Informatics and Telecommunications, University of Peloponnese, Terma Karaiskaki, 22100 Tripoli, Greece. Email: tmalamat@uop.gr

§Department of Mathematics, J.J. Strossmayer University, 31000 Osijek, Croatia. Email: domagoj@mathos.hr

¶Faculty of Geomatics, Computer Science and Mathematics, Stuttgart University of Applied Sciences, Schellingstr. 24, 70174 Stuttgart, Germany. Email: nicola.wolpert@hft-stuttgart.de

with substantially improved query times with respect to ε-dependence, e.g., by Har-Peled [18] and Arya and Malamatos [5]. Here each cell u of this decomposition is assigned a point $p_u \in S$ such that for all $q \in u$, p_u is an approximate nearest neighbor for q. Assigning extra points to each cell leads further to space-time tradeoffs [3, 4, 6].

In our paper we consider a special version of nearest neighbor searching, which we will refer to as the *conic nearest neighbor ($\triangleleft$NN)* problem: given a cone C we want to preprocess S so that for any query point $q \in \mathbb{R}^d$ we can determine a nearest neighbor to q among the points of S contained in cone C with apex at q.

1.1 Related Work

The broad applications of answering $\triangleleft$NN queries were our main motivation to work on this problem. Yao [24] first introduced the problem and used $\triangleleft$NN queries to construct in subquadratic time the Euclidean minimum spanning tree (EMST) of a set S of n points in $\mathbb{R}^d$. Czumaj et al. [13] gave an algorithm that estimates the weight of an EMST with high probability to within a $(1+\epsilon)$ factor in sublinear time. Their algorithm assumes that the input S is supported by an efficient data structure for answering approximate $\triangleleft$NN queries. Arya, Mount and Smid [8] presented a method for solving $\triangleleft$NN queries based on nested range trees in polylogarithmic time and used it in maintaining geometric spanners. In the context of motion planning, Clarkson [10] considered the conic Voronoi diagram in two dimensions and showed how it can be built by a simple sweep-line algorithm. Agarwal, Arge and Erickson [1] presented a method for answering an approximate nearest neighbor query among moving points in any fixed dimension d by a reduction to a fixed number of $\triangleleft$NN queries. Further important applications for $\triangleleft$NN queries stem from problems in surface reconstruction, analysis of point cloud data, and dimension detection for manifolds of unknown dimension (see e.g., Funke and Ramos [16], Dey et al. [14], and Giesen and Wagner [17]). In particular, Funke and Ramos [16] used the well-separated pair decomposition to determine in three dimensions a $\triangleleft$NN to each point of S in $O(n \log n)$ time. Finally, in a related constrained geometric search problem, Aronov et al. [2] present a data structure of $O(n \log^3 n)$ size that given a query point q and a halfplane h reports the nearest or farthest point to q but only among the points of $S \cap h$ in $O(\log n)$ time.

1.2 Our Contribution

Given a fixed set S of n points, a cone C of fixed direction and angle, and a fixed approximation factor $\varepsilon > 0$ we show how to construct an approximate conic Voronoi diagram or $\mathcal{ACVD}$ of size $O((n/\varepsilon^d)\log(1/\varepsilon))$ that can be used to answer approximate $\triangleleft$NN queries in $O(\log(n/\varepsilon))$ time. Preprocessing time is $O((n/\varepsilon^d)\log(n/\varepsilon)\log(1/\varepsilon))$. Our result, presented in Sec. 3, assumes standard d-dimensional cones but it extends also to simplicial cones. (See Sec. 2.1 for the definition of a standard cone and see, for example, [22] for the definition of a simplicial cone.) Note that all previous results assumed only simplicial cones. Following ideas from approximate Voronoi diagrams, we partition the space into cells such that each cell u has a representative point p_u associated with it. For any point $q \in u$ and the cone C with apex at q, the representative point p_u will have the following properties:

(i) the distance from q to p_u could be slightly larger than the distance from q to its exact $\triangleleft$NN (by at most a factor of $1+\varepsilon$),

(ii) p_u lies either in the cone C or slightly outside of C (by an angle of at most ε).

We give a short overview of the construction. First the cells of the $\mathcal{ACVD}$ are generated and stored in quadtree-like data structure for fast access. Then each cell u is assigned an appropriate

point p_u satisfying the above two requirements. This is achieved by an original top-down method of propagation of representatives. By contrast, the constructions for $\mathcal{AVD}$s in order to select and assign efficiently representatives to the cells had to resort to an existing, separate approximate nearest neighbor search structure. That complicates the constructions and was clearly not an option in our problem. After presenting the construction of an $\mathcal{ACVD}$, we show how the processing of queries is performed and prove correctness. In addition we show that the fixed direction and fixed angle cone restriction can be easily removed with an increase in space by a factor of $O(1/\varepsilon^d)$. We conclude with open problems for future research in Sec. 4.

2 Preliminaries

2.1 Approximate Conic Nearest Neighbors

Let f be a point in $\mathbb{R}^d$ which is different from the origin, and let θ be an angle with $0 < \theta < 2\pi$. Let p be any point in $\mathbb{R}^d$. We define the *cone* with apex p, direction f, and angle θ to be the set of points

$$C(p, f, \theta) = \{q \in \mathbb{R}^d \backslash \{p\} : \angle(q - p, f) \leq \theta/2\},$$

where $\angle(x, y)$ is the angle between two vectors $\vec{x}$ and $\vec{y}$. Given a real parameter $0 < \varepsilon \leq 1$ we define the *expanded cone* with apex p, direction f, and angle θ to be the set of points

$$C^{\varepsilon}(p, f, \theta) = \{q \in \mathbb{R}^d \backslash \{p\} : \angle(q - p, f) \leq \theta/2 + \varepsilon\}.$$

Note that $\varepsilon < \pi/3$. The *reverse cone* and the *reverse expanded cone* with apex p are defined to be $C_{\text{rev}}(p, f, \theta) = C(p, -f, \theta)$ and $C^{\varepsilon}_{\text{rev}}(p, f, \theta) = C^{\varepsilon}(p, -f, \theta)$ respectively. Further on we assume f and θ are fixed and we omit the last two parameters in all cone definitions. In the figures, where needed we set the vector f implicitly to have the positive direction of the x-axis.

Let S be a set of points in $\mathbb{R}^d$, and let $0 < \varepsilon \leq 1$ be the approximation factor. Let q, x be two points in $\mathbb{R}^d$, and let $d(q, x)$ or $\|qx\|$ denote the Euclidean distance between q and x. Given q and a set of points X let $d(q, X)$ be the distance of a nearest point to q among the points in X. Given f and θ, we say that a point p in S is a *conic nearest neighbor ($\triangleleft$NN)* to a query point q if $d(q, p) = d(q, S \cap C_{\text{rev}}(q))$, that is, p is a nearest neighbor to q among the points in $C_{\text{rev}}(q)$. Using $C_{\text{rev}}(q)$ instead of $C(q)$ in the definition of $\triangleleft$NN simplifies the description of our data structure later. We say that a point p in S is an *ε-approximate conic nearest neighbor (ε-$\triangleleft$NN)* to a query point q if $d(q, p) \leq (1 + \varepsilon) \cdot d(q, S \cap C_{\text{rev}}(q))$ and p lies in $C^{\varepsilon}_{\text{rev}}(q)$. We will refer to the first condition of the above definition as the *distance constraint* and to the second condition as the *angle constraint*. Note that if p does not lie in $C_{\text{rev}}(q)$ then it is possible that $d(q, p) < d(q, S \cap C_{\text{rev}}(q))$. Also, if $S \cap C_{\text{rev}}(q) = \emptyset$ and $S \cap C^{\varepsilon}_{\text{rev}}(q) \neq \emptyset$, we define any point in $S \cap C^{\varepsilon}_{\text{rev}}(q)$ to be an ε-$\triangleleft$NN to q.

Our data structure for answering ε-$\triangleleft$NN queries is a d-dimensional compressed quadtree constructed with the help of a well-separated pair decomposition, similarly to [5, 6]. We define and we give the main properties of well-separated pair decompositions and compressed quadtrees below.

2.2 Well-Separated Pair Decompositions

Two sets of points X and Y in $\mathbb{R}^d$ are *well-separated* if there exist two disjoint balls b_X and b_Y of radius r, such that $X \subset b_X$ and $Y \subset b_Y$ and the distance between the centers of the balls is at least $\alpha \cdot r$, where $\alpha > 2$ is a real number. The balls are called the *heads* of the pair, the distance between the centers of the heads is called the *length* of the pair, and α is called the *separation factor*. Also the midpoint of the two centers is called the *center* of the pair.

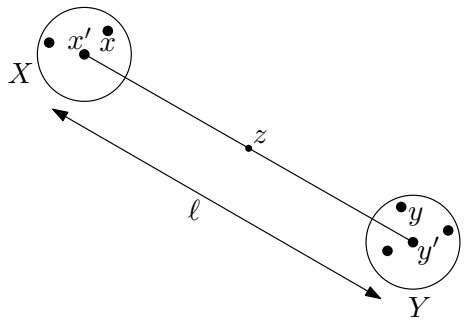

Fig. 1: A well-separated pair, its two heads X and Y, its center z, its two center points x' and y' at distance ℓ, and two arbitrary points x and y, one in each head.

Let S be a set of n points in $\mathbb{R}^d$. A *well-separated pair decomposition* (*WSPD*) of S is a set

$$\mathcal{P} \;=\; \{(X_1, Y_1), \cdots, (X_m, Y_m)\}$$

such that

(i) for any $1 \leq i \leq m$, $X_i, Y_i \subset S$,

(ii) for any $1 \leq i \leq m$, X_i and Y_i are well separated (for the same separation factor α), and

(iii) for any two different points $x, y \in S$, there is a unique pair (X_i, Y_i) with $x \in X_i$ and $y \in Y_i$ or vice versa.

We say that the pair (X_i, Y_i) *separates* x and y.

Callahan and Kosaraju [9] showed that there exists a WSPD that has only $O(\alpha^d n)$ pairs and gave a method to construct it in $O(n \log n + \alpha^d n)$ time. We note that their construction provides also the corresponding heads of each pair.

In our construction of an approximate conic Voronoi diagram the separation factor α is a constant greater than four and independent of ε. A value $\alpha > 4$ implies that for any pair, any two points in different heads have strictly greater distance than any two points in the same head. We also use a special WSPD with the property that at the center of each head lies a point from the head, called the *center point*, which is stored with the corresponding pair.

Lemma 2.1 *Let a WSPD $\mathcal{P}$ with separation factor $\alpha > 6$. Then $\mathcal{P}$ can be converted into a special WSPD as defined above with separation factor $\alpha/2 - 1$.*

Proof: It suffices to determine which are the center points, the heads, and the separation factor for each pair in the special WSPD. For each head of a pair in $\mathcal{P}$, we pick any point associated with it as the center point. Let r be the radius of the two heads associated with the pair. We select as new heads the balls which are centered at the two center points and have radius $r' = 2r$. Clearly because $\alpha > 6$ these balls are disjoint and each contains the same points as the balls of the pair. Since all points in a head are at distance at most r from its center the distance between the center points is clearly at least $\ell - 2r$ and thus the separation factor of the new pair is at least $(\ell - 2r)/r' \geq (\alpha r - 2r)/2r \geq a/2 - 1 > 2$, which completes the proof. □

The use of the WSPD with the center points will help us in some of our subsequent proofs. Using the triangle inequality we can easily show the following additional properties, which are taken from [6]: (See Fig. 1.)

Lemma 2.2 *Consider a well-separated pair $Z = (X, Y)$ of a WSPD with separation factor $\alpha > 4$. Let ℓ be the distance between the two center points of Z and let z be the center of Z. Then for any $x \in X$ and $y \in Y$ and $p \in X \cup Y$ we have:*

(*i*) $\ell/2 < \|xy\| < 3\ell/2$.

(*ii*) $\ell/4 < \|pz\| < 3\ell/4$.

2.3 Compressed Quadtrees

Let $\mathbb{U}^d = [0,1]^d$ denote a unit hypercube in $\mathbb{R}^d$. A quadtree is a rooted tree that represents a hierarchical partition of space into quadtree boxes. A quadtree box is either the hypercube $\mathbb{U}^d$ or is one of the hypercubes obtained by splitting a quadtree box into 2^d equal parts. We define the *size* of a quadtree box to be the side length of its corresponding hypercube. Each node of the quadtree is associated with a quadtree box. If a quadtree box is obtained by the splitting of another quadtree box as described above then the corresponding nodes in the quadtree are associated with a child-parent relation and they are put at successive levels in the quadtree. The root of the quadtree is associated with $\mathbb{U}^d$. An internal node is associated with a quadtree box that is split. Each internal node has 2^d children and there is a 1-1 correspondence between these children and the 2^d quadtree boxes produced by splitting the internal node. A leaf is associated with a quadtree box that is not split. Note that the quadtree has two types of nodes: internal nodes that correspond to quadtree boxes that are split and leaves that correspond to quadtree boxes where the splitting stops.

Given a set of quadtree boxes it is easy to see that we can recursively construct the minimum quadtree having these quadtree boxes as leaves. However if a quadtree box is arbitrarily small then the size of the quadtree can become arbitrary large. This can be fixed by using a compressed quadtree. A compressed quadtree [19] is a quadtree with two extra types of nodes. There is a second type of internal node that has only two children. Let u be the quadtree box associated with such an internal node. The one of the two children of this node is associated with a quadtree box v that is contained inside u. Its other child is associated with the difference $u \backslash v$ and it forms a second type of leaf. The advantage of a compressed tree is that given u and $v \subset u$ one level suffices to reach the node for v from the node for u whereas in a non-compressed tree several levels and successive splitting might be required.

We call *cell* the region of space associated with any type of node of a compressed tree. Note that a cell corresponds to the difference between a quadtree box, called the *outer* box, and a second quadtree box, contained in the outer box, called the *inner* box. The inner box is allowed to be nil. Throughout we use the following notation. Given a cell w, let s_w denote the size of the outer box of w, let $r_w = s_w d$, and let b_w be the ball of radius r_w whose center coincides with the center of w's outer box. Clearly, it follows that $b_w \supseteq w$. Note that all the leaves of a compressed tree are disjoint and the cells associated with nodes in the same downward path are nested within each other. See Fig. 2.

A compressed quadtree for a set $\mathcal{U}$ of m quadtree boxes is guaranteed to have size $O(m)$. Its height however could be $\Omega(m)$. It can be reduced to $O(\log m)$ by a standard method that balances the tree with the help of separator nodes. We give below the properties of the compressed tree that we will need in our data structure. For their proof see [19].

Lemma 2.3

(i) Given a collection $\mathcal{U}$ of m quadtree boxes in $\mathbb{U}^d$, we can construct in $O(m \log m)$ time a compressed quadtree with $O(m)$ nodes such that each quadtree box in $\mathcal{U}$ is a cell of this structure.

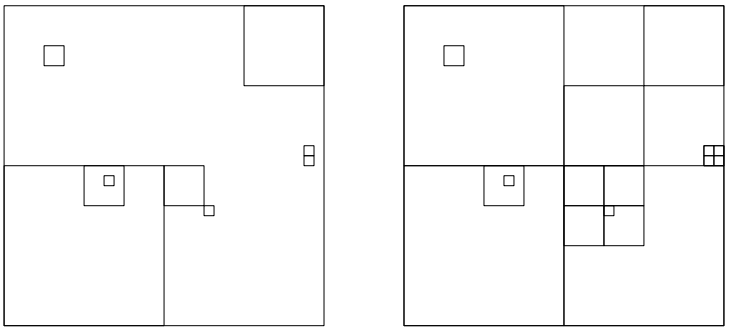

Fig. 2: (a) A set of quadtree boxes and (b) the cells of the compressed tree for this set.

(ii) Given the structure of (i), we can preprocess it in a second structure of $O(m)$ size such that we can determine the leaf cell of the structure (i) containing any query point $q \in \mathbb{R}^d$ in $O(\log m)$ time.

For the case (ii) above we note that when q lies on the boundary of more than one cell, any of them is assumed to contain q and a single leaf is always returned. Also, the above bound on construction time holds under the assumption that certain bit operations take constant time. To avoid this, we could alternatively use the *balanced box-decomposition tree* [7].

3 Approximate Conic Voronoi Diagram

Let S be an n-element point set in $\mathbb{R}^d$ and let $0 < \varepsilon \leq 1$ be the approximation factor. In this section we give the construction of the data structure for answering approximate conic nearest neighbor queries over S and the associated approximate conic Voronoi diagram. Our construction uses ideas from the construction of approximate Voronoi diagram in [5]. The construction of an $\mathcal{ACVD}$ is more complex than that of an $\mathcal{AVD}$. Intuitively this is due to the fact that nearby query points have always nearest neighbors at similar distances while it is possible that they have conic nearest neighbors that are very far apart.

We assume that S has been scaled to lie within a ball of diameter $\varepsilon/17$ at the center of the unit hypercube. We first show how to answer in constant time a query for a point q that lies outside the unit hypercube. Select any point p of S and check whether the query point q lies in $C^{\varepsilon}(p)$. If it does then p is an ε-$\lhd$NN for q otherwise there is no $\lhd$NN for q. In other words, the intersection of the complement of the unit hypercube and $C^{\varepsilon}(p)$ is an approximate conic Voronoi cell associated with p. Correctness will follow from Lemma 3.2.

Before Lemma 3.2, we prove the following auxiliary lemma. Roughly, it states that any point of S close to a conic nearest neighbor of a query point q is also an approximate conic nearest neighbor of q. Formally, we have:

Lemma 3.1 *Let $0 < \varepsilon \leq 1$ be a real parameter and let S be a set of points in $\mathbb{R}^d$. Let q be any point in $\mathbb{R}^d$ that has a $\lhd$NN with respect to S and let x be a $\lhd$NN for q with $d(q,x) = D$. Let b be a ball of diameter $\varepsilon D/4$ centered at x. Then any point in $S \cap b$ is an ε-$\lhd$NN for q.*

Proof: Let x' be any point in $S \cap b$. Using the triangle inequality we have $\|qx'\| \leq \|qx\| + \|xx'\| \leq \|qx\| + \varepsilon D/8 \leq (1 + \varepsilon/8)\|qx\|$. This means x' satisfies the distance constraint in the definition of an ε-$\lhd$NN. We next show that the angle $\angle xqx' = \phi$ is at most ε. Clearly ϕ is maximized when the segment qx' is tangent to the ball b. It follows that $\sin\phi \leq (\varepsilon D/8)/\|qx\| \leq \varepsilon/8$. Since $\varepsilon \leq 1$ and $\sin(\pi/6) = 1/2 > \varepsilon/8$ we have $\sin\phi \leq 1/2$ and $\phi < \pi/6$. It is easy to see then that

$(3/\pi)\phi < \sin\phi$ and thus $(3/\pi)\phi < \varepsilon/8$. Therefore $\phi < \varepsilon/7 < \varepsilon$ and the angle constraint is satisfied as well. □

Lemma 3.2 *Let $0 < \varepsilon \leq 1$ be a real parameter, let b' be a ball of diameter $\varepsilon/17$ located at the center of the unit hypercube $\mathbb{U}^d$. Given a point set $S \subset b'$, and a point q in the complement of the unit hypercube that has a $\lhd$NN with respect to S then any point of S is an ε-$\lhd$NN for q.*

Proof: Let x be a $\lhd$NN of q and let $\|qx\| = D$. Since $D > 1/2 - (\varepsilon/17)/2$ it follows that the ball b centered at x with radius $\varepsilon D/8 > \varepsilon(1/2 - 1/34)/8 \geq \varepsilon/17$ contains ball b' and thus by Lemma 3.1 any point in S is an ε-$\lhd$NN of q. □

It remains to construct the approximate Voronoi cells for the unit hypercube. These cells will be the leaf cells of a compressed quadtree. The compressed quadtree is also the data structure that we will use for answering any ε-$\lhd$NN query with a point that lies within the unit hypercube. Next we show how to build this data structure.

3.1 Construction of the Data Structure

We present the construction algorithm for the desired compressed quadtree T. Our construction depends on the approximation factor ε, and the constants c_1, c_2 and c_3 whose specific values will be determined later. We define $L(s) = 2^{\lfloor \log s \rfloor}$. We use the function L to describe the size of quadtree boxes. Note that $s/2 < L(s) \leq s$.

We give a short overview of the construction. We first generate a number of quadtree boxes based on a WSPD for S and then assign to each quadtree box up to two points from S. These points are potential ε-$\lhd$NNs for the points lying in each quadtree box. If two points are assigned, one point will be relatively close or inside the quadtree box. The other point could be quite far from the quadtree box and it could be an answer for points of the quadtree box that they do not lie in the expanded cone of the previous point. Then all the quadtree boxes generated are processed into a compressed quadtree and a top-down procedure assigns to each leaf the necessary points.

We begin by computing the well-separated pair decomposition with center points $\mathcal{P}$ for point set S using any constant separation factor $\alpha > 4$. For a fixed well-separated pair $Z \in \mathcal{P}$, let ℓ denote its length and z denote its center. Let x' and y' denote the two center points stored with this pair. Next, we compute a set of quadtree boxes $\mathcal{U}(Z)$ as follows.

Let $C^\varepsilon(x')$ and $C^\varepsilon(y')$ be the two expanded cones with apexes x' and y' respectively. For $0 \leq i \leq \lceil \log(c_1/\varepsilon) \rceil$, let $b_i(Z)$ denote the ball centered at z of radius $r_i = 2^i \ell$. Also let $b_{-1}(Z) = \emptyset$. Let $\mathcal{B}(Z)$ denote the resulting set of balls. These balls involve radius values ranging from ℓ to $\Theta(\ell/\varepsilon)$. For each such ball $b_i(Z)$, let $\mathcal{U}_i(Z)$ be the set of quadtree boxes of size $L(\varepsilon r_i/c_2)$ that overlap the annulus $b_i(Z) \backslash b_{i-1}(Z)$ and at least one of the cones $C^\varepsilon(x')$ and $C^\varepsilon(y')$ (see Fig. 3). Let $\mathcal{U}(Z)$ denote the union of all these quadtree boxes over all the $O(\log(1/\varepsilon))$ values of i.

With each quadtree box v we store either point x' or point y' or both. We call these points as *representatives* for v because they are potential ε-$\lhd$NNs for queries lying within v. Below we use z' to denote either of the points x' or y'. We say that a quadtree box v is *covered* by $C^\varepsilon(z')$ if v is contained inside $C^\varepsilon(z')$. We say that a quadtree quadtree box v is *stabbed* by $C^\varepsilon(z')$ if v intersects $C^\varepsilon(z')$ but v is not covered by $C^\varepsilon(z')$.

We decide which representatives to store with v depending on the cones that cover or stab it. We call a point z' that is stored with v a *far* representative if $C^\varepsilon(z')$ covers v and a *close* representative if $C^\varepsilon(z')$ stabs v. We distinguish two cases which we examine separately for $z' = x'$ and $z' = y'$:

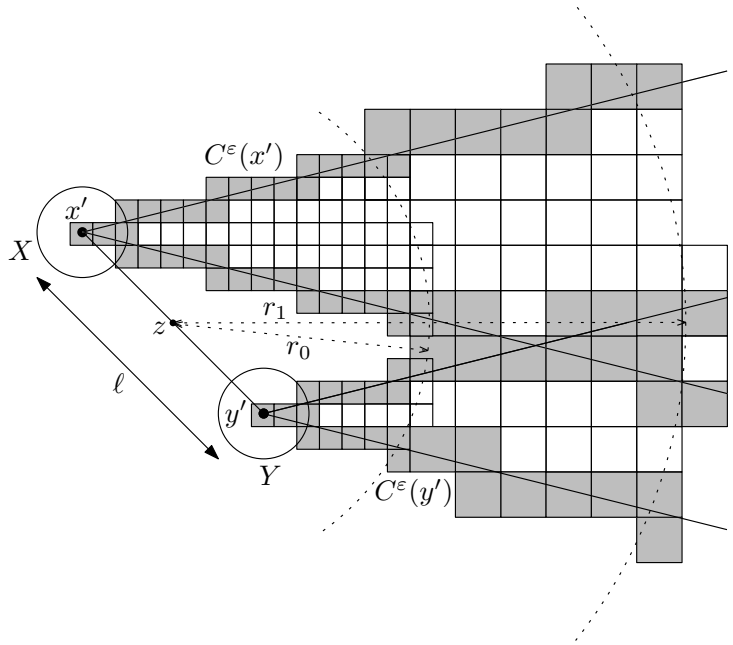

Fig. 3: Construction. The quadtree boxes generated for $i = 0, 1$ are only shown. The stabbed quadtree boxes are gray.

(i) v is covered by $C^\varepsilon(z')$. In this case store z' with v as a far representative.
(ii) v is stabbed by $C^\varepsilon(z')$. If $d(z', v) \leq c_3\ell$ then store z' with v as a close representative.

For case (ii) we note that if a quadtree box v is relatively far from z', $C^\varepsilon(z')$ may stab v but $C(z')$ may not stab it. Intuitively this justifies our choice in this case to store z' only for quadtree boxes near z'.

As a result of the above procedure quadtree box v may store up to two representatives. If the quadtree box v has stored two far representatives, that is both x' and y', then we keep only one far representative, the one closest to the center of v (ties broken arbitrarily). We do similarly if the quadtree box v has stored two close representatives. (Note that this case occurs only for some values of c_3.) See Fig. 4 for an example where the quadtree box v stores both a

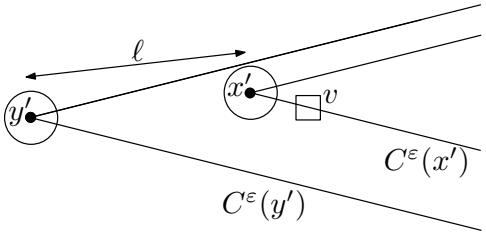

Fig. 4: A case where the quadtree box v stores two representatives.

far and a close representative (assuming c_3 is chosen large enough). Note that given v, $C^\varepsilon(x')$ and $C^\varepsilon(y')$, we can easily check which of the above cases holds in constant time. (Note that this checking implicitly assumes that we can compute function cos in constant time. To remove this assumption we can simply define the cones based on the inner product of vectors.)

This process is performed for each well-separated pair of $\mathcal{P}$. Observe that one quadtree box could have been generated by more than one well-separated pair and each instance of the same quadtree box could store different representatives. However for any quadtree box v, we always

keep at most two representatives. From all far representatives encountered for v (if any) we keep a far representative that is closest to the center of v, and from all close representatives encountered for v (if any) we keep a close representative that is closest to the center of v. If at the end of this process a quadtree box v stores both a far and a close representative, but the far representative for any point of v is not farther than the close representative, we remove the close representative from v. (It is relatively easy to see that this checking can be reduced to solving a linear program, for the case of simplicial cone, and to solving a quadratic program, for the case of standard cone. And because both programs are of fixed size, it can be completed in constant time.)

Let $\mathcal{U} = \bigcup_{Z \in \mathcal{P}} \mathcal{U}(Z)$ denote the union of all the quadtree boxes. If no point is stored with a quadtree box v in $\mathcal{U}$ we remove it from $\mathcal{U}$. Then we apply Lemma 2.3 (i) to construct a compressed tree T storing all these quadtree boxes.

3.1.1 Propagation and Assignment of Representatives

To finish the construction it remains to assign representatives to the leaves of T. To do this we use the representatives stored with the quadtree boxes as follows.

First each node in T associated with an outer box in $\mathcal{U}$ receives the representatives stored with this quadtree box. Then starting from the root of the tree we propagate the far representative stored with each node (if present) down to all of its children. Each node in T checks if the far representative stored with its parent (if any) is closer to the center of its outer box than its current far representative (if any). If it is closer or if it has no far representative, the node inherits the far representative from its parent. The same procedure continues recursively to the children of the node until we reach the leaves.

Next we propagate down the close representatives again in a top-down manner as follows. If a node has a close representative it passes it down to its children. If the child node has already a close representative stored with it, no change is made. If it does not have a close representative and the expanded cone with apex the close representative intersects its associated outer box, it stores this representative. Note that the close representative inherited from a parent may cover the outer box, which turns it into a far representative for this node. In this case we check if the node has a far representative and if so we keep the one that is closest to the center of its outer box. The propagation down the T of the close representatives, as well as of any far representatives that resulted from them, continues recursively until we reach the leaves.

The above two propagation stages ensure that each leaf of T has been assigned the required at most two representatives. The example in Fig. 5 illustrates the usefulness of propagation. We note that the balancing of T's height as described in [19] follows just after the propagation and that at the end we only keep the representatives stored in the leaves. This completes the construction of the compressed tree.

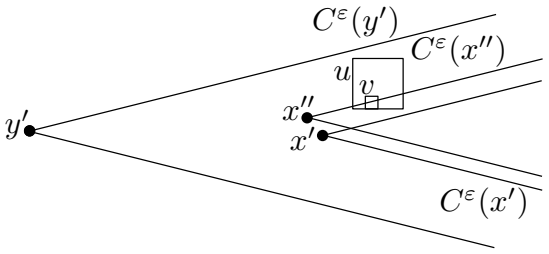

Fig. 5: Quadtree box v is generated by pair $(\{x'\}, \{x''\})$ and u by pair $(\{x', x''\}, \{y'\})$. Assume that x' and y' are the center points of the second pair. As needed y' is stored by propagation in v.

3.1.2 Space and Preprocessing Time

Lemma 3.3 *Let $m = n\log(1/\varepsilon)/\varepsilon^d$. The construction of the compressed quadtree requires $O(m\log m) = \widetilde{O}(n/\varepsilon^d)$ time and occupies $O(m)$ space. The compressed quadtree has $O(m)$ nodes and $O(m)$ representatives in total.*

Proof: The construction of a WSPD for S takes $O(n\log n)$ time. For each well-separated pair $Z \in \mathcal{P}$ and for each i, we first consider the number of quadtree boxes that overlap the annulus $b_i(Z)\backslash b_{i-1}(Z)$ in the construction. This number is bounded by the number of quadtree boxes that overlap the ball $b_i(Z)$ of radius r_i. Since the quadtree boxes are of size $L(\varepsilon r_i/c_2)$, by a simple packing argument the number of overlapping quadtree boxes is

$$O\left(\left(1+\frac{r_i}{\varepsilon r_i/c_2}\right)^d\right) \;=\; O((1/\varepsilon)^d).$$

Since the number of balls for each well-separated pair is $O(\log(1/\varepsilon))$, the total number of quadtree boxes in $\mathcal{U}(Z)$ is $O((1/\varepsilon)^d\log(1/\varepsilon))$. The total number of well-separated pairs is $O(n)$, and thus $|\mathcal{U}| = O((n/\varepsilon^d)\log(1/\varepsilon)) = O(m)$.

By standard results [23], we can define a linear ordering on all quadtree boxes. For example we can order them according to the lexicographic order of the coordinates of their center points. This allows us to check and locate if a quadtree box has been generated before in $O(\log m)$ time per quadtree box [12]. Thus we can maintain for each quadtree box the required at most two representatives in overall time $O(m\log m)$. By Lemma 2.3 (i) of the compressed trees the number of nodes of T is $O(|\mathcal{U}|) = O(m)$, and it can be constructed in time $O(m\log m)$. Using inequality $x \geq \ln(x+1)$ for $x = 1/\varepsilon$, we have $O(\log(n/(\varepsilon^d)\log(1/\varepsilon))) = O(\log(n/\varepsilon) + \log\log(1/\varepsilon)) = O(\log(n/\varepsilon))$. Thus $O(m\log m) = O((n/\varepsilon^d)\log(n/\varepsilon)\log(1/\varepsilon))$.

The assignment of the representatives to the leaves of T involves two level order traversals and clearly takes only $O(m)$ time. The space requirement for T is similarly $O(m)$. □

3.2 Query Answering

Next we will show how to answer an ε-$\lhd$NN query using T and prove correctness. Let q be a query point. To find an ε-$\lhd$NN for q, we first locate the leaf cell w of tree T that contains q. By Lemma 2.3 (ii), this takes time $O(\log m)$ which as shown above is $O(\log(n/\varepsilon))$.

If w has no representatives stored with it, we report none. If w has just one representative x, we check whether q is contained in $C^\varepsilon(x)$, and if so we report this single representative as an answer, otherwise we report none. If w has two representatives, a close x and a far y, we report y unless $C^\varepsilon(x)$ contains q and x is closer to q than y in which case we report x. Total query time is clearly $O(\log(n/\varepsilon))$.

Suppose that q does not have a $\lhd$NN. If we have reported some representative as an answer then clearly this representative satisfies the angle constraint of an ε-$\lhd$NN and thus by definition the answer is correct. Assume now that q has a $\lhd$NN. Let x be a $\lhd$NN of q and let $d(q,x) = D$. Let b be a ball of diameter $\varepsilon D/4$ centered at x.

3.2.1 Case $S \subseteq b$

We consider first the case where all points of S lie within b. In this case by Lemma 3.2 any point in S is an ε-$\lhd$NN to q. Thus it suffices to show that at least one representative was stored with leaf cell w.

Observe that b must lie inside the ball of diameter $\varepsilon/17$ containing S. Clearly, this implies $D \leq 4/17$. Without loss of generality we may assume that the diameter of S is at least

$\varepsilon/(2 \cdot 17)$. Let y be the farthest point in S from x according only to the Euclidean distance. We have $\|yx\| \geq \varepsilon/(4 \cdot 17)$, Consider the pair Z in $\mathcal{P}$ that separates points x and y. Let x' and y' be the corresponding two center points. Note that by Lemma 3.2 both $C^{\varepsilon}(x')$ and $C^{\varepsilon}(y')$ contain q. Let ℓ denote the length of the pair and let z denote its center.

By Lemma 2.2, we have $\|yx\| < 3\ell/2$. Combining the two inequalities for $\|yx\|$, we get $\ell > (2/51)\varepsilon$. We next show two different upper bounds for $\|zq\|$. By triangle inequality and Lemma 2.2, $\|zq\| \leq \|zx\| + \|xq\| \leq 3\ell/4 + D$. Using the above inequalities for D and ℓ/ε, we get $\|zq\| < \ell + 4/17 < \ell/\varepsilon + 6\ell/\varepsilon \leq 7\ell/\varepsilon$. Clearly $\ell < \varepsilon D/4 < D$, since the center points of the pair Z lie in b. Thus we also have $\|zq\| < \ell + D < 2D$. In summary, $\|zq\| < \min(2D, 7\ell/\varepsilon)$.

This implies that for $c_1 > 7$, when we processed the pair Z a ball $b_i(Z)$ contained q, therefore a quadtree box v of size at most $\varepsilon(2D)/c_2$ and $r_v \leq d\varepsilon(2D)/c_2$ was generated. For $c_2 \geq 8d$, we have $s_v \leq \varepsilon D/(4d)$ and $r_v \leq \varepsilon D/4$. Lemma 3.4 below shows that $C^{\varepsilon}(x')$ must cover v.

Lemma 3.4 *Let $0 < \varepsilon \leq 1$ be a real parameter. Let q be any point in $\mathbb{R}^d$ contained in $C(x)$ and let $d(q, x) = D$. Let v be a cell that contains q and has size $s_v \leq \varepsilon D/(4d)$. Let x' be any point at distance at most $\varepsilon D/8$ from x. Then $v \subseteq C^{\varepsilon}(x')$.*

Proof: Let q' any point of v. It suffices to show that $\angle q'xq + \angle xqx' \leq \varepsilon$. See Fig. 6. By the proof of Lemma 3.1 we have $\angle xqx' < \varepsilon/7$. By triangle inequality $\|x'q\| \geq D - \|xx'\| \geq D - \varepsilon D/8$. Also $r_v = s_v d \leq \varepsilon D/4$. Clearly $\phi' = \angle q'x'q$ is maximized when the segment $q'x'$ is orthogonal to $q'q$. It follows that since $\varepsilon \leq 1$, $\sin\phi' \leq 2r_v/\|x'q\| \leq (\varepsilon D/2)/(D - \varepsilon D/8) \leq 4\varepsilon/7$. Since $\varepsilon \leq 1$ and $\sin(\pi/3) = \sqrt{3}/2 > 4\varepsilon/7$, we have $\phi' < \pi/3$. It is easy to see then that $(3\sqrt{3}/2\pi)\phi' < \sin\phi'$ and thus $0.8\phi' < 4\varepsilon/7$. Therefore $\phi' < 5\varepsilon/7$ and $\phi + \phi' < \varepsilon/7 + 5\varepsilon/7 < \varepsilon$ as desired. □

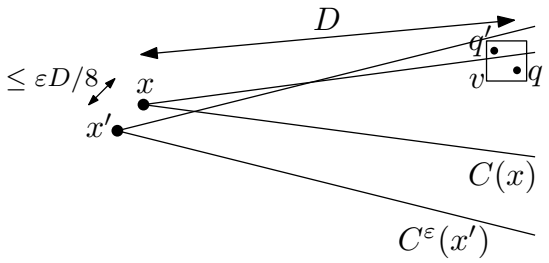

Fig. 6: Proof of Lemma 3.4.

The lemma implies that the quadtree box v stores at the end of construction a far representative, i.e., x' or some other point of S closer to its center. Clearly it follows that $w \subseteq v$, otherwise w would not be a leaf cell. Notice that in the tree T there must be an ancestor of the leaf associated with w which is associated with v. Therefore because of the propagation of representatives towards the leaves, leaf cell w must store a far representative, which completes the proof for the case $S \subseteq b$.

We now consider what happens if $S \cap b \neq S$. We will show that again w stores an ε-$\lhd$NN of q. Let $\overline{b}$ denote the set of points outside b. Let y be a point in $S \cap \overline{b}$ that is nearest to x. Consider the pair $Z \in \mathcal{P}$ that separates points x and y and let x' and y' be the corresponding center points of Z. Let ℓ denote the length of the pair and let z denote its center. By Lemma 2.2, we have $\|xz\| < \ell$ and $\ell/2 < \|xy\| < 2\ell$. We will make implicit or explicit use of these inequalities below. We will now show that $\|xx'\| \leq \varepsilon D/8$. Clearly if $x = x'$, this is true. If $x \neq x'$ and $\|xx'\| > \varepsilon D/8$ we have a contradiction since then point y should not have been chosen in the first place. By Lemma 3.1, x' is an ε-$\lhd$NN for q. This implies that $C^{\varepsilon}(x')$ contains q and it intersects cell w.

Based on the relationship between D and ℓ we distinguish two more cases, case two when $D \geq \ell/10$ and case three when $D < \ell/10$.

3.2.2 Case $D \geq \ell/10$

We start by showing an upper bound on the distance between z and q. We note that $\|xy\| > \varepsilon D/8$ and thus by Lemma 2.2 $\ell > 2\|xy\|/3 > \varepsilon D/12$ or equivalently $D < 12\ell/\varepsilon$. By triangle inequality, $\|zq\| \leq \|zx\| + \|xq\| \leq \ell + D < 13\ell/\varepsilon$. We have also $\|zq\| \leq \ell + D \leq 10D + D \leq 11D$. Thus $\|zq\| \leq \min(13\ell/\varepsilon, 11D)$.

This implies that for $c_1 > 13$ when we processed the pair Z a ball $b_i(Z)$ and the cone $C^\varepsilon(x')$ contained q and therefore a quadtree box v of size at most $\varepsilon(11D)/c_2$ and $r_v \leq d\varepsilon(11D)/c_2$ was generated. For $c_2 \geq 44d$, we have $s_v \leq \varepsilon D/(4d)$ and $r_v \leq \varepsilon D/4$.

By Lemma 3.4, $C^\varepsilon(x')$ covers v. Therefore the quadtree box v of size at most $\varepsilon D/(4d)$ during the construction considers x' to be stored as a far representative. Let D' be the distance from the center of quadtree box v to x'. We have $D' \leq \|x'x\| + D + r_v \leq \varepsilon D/8 + D + \varepsilon D/4 \leq D + 3\varepsilon D/8$. Clearly x' can be replaced in v only by a closer far representative (including y'). Thus the final far representative for quadtree box v will be at distance from the center of the leaf cell $w \subseteq v$, and from any other point in v, at most $D' + r_v$.

The associated node of v in T is an ancestor of the leaf associated with w. Because of propagation, the cell w receives and stores as a far representative some point p in S that is no farther from the center of w than $D' + r_v$. Thus, by triangle inequality

$$\|pq\| \leq (D' + r_v) + r_w \leq D' + 2r_v \leq D + 3\varepsilon D/8 + 2\varepsilon D/4 < (1+\varepsilon)D,$$

and representative p is an ε-$\triangleleft$NN for q.

3.2.3 Case $D < \ell/10$

We will now show that when $D < \ell/10$, w stores an ε-$\triangleleft$NN for q. The proof idea here is that the leaf cell w that contains q must have stored at least one representative from ball b given that all points outside b are relatively far from q.

Since y is a nearest point to x outside the ball b, we have $\|xy\| > \ell/2 > 5D$. By Lemma 3.1, any point inside b is an ε-$\triangleleft$NN for q. If one of them was stored with w either as a far or a close representative, the returned answer is correct. Next we will show that a point in b is always stored as a representative with w.

We show first that a quadtree box $v \in \mathcal{U}$ with $v \supseteq w$ stores an ε-$\triangleleft$NN for q. By the triangle inequality and since $D < \ell/10$, $\|zq\| \leq \|zx\| + \|xq\| \leq 3\ell/4 + D < \ell$. Since x' lies in b, q is contained in $C^\varepsilon(x')$. Clearly because $\|zq\| < \ell$, a quadtree box v containing q was generated of size at most $\varepsilon\ell/c_2$ and $r_v \leq d\varepsilon\ell/c_2$. For $c_2 \geq 20d$, $r_v \leq \varepsilon\ell/20$. Clearly $C^\varepsilon(x')$ stabs or covers v. Since $D < \ell/10$, $d(x', v) \leq \|qx\| + \|xx'\| \leq D + \varepsilon D/8 \leq \ell/10 + \varepsilon\ell/80 \leq \ell/8$. Thus for $c_3 \geq 1/8$, x' was considered for being stored with v as a close or a far representative.

Let $r' = \|xy\|$ and b' the ball centered at x with radius r'. Recall that the open annulus $b' \backslash b$ contains no point. Let $\overline{b'}$ denote all points outside b'. We claim that any point p' in $\overline{b'}$ cannot replace a representative for quadtree box v that is contained in b. Intuitively, this is because any such point is at a relatively large distance from the center of quadtree box v. See Fig. 7.

Lemma 3.5 *Let p be any point in ball b, let p' be any point in $\overline{b'}$, and let q' be any point in quadtree box v. Then $\|q'p\| < \|q'p'\|$.*

Proof: By the triangle inequality, $\|qp'\| \geq \|xp'\| - \|xq\| \geq \|xy\| - \|xq\| \geq \ell/2 - D$. For $\|q'p'\|$ we now have:

$$\|q'p'\| \geq \|qp'\| - \|qq'\| \geq \|qp'\| - 2r_v \geq (\ell/2 - D) - 2(\varepsilon\ell/20) \geq \ell/2 - D - \varepsilon\ell/10.$$

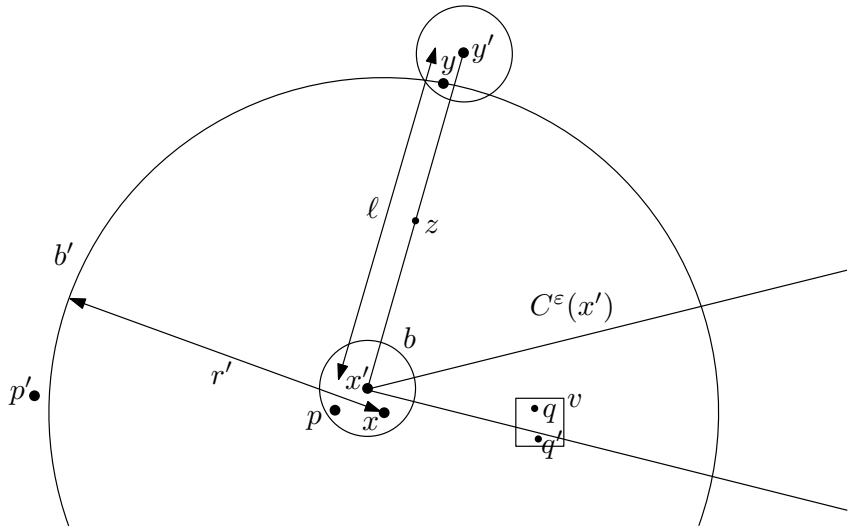

Fig. 7: Case $D < \ell/10$. (Distances are not scaled for clarity.)

On the other hand for $\|q'p\|$ again by the triangle inequality we have:

$$\|q'p\| \leq \|q'q\| + \|qx\| + \|xp\| \leq 2(\varepsilon\ell/20) + D + \varepsilon D/4.$$

To prove the lemma, it suffices that

$$D + \varepsilon D/4 + \varepsilon\ell/10 < \ell/2 - D - \varepsilon\ell/10.$$

Since $\varepsilon \leq 1$ and $D < \ell/10$, it is straightforward to show that the inequality is true. □

We set q' to be the center of quadtree box v, apply Lemma 3.5 and we derive that after generating all quadtree boxes of $\mathcal{U}$, quadtree box v has stored a close or far representative p that lies in b (e.g., x') and is an ε-$\lhd$NN for q. Again by Lemma 3.5 it is easy to see that after the two propagation stages, v stores as one of its representatives p or another point in b.

We will now show that some point from b is stored as one of the representatives of w. We consider two cases depending on whether v stored a point from ball b as a far or as a close representative. Suppose first that v stored this point as a far representative. Let t be the path in T (before its balancing) from the node associated with v to the leaf associated with $w \subseteq v$. Stage one propagation (or stage two for a resulting far representative for v) guarantees that some point contained in b will be stored with w as a far representative and be an ε-$\lhd$NN for q. This is because by Lemma 3.5 a far representative from b will replace any far representative from $\overline{b'}$ in any node in path t.

We consider now the remaining case where v stored a point from ball b as a close representative. Clearly the corresponding expanded cone of this point stabs or covers w. Consider of all the nodes in path t that before propagation stored a close representative the node u' that is closest to the leaf of t. Let p' denote the close representative and u the quadtree box that are associated with u'. Note that $w \subseteq u \subseteq v$. Clearly by construction, p' is a center point of some well-separated pair and $C^\varepsilon(p')$ stabs u. Let ℓ' denote the length of the corresponding pair. By construction for $c_3 = 1/8$, p' must be at distance $d(p', u) \leq \ell'/8$. Clearly,

$$(\varepsilon\ell'/c_2)/2 \leq s_u \leq s_v \leq \varepsilon\ell/c_2.$$

Hence $\ell' \leq 2\ell$ and $d(p', u) \leq \ell/4$. Assume that p' is in $\overline{b'}$. By the proof of Lemma 3.5 for any q' in v and thus for any q' in u

$$\|q'p'\| \geq \ell/2 - D - \varepsilon\ell/10 > \ell/2 - \ell/10 - \ell/10 \geq 2\ell/5.$$

By choosing q' as the point that minimizes the distance of u from p', we get that $d(p', u) > 2\ell/5 > \ell/4$ which leads to a contradiction. Thus p' lies in b. Recall that $C^{\varepsilon}(p')$ intersects w. By the propagation of close representatives, clearly p' must reach leaf cell w and be stored with it either as a close or a far representative. Note that by the selection of node u', none of its descendants in the path t stores a close representative before propagation. If during the second propagation stage p' becomes a far representative for some node in the path t by Lemma 3.5 it replaces any other far representative stored in a descendant or it stops to propagate because of another far representative which should also lie in b. This implies that in all cases a representative from ball b is stored with w. We conclude that the returned answer is correct.

3.3 Construction of an $\mathcal{ACVD}$

The leaf cells of tree T (which we will simply call cells) clearly form an $\mathcal{ACVD}$ for S assuming that a cell can store up to two representatives and where the guarantee is that for any point in each cell at least one of the stored representatives is an ε-$\lhd$NN.

If we desire each cell of the $\mathcal{ACVD}$ to store at most one representative then it is possible to further split the cells in our $\mathcal{ACVD}$ that store two representatives into a fixed number of parts. Specifically, the two (expanded) cones of the two representatives split the cell into at most four regions: one region that corresponds to the region of the cell outside the two cones, one region for the points of the cell that lie in only the cone of the close representative, one region for the points of the cell that lie in only the cone of the far representative, and one region that is the intersection of the two cones and the cell.

The last region needs to be split further and this is done by simply using the bisector between the two representatives and assigning each of the resulting regions to just one of the two representatives based on which side of the bisector the region lies. Locating in which of the five regions of the cell a query point lies clearly requires $O(1)$ time. We note that all the above regions in which a cell is split need not be connected and that their shape is not that of a cell.

In summary, we have the following theorem:

Theorem 3.1 *Given a set S of n points in $\mathbb{R}^d$, a cone C of fixed direction and angle, and an approximation factor $0 < \varepsilon \leq 1$, we can construct in $O((n/\varepsilon^d)\log(n/\varepsilon)\log(1/\varepsilon))$ time an $\mathcal{ACVD}$ for S of size $O(n\log(1/\varepsilon)/\varepsilon^d)$ that supports ε-$\lhd$NN queries in $O(\log(n/\varepsilon))$ time.*

It is easy to see that by building the data structure of Theorem 3.1 for several different cones it is possible to obtain a data structure of $\widetilde{O}(n/\varepsilon^{2d})$ size that can answer ε-$\lhd$NN queries with the same time bound as above for any given cone with any direction and with any angle in the interval $[\varepsilon, 2\pi)$. (Note there are $O(1/\varepsilon^{d-1})$ different directions and $O(1/\varepsilon)$ different angles that we have to consider in order to cover all possible cones.)

Thus, a corollary of Theorem 3.1 is the following:

Corollary 3.1 *Given a set S of n points in $\mathbb{R}^d$ and an approximation factor $0 < \varepsilon \leq 1$, we can construct in $O((n/\varepsilon^{2d})\log(n/\varepsilon)\log(1/\varepsilon))$ time a data structure for S of size $O(n\log(1/\varepsilon)/\varepsilon^{2d})$ that supports ε-$\lhd$NN queries for any cone with any angle in the interval $[\varepsilon, 2\pi)$ in $O(\log(n/\varepsilon))$ time.*

It is worth mentioning that for $\mathcal{AVD}$s, Arya and Malamatos [5] have showed a lower bound on space of $\Omega(n/\varepsilon^{d-1})$ assuming that the cells are allowed to store only one representative and that they are either axis-aligned hyperrectangles or differences of two axis-aligned hyperrectangles. Clearly, this lower bound applies to the case of $\mathcal{ACVD}$s as well assuming that the cone angle is a constant independent of ε.

4 Conclusion

In this paper we gave an efficient construction of an approximate conic Voronoi diagram that can be used to solve the conic nearest neighbor problem which arises in a wide range of applications. We pose two open questions. The first is to solve this problem in a dynamic setting perhaps by using the results in [21]. The second question is to provide space-time tradeoffs for approximate conic Voronoi diagrams analogous to those that are known for approximate Voronoi diagrams.

Acknowledgments

Part of this work was done while the authors were at Max-Planck-Institute for Informatics. We thank the anonymous referees for suggesting a generalization of our result and numerous ways to improve its presentation.

References

[1] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. *J. Comput. Sys. Sci.*, 66:207–243, 2003.

[2] B. Aronov, P. Bose, E. D. Demaine, J. Gudmundsson, J. Iacono, S. Langerman, and M. Smid. Data structures for halfplane proximity queries and incremental Voronoi diagrams. In *Proc. of the 7th Latin American Symposium on Theoretical Informatics*, 2006.

[3] S. Arya, G. D. Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd Annu. ACM Sympos. Theory Comput.*, pages 579–586, 2011.

[4] S. Arya, G. D. Fonseca, and D. M. Mount. Polytope approximation and the Mahler volume. In *Proc. 23rd Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 29–42, 2012.

[5] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.

[6] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57:1–54, 2009.

[7] S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach.*, 45:891–923, 1998.

[8] S. Arya, D. M. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: randomized solutions. *Comput. Geom. Theory Appl.*, 13(2):91–107, 1999.

[9] P. B. Callahan and S. R. Kosaraju. Algorithms for dynamic closest pair and n-body potential fields. In *Proc. 6th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 263–272, 1995.

[10] K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 56–65, 1987.

[11] K. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivers, and C. Stein. *Introduction to Algorithms.* MIT Press, third edition, 2009.

[13] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM J. Comput.*, 35(1):91–109, 2005.

[14] T. K. Dey, J. Giesen, S. Goswami, and W. Zhao. Shape dimension and approximation from samples. In *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 772–780, 2002.

[15] C. A. Duncan, M. T. Goodrich, and S. Kobourov. Balanced aspect ratio trees: combining the advantages of k-d trees and octrees. In *Proc. 10th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 300–309, 1999.

[16] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 781–790, 2002.

[17] J. Giesen and U. Wagner. Shape dimension and intrinsic metric from samples of manifolds with high co-dimension. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 329–337, 2003.

[18] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.

[19] S. Har-Peled. *Geometric approximation algorithms.* American Mathematical Society, 2011. `http://sarielhp.org/book/`.

[20] S. Meiser. Point location in arrangements of hyperplanes. *Inform. Comput.*, 106:286–303, 1993.

[21] D. M. Mount and E. Park. A dynamic data structure for approximate range searching. In *Proc. 26th Annu. ACM Sympos. Comput. Geom.*, pages 247–256, 2010.

[22] G. Narasimhan and M. Smid. *Geometric spanner networks.* Cambridge University Press, 2007.

[23] H. Samet. *Foundations of multidimensional and metric data structures.* Morgan Kaufmann, 2006.

[24] A. C. Yao. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.*, 11:721–736, 1982.