(Approximate) Conic Nearest Neighbors and the induced Voronoi Diagram

Stefan Funke[†*] Theocharis Malamatos[†] Domagoj Matijevic[†] Nicola Wolpert[†]

Abstract

For a given point set in Euclidean space we consider the problem of finding (approximate) nearest neighbors of a query point but restricting only to points that lie within a fixed cone with apex at the query point. Apart from being a rather natural question to ask, solutions to this problem have applications in surface reconstruction and dimension detection.

We investigate the structure of the Voronoi diagram induced by this notion of proximity and present approximate and exact data structures for answering cone-restricted nearest neighbor queries. In particular we develop an approximate Voronoi diagram of size $O((n/\varepsilon^d)\log(1/\varepsilon))$ that can be used to answer cone-restricted nearest neighbor queries in $O(\log(n/\varepsilon))$ time.

1 Introduction

Answering nearest neighbor queries for a given point set S in a (low-dimensional) Euclidean space is a classical problem in computational geometry and many algorithms have been proposed to solve that problem. The natural datastructure to solve it is the so-called *Voronoi diagram* $VD(S)$. $VD(S)$ is a decomposition of $\mathbb{R}^d$ into *Voronoi cells* such that for the cell $V(s,S)$ of point $s \in S$ we have $V(s,S) := \{p \in \mathbb{R}^d | d(p,s) \leq d(p,s') \forall s' \in S\}$. Unfortunately it is easy to come up with point sets for which the Voronoi diagram has size $\Omega(n^{\lceil d/2 \rceil})$, so its use to answer nearest neighbor queries in dimensions $d > 2$ is rather unattractive. The lack of other methods that guarantee efficient (i.e. poly-logarithmic) query times has led to the introduction of the notion of *approximate nearest neighbors*. For a query point q, a point $s \in S$ is a $(1+\varepsilon)$ approximate nearest neighbor (ANN) for q if $d(q,s) \leq d(q,s') \cdot (1+\varepsilon) \forall s' \in S$. Not insisting on the *exact* nearest neighbor has allowed for datastructures of near-linear size and logarithmic query time, e.g. [AMN$^+$98], [DGK99]. Similarly, the notion of an *approximate Voronoi diagram* (AVD) has allowed for space decompositions of near-linear size, e.g. by Har-Peled [Har01] and Arya and Malamatos [AM02]. Here each cell $\mathcal{C}$ of this decomposition has a point $N(\mathcal{C}) \in S$ assigned such that $\forall q \in \mathcal{C}$, $N(\mathcal{C})$ is an approximate nearest neighbor (ANN) for q. In all these papers as well as in our work the dimension d is treated as a constant.

In this paper we consider a variant of the nearest neighbor search problem that has not been studied as extensively before: given a set of points S and a cone C we want to preprocess S such that for a query point q we can determine the (approximate) nearest neighbor $s_q \in S$ that is contained in the cone C with apex at q. In Section 2 we describe one application where this type of nearest neighbor query can be of use. The only related result on this problem that we are aware of is the work by Funke and Ramos [FR02], where the authors process S in $O(n \text{ polylog } n)$ time and determine for each $s \in S$ its conic nearest neighbor in S. We want to note though, that their approach does *not* provide a query datastructure which can determine the conic nearest neighbor for a point $q \notin S$. Attempts to instrument directly some of the known datastructures for approximate nearest neighbor queries ([AMN$^+$98], [DGK99]) to solve the case of cone queries failed so far, since a certain crucial packing property necessary for these approaches does not seem to hold in case of cones.

1.1 Our contribution In this paper we consider the problem of computing (approximate) nearest neighbors in a point set S with the additional restriction that returned points have to lie in a cone with apex at the query point, in particular we

- develop datastructures for answering *exact* cone queries in sublinear time (Section 3).
- examine the structure of the Voronoi diagram induced by the notion of conic proximity in 2 and 3 dimensions (Section 4).

*This work was done while the first author was member of Prof. Leonidas Guibas' group at the Computer Science Department, Stanford University, USA

†Max-Planck-Institut f. Informatik, Saarbrücken, Germany, {funke,tmalamat,dmatijev,wolpert}@mpi-sb.mpg.de

- propose near-linear size datastructures for answering cone queries for a fixed cone approximately and construct an approximate conic Voronoi diagram of size $O((n/\varepsilon^d)\log(1/\varepsilon)$ which allows for query time $O(\log(n/\varepsilon))$ (Section 5). This is the main contribution of our paper.

Before we get to the more technical part of the paper we illustrate in Section 2 an actual application for the type of datastructures developed in this paper.

2 Motivation

The original motivation for this work stems from a problem in surface reconstruction/analysis of point cloud data. Here one is given a set S of sample points (point cloud) that are taken from an unknown surface Γ. The goal is now to reason about or in the best case even *reconstruct* the original surface Γ, e.g. find a faithful polyhedral approximation which has the sample points in S as vertices. See Figure 1 for a sample set taken from a cactus (model).

One important primitive when working with point cloud data is the estimation of surface normals for a given point on the surface (but only using the set S and not the original surface Γ). A common strategy for estimating the surface normal of a point $p \in \Gamma$ is to first collect the k-nearest neighbors of p (for some constant k, this can be done efficiently in $O(k \log n)$ if one is satisfied with approximate nearest neighbors) and then either use some sort of plane fitting (and use the normal of the fitting plane as estimation) or connect p to two of its k-nearest neighbors to form a 'nice' triangle using the triangle normal as an estimation for the surface normal.

In fact for the latter one can prove that if the sample set S is a ε-sample (intuitively: it is sampled densely enough), then the triangle Δpqr formed by p and two of its k-nearest neighbors p,q yields a faithful approximation (up to an angle of $O(\varepsilon)$) to the surface normal at p *if* Δ is non-skinny, i.e. it is not formed by three almost collinear points, see [FR02]. Essentially, if p,q,r are almost collinear, a slight height change of the surface in one of the points changes the triangle normal by an arbitrarily large amount.

Now, the problem is that even if S is a ε-sample, there is no constant k for which we can guarantee that amongst the k-nearest neighbors of p are two points q,r which form a 'nice' triangle with p. That is due to the fact that the ε-sample condition is only a *lower* bound on the sampling density but allows for arbitrary oversampling in arbitrary patterns. For example, in Figure 2, as long as $k \leq 8$ collecting k-nearest neighbors from the query point will never yield sample points suitable for a faithful normal estimation in the query point (also when using plane fitting). Observe, that this threshold $k \leq 8$ can be made arbitrarily high by just sampling further points along the already densely sampled lines. The sample set obviously remains a ε-sample, but normal estimation via simple collection of k-nearest neighbors gets increasingly difficult[1]. The solution to this problem is to allow for nearest neighbor searches that are restricted to some direction, see [FR02]. Here, the authors divide the space into a *constant* number of *cones* with a common apex and then determine for a point p nearest neighbors within each cone. Using this approach it is easy to extract a 'nice' triangle with nearby points of p.

The solution presented in [FR02] computes in $O(n \text{ polylog } n)$ for each point $p \in S$ its nearest neighbors within each of the cones with apex at p. What is important to note here: their algorithm *does not* provide a query data structure such that when interested in the surface normal at some point $q \in \Gamma$, but $q \notin S$, one cannot use their method to answer this query. It only applies for points that are already in the point set S. Apart from being a natural question to ask, a genuine query data structure for nearest neighbors in a cone might be of interest when refining/resampling a surface that is only represented as a point cloud. Further applications that could be thought of are in the context of *dimension detection* for manifolds of unknown dimension in some Euclidean space, see e.g. [DGGZ02], [GW03].

The contribution of this paper is to provide datastructures of near-linear size which can answer (approximate) nearest neighbor queries restricted to a cone.

3 Exact cNN Queries for arbitrary Query, Angle, and Direction

Let S be a set of n distinct points (*sites*) in $\mathbb{R}^d$. Furthermore, let V be a set of d linearly independent vectors $v_1,\ldots,v_d \in \mathbb{R}^d$. We define the set $C(V) := \{v \in \mathbb{R}^d | v = \sum_i \lambda_i v_i,\ \lambda_i \geq 0\}$. For any point $q \in \mathbb{R}^d$ we define the *cone* of q as $C(q,V) := \{x \in \mathbb{R}^d \mid x = q+v,\ v \in C(V)\}$. We also define the *reverse cone* of q as $\overline{C}(q,V) := \{x \in \mathbb{R}^d \mid x = q-v,\ v \in C(V)\}$. Note that we use *simplicial* cones.

Suppose we want to construct a data structure such that for any cone $C(q,V)$ we can efficiently report the site $s_q \in S \cap C(q,V)$ such that $d(q,s_q) \leq d(q,s)$ for any other $s \in S \cap C(q,V)$. We say that s_q is a *conic nearest neighbor* (cNN) of q with respect to $C(q,V)$.

[1] If one assumes that S is a *tight* ε-, (ε,δ)-, or (locally) uniform sample, k can be determined easily. But if S is neither of those but only an ε-sample, making S tight itself seems to require reliable normal estimation.

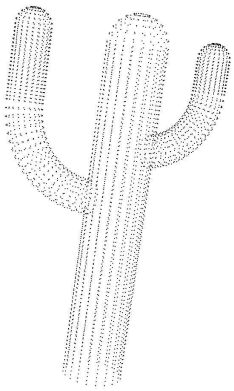

Figure 1: Point sample from the surface of a cactus

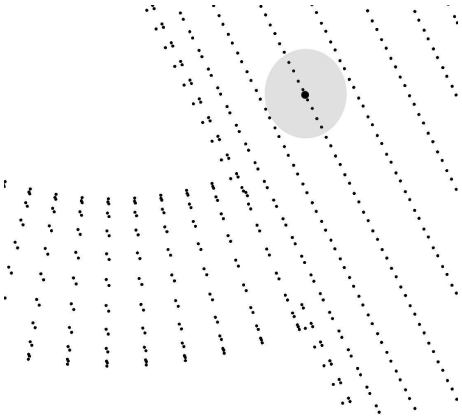

Figure 2: Zoom-In on one of the branches of the cactus: Neighborhood of a sample point that does not allow for reliable normal estimation.

The solution to our problem mainly relies on the well known *Partition theorem* which has been used in the context of range searching ([Mat92]). We cite the theorem in the following:

THEOREM 3.1. *Any set S of n points in $\mathbb{R}^d$ can be partitioned into $O(r)$ simplices, such that every simplex contains between n/r and $2n/r$ points and every hyperplane crosses at most $O(r^{1-1/d})$ simplices (crossing number). Moreover, for any $\psi > 0$ such a simplicial partition can be constructed in $O(n^{1+\psi})$ time.*

Using this theorem recursively one can construct a tree which is called a *partition tree* (e.g. the root of the tree, associated with S, has $O(r)$ children, each associated with a simplex from the first level, and so on). Observe that if r is a constant, partition tree is of $O(n)$ size and it can be constructed in time $O(n^{1+\psi})$ for any $\psi > 0$. With the help of such tree it is well-known that one can answer range counting queries in $O(n)$ space and $O(n^{(1-1/d)+\psi})$ time in $\mathbb{R}^d$, which is very close to the best possible.

The good news for us is that we can use almost the same data structure to answer cone queries. We first deal with the 2D case and then show that a similar approach works in higher dimension as well.

LEMMA 3.1. *Let $S \subset \mathbb{R}^2$ be a set of points. For any $\psi > 0$, there is a data structure of $O(n \log n)$ size and $O(n^{1+\psi})$ construction time such that for any point q and an arbitrary cone $C(q,V)$ one can compute cNN of q in time $O(n^{1/2+\psi})$.*

Proof: Assume we are given a partition tree and suppose that we have some conic query to answer. Clearly by Theorem 3.1 we have the bound on the number of triangles that intersect the two semi-lines which is $O(\sqrt{r})$. On those triangles we recur, which leads to a total of $O(\sqrt{n})$ triangles intersected by the semi-lines. But still there might be $O(r)$ triangles lying completely inside the cone. To overcome this problem, one can precompute a normal Voronoi diagram (VD) for the points inside each triangle (the VD extends outside each triangle as well). This will increase the total space of the partition tree by a $O(\log n)$ factor since every level in the tree now will be of $O(n)$ size. However, by doing this we avoid recursing on the triangles that lie completely inside the cone. Namely, for every triangle that lies inside the cone the point closest to q can be found by point-location in $O(\log n)$ time. □

In principle our ideas developed so far for computing cNN of a point q in the plane work for cNN in arbitrary dimension. The disadvantage is the high space complexity for storing the Voronoi diagrams associated with each triangle. The space complexity for a Voronoi diagram of n points is $\Omega(n^{\lceil d/2 \rceil})$. However, one can avoid storing the Voronoi diagrams at the cost of moving to the higher-dimensional space $\mathbb{R}^{d+1}$ instead of $\mathbb{R}^d$.

LEMMA 3.2. *Let $S \subset \mathbb{R}^d$ be a set of points. For any $\psi > 0$, there is a data structure of $O(n)$ size and $O(n^{1+\psi})$ construction time such that for any point $q \in \mathbb{R}^d$ and an arbitrary cone $C(q,V)$ one can compute cNN of q in time $O(n^{(1-1/(d+1))+\psi})$.*

Proof: The main idea is to build a partition tree for S, additionally storing a representative point for every simplex. For the query cone $C(q,V)$ we recurse on all simplices that are intersected by the faces of the cone. What again remains to do is

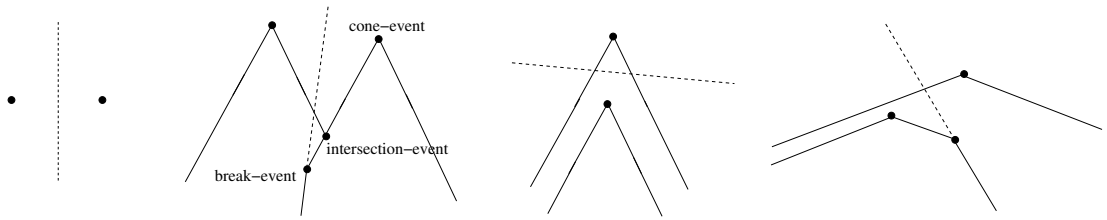

Figure 3: Left most figure denote the classic VD cells whereas rest of the figures depict conic VD cells depending on the relative position of two sites and on the fixed angle α. Observe that the hashed lines denote the standard Voronoi separator between two sites.

the handling of the inner simplices, i.e simplices which are completely inside the cone. Among these we determine the one with the representative point closest to q. Let s_i be this nearest representative point. We draw a sphere $B_{q,r}$ around q with radius $r = |q - s_i|$. It is easy to see that only the inner simplices intersecting B are candidates to contain the cNN of q. So we have to recurse also for these.

In order to be able to bound the number of inner simplices intersecting B we build the partition tree not in d- but in $(d+1)$-dimensional space. We make use of a well-known lifting step and project all points of S onto the unit-paraboloid P in $\mathbb{R}^{d+1}$. This gives us a new set $\widetilde{S}$. We build the partition tree for $\widetilde{S}$. This of course also leads to a partition of S into convex $\leq (d+2)$-gons. By construction and by the fact that a sphere in $\mathbb{R}^d$ is the projected intersection of a hyperplane and the unit paraboloid in $\mathbb{R}^{d+1}$ we know that every hyperplane and every sphere in $\mathbb{R}^d$ intersects $O(r^{1-1/(d+1)}) \leq (d+2)$-gons.

□

The techniques presented in this section only allow for sublinear but not polylogarithmic query times. Since in practice this is rather prohibitive, we will later present datastructures that guarantee polylogarithmic query times at the cost of only approximate answers.

4 Exact Conic Voronoi diagrams

Let S be a set of n distinct points (sites) in the plane. In the following we fix a set of linearly independent vectors $V = \{v_1, v_2\}$ and assume that the cone angle $0 < \alpha(v_1, v_2) < \pi$. Since V is fixed, let $C(q) := C(q, V)$. We want to precompute the set of sites in S such that we can efficiently answer cNN queries for q with respect to $C(q)$. The precomputation step is based on the idea to compute a Voronoi-like diagram on the set of sites which we will call *Conic Voronoi diagram* (cVD). Answering the cNN for an arbitrary query point q will then reduce to the classical point location problem in the cVD.

For a fixed set S of sites, fixed angle and fixed direction we define a Conic Voronoi diagram of S as the subdivision of the plane into n cells, one for each site in S, with the property that a point q lies in the cell corresponding to a site $s_i \in S$ if and only if s_i is a conic nearest neighbor of q. That is, the conic Voronoi cell for $s_i \in S$ is defined to be:

$$\mathcal{V}_C(s_i) = \{p \in \mathbb{R}^2 \mid s_i \in C(p) \text{ and } d(p, s_i) \leq d(p, s_j) \text{ for all } s_j \in C(p) \cap S\}.$$

Note that the cVD is a bit more complicated than the normal Voronoi diagram. W.l.o.g. we assume from now on that the cone direction is parallel to the $(0,1)$ vector. In Figure 3 three different kind of cVDs for two points are depicted (the cVD for one point $s = (s_x, s_y)$ is just the reversed cone $\overline{C}(s)$ mirrored at the line $y = s_x$). In the cVD for a set $S = \{s_1, \ldots, s_n\}$ of sites there are two different kind of edges: Edges that lie on the reversed cones $\overline{C}(s_i)$ (*cone-edges*) and edges that lie on the standard Voronoi edges (*voronoi-edges*) between sites.

The cVD is planar and connected. A cell in the cVD belonging to site s_i has s_i as the unique point with highest y-coordinate. The latter allows us to compute the cVD with a sweep-line algorithm where we sweep the scene from top to bottom. We always maintain the invariant that above the sweep-line the cVD is correctly computed. For the current sweep-line we know in which order it is intersected by the edges (which may be cone- or voronoi-edges) and by the faces of the cVD. This is stored in an X-structure. The order changes only at three kind of event points: The sweep-line encounters a new site (*site-event*), a cone-edge becomes a voronoi-edge (*break-event*), or two neighbouring edges along the sweep-line intersect (*intersection-event*). A Y-structure stores the event-points discovered so far. It is initialized with the site-events.

At each event-point the X- and Y-structure can be updated in $O(\log n)$ time. An event-point causes only local changes along the sweep-line. Also the Y-structure needs only constantly many updates. The concrete implementation is straightforward and left to the reader.

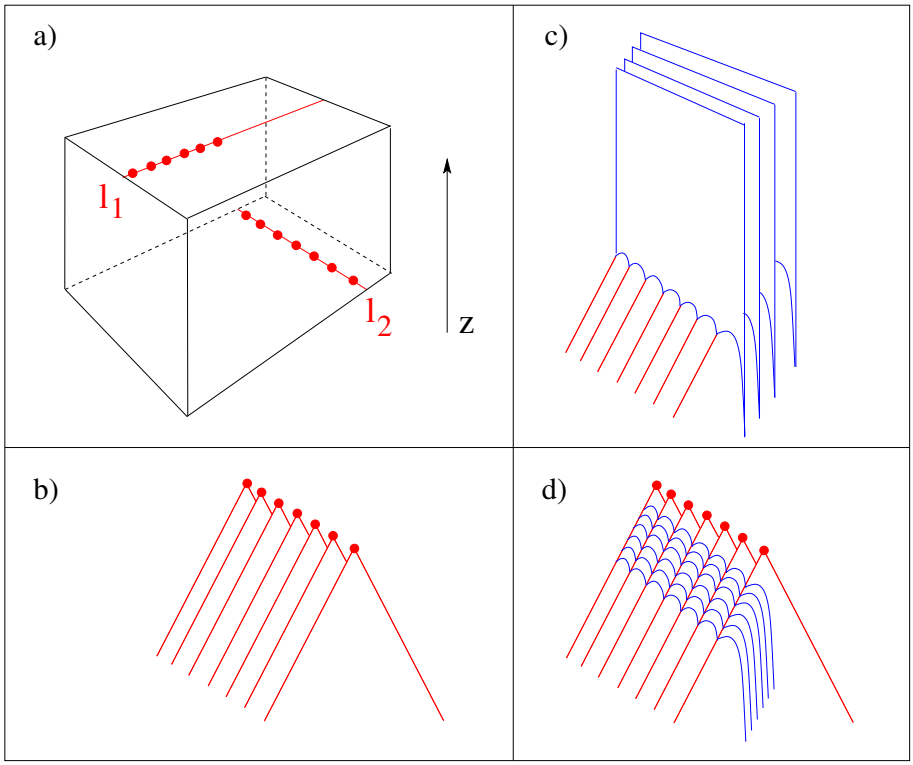

Figure 4: The construction of a Conic Voronoi diagram of quadratic complexity in 3-space.

LEMMA 4.1. *Let P be a set of n sites. The conic Voronoi diagram of P is of size $O(n)$.*

Proof: If we imagine all the unbounded edges of cVD as going to a common point infinity, we can prove the lemma with the help of Euler's formula, which states that for any connected planar graph with n_v nodes, n_e edges and n_f cells $n_v - n_e + n_f = 2$. Note that in our case $n_f = n + 1$ since every cell corresponds to a unique site. Furthermore, let $n_v = n'_v + n''_v + 1$ where n'_v denotes the number of site- and break-points and n''_v denotes the number of intersection-points of cVD. Since every site can produce at most two break-points, we have that $n'_v \leq 3n$. Also, observe that the rest of the points in the cVD have degree at least three (including node infinity) which implies that $3(n''_v + 1) \leq 2n_e$. Together with Euler formula this implies the statement of our lemma. □

Next we will show a lower bound for the Conic Voronoi Diagram in 3-dimensional space:

LEMMA 4.2. *There exists a set of n points in 3-space which has a Conic Voronoi Diagram of size $\Omega(n^2)$.*

Proof: This proof and the pictures are based on 'true' cones, but translate directly to the case of simplicial cones. The main idea of construction is the same as for Voronoi diagrams. We assume that the direction for which we want to compute the cVD is parallel to the $(0,0,1)$ vector. We take two planes parallel to the (x,y)-plane. On the upper such plane we place $n/2$ points along a line l_1. We do the same on the lower plane but along a line l_2 which is perpendicular to l_1. For illustration have a look at Figure 4 a). The cVD of the lower $n/2$ points is depicted in Figure 4 b). The separators of this cVD between neighbouring cones are planar patches parallel to the z-axis bounded by parabolas. The cVD of all n points consists of two of these kind of cVD's just described: one for the lower and one for the upper $n/2$ points. The cells of the upper points are delimited from below by the cVD of the lower points. If we choose the upper points sufficiently high above the lower points the separators between neighbouring cones are stopped by the lower cVD as shown in Figure 4 c). Each separator causes $n/2$ parabolic segments of the overall cVD, resulting in $\Omega(n^2)$ parabolic segments, Figure 4 d). □

Like for the regular Euclidean distance, the size of the respective cone Voronoi diagram in dimensions larger than 2 turns out to be prohibitive for its use in answering cone queries.

5 Approximate Conic Voronoi diagrams and NN Queries

In this section we relax the problem in a sense that when querying with a point q we do not insist on receiving the *exact* nearest neighbor from the datastructure. In particular we allow as output a point that is slightly (by a factor of $(1+\varepsilon)$) further than the true conic nearest neighbor and – for our second approach – slightly outside (by an angle of $O(\varepsilon)$) of the cone with apex at q.

Let V be a set of d linear ind. vectors $v_1,\ldots,v_d \in \mathbb{R}^d$. Let $s \in \mathbb{R}^d$ be some point and $b_i = v_i^T s$ for $i = 1,\ldots d$. We define the *cone* of s w.r.t. V as $\text{cone}(s) := \{p \in \mathbb{R}^d : \forall i : v_i^T p \leq b_i\}$. We also define the *reverse cone* of s w.r.t. V as $\overline{\text{cone}(s)} := \{p \in \mathbb{R}^d : \forall i : v_i^T p \geq b_i\}$. Note that we use here a different definition of cone – it is expressed wrt to the *normals* of the bounding hyperplanes. As before we call the cone in which answers to a query q should lie the *reverse cone*.

In the following we fix a set of l.i. vectors V and aim to preprocess a set S of points in $\mathbb{R}^d$ such that for any given query point q we can find an *approximate conic nearest neighbor* (cANN) s_q with the following properties:

- if $d_{\min} = \min\{d(s,q) : s \in S \cap \overline{\text{cone}}(q)\}$, then $d(s_q,q) \leq (1+\varepsilon)d_{\min}$
- either $s_q \in S \cap \overline{\text{cone}}(q)$ or the angle between $\overline{s_q q}$ and $\overline{\text{cone}}(q)$[2] is $O(\varepsilon)$

Intuitively the former guarantees that the returned point is not too far away compared to the 'true' conic nearest neighbor, the latter guarantees that it 'almost' lies within the desired reverse cone of the query point.

The first approach is based on a simple construction of nested range trees and returns points that lie exactly in the cone of query point q, and only approximates the distance. The second approach is based on a conic approximate Voronoi diagram (cAVD) – a decomposition of the underlying space – and allows for very fast query times and rather low space requirements.

5.1 Reduction to 'orthogonal' Range Queries in a skewed Coordinate System Here we briefly sketch a method to solve approximate conic nearest neighbor queries using nested range trees. For convenience we assume that the respective cone which we want to query is reasonably small, i.e. $\forall v_i, v_j \in V : \angle v_i v_j \geq \pi/4$. If this is not the case we could always subdivide the desired cone into a constant number of smaller cones.

The idea is to derive new coordinates for all $s \in S$ based on the set of hyperplanes V forming the cone. We set the new ith coordinate s_i^N of a point $s = (s_1,\ldots,s_d)$ to be $s_i^N := v_i^T s$. Now, if we have a query point q with respective new coordinates $(q_1^N,\ldots q_d^N)$ all points p within its reverse cone have new coordinates $p_i^N \geq q_i^N$. That is we can determine them using a nested range query for points of the form $[q_1^N,\infty],\ldots,[q_d^N,\infty]$. So we build a 1-dimensional range tree on the first (new) coordinate. Each internal node stores the points in its respective subtree as a 1-dimensional range tree on the second coordinate and so on. The overall space requirement is $O(n\log^d n)$. The result (a set of points) for the query in the first tree is returned in $O(\log n)$ batches, we query each of these batches according to the 2nd coordinate and so on. Finally, in the last level (d) we obtain all points within the reverse cone of p in $O(\log^d n)$ batches. It remains to extract an (approximately) closest amongst them.

Let $\overrightarrow{r}$ be a vector starting at $(0,\ldots,0)$ that is contained in the reverse cone of $(0,\ldots,0)$. We order the sets associated with the internal nodes of the last level of our tree hierarchy according to the direction $\overrightarrow{r}$. Let p_1 be the point amongst the $O(\log^d n)$ batches that minimizes $\overrightarrow{r}^T p$ (this can be found by inspecting the *first* element in each batch, as they are stored sorted according to $\overrightarrow{r}^T p$). It's easy to see that for cone angles of less than $\pi/4$, $d(q,p_1) \leq d(q,p)\cdot 3/2$ for all $p \in \overline{\text{cone}}(q)$, so p_1 is already a $3/2$ approximation. That is we have a upper bound of $d_{\text{up}} = d(q,p_1)$ and a lower bound of $d_{\text{low}} = d(q,p_1)\cdot 2/3$ for the distance of the conic nearest neighbor of q. Now consider the following part of the reverse cone of q: $G := \overline{\text{cone}}(q) \cap B(q,d_{\text{up}}/(1+\varepsilon)) - B(q,d_{\text{low}})$. Using a standard packing argument it is easy to see that we can certify using $O(1/\varepsilon^d)$ many cone queries whether G does not contain any point (in which case the point determining the current upper bound is a $(1+\varepsilon)$ cANN) or whether G contains a point (in which case we improve the upper bound by a factor of at least $(1+\varepsilon)$). Hence after at most $O(\log_{1+\varepsilon}(3/2)) = O(1/\varepsilon)$ iterations we arrive at a $(1+\varepsilon)$ cANN. Therefore the overall query time is $O((1/\varepsilon^{d+1})\log^d n)$.

THEOREM 5.1. *Given a set S of points in $\mathbb{R}^d$ and a cone defined by a set of vectors V, we can preprocess them in a datastructure of size $O(n\log^d n)$ such that we one determine an cANN (with no angle error) in time $O((1/\varepsilon^{d+1})\log^d n)$.*

We note that using the standard technique of fractional cascading one can improve the query time by a $\log n$ factor.

[2] This angle is $\min_{p\in\overline{\text{cone}}(q)} \angle s_q q p$.

5.2 An approximate conic Voronoi diagram of near-linear size For the first part we will borrow some ideas first presented in [AM02] for construction of ('normal') approximate Voronoi diagrams but following more the the presentation in [HP].

The overall picture of our construction is as follows: based on the well-separated pair decomposition (WSPD, see [CK95]) of the point set S we generate a set of $O((n/\varepsilon^d)\log(1/\varepsilon))$ many grid cells. Some of the grid cells are marked *critical*, and some of the cells have a point from S associated, such that if the smallest of the generated cells containing a query point q is non-critical and has a point associated, this point is an approximate conic nearest neighbor for q (where the approximation is both w.r.t. the angle as well as the distance). Then in a second step, we treat the critical cells individually and partition them using a constant number of hyperplanes. The set of all generated (and possibly split) cells can then be transformed into a space decomposition and/or stored in a *compressed quadtree* (e.g. in [HP]) to allow for efficient point location in the space decomposition (query time $O(\log(n/\varepsilon))$ and space of $O((n/\varepsilon^d)\log(1/\varepsilon))$).

A central component of our construction is the so-called *well-separated pair decomposition* (WSPD) of a point set. For a point set S in $\mathbb{R}^d$ and a *separation constant* s, the WSPD is a collection of $O(ns^d)$ cluster pairs (A_i,B_i), with $A_i,B_i \subset S$ and *centers* $a_i \in A_i, b_i \in B_i$ such that $\forall p \in A_i : d(p,a_i) \leq |ab|/s$ (and likewise for points in B_i). Furthermore, for any two points $s,t \in S$, there exists a unique pair (A_i,B_i) with $s \in A_i$ and $t \in B_i$. Intuitively a WSPD approximates the $\Omega(n^2)$ distance relationships using only $O(n)$ pairs of clusters (for constant s).

In contrast to the constructions in [AM02] or [HP] we do not rely on a separate query datastructure to determine points associated with single cells, but rather determine them directly during the construction via a WSPD of the point set. Assume as in [HP] that the point set S is contained in a cube of dimensions $[0.5-\varepsilon, 0.5+\varepsilon]^d$, and this cube is a minimum axis-aligned cube for S. For the rest of the paper we only consider a decomposition or conic nearest neighbor relationships for points $q \in [0,1]^d$ since for points outside this unit cube, we can determine easily whether they're contained in an enclosing cone of the point set S, and then any point in S is a conic approximate nearest neighbor. Our algorithm constructs cells that arise in the quadtree (or its higher dimensional equivalent) when decomposing the unit cube $[0,1]^d$ recursively – we call this the *canonical grid* and the cells the *canonical cells*. The canonical grid G_{α_i} consists of cubes of width/side length α_i (for $\alpha_i = 2^{-i}, i \in \mathbb{N}$).

Hence the constructed cells in the algorithm are either disjoint, identical or one is contained in the other.

We use the notion of an *exponential grid*[3] $G_E(p,r,R,\varepsilon)$ around p. Let $b_i = b(p,r_i)$, $i = 0,\ldots\lceil \log R/r \rceil$ be the ball of radius $r_i = r2^i$. Define G'_i to be the set of cells of the canonical grid G_{α_i} that intersect b_i with $\alpha_i = 2^{\lfloor \log(\varepsilon r_i/(16d)) \rfloor}$. Obviously $|G'_i| = O(1/\varepsilon^d)$. We remove from G'_i all cells completely covered by cells of G'_{i-1}. Cells that are partially covered by cells in G'_{i-1} are replaced by the cells covering them in G'_{i-1}. Let G_i be the resulting set of canonical cells. And let $G_E(p,r,R,\varepsilon) = \cup_i G_i$. We have $|G_E(p,r,R,\varepsilon| = O(\varepsilon^{-d}\log(R/r))$. It can be computed in linear time in its size.

5.2.1 Stage I of the construction We first construct a WSPD for the point set with separation constant 32 and then consider all pairs of the WSPD. A pair (A,B) with representatives $(a,b) \in P \times P$ in the WSPD has the property that $\forall p \in A$ we have $d(p,a) \leq |ab|/32$ (and likewise for B and b). For each pair (A,B) with representatives a and b, construct the exponential grid $G_E(a,|ab|/8,64|ab|/\varepsilon,\varepsilon) \cup G_E(b,|ab|/8,64|ab|/\varepsilon,\varepsilon)$. The idea is now to associate either a or b with some of the cells but maintaining the invariant that if a cell $\mathcal{C}$ has a (b respectively) associated, then any point $q \in \mathcal{C}$ can see a (b respectively) within its reverse cone or the line between q and a makes an angle of at most $O(\varepsilon)$ with the reverse cone of q. Furthermore some cells (which might or might not have a or b associated with it, could be marked as 'critical'). Only cells that are marked as critical or have a point associated with them are remembered for further processing.

The construction proceeds as follows: Partition the set of grid cells into C_a (cells that intersect the ball of radius $|ab|/8$ around a), C_b (cells that intersect the ball of radius $|ab|/8$ around b) and C_x (the remaining cells). Now determine which cells are 'close' to the cone for a (likewise to the cone for b) as follows: A cell $\mathcal{C}$ is said to be 'close' to the cone of a if $\exists p_1 \in \mathcal{C}, p_2 \in \text{cone}(a) : \angle p_1 a p_2 \leq \varepsilon$, Now for all cells $\mathcal{C} \in C_x$:

- if $\mathcal{C}$ is only close to the cone of a, store a with $\mathcal{C}$
- if $\mathcal{C}$ is only close to the cone of b, store b with $\mathcal{C}$
- if $\mathcal{C}$ is not close to either ... don't store anything
- if $\mathcal{C}$ is close to both, store either a or b (whichever is closer to the center of $\mathcal{C}$)

For all cells $\mathcal{C} \in C_b$ (that is, cells near b)

[3] Taken from [HP]

- if C is close to the cone of a, store a with it
- furthermore, if C is close to the cone of b, mark C as 'critical' and remember the WSPD pair (A,B) with it

Do the symmetric thing for all $C \in C_a$.

The following lemma should now be easy to see.

LEMMA 5.1. *Let C be a cell created during the processing of WSPD pair (A,B), $N(C)$ the point stored with it. Then for all points $q \in C$, $N(C)$ either lies in the reverse cone of q or is at most an angle of 2ε away from it.*

Proof: Assume w.l.o.g. $N(C) = a$. By construction, there exists a point $q' \in C$ and a point $p \in \text{cone}(a)$ (not necessarily from S) such that $\angle q'ap \leq \varepsilon$. Furthermore we also have $|qq'| \leq \varepsilon |aq'|$ (remember that C is far away from a otherwise we would not have associated a with it). But then it follows immediately that $\angle qap \leq \angle q'ap + \angle q'aq \leq \varepsilon + \arctan\varepsilon \leq 2\varepsilon$. So we get that the angle between the reverse cone of q and the ray $\overrightarrow{qa}$ is at most 2ε. □

At this point we have not claimed anything about the distances of the associated points. **Construction (continued):** We collect all the cells created (i.e. that have a point associated or have been marked critical) in the above step (some cells might be created several times) and aggregate the conic neighbor for some cell C as follows: cell C keeps the closest (to its center) point stored with C or one of its ancestors (i.e. cells that contain C). This step completely ignores the fact whether cells are marked 'critical' or not. 'Criticality' is also *not* inherited, i.e. a cell C is called critical iff *all* of its instances created were critical (no dependence on ancestors or children). Clearly the Lemma above still remains true and the distance of the point associated with a cell can only decrease. This finishes the first stage of the construction.

Let q be a query point, and C the *smallest* cell generated which contains q. We claim that if C is not marked 'critical', the point $N(C)$ stored with C is a $(1+O(\varepsilon))$ cANN with angle error of $O(\varepsilon)$.

LEMMA 5.2. *Let $q \in [0,1]^d$ be a query point, C be the smallest cell created which contains q. If C is not critical, the point $N(C)$ stored with C has distance at most $(1+\varepsilon)d^C_{\min}(q)$ where $d^C_{\min}(q)$ denotes the distance of the exact conic nearest neighbor of q.*

Proof: Let us first show that if q has an (exact) conic nearest neighbor $T(q)$, a cell containing q was created during our construction. Let us denote by x the point closest to $T(q)$ with $|T(q)x| \geq \varepsilon |T(q)q|$. Such a point x certainly exists (in the 'worst' case just take the furthest point from $T(q)$), and therefore a cell containing q was considered; but we still need to show that it had either a point associated or was marked critical (and hence created). Let C be the cell considered when looking at the WSPD pair (A,B) with representatives (a,b) which separates $T(q)$ from x (with $T(q) \in A$, $x \in B$). Let us denote by $l = |T(q)q|$ the distance between q and its true cNN. If we have $|aT(q)| \leq \varepsilon l$, C was 'close' to the cone of a (as q has a point p' in the cone of a at distance of at most $|aT(q)|$ and then $\angle qap' \leq \arctan\varepsilon \leq \varepsilon$) and therefore was created. So assume $|aT(q)| > \varepsilon l$. But since on the other hand $|aT(q)| \leq |ab|/32$, $x = a$ would have been a better (i.e. closer) choice with $|T(q)x| \geq \varepsilon |T(q)q|$ which contradicts our initial choice of x.

Let us now turn to the second part of the proof: we want to show that if the smallest cell C containing a query point q is not critical, its associated point is a cANN for q. But this can be argued by essentially the same construction as above; again we consider point x that is closest to $T(q)$ with $|T(q)x| \geq \varepsilon |T(q)q|$. We have argued that when considering the WSPD pair (A,B) separating $T(q)$ from x (with $T(q) \in A$, $x \in B$), we will create a cell C' containing q. We also showed that $|aT(q)| \leq \varepsilon l$ where $l = |T(q)q|$. Hence if the cell C' was not 'too close' to a, i.e. $d(C',a)) > |ab|/8$, a was stored with C' and $|qa| \leq |T(q)q| + |aT(q)| \leq (1+\varepsilon)|T(q)q|$, that is a is a $(1+\varepsilon)$ cANN. For smaller cells contained in C' containing q, only better points could have been found, so the point associated with the smallest cell C containing q is certainly a cANN.

It remains to treat the case that $d(C',a)) \leq |ab|/8$, that is, when considering (A,B), C' was marked critical (and a was not associated with it), By the choice of x and the fact that C' was marked critical, we know that $\nexists s \in S$: $\varepsilon |qT(q)| < |sT(q)| < 15|ab|/16$ (o.w. $x = s$ would have been chosen). Let w' be the width of cell C'. We have $w' \leq \varepsilon |ab|/(8 \cdot 16 \cdot d)$ since C' was critical. Now assume the smallest non-critical cell $C \neq C'$ containing q was created by WSPD pair (E,F) with representatives e, f. Let w be the width of C; w.l.o.g. assume C was closer to e than to f. If $|eT(q)| \leq \varepsilon |T(q)q|$ we're done since e is an cANN for q. Otherwise we have $|eT(q)| > \varepsilon |T(q)q|$ but also by construction of the grid for (E,F) $|eq| \leq 32dw/\varepsilon \leq 32dw'/\varepsilon \leq |ab|/4$. Furthermore we have $|T(q)q| \leq |ab|/2$ otherwise C' would not have been critical and hence $|T(q)e| \leq |T(q)q| + |eq| \leq |ab|/2 + |ab|/4 \leq 3|ab|/4$ which contradicts the fact that $\nexists s \in S : \varepsilon |qT(q)| < |sT(q)| < |ab|15/16$.

□

The previous two Lemmas imply that if the smallest generated cell C containing a query point q is not critical, the point stored with that cell is a valid cANN, i.e. it has distance at most $(1+\varepsilon)$ times the distance of the exact cNN, and the returned point lies at most an angle of 2ε off the reverse cone with apex at q.

5.2.2 Stage II of the Construction In the following we will refine the construction to cope with the case that the query point ends up in a critical cell. Before we can explain the respective refinement of our construction, let us first prove some properties about critical cells.

LEMMA 5.3. *Let C be a critical cell that is also the smallest cell containing some query point q, assume w.l.o.g. C was generated and declared critical while processing WSPD pair (A,B) with representatives (a,b) and $d(C,a) < |ab|/8$. If $\perp \neq cNN(q) \notin A$, that is, the true conic nearest neighbor of q is not in the set A but exists, then C has already an approximate conic nearest neighbor for q associated.*

Proof: Consider the WSPD pair (E,F) with representatives (e,f) separating $T(q)$ and a ($T(q) \in E$, $a \in F$), let C' be the respective cell containing q. If $d(C',e) \geq |ef|/8$, the cell containing q got a point associated that is a cANN (proof along the lines of the above: if $|T(q)e| < \varepsilon|T(q)q|$ we don't have anything to show, otherwise consider the WSPD pair separating $T(q)$ and $e\ldots$). But if $d(C',e) < |ef|/8$ the cell C' has to be smaller than the cell C (as it is closer to e than to a). Furthermore either C' or one of its children containing q has been marked or associated with a point (again the argument is that if $|T(q)e| < \varepsilon|T(q)q|$, C' would have been marked, otherwise consider the WSPD pair separating $T(q)$ and $e\ldots$). This contradicts the assumption that C was the smallest cell created containing q. □

So we know that for points q for which a critical cell C is the smallest containing cell, the conic nearest neighbor is in A (the cluster whose representative a marked C as critical) or it is already associated with C.

Let us distinguish two cases: $|A| = 1$: we can simply split the relevant region of C by the planes of the cone of a and assign a to one part, the 'old' representative (if existant) to the other part.

$|A| > 1$: Let δ be the minimum distance of a part of C to a, that is not covered by smaller cells. If $\delta = 0$ (i.e. a is not covered by a smaller cell), another point in A together with a would have induced a finer grid cell at a. Hence assume $\delta > 0$. Then the diameter of point set A must be less than $\delta\varepsilon$. Otherwise consider the WSPD pair (E,F) separating a and the point $x \in A$ furthest from a. Then either the cell C is non-critical for (E,F) (contradiction to the initial assumption) or it is covered by smaller cells induced by (E,F). So for $|A| > 1$ we can construct an enlarged cone $\text{econe}(a) := \{p : \angle pap' \leq \varepsilon | p' \in \text{cone}(a)\}$ use this to partition the relevant part of cell C. Any point in C uncovered by smaller cells but contained in the enlarged cone of a has a as a cANN: distance wise and angle-wise. We note that the complexity of the intersection of relevant parts of C with the enlarged cone can be quite considerable (in particular, if C has many direct descendants). But since every cell has only one direct parent and the complexity of the intersection between a cube and d hyperplanes is constant for constant d, the overall complexity of the resulting decomposition remains linear in the number of original cubic cells.

THEOREM 5.2. *For a set of points $S \subset \mathbb{R}^d$ and a cone defined by d halfspaces, one can compute a decomposition of $\mathbb{R}^d$ into $O(\frac{n}{\varepsilon^d}\log\frac{1}{\varepsilon})$ regions with one associated point $\in S$ each, such that for any point $q \in \mathbb{R}^d$, the point associated with the cell C containing q is a cANN. If C has no point associated, q has no cNN in S. The space decomposition can be queried in time $O(\log(n/\varepsilon))$ and constructed in $O(\frac{n}{\varepsilon^d}\log^2\frac{n}{\varepsilon})$.*

Proof: The size follows from the fact that the WSPD created has $O(n)$ pairs and each pair induces $O((1/\varepsilon^d)\log(1/\varepsilon))$ cells. These are then stored in a compressed quadtree as in [HP] which yields the claimed space, construction, and query time. Correctness of the returned points follows from the previous Lemmas. □

6 Open Problems

Our (approximate) datastructures that allow for polylogarithmic query time all require a *fixed* cone during their construction. Even though in actual applications one is typically interested only in a constant number of cones – hence building our datastructures a constant number of times does not change the asymptotic space and running time – practicability would be greatly increased if we could design a datastructure that could answer queries for *variable* cones (like the partition-tree based approach, but the latter has almost-linear query time).

References

[AM02] S. Arya and T. Malamatos. Linear-size approximate voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[CK95] Paul B. Callahan and S. Rao Kosaraju. Algorithms for dynamic closest pair and n-body potential fields. In *Proc. 6th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 263–272, 1995.

[DGGZ02] Tamal K. Dey, Joachim Giesen, Samrat Goswami, and Wulue Zhao. Shape dimension and approximation from samples. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 772–780, 2002.

[DGK99] Christian A. Duncan, Michael T. Goodrich, and Stephen Kobourov. Balanced aspect ratio trees: combining the advantages of k-d trees and octrees. In *Proc. 10th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 300–309, 1999.

[FR02] Stefan Funke and Edgar A. Ramos. Smooth-surface reconstruction in near-linear time. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 781–790, 2002.

[GW03] Joachim Giesen and Uli Wagner. Shape dimension and intrinsic metric from samples of manifolds with high co-dimension. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 329–337, 2003.

[Har01] 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.

[HP] Sariel Har-Peled. Geometric approximation algorithms. Lecture Notes for CS598, UIUC.

[Mat92] Jirí Matousek. Efficient partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.