(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.

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 C of this decomposition has a point $N(C) \in S$ assigned such that $\forall q \in C, N(C)$ 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: 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. This class of problems goes back to Yao [Yao82] who showed that the conic nearest neighbors (cNN), using a fan of cones, can be used to find an $O(n)$-size supergraph of the minimum spanning tree. This construction is known today in the wireless literature as the Yao-graphs or somethimes also as Θ-graphs. We want to note though, that this approach answers cNN in any fixed dimension in sublinear time for arbitrary query point and direction but fixed angle. Several other authors showed that a similar construction, using a fan of cones of small angular diameter, yields a geometric spanner (see [KG92], [RS91], [AMS94], [Cla87]). Funke and Ramos [FR02] showed how to preprocess S in $O(n \text{ polylog } n)$ time and determine for each $s \in S$ its cNN in S. 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. In this paper we develop datastructures for answering *exact* cone queries in sublinear time (Section 2) and 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 3). The latter is the main contribution of our paper.

The original motivation for this work stems from a problem in surface reconstruction/analysis of point cloud data (see e.g. [FR02]).

This extended abstract omits most proofs and details which can be found in the long version of the paper[1].

2 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

*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

[1]For the long version of the paper see http://www.mpi-inf.mpg.de/ dmatijev/papers/ConeQueries.ps.gz

$C(q,V)$. 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 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 2 *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 $O(n^{1/2+\psi})$.*

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 *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 $O(n^{(1-1/(d+1))+\psi})$.*

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

3 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. 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 $\min_{p \in \overline{\text{cone}}(q)} \angle s_q q p$. 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.

3.1 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 $[q_1^N, \infty], \ldots, [q_d^N, \infty]$ and obtain all points within the $\overline{\text{cone}}(q)$ 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 range 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 4 *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 one can 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.

3.2 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 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 $t \in B_i$. 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* $G_E(p,r,R,\varepsilon)$ introduced in [HP] 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.

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), if $\mathcal{C}$ is close to the cone of a, store a with it and if $\mathcal{C}$ is close to the cone of b, mark $\mathcal{C}$ as 'critical' and remember the WSPD pair (A,B) with it. Do the symmetric thing for all $\mathcal{C} \in C_a$. Let $\mathcal{C}$ be a cell created during the processing of WSPD pair (A,B), $N(\mathcal{C})$ the point stored with it. Then for all points $q \in \mathcal{C}$ it should be possible to see that $N(\mathcal{C})$ either lies in the reverse cone of q or is at most an angle of 2ε away from it.

At this point we have not claimed anything about the distances of the associated points: 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 $\mathcal{C}$ as follows: cell $\mathcal{C}$ keeps the closest (to its center) point stored with $\mathcal{C}$ or one of its ancestors (i.e. cells that contain $\mathcal{C}$). This step completely ignores the fact whether cells are marked 'critical' or not. 'Criticality' is also *not* inherited, i.e. a cell $\mathcal{C}$ 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.

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

If the smallest generated cell $\mathcal{C}$ 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.

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.

Lemma 5 *Let $\mathcal{C}$ be a critical cell that is also the smallest cell containing some query point q, assume w.l.o.g. $\mathcal{C}$ was generated and declared critical while processing WSPD pair (A,B) with representatives (a,b) and $d(\mathcal{C},a) < |ab|/8$. If the true conic nearest neighbor of q is not in the set A but exists, then $\mathcal{C}$ has already an approximate conic nearest neighbor for q associated.*

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

Let us distinguish two cases: $|A| = 1$: we can simply split the relevant region of $\mathcal{C}$ 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 $\mathcal{C}$ 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 $\mathcal{C}$ 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 $\mathcal{C}$. Any point in $\mathcal{C}$ 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 $\mathcal{C}$ with the enlarged cone can be quite considerable (in particular, if $\mathcal{C}$ 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 6 *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 $\mathcal{C}$ containing q is a cANN. If $\mathcal{C}$ 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 $O(\frac{n}{\varepsilon^d}\log^2\frac{n}{\varepsilon})$.*

Observation: Our (approximate) datastructures that allow for polylogarithmic query time require a *fixed* cone during their construction. One interesting question is to design a datastructure that could answer queries for *variable* cones (like the part.-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] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[AMS94] S. Arya, D. M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *IEEE Symposium on Foundations of Computer Science*, pages 703–712, 1994.

[CK95] P. B. Callahan and S. R. 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.

[Cla87] K. Clarkson. Approximation algorithms for shortest path motion planning. In *STOC '87: Proc. 19th annual ACM conference on Theory of computing*, pages 56–65. ACM Press, 1987.

[DGK99] 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 annual ACM-SIAM Symp. on Discrete Algorithms*, pages 300–309, 1999.

[FR02] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. 13th annual ACM-SIAM Symposium on Discrete algorithms (SODA'02)*, pages 781–790, 2002.

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

[KG92] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Comput. Geom.*, 7(1):13–28, 1992.

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

[RS91] J. Ruppert and R. Seidel. Approximating the d-dimensional complete euclidean graph. In *Canadian Conf. on Comp. Geometry*, pages 207–210, 1991.

[Yao82] A. Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.