Approximate Nearest Neighbor Search Amid Higher-Dimensional Flats*

Pankaj K. Agarwal[1], Natan Rubin[2], and Micha Sharir[3]

1 **Department of Computer Science, Duke University, Durham, NC, USA**
pankaj@cs.duke.edu
2 **Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel**
rubinnat.ac@gmail.com
3 **School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
michas@tau.ac.il

Abstract

We consider the *approximate nearest neighbor* (ANN) problem where the input set consists of n k-flats in the Euclidean $\mathbb{R}^d$, for any fixed parameters $0 \leq k < d$, and where, for each query point q, we want to return an input flat whose distance from q is at most $(1+\varepsilon)$ times the shortest such distance, where $\varepsilon > 0$ is another prespecified parameter. We present an algorithm that achieves this task with $n^{k+1}(\log(n)/\varepsilon)^{O(1)}$ storage and preprocessing (where the constant of proportionality in the big-O notation depends on d), and can answer a query in $O(\text{polylog}(n))$ time (where the power of the logarithm depends on d and k). In particular, we need only near-quadratic storage to answer ANN queries amid a set of n lines in any fixed-dimensional Euclidean space. As a by-product, our approach also yields an algorithm, with similar performance bounds, for answering *exact* nearest neighbor queries amid k-flats with respect to any polyhedral distance function. Our results are more general, in that they also provide a tradeoff between storage and query time.

1998 ACM Subject Classification E.1 Data Structures, F.2.2 Nonnumerical Algorithms and Problems, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Approximate nearest neighbor search, k-flats, Polyhedral distance functions, Linear programming queries

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.4

1 Introduction

Nearest neighbor search is one of the most fundamental problems in computational geometry and has been studied extensively in many different fields, including computational geometry,

* Work by P. A. and M. S. was supported by Grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by P. A. was also supported by NSF under grants CCF-11-61359, IIS-14-08846, and CCF-15-13816, and by an ARO grant W911NF-15-1-0408. Work by N. R. was supported by grant 1452/15 from Israel Science Foundation by grant 2014384 from the U.S.-Israeli Binational Science Foundation, and by Ralph Selig Career Development Chair in Information Theory; the project leading to this application has received funding from European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 678765. Work by M. S. has also been supported by Grant 892/13 from the Israel Science Foundation, by the Blavatnik Research Fund in Computer Science at Tel Aviv University, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by the Hermann Minkowski-MINERVA Center for Geometry at Tel Aviv University.

databases, machine learning, and data mining; see [4, 11] for comprehensive surveys. The very basic scenario, referred to as the *post-office problem* in [23], asks to preprocess a collection S of n points in $\mathbb{R}^d$ (called *sites*), where d is a fixed parameter, into a data structure, so that the site in S nearest to a query point $q \in \mathbb{R}^d$, i.e., the site $\mathsf{NN}(q, S) = \arg\min_{s \in S} \mathsf{dist}(q, s)$, where $\mathsf{dist}(\cdot, \cdot)$ is the Euclidean distance, can be reported quickly.[1] This basic version has been extended in numerous ways over the last four decades. Most notably, in such extensions the sites and/or the queries can be chosen from richer families of geometric objects (say, lines, k-flats, or even convex polyhedra, not to mention curved objects like balls), and $\mathsf{dist}(\cdot, \cdot)$ can be another distance function, such as an l_p-norm, a polyhedral distance function, or the Hausdorff distance (for non-point sites or queries) [4, 11]. The best known solution for the post-office problem requires roughly $n^{\lceil d/2 \rceil}$ storage in the worst case, for answering queries in $O(\log n)$ time, in any fixed dimension $d \geq 2$. The extended versions of the problem, for non-point sites and/or for other metrics/distance-functions, are naturally even more challenging. In the search for more efficient data structures, we therefore give up the goal of finding the exact nearest neighbor, and settle for structures that can answer efficiently *approximate nearest neighbor* (or, shortly, ANN) queries. That is, given a prespecified error parameter $\varepsilon > 0$, an ε-ANN query returns a site $s \in S$ satisfying $\mathsf{dist}(q, s) \leq (1 + \varepsilon)\mathsf{dist}(q, \mathsf{NN}(q, S))$. In what follows we use $\mathsf{ANN}(q, S)$ to denote the set of all sites with this property, i.e.,

$$\mathsf{ANN}(q, S) = \{s \mid \mathsf{dist}(q, s) \leq (1 + \varepsilon)\mathsf{dist}(q, \mathsf{NN}(q, S))\}.$$

This paper focuses on the scenario in which S is a collection of n k-flats lying in the Euclidean space $\mathbb{R}^d$, of any fixed dimension $d > k$ (where d and k are treated as constants), and the queries are points $q \in \mathbb{R}^d$. For a point $q \in \mathbb{R}^d$ and a site $s \in S$ (assumed to be closed), $\mathsf{dist}(q, s)$ is the minimum Euclidean distance between q and a point of s, i.e., $\mathsf{dist}(q, s) = \min_{p \in s} \mathsf{dist}_{p \in s}(q, p)$. Given S and a parameter $\varepsilon > 0$, the goal is to preprocess S into a data structure so that, for any query point $q \in \mathbb{R}^d$, a k-flat $s \in \mathsf{ANN}(q, S)$ can be reported quickly.

Related work. As mentioned above, nearest-neighbor (NN) searching, especially when the input sites are points, has been studied extensively. It is beyond the scope of this paper to review all the related work on nearest-neighbor searching, so we focus on the most relevant work, and refer the reader to [4, 11, 30] for more comprehensive reviews.

The most obvious approach to answering NN queries is to construct the *Voronoi diagram* of the set S of input objects, and perform point location in the diagram with the query point. Recall that the Voronoi diagram of S is the decomposition of space into cells, where each cell, associated with one of the input sites, consists of all points whose nearest site in S is that site. It is well known that the complexity of the Euclidean Voronoi diagram of a set of n points in $\mathbb{R}^d$ is $\Theta(n^{\lceil d/2 \rceil})$ in the worst case. Better bounds on the complexity of the diagram are known, though, in some special cases. For example, the expected complexity of the Voronoi diagram of a set of n random points chosen uniformly in $[0, 1]^d$ is linear; see [19].

Recently, there has been some work on Voronoi diagrams of non-point sites. For example, Chew *et al.* [17] have shown that the complexity of the Voronoi diagram of a set of n lines in $\mathbb{R}^3$ under the polyhedral metric (or distance function) defined by a convex polytope Q of constant complexity (see Section 2 for the definition of polyhedral distance functions) is $O(n^2\alpha(n)\log n)$, where the constant of proportionality depends on the complexity of

[1] The site $\mathsf{NN}(q, S)$ is uniquely defined, unless q belongs to a set of measure zero (namely, if q lies on the boundaries of two or more cells in the Voronoi diagram of S).

Q. The near-quadratic bound was subsequently extended to the case when the input sites are constant-complexity convex polyhedra in $\mathbb{R}^3$ [25]. It is an open question whether the complexity of the Euclidean Voronoi diagram of a set of lines in $\mathbb{R}^3$ is nearly quadratic; so far, the near-quadratic upper bound has been confirmed only for lines with constantly many orientations [24]. See the book by Aurenhammer *et al.* [13] for comprehensive studies of Voronoi diagrams.

Because of the potentially large complexity of the Voronoi diagram, there has also been work on constructing a data structure for answering NN queries directly, that does not require the construction of the Voronoi diagram. For example, an NN query amid a set of n points in $\mathbb{R}^d$ can be answered in $\widetilde{O}(n^{1-1/\lceil d/2\rceil})$ time using linear space.[2] More generally, for a given parameter $n \leq m \leq n^{\lceil d/2\rceil}$, a query can be answered in $\widetilde{O}(n/m^{1/\lceil d/2\rceil})$ time using $O(m)$ space. The known lower-bound results on range searching [2] suggest that this is the best bound one can hope for.

Consequently, attention has focused on answering ANN queries, as described above (see, e.g., [7, 12, 15, 20, 22], to name a few works that follow this paradigm). Earlier methods for answering ANN queries stored the input points in a (compressed) quad tree, k-tree, or their variants, and performed a spiral search to return a point in $\mathsf{ANN}(q, S)$ for a given query point q; see, e.g., [21]. More recently, the notion of an *approximate Voronoi diagram* (AVD for short) has been introduced; similar to a Voronoi diagram, AVD is a decomposition of space into cells, each associated with a site s, so that s is an approximate nearest neighbor for all query points in that cell. Har-Peled [20] constructed an approximate Voronoi diagram (AVD) of a set of n points in $\mathbb{R}^d$ of size $\widetilde{O}\left(\frac{1}{\varepsilon^d}n\right)$. Another AVD was proposed by Arya, Malamitos and Mount [8, 9]; its size is linear in n, and it can be constructed in near-linear time.

A more elaborate approach yields a data structure for ANN queries that can answer an ε-ANN query in $O(\log(n/\varepsilon))$ time using $O(n/\varepsilon^{d/2})$ space; more generally, for a parameter $\log\frac{1}{\varepsilon} \leq \theta \leq \frac{1}{\varepsilon^{d/2}\log(1/\varepsilon)}$, a query can be answered in $O(\log n + \frac{1}{\theta\varepsilon^{d/2}})$ time using $O(n\theta)$ space [7].

The performance of these and of many earlier data structures for answering ANN queries depends exponentially on d, so they are not efficient for large values of d. This has lead to extensive work on data structures for ANN-queries whose query time and size have polynomial dependence on d, most notably using the locality-sensitive-hashing (LSH) technique and its variants [3, 6]. The best-known data structure of this kind computes in $n^{7/(8c^2)+O(1/c^3)}$ time a $(c-1)$-ANN with high probability, for $c > 1$, using $n^{1+7/(8c^2)+O(1/c^3)}$ space. See [4] for a survey of higher-dimensional NN problems and techniques.

Relatively little is known about ANN-queries for non-point input sites (e.g., lines, k-flats, or even convex polyhedra); see, e.g., [5, 27, 29]. The best structures obtained for ANN-search in such extended setups are typically more expensive than those obtained for the point-to-point problem. The result of Koltun and Sharir [25] implies that an AVD for a set of n pairwise disjoint triangles in $\mathbb{R}^3$, of size $\widetilde{O}(n^2)$, can be constructed in near-quadratic time, and thus an ANN-query for this setting can be answered in $O(\log n)$ time using $\widetilde{O}(n^2)$ space. A simple grid-like construction shows that any AVD for a set of n lines in any dimension $d \geq 2$ has $\Omega(n^2)$ complexity [20], which suggests that a near-linear-size data structure with $O(\log n)$ query time is unlikely. For higher dimensions, the best known data structure for lines is by Mahabadi [27]; it answers an ε-ANN query for lines in $\mathbb{R}^d$ in $(d + \log n + 1/\varepsilon)^{O(1)}$ time, using $(n+d)^{O(1/\varepsilon^2)}$ space; see also [14, 26].

[2] Throughout this paper, we use $\widetilde{O}(f(n))$ to denote $O(f(n)\,\mathrm{polylog}(n))$.

There is also some work on the dual problem, in which the input sites are points but the query objects are k-flats. For the case when the query is a line, i.e., a 1-flat, Andoni *et al.* [5] proposed a data structure that answers an ε-ANN query in $O\left(d^3 n^{1/2+\delta}\right)$ time, using $d^2 n^{O(1/\varepsilon^2+1/\delta^2)}$ space, for any constant $\delta > 0$. Later, Mulzer *et al.* [29] proposed a data structure for the case where the query objects are k-flats. Assuming there is an ANN data structure, when both input sites and query objects are points in $\mathbb{R}^d$, with $O(n^\rho)$ query time and $O(n^\sigma)$ space, for some parameters $\rho, \sigma > 0$, their data structure answers a query with a k-flat in time $O\left(n^{k/(k+1-\rho)+\delta}\right)$, using $O\left(n^{1+\sigma k/(k+1-\rho)} + n \log^{O(1/\delta)} n\right)$ space, for any constant $\delta > 0$.

Our results. We present an efficient data structure for answering ANN-queries when the input sites are k-flats in $\mathbb{R}^d$. The main results are summarized in the following theorem.

▶ **Theorem 1.** *Let d be a constant, let $0 \leq k \leq d-1$ be an integer, let $\varepsilon > 0$ be a given error parameter, and let $\gamma = O\left((1/\varepsilon)^{\frac{d-1}{2}\min(d-k,k+1)}\right)$. For a given parameter m with $n \leq m \leq n^{k+1}$, a given set S of n k-flats in $\mathbb{R}^d$ can be preprocessed in $\widetilde{O}(\gamma m)$ expected time into a data structure of $\widetilde{O}(\gamma m)$ size, so that, for a query point $q \in \mathbb{R}^d$, a flat $f \in \mathsf{ANN}(q, S)$, with respect to the Euclidean metric, can be reported in $\widetilde{O}\left(\gamma n / m^{\frac{1}{k+1}}\right)$ time.*

In particular, in the high-storage/fast-query regime, choosing $m = n^{k+1}$, we can perform, in *any* dimension $d > k$, ANN search with $\widetilde{O}(1)$ query cost (a) amid points ($k = 0$), using a near-linear structure, or (b) amid k-flats, for $k \geq 1$, using a structure of size $\widetilde{O}\left(\gamma n^{k+1}\right)$. For $k = 1$, i.e., for lines in $\mathbb{R}^d$ ($d \geq 2$), our data structures requires storage that is nearly quadratic in n in order to answer a query in $\widetilde{O}(1)$ time. For $d = 3$, our bound nearly coincides with that obtained from the three-dimensional AVD construction of Chew *et al.* [17], but no near-quadratic data structure was known for $d > 3$.

Unlike some of the recent ANN data structures for point sites [8, 9], we do not explicitly maintain the AVD of S. Instead, we approximate the Euclidean distance by a suitable polyhedral metric (see Section 2 for the definition), and use multi-level partition trees (designed for simplex range searching) [2] to answer (exact) NN-queries amid the flats of S with respect to the approximating polyhedral metric. As a byproduct, we obtain a simple and efficient data structure for answering *exact* NN queries amid k-flats with respect to a polyhedral distance function; see Theorem 2. An advantage of this approach is that it allows a trade-off between the size of the data structure and the query time, as stated in Theorem 1. In particular, an ANN query amid k-flats can be answered in $\widetilde{O}\left(n^{1-1/(k+1)}\right)$ time with near-linear storage.

2 Warm-up: Lines in $\mathbb{R}^3$

In this section we establish Theorem 1 for a set of lines in $\mathbb{R}^3$. Let L be a set of n lines in $\mathbb{R}^3$, and let $\varepsilon > 0$ be a parameter. We wish to preprocess L into a data structure that answers efficiently queries of the form: given a point $q \in \mathbb{R}^3$, find a line $\ell \in L$ such that

$$\mathsf{dist}(q, \ell) \leq (1+\varepsilon)\mathsf{dist}(q, L), \qquad \text{where} \qquad \mathsf{dist}(q, L) := \min_{\ell' \in L} \mathsf{dist}(q, \ell'),$$

and where dist denotes the Euclidean distance.

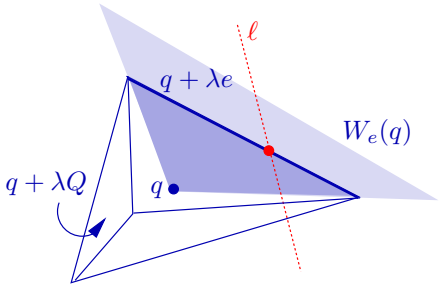

Figure 1 The Q-distance $\mathsf{dist}_Q(q, \ell)$ is the scaling factor λ for which the line ℓ touches $q + \lambda Q$, at some edge $q + \lambda e$ (and misses the interior of $q + \lambda Q$).

Polyhedral distance functions. In the general d-dimensional case, given a centrally symmetric convex polytope $Q \subset \mathbb{R}^d$, the *polyhedral distance* (with respect to Q) $\mathsf{dist}_Q(p, q)$, for a pair of points $p, q \in \mathbb{R}^d$, is defined as[3]

$$\mathsf{dist}_Q(p, q) = \sup \{t \mid q \notin p + tQ\},$$

and, more generally, for a point q and a convex object c not containing q,

$$\mathsf{dist}_Q(q, c) = \sup \{t \mid c \cap (q + tQ) = \emptyset\}.$$

The classical result of Dudley [18] implies that, for any $\varepsilon > 0$, there exists a convex polytope Q_ε, which is an intersection of $O\left(\frac{1}{\varepsilon^{(d-1)/2}}\right)$ halfspaces, or, alternatively, the convex hull of a similar number of vertices, such that $\mathsf{dist}_{Q_\varepsilon}$ approximates the Euclidean metric up to a factor of $1 + \varepsilon$; that is, for any pair of points p, q, we have

$$\mathsf{dist}(p, q) \leq \mathsf{dist}_{Q_\varepsilon}(p, q) \leq (1 + \varepsilon)\mathsf{dist}(p, q). \tag{1}$$

The advantage of using polyhedral distance functions for answering ANN-queries is that, when q is a point and f is a k-flat, $\mathsf{dist}_Q(q, f)$ can be characterized as the smallest expansion factor t for which f makes contact with some $(d-k-1)$-face of the expanding polytope $q+tQ$. This allows us to process each of the $O(1)$ $(d - k - 1)$-faces σ of Q for fast *face-shooting* queries, in which, given a query point q, we seek the smallest t for which $q + t\sigma$ hits a flat in S, and return that flat. For example, for the case of line sites in $\mathbb{R}^3$, the case studied in this section, each such query shoots a fixed segment from the query point q; the expanding segment traces a flat (two-dimensional) wedge that emanates from q and is a translate of some fixed wedge (that depends on the edge of Q that we shoot). We seek the first time at which the expanding wedge hits an input line and return that line.[4] Hence, in the general case of k-flats in $\mathbb{R}^d$, we prepare a constant number of face-shooting structures, one for each face of Q of the appropriate dimension, search with the query point q in each of them, and return the smallest expansion factor that the queries output, and the corresponding flat of S as the nearest neighbor.

In what follows we return to the special case of lines in $\mathbb{R}^3$. Given the error parameter $\varepsilon > 0$, we approximate the Euclidean unit ball by a centrally symmetric convex polytope $Q = Q_\varepsilon$, of complexity $O(1/\varepsilon)$ (using Dudley's bound). We then solve the *exact* NN problem

[3] In fact, the polyhedral metric also can be defined for non-centrally-symmetric polytopes (or, for that matter, for any compact convex body Q), but, to simplify matters in this presentation and to ensure that the distance function is a metric, we take Q to be centrally symmetric.

[4] Note that the wedge might miss the line completely, in which case we output $+\infty$.

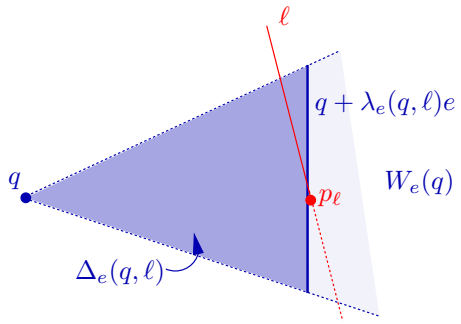

Figure 2 The wedge $W_e(q)$ is the union of all the copies $q + \lambda\Delta_e$, for $\lambda \geq 0$. $\lambda_e(q,\ell)$ is the scaling factor λ for which the triangle $q + \lambda\Delta_e$ touches ℓ at its edge $q + \lambda e$, and $\Delta_e(q,\ell)$ is that triangle. For each boundary edge e of Q, we seek the line $\ell \in L$ which minimizes the scaling distance $\lambda_e(q,\ell)$.

for the lines of L with respect to dist_Q, that is, for any query point q, the algorithm computes $\mathsf{dist}_Q(q, L)$, and returns the line of L that is nearest to q under this metric. In fact, the procedure presented next solves the exact NN problem for any convex polytope Q, not even assuming that it is centrally symmetric with respect to the origin.

Exact NN-search for L with respect to an arbitrary polytope Q. Given a point q and a line ℓ, there exists at least one edge e of Q (that depends on q and ℓ), such that the distance $\mathsf{dist}_Q(q,\ell)$ is the scaling factor λ for which (i) $q + \lambda e$ and ℓ touch one another, and (ii) ℓ does not meet the interior of $q + \lambda Q$. See Figure 1.

To decompose the problem, we consider, for each edge e of Q, the triangle $\Delta_e \subset Q$ spanned by the origin o and e. Assume with no loss of generality that no $\ell \in L$ is parallel to Δ_e.[5] Thus, for each $\ell \in L$ there exists a unique scaling factor $\lambda_e(q,\ell) \in \mathbb{R} \cup \{\infty\}$, such that the homothetic placement $q + \lambda_e(q,\ell)\Delta_e$ touches ℓ at a point of $q + \lambda_e(q,\ell)e$ (we put $\lambda_e(q,\ell) := +\infty$ when there is no such placement). We have $\lambda_e(q,\ell) < \infty$ if and only if ℓ intersects the planar wedge $W_e(q)$ which is the union of all the copies $q + \lambda\Delta_e$, for $\lambda \geq 0$. In what follows, we denote the resulting placement $q + \lambda_e(q,\ell)\Delta_e$ by $\Delta_e(q,\ell)$; see Figure 2.

As already noted, our strategy for computing $\mathsf{dist}_Q(q, L)$ is to design a separate data structure $\mathcal{D}_e$ for each edge e of Q, which answers efficiently queries of the form: Given a point q, find the smallest scaling factor $\lambda_e(q) := \min_{\ell \in L} \lambda_e(q,\ell)$, and report the corresponding line ℓ^* that attains $\lambda_e(q) = \lambda_e(q,\ell^*)$.

With this machinery available, we return to our approximating polytope Q_ε, query each of its $O(1/\varepsilon)$ edge-structures $\mathcal{D}_e$ with q, and report the minimum of the corresponding output values $\lambda_e(q)$, and the line attaining that minimum. As is easily checked, the output gives a $(1+\varepsilon)$-approximation to the Euclidean $\mathsf{dist}(q, L)$.

The edge structures $\mathcal{D}_e$. Let e be a fixed edge of Q. Given a point q, we wish to return $\lambda_e(q) := \min_{\ell \in L} \lambda_e(q,\ell)$ as well as the corresponding line ℓ_e^* that attains $\lambda_e(q) = \lambda_e(q,\ell^*)$. To simplify the presentation, and with no loss of generality, assume that e is the edge $z = 0$, $x = 1$, $-a \leq y \leq a$, for some $a > 0$.

Let us express a query in algebraic terms. Recall that we assume no line in L to be parallel to Δ_e, i.e., no line in L is normal to the z-axis. Hence, we parametrize such a line ℓ

[5] If L contains lines that are parallel to Δ_e, we apply an infinitesimally small rotation to Q which preserves all of its essential properties.

in $\mathbb{R}^3$ by the pair of equations

$$x = u_x(\ell)z + v_x(\ell), \qquad y = u_y(\ell)z + v_y(\ell),$$

for a suitable quadruple of real parameters $(u_x(\ell), v_x(\ell), u_y(\ell), v_y(\ell))$.

Let $q = (x_0, y_0, z_0)$ be the query point, and let ℓ be a line in L with the parameters $(u_x(\ell), v_x(\ell), u_y(\ell), v_y(\ell))$. Notice that $W_e(q)$ is contained in the plane $z = z_0$, and this plane meets ℓ at the point

$$p_\ell = (u_x(\ell)z_0 + v_x(\ell),\ u_y(\ell)z_0 + v_y(\ell),\ z_0).$$

The condition that p_ℓ lies in the wedge $W_e(q)$ can be expressed as

$$-a\left(u_x(\ell)z_0 + v_x(\ell) - x_0\right) \le u_y(\ell)z_0 + v_y(\ell) - y_0 \le +a\left(u_x(\ell)z_0 + v_x(\ell) - x_0\right),$$

$$\begin{aligned} \text{or} \quad & (u_y(\ell) + au_x(\ell))z_0 + (v_y(\ell) + av_x(\ell)) \ge y_0 + ax_0 \\ & (u_y(\ell) - au_x(\ell))z_0 + (v_y(\ell) - av_x(\ell)) \le y_0 - ax_0. \end{aligned} \tag{2}$$

Both constraints in (2) are linear in $u_x(\ell), v_x(\ell), u_y(\ell), v_y(\ell)$, with coefficients depending on the query q and the constant a (that is, on the edge e). Among the lines that satisfy these inequalities, our goal is to return the one that minimizes the scaling factor $\lambda_e(q, \ell)$, given by

$$\lambda_e(q, \ell) = u_x(\ell)z_0 + v_x(\ell) - x_0,$$

which is also linear in the chosen parameterization of ℓ.

In view of the above observations, we construct a three-level partition tree (see [1, 2, 16, 28]) on the lines of L. The first two levels are used to collect, for any given query point $q = (x_0, y_0, z_0)$, the lines that satisfy both conditions in (2), as the (disjoint) union of a small number of pre-stored "canonical" subsets, and the third level supports linear-programming-like queries, where each query specifies a linear objective function and asks for the point in the canonical subset that attains its minimum.

In more detail, we represent each line $\ell \in L$, parametrized by $(u_x(\ell), v_x(\ell), u_y(\ell), v_y(\ell))$, by the triple of points $p^+(\ell), p^-(\ell), p^\circ(\ell) \in \mathbb{R}^2$, where

$$\begin{aligned} p^+(\ell) &:= (u_y(\ell) + au_x(\ell),\ v_y(\ell) + av_x(\ell)) \\ p^-(\ell) &:= (u_y(\ell) - au_x(\ell),\ v_y(\ell) - av_x(\ell)), \quad \text{and} \\ p^\circ(\ell) &:= (u_x(\ell),\ v_x(\ell)), \end{aligned}$$

and put

$$P^+ := \{p^+(\ell) \mid \ell \in L\}, \qquad P^- := \{p^-(\ell) \mid \ell \in L\}, \qquad P^\circ := \{p^\circ(\ell) \mid \ell \in L\}.$$

A line ℓ satisfies (2) if and only if $p^+(\ell)$ lies above the line $z_0x + y = y_0 + ax_0$ and $p^-(\ell)$ lies below the line $z_0x + y = y_0 - ax_0$.

Following the standard methodology of multi-level data structures, each of the three levels of our partition tree, each of whose levels supports halfplane range searching queries amid points of one of the planar sets P^+, P^- or P°. This is done as follows. We fix a parameter $n \le m \le n^2$. The first level is a partition tree T, as described in [16], over the set P^+. Each node of T is associated with some *canonical subset* P_v^+. For a query halfplane h, $h \cap P^+$ can be represented as the disjoint union of $O(n/\sqrt{m} + \log n)$ canonical subsets (those stored at the nodes of T that the query with h reaches). Next, for each node v of T,

we construct a similar partition tree $T^{(v)}$, as a second-level structure, on the corresponding subset $P_v^- = \{p^-(\ell) \mid p^+(\ell) \in P_v^+\}$ of P^-. Again, each node $z \in T^{(v)}$ is associated with a suitable canonical subset $P_{z,v}^- \subset P_v^-$. Finally, at the third level, we preprocess the point set $P_{z,v}^\circ = \{p^\circ(\ell) \mid p^-(\ell) \in P_{z,v}^-\}$ into a data structure so that, for a query vector $u \in \mathbb{R}^2$, the point of $P_{z,v}^\circ$ that is minimal in direction u can be computed efficiently. Using a linear-size halfplane range reporting data structure[6] (see, e.g., [2]), such an extremal query can be answered in $O(\log n)$ time. Putting everything together, we obtain a data structure of $\widetilde{O}(m)$ size, so that, for a query point $q \in \mathbb{R}^3$, $\lambda_e(q)$, and the corresponding line $\ell_e^* \in L$, can be computed in $\widetilde{O}(1 + n/\sqrt{m})$ time. The further details, omitted here, can be found in the aforementioned papers.

Hence, for any choice of $n \le m \le n^2$, and for each of the $O(1/\varepsilon)$ edges e of Q_ε, we construct, in $\widetilde{O}(m)$ time, the data structure $\mathcal{D}_e$, as just described. This takes a total of $\widetilde{O}(m/\varepsilon)$ storage and preprocessing. Given a query point $q \in \mathbb{R}^3$, we query with q in each of these structures, and output the smallest scaling factor $\lambda_e(q)$, over all edges e, and the line $\ell \in L$ that attains this minimum. The total cost of a query is $\widetilde{O}(\frac{1}{\varepsilon}(n/m^{1/2}))$.

In particular, we can answer ANN queries amid a set of n lines in $\mathbb{R}^3$ under the Euclidean distance, in $\widetilde{O}(1)$ time using a data structure that requires only $\widetilde{O}(n^2/\varepsilon)$ storage and preprocessing time.

3 Proof of Theorem 1

The preceding algebraic approach can be extended, in a fairly straightforward manner, to nearest-neighbor problems involving k-flats, for $k \ge 1$, in $\mathbb{R}^d$, for $d \ge 3$ (and $d > k$). This is done as follows.

Let F be a collection of n k-flats, in general position, in $\mathbb{R}^d$, for some fixed $d > k \ge 1$ and $d \ge 3$. We approximate the Euclidean unit ball by a fixed convex polytope $Q = Q_\varepsilon$, which is centrally symmetric with respect to the origin o, so that the resulting Q-distance function d_Q, satisfies (1). By Dudley's theorem [18], this can be achieved by a polytope Q_ε that has either $O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$ vertices, or $O\left(1/\varepsilon^{\frac{d-1}{2}}\right)$ facets.

As in Section 2, we next present a solution of the (exact) NN search problem for F with respect to the Q-distance function dist_Q, for an arbitrary fixed convex polytope Q, not even requiring it to be centrally symmetric.

Exact nearest neighbor search with respect to Q. For a k-flat $f \in F$ and a point q, the distance $\lambda = \mathsf{dist}_Q(q, f)$ is attained at a point $v \in f$ such that v lies on a $(d-k-1)$-face of $q + \lambda Q$. Thus, in complete analogy to the preceding treatment, we construct, for each $(d-k-1)$-face σ of Q, a data structure that supports queries of the form: given a query point q, find the smallest λ such that $q + \lambda\sigma$ touches a flat of F.

By triangulating Q, if necessary, we may assume that σ is a simplex. Let E_σ be the $(d-k)$-dimensional affine space spanned by o and σ, and let $K_\sigma := \bigcup_{\lambda \ge 0} \lambda\sigma$ be the $(d-k)$-dimensional wedge contained in E_σ.

The region $K_\sigma(q) := q + K_\sigma = \bigcup_{\lambda \ge 0}(q + \lambda\sigma)$ is a $(d-k)$-dimensional simplicial wedge whose (also $(d-k)$-dimensional) affine hull $E_\sigma(q)$ is $q + E_\sigma$. Assuming general position, each flat f of F intersects $E_\sigma(q)$ at a unique point, denoted as $f_\sigma(q)$.

[6] In this very special case, the structure is simply the convex hull of the underlying set.

For each $(d-k)$-face σ of Q, we collect those points $f_\sigma(q)$ that lie in $K_\sigma(q)$, and choose among them a point that minimizes the scaling factor $\lambda_\sigma(q) = \lambda_\sigma(q, f)$ at which $q + \lambda\sigma$ touches the point $f_\sigma(q)$. As in Section 2, this is done by constructing a separate data structure $\mathcal{D}_\sigma$ for each $(d-k-1)$-face σ of Q.

The face structures $\mathcal{D}_\sigma$. Without loss of generality, assume that the coordinate system is such that the linear subspace E_σ spanned by σ is the $x_1x_2\cdots x_{d-k}$-space $\mathbb{R}^{d-k}$ (given by $x_{d-k+1} = x_{d-k+2} = \cdots = x_d = 0$). Regard K_σ as the intersection of $d-k$ fixed halfspaces (through the origin) $h_1^+, h_2^+, \ldots, h_{d-k}^+$ within $\mathbb{R}^{d-k}$, and write $h_j^+ = \{x \in E_\sigma \mid x \cdot u_i \geq 0\}$, for fixed respective vectors $u_1, \ldots, u_{d-k}$ in E_σ.

We now cast the preceding observations in algebraic form. In general position, each k-flat $f \in F$ can be expressed by $d-k$ linear equations of the form

$$x_i = \sum_{j=1}^{k} a_{ij}(f)x_{d-k+j} + b_i(f), \tag{3}$$

for $i = 1, \ldots, d-k$. Let $A(f)$ denote the $(d-k) \times k$ matrix of the coefficients $a_{ij}(f)$, and let $b(f)$ denote the $(d-k)$-dimensional vector $(b_1(f), \ldots, b_{d-k}(f))$.

For each flat $f \in F$, the condition that $f_\sigma(q)$ lies in $K_\sigma(q)$ is equivalent to the condition that $f_\sigma(q) - q$ lies in K_σ (within E_σ). The point $f_\sigma(q)$ is obtained by substituting in (3) the last k coordinates of q. To simplify the notation, add the vector $b(f)$ as a last column of $A(f)$ (and continue to denote the matrix as $A(f)$). Then $f_\sigma(q) = A(f)q^*$, where q^* is the $(k+1)$-dimensional vector whose first k coordinates are the last k coordinates of q, and whose last coordinate is 1.

Hence, the condition that $f_\sigma(q) - q$ lies in K_σ is

$$u_j^T(A(f)q^* - q) \geq 0, \qquad \text{for } j = 1, \ldots, d-k. \tag{4}$$

Let u_{d-k+1} denote the outward normal of σ within E_σ. In analogy with Section 2, we construct a $(d-k+1)$-level partition tree, whose first $d-k$ levels are used to collect the set F_q of k-flats f that satisfy (4), and whose bottommost level is used to determine the flat $f \in F_q$ that minimizes the linear function $f_\sigma(q) \cdot u_{d-k+1} = u_{d-k+1}^T A(f) \cdot q^*$ in F_q.

Notice that the intrinsic dimension at each level is only $k+1$, as we represent each $f \in F$ by the $d-k+1$ $(k+1)$-dimensional vectors:

$$c_j(f) = u_j^T A(f), \quad \text{for } j = 1, \ldots, d-k+1.$$

Since the vectors u_j, for $1 \leq j \leq d-k+1$, are fixed, each vector $c_j(f)$ is a linear expression in $A(f)$, independent of the query q.

We thus prepare a $(d-k+1)$-level $(k+1)$-dimensional partition tree, each of whose levels corresponds to a $(k+1)$-dimensional halfspace range-searching data structure. Again, we fix a parameter m with $n \leq m \leq n^{k+1}$, and construct a partition tree of size $O(m)$ in $O(m \log m)$ expected time, using Chan's algorithm [16] over the set $\{c_1(f) \mid f \in F\} \subset \mathbb{R}^{k+1}$. Suppose we have constructed $j-1$ levels of the data structure, for $2 \leq j \leq d-k$. For each canonical subset F' of the $(j-1)$-level of the data structure, we construct a partition tree, using Chan's algorithm, over the set $\{c_j(f) \mid f \in F'\} \subset \mathbb{R}^{k+1}$. Finally, for each canonical node F' of level $d-k$, we again construct a partition tree on the point set $\{c_{d-k+1}(f) \mid f \in F'\} \subset \mathbb{R}^{k+1}$ so that, for a query vector $u \in \mathbb{R}^{k+1}$, the minimal point in direction u, i.e., $f_u = \arg\min_{f \in F'} c_{d-k+1}(f) \cdot u$ is returned. The overall preprocessing time and size of the data structure are $\widetilde{O}(m)$ [2, 16].

Answering queries. Given a query point q, we query with q, for each $(d-k-1)$-face σ of Q, the corresponding structure $\mathcal{D}_\sigma$, so as to find the flat $f \in F$ that satisfies (4) and (among all such flats) attains the minimum value $c_{d-k+1}(f) \cdot q^* - u_{d-k+1} \cdot q$.

Specifically, at each level $1 \le j \le d-k$, we access each of its structures, built over the canonical sets that the query retrieved at the preceding level $j-1$, and query it with the halfspace $c_j(f) \cdot q^* \ge u_j \cdot q$. As a result, after accessing all levels $j = 1, \ldots, d-k$, we obtain a compact representation of the above set F_q of flats $f \in F$ that satisfy (4), as a union of canonical sets that are stored at the $(d-k)$-level. We then query, for each of these canonical sets $F' \subseteq F_q$, its $(d-k+1)$-level structure, so as to find the flat $f \in F'$ that minimizes the objective function $c_{d-k+1}(f) \cdot q^* - u_{d-k+1} \cdot q$, and return the flat f_σ that attains the overall minimum value, along with that value, which is in fact equal to $\lambda_\sigma(q) = \lambda_\sigma(q, f_\sigma)$, as defined above. Note that f_σ exists if and only if (4) is satisfied for at least one $f \in F$. If this process has failed to find any flat $f \in F_q$, we make f_σ undefined, and return $\lambda_\sigma(q) = +\infty$. Nevertheless, there always exists at least one $(d-k-1)$-face σ of Q for which f_σ exists, so at least one of the output values $\lambda_\sigma(q, f_\sigma)$ will be finite.

We iterate this process for each $(d-k-1)$-face σ of Q, and return the flat f_σ with the minimum corresponding scaling factor $\lambda_\sigma(q, f_\sigma)$; as just observed, this minimum is always finite, so the output flat is always well defined (and is unique for a generic query q).

Using the standard results on multi-level partition-trees and on halfspace range searching [2, 16, 28], the overall size and preprocessing time of the data structure are $\widetilde{O}(m)$ and a query can be answered in $\widetilde{O}(n/m^{1/d})$ for every face of Q. Summing this bound over all faces of Q, we obtain the following general result for exact NN-search with respect to a polyhedral distance functions.

▶ **Theorem 2.** *Let $d \ge 2$ be a constant, let $0 \le k \le d-1$, let Q be a convex polytope in $\mathbb{R}^d$ with γ faces of dimension $d-k-1$. For a given parameter m with $n \le m \le n^{k+1}$, a given set F of n k-flats in $\mathbb{R}^d$ can be preprocessed in $\widetilde{O}(\gamma m)$ expected time into a data structure of $\widetilde{O}(\gamma m)$ size, so that, for a query point $q \in \mathbb{R}^d$, a flat $f \in F$ that attains the smallest Q-distance $d_Q(q, f)$ can be reported in $\widetilde{O}\left(\gamma\left(n/m^{\frac{1}{k+1}}\right)\right)$ time.*

Back to Euclidean ANN-search. We now apply the machinery just derived to obtain an efficient solution to the Euclidean ANN-search problem. Given $\varepsilon > 0$, we take a convex centrally symmetric polytope Q_ε that approximates the Euclidean ball, in the sense that its corresponding Q_ε-distance function satisfies (1). Recall that, using Dudley's bound, we can take Q_ε to have either $O\left((1/\varepsilon)^{\frac{d-1}{2}}\right)$ vertices or $O\left((1/\varepsilon)^{\frac{d-1}{2}}\right)$ facets.

The maximum number γ of $(d-k-1)$-faces of such a polytope Q_ε satisfies

$$\gamma = O\left((1/\varepsilon)^{\frac{d-1}{2}\min(d-k,k+1)}\right).$$

In this bound, we use a polytope Q_ε with a small number of facets (resp., vertices) when $k+1 \le d-k$ (resp., $k+1 > d-k$).

Plugging this into Theorem 2 finally yields Theorem 1. □

4 Discussion and Open Problems

Our data structure answers ANN queries amid a set F of k-flats in $\mathbb{R}^d$ by answering exact NN queries amid F with respect to a suitable polyhedral Q-metric. The most obvious direction towards further improving the bounds of Theorem 1 is to replace the exact NN-search under the Q-norm by some approximate version. Ideally, this would allow us to avoid the use of the

fairly expensive halfspace range searching structures. Unfortunately, our parametrization of k-flats by $(k+1)$-dimensional points does not preserve distances, so the existing machinery of approximate range searching, such as in [10], does not directly apply.

The most interesting instance of the problem involves lines in $\mathbb{R}^d$. Notice that our fast structure, using only nearly quadratic storage in n, does not yield an approximate Voronoi diagram whose description complexity is also nearly quadratic. A challenging open problem is whether an approximate Voronoi diagram of near-quadratic size exists for a set of lines in $\mathbb{R}^d$, for $d > 3$. More generally, does an approximate Voronoi diagram of size $O(n^{k+1})$ exist for a set of k-dimensional flats in $\mathbb{R}^d$, for $d > k$?

Acknowledgement. The authors thank Sariel Har-Peled for helpful discussions. We also thank ESA referees for the helpful suggestions which will be incorporated into the full version of the paper.

References

1 Pankaj K. Agarwal. Geometric range searching. In J. E. Goodman, J. O'Rourke, and Cs. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 3rd edition, 2017 (to appear).

2 Pankaj K. Agarwal. Simplex range searching. In M. Loebl, J. Nešetril, and R. Thomas, editors, *Journey Through Discrete Mathematics*. Springer, Heidelberg, to appear.

3 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008. doi:10.1145/1327452.1327494.

4 Alexandr Andoni and Piotr Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman, J. O'Rourke, and Cs. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, Boca Raton, FL, 3rd edition, 2017 (to appear).

5 Alexandr Andoni, Piotr Indyk, Robert Krauthgamer, and Huy L. Nguyen. Approximate line nearest neighbor in high dimensions. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'09, pages 293–301, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1496770.1496803.

6 Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, pages 1018–1028, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2634074.2634150.

7 Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Optimal approximate polytope membership. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'17, pages 270–288, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=3039686.3039704.

8 Sunil Arya and Theocharis Malamatos. Linear-size approximate voronoi diagrams. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'02, pages 147–155, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=545381.545400.

9 Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-efficient approximate voronoi diagrams. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC'02, pages 721–730, New York, NY, USA, 2002. ACM. doi:10.1145/509907.510011.

10 Sunil Arya and David M. Mount. Approximate range searching. *Computational Geometry*, 17(3):135–152, 2000. doi:10.1016/S0925-7721(00)00022-5.

11 Sunil Arya and David M. Mount. Computational geometry: Proximity and location. In D. P. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, chapter 3. Chapman and Hall/CRC, Boca Raton, FL, 2004.

12 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, November 1998. doi:10.1145/293347.293348.

13 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1st edition, 2013.

14 Ronen Basri, Tal Hassner, and Lihi Zelnik-Manor. Approximate nearest subspace search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):266–278, February 2011. doi:10.1109/TPAMI.2010.110.

15 Marshall Bern. Approximate closest-point queries in high dimensions. *Information Processing Letters*, 45(2):95–99, 1993. doi:10.1016/0020-0190(93)90222-U.

16 Timothy M. Chan. Optimal partition trees. *Discrete Comput. Geom.*, 47(4):661–690, June 2012. doi:10.1007/s00454-012-9410-z.

17 L. Paul Chew, Klara Kedem, Micha Sharir, Boaz Tagansky, and Emo Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *Journal of Algorithms*, 29(2):238–255, 1998. doi:10.1006/jagm.1998.0957.

18 R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974. doi:10.1016/0021-9045(74)90120-8.

19 Rex A. Dwyer. Higher-dimensional voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(3):343–367, Sep 1991. doi:10.1007/BF02574694.

20 S. Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS'01, pages 94–, Washington, DC, USA, 2001. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=874063.875592.

21 Sariel Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, Boston, MA, USA, 2011.

22 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC'98, pages 604–613, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276876.

23 Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

24 Vladlen Koltun and Micha Sharir. 3-dimensionall euclidean voronoi diagrams of lines with a fixed number of orientations. *SIAM J. Comput.*, 32(3):616–642, March 2003. doi:10.1137/S0097539702408387.

25 Vladlen Koltun and Micha Sharir. Polyhedral voronoi diagrams of polyhedra in three dimensions. *Discrete & Computational Geometry*, 31(1):83–124, Jan 2004. doi:10.1007/s00454-003-2950-5.

26 Avner Magen. Dimensionality reductions in ℓ2 that preserve volumes and distance to affine spaces. *Discrete Comput. Geom.*, 38(1):139–153, July 2007. doi:10.1007/s00454-007-1329-4.

27 Sepideh Mahabadi. Approximate nearest line search in high dimensions. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 337–354, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2722129.2722154.

28 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(1):157–182, December 1993. doi:10.1007/BF02573972.

29 Wolfgang Mulzer, Huy L. Nguyên, Paul Seiferth, and Yannik Stein. Approximate k-flat nearest neighbor search. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, pages 783–792, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746559.

30 Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.