Dynamic Nearest-Neighbor Searching Under General Metrics in $\mathbb{R}^3$ and Its Applications

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham, NC, USA

Matthew J. Katz
The Stein Faculty of Computer and Information Science,
Ben-Gurion University of the Negev, Beer Sheva, Israel

Micha Sharir
School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Abstract

Let K be a compact, centrally-symmetric, strictly-convex region in $\mathbb{R}^3$, which is a semi-algebraic set of constant complexity, i.e. the unit ball of a corresponding metric, denoted as $\|\cdot\|_K$. Let $\mathcal{K}$ be a set of n homothetic copies of K. This paper contains two main sets of results:

(i) For a storage parameter $s \in [n, n^3]$, $\mathcal{K}$ can be preprocessed in $O^*(s)$ expected time into a data structure of size $O^*(s)$, so that for a query homothet K_0 of K, an intersection-detection query (determine whether K_0 intersects any member of $\mathcal{K}$, and if so, report such a member) or a nearest-neighbor query (return the member of $\mathcal{K}$ whose $\|\cdot\|_K$-distance from K_0 is smallest) can be answered in $O^*(n/s^{1/3})$ time; all k homothets of $\mathcal{K}$ intersecting K_0 can be reported in additional $O(k)$ time. In addition, the data structure supports insertions/deletions in $O^*(s/n)$ amortized expected time per operation. Here the $O^*(\cdot)$ notation hides factors of the form n^ε, where $\varepsilon > 0$ is an arbitrarily small constant, and the constant of proportionality depends on ε.

(ii) Let $\mathcal{G}(\mathcal{K})$ denote the intersection graph of $\mathcal{K}$. Using the above data structure, breadth-first or depth-first search on $\mathcal{G}(\mathcal{K})$ can be performed in $O^*(n^{3/2})$ expected time. Combining this result with the so-called shrink-and-bifurcate technique, the reverse-shortest-path problem in a suitably defined proximity graph of $\mathcal{K}$ can be solved in $O^*(n^{62/39})$ expected time. Dijkstra's shortest-path algorithm, as well as Prim's MST algorithm, on a $\|\cdot\|_K$-proximity graph on n points in $\mathbb{R}^3$, with edges weighted by $\|\cdot\|_K$, can also be performed in $O^*(n^{3/2})$ time.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Homothets, Minkowski metric, Shallow cuttings, Nearest-neighbor searching, Intersection and proximity graphs, Reverse-shortest-path problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.4

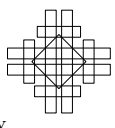
Related Version *Full Version*: `https://arxiv.org/abs/2603.26585`

Funding *Pankaj K. Agarwal*: Partially supported by NSF grants CCF-2223870 and IIS-2402823, and by a US-Israel Binational Science Foundation Grant 2022/131.
Matthew J. Katz: Partially supported by Israel Science Foundation Grant 495/23.
Micha Sharir: Partially supported by Israel Science Foundation Grant 495/23.

1 Introduction

Problem statement. Let K be a compact, centrally-symmetric, strictly-convex region in $\mathbb{R}^3$, which is a semi-algebraic set of constant complexity.[1] K is the unit ball of a norm $\|\cdot\|_K$, and we denote its metric as dist_K.

[1] Roughly speaking, a semi-algebraic set in $\mathbb{R}^d$ is the set of points in $\mathbb{R}^d$ that satisfy a Boolean formula over a set of polynomial inequalities; the complexity of a semi-algebraic set is the number of polynomials defining the set and their maximum degree. See [12] for formal definitions of a semi-algebraic set and its complexity.



A *homothet* (or *homothetic copy*) of K is a scaled and translated copy of K, expressed as $K(c,\rho) := \rho K + c$, for $c \in \mathbb{R}^3$ and $\rho \geq 0$; c is the *center* of $K(c,\rho)$ and ρ is its *size*. We represent $K(c,\rho)$ as the point $(c,\rho) \in \mathbb{R}^3 \times \mathbb{R}_{\geq 0}$. Let $\mathbb{K}$ be the set of all homothets of K, represented as the halfspace $x_4 \geq 0$ of $\mathbb{R}^4$. The *K-distance* $\mathrm{dist}_K(u,v) = \|u-v\|_K$ between points $u, v \in \mathbb{R}^3$ is the minimum value of ρ for which $v \in K(u,\rho)$ (or, equivalently, $u \in K(v,\rho)$). Since K is strictly convex, the triangle inequality $\mathrm{dist}_K(u,v) \leq \mathrm{dist}_K(u,w) + \mathrm{dist}_K(w,v)$ is sharp unless w lies on the segment uv. The K-distance of a point $x \in \mathbb{R}^3$ from a homothet $K(c,\rho)$ is defined as $f_{c,\rho}(x) = \mathrm{dist}_K(c,x) - \rho$, and the K distance of another homothet $K(c',\rho')$ from $K(c,\rho)$ is $f_{c,\rho}(c') - \rho' = \mathrm{dist}_K(c,c') - \rho - \rho'$. Since K is a constant-complexity semi-algebraic set, $f_{c,\rho}$ is a semi-algebraic function of constant complexity. The homothets $K(c,\rho)$ and $K(c',\rho')$ intersect if the distance between them is non-positive, i.e., $\mathrm{dist}_K(c,c') \leq \rho + \rho'$, or equivalently, the point (c',ρ') lies above the graph of $f_{c,\rho}$ in $\mathbb{R}^4$, or vice-versa.

Let $\mathcal{K} = \{K_1, \ldots, K_n\} \subset \mathbb{K}$ be a set of n homothets of K, where $K_i = K(c_i, \rho_i)$. In this paper we study two classes of proximity problems related to $\mathcal{K}$.

Proximity queries. We wish to preprocess $\mathcal{K}$ into a data structure so that various proximity queries for a query homothet K_0, such as intersection-detection (determine whether K_0 intersects any homothet of $\mathcal{K}$; if the answer is yes, return one such intersecting homothet of $\mathcal{K}$), intersection reporting (report all homothets of $\mathcal{K}$ that intersect K_0), and nearest-neighbor queries (report the homothet of $\mathcal{K}$ at the minimum K-distance from K_0), can be performed efficiently. In addition, the data structure should support insertions and deletions of homothets of K.

Searching in the intersection graph and related problems. Let $\mathcal{G}(\mathcal{K})$ denote the intersection graph of $\mathcal{K}$, whose vertices are the homothets of $\mathcal{K}$ and (K_i, K_j) is an edge if $K_i \cap K_j \neq \emptyset$. Similarly, for a threshold parameter r_0, the *r_0-proximity graph* $\mathcal{G}_{r_0}(\mathcal{K})$ of $\mathcal{K}$ consists of those edges (K_i, K_j) for which $\mathrm{dist}_K(K_i, K_j) \leq r_0$. We study efficient implementation of graph-search problems on $\mathcal{G}(\mathcal{K})$, like BFS or DFS. We also study the *reverse shortest path* (RSP) problem on $\mathcal{K}$: given two elements $K_i, K_j \in \mathcal{K}$ and a parameter $1 \leq k < n$, compute the smallest value r^* for which $\mathcal{G}_{r^*}(\mathcal{K})$ contains a path from K_i to K_j with at most k edges.

Related work. There is extensive work on nearest-neighbor (NN) searching in many different fields including computational geometry, database systems, and machine learning. We refer the readers to various survey papers [4, 10, 20] for a general overview of known results on this topic. Here we briefly mention a few results that are most closely related to the problems studied in this paper. Agarwal and Matoušek [9] presented a dynamic NN data structure for a point set in $\mathbb{R}^d$, under the Euclidean metric, that answers a query in $O^*(n/s^\phi)$ time, where $\phi = 1 - \frac{1}{\lceil d/2 \rceil}$ and $s \in [n, n^{\lceil d/2 \rceil}]$ is a storage parameter, using $O^*(s)$ space and preprocessing; the (amortized) update time for insertions/deletions of points is $O^*(s/n)$. The bounds were slightly improved – n^ε factors were replaced by $\log^{O(1)} n$ factors – in [15,26]. Kaplan *et al.* [26] presented a dynamic NN data structure for a set S of points in $\mathbb{R}^2$ under fairly general distance functions. Its performance depends on the complexity of the Voronoi diagram of S under that distance function. For the Minkowski metric induced by a centrally-symmetric convex region, which is a semi-algebraic set of constant complexity, their data structure can answer an NN query in $O(\log^2 n)$ time, and handle insertion and deletion of a point in $O(\beta(n)\log^5 n)$ and $O(\beta(n)\log^9 n)$ amortized expected time, respectively, where $\beta(n)$, a variant of inverse Ackermann's function, is an extremely slowly growing function; see also [31] for slightly improved bounds. These data structures, however, do not extend to $\mathbb{R}^3$.

The problem of NN searching under general distance functions in $\mathbb{R}^3$ is much less understood. By reducing NN queries to ball-intersection queries [8] and using semi-algebraic range-searching data structures [1, 2, 33], an NN query amid the set $\mathcal{K}$ of n homothets in $\mathbb{R}^3$, as above, can be answered in $O^*(1)$ time using $O^*(n^4)$ space and preprocessing, or in $O^*(n^{3/4})$ time using $O(n)$ space and preprocessing. Using recent standard techniques (see, e.g., the Appendix in [1]), one can obtain a space/query-time trade-off, as well as handle insertions/deletions of objects. Using a recent result by Agarwal *et al.* [5] on the vertical decomposition of the lower envelope of trivariate functions, an NN query amid $\mathcal{K}$ can be answered in $O(\log^2 n)$ time using only $O^*(n^3)$ space, but this data structure does not support efficient deletions (insertions can be handled in a standard manner, using the dynamization technique of Bentley and Saxe [13]). Furthermore, it was not known whether an NN query amid $\mathcal{K}$ can be answered in $O^*(n^{2/3})$ time using $O^*(n)$ space.

Motivated by numerous applications, there has been work on developing fast algorithms for intersection and proximity graphs of n geometric objects. Although the intersection graph may have $\Theta(n^2)$ edges, the goal in this line of work is to develop algorithms, by exploiting the underlying geometry, that run in $O^*(n)$ time (or at least in subquadratic time). Cabello and Jejčič [14], and subsequently Chan and Skrepetos [21], presented $O(n \log n)$ implementations of BFS in unit-disk intersection graphs, which was recently extended to general disk-intersection graphs by de Berg and Cabello [22]; see also [25, 26, 29]. The algorithm in [22] can also perform DFS and Dijkstra's algorithm in $O(n \log n)$ time. Katz *et al.* [28] showed that BFS in ball-intersection graphs (for congruent or arbitrary Euclidean balls) can be implemented in $O^*(n^{2\phi/(\phi+1)})$ time, where $\phi = \lfloor d/2 \rfloor + 1$, so in $O^*(n^{4/3})$ time for $d = 3$. The problem of devising efficient implementations for BFS/DFS in more general graphs in the plane also has received some attention, see, e.g., [6, 7, 27].

The reverse shortest path (RSP) problem for unit disks was studied by Wang and Zhao [36], who gave an $O^*(n^{5/4})$-time solution. Kaplan *et al.* [25] improved the runtime to $O^*(n^{6/5})$ using the *shrink-and-bifurcate* technique (see [11]), and it was recently improved to $O^*(n^{9/8})$ by Chan and Huang [17]. The best-known RSP algorithm for arbitrary disks runs in $O^*(n^{6/5})$ expected time. Katz *et al.* [28] presented RSP algorithms for balls in $\mathbb{R}^3$ – with $O^*(n^{29/21})$ time for congruent balls and $O^*(n^{56/39})$ for arbitrary balls. The RSP problem also has been studied for other geometric objects in $\mathbb{R}^2$ [7, 17, 28].

Our results. Our main result is a dynamic data structure for answering intersection and NN queries with a homothet of $\mathbb{K}$ amid the set $\mathcal{K}$ of homothets:

▶ **Theorem 1.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. For a storage parameter $s \in [n, n^3]$, $\mathcal{K}$ can be maintained in a dynamic data structure with $O^*(s)$ storage, that can answer an intersection-detection or NN-query (under the K-distance) with a homothet K_0 of K in $O^*(n/s^{1/3})$ time and that supports insertions/deletions in $O^*(s/n)$ amortized expected time. It can report all k objects of $\mathcal{K}$ intersecting K_0 in additional $O(k)$ time.*

Plugging Theorem 1 into Eppstein's technique [23], or its enhancement by Chan [16], for maintaining bichromatic closest pairs, using $s = n^{3/2}$, we obtain:

▶ **Corollary 2.** *Let K be a compact, centrally-symmetric strictly-convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}, \mathcal{K}'$ be two sets of homothets of K of combined size n. The K-closest pair between $\mathcal{K}$ and $\mathcal{K}'$, under insertions/deletions of homothets (in $\mathcal{K}$ and $\mathcal{K}'$), can be maintained in $O^*(n^{1/2})$ amortized expected time per update.*

Theorem 1 is obtained by presenting two data structures. The first one (see Section 4.1) uses $O^*(n^3)$ space, answers a query in $O^*(1)$ time, and handles an insertion/deletion in $O^*(n^2)$ amortized expected time. (The data structure in [5] also obtains a similar space/query-time bound but it cannot handle deletions efficiently.) This data structure is based on constructing *vertical shallow cuttings*, a notion originally introduced in [15,18], of the graphs of the distance functions of $\mathcal{K}$. Roughly speaking, for a parameter $t \geq 1$, a *vertical* (n/t)*-shallow cutting* of $\mathcal{K}$ is a collection of pairwise openly disjoint semi-unbounded *pseudo-prisms* that covers the region lying below the k-level of the arrangement of $\mathcal{K}$, where each prism, consisting of all points that lie vertically below a constant-complexity semi-algebraic 3-dimensional region, intersects the graphs of only $O(n/t)$ functions of $\mathcal{K}$. The random-sampling based technique used in [15, 26] to construct a vertical shallow cutting does not immediately extend to our setting because one needs to decompose the q-level, for some parameter $q > 0$, in the 4D arrangement of the distance functions of $\mathcal{K}$ into $O^*(n^3(q+1))$ constant-complexity cells. Although a recent result [5] constructs such a decomposition of the 0-level (with $O^*(n^3)$ cells), it does not extend to larger values of q. This problem remains elusive for arrangements of arbitrary trivariate semi-algebraic functions of constant complexity, but we develop a desired decomposition technique for our setting (in Section 2) by exploiting the structural geometric properties of the distance functions of $\mathcal{K}$. This result is one of the main technical contributions of the paper. Using our result on the decomposition of the q-level, we construct a vertical (n/t)-shallow cutting of $\mathcal{K}$ of size $O^*(t^3)$, for any t (Section 3).

The second data structure is a linear-size partition tree, with $O^*(n^{2/3})$ query time, constructed on the point set $\mathcal{K}^* = \{(c_i, \rho_i) \mid 1 \leq i \leq n\} \subset \mathbb{R}^4$, that answers intersection-detection queries on $\mathcal{K}$ with a homothet of $\mathbb{K}$ as a query. The main technical challenge we face here is the construction of a so-called *test set* of $\mathcal{K}$ (see [35] and below) of size $r^{O(1)}$, which represents well the distance functions of all (n/r)*-shallow* homothets in $\mathbb{K}$, i.e., the homothets that intersect at most n/r elements of $\mathcal{K}$. By adapting the technique in [35] and again exploiting the structural geometric properties of the distance functions, we show that a test set, of size $O^*(r^4)$, with the desired properties can be constructed efficiently.

Using Theorem 1 and Corollary 2, we obtain our second set of results:

▶ **Theorem 3.**

(a) *Given a set $\mathcal{K}$ of n homothets of a constant-complexity, strictly convex, centrally-symmetric semi-algebraic set K, BFS or DFS in the intersection graph of $\mathcal{K}$ can be performed in $O^*(n^{3/2})$ expected time.*

(b) *The single-source shortest-path tree as well as the minimum spanning forest of the r_0-proximity graph of a set of n points, for any threshold parameter r_0, weighted by the pairwise K-distances, can be computed in $O^*(n^{3/2})$ time.*

Combining Theorem 3(a) with the shrink-and-bifurcate technique [11, 25] yields:

▶ **Theorem 4.** *Given a set $\mathcal{K}$ of n homothets of a constant-complexity, strictly convex, centrally-symmetric semi-algebraic set K, two designated elements $K, K' \in \mathcal{K}$, and an integer $k < n$, the RSP problem on $\mathcal{K}$, of finding the smallest r^* for which $G_{r^*}(\mathcal{K})$ contains a path between K and K' of length at most k, can be solved in $O^*(n^{62/39})$ expected time.*

Because of lack of space, the proofs of Theorems 3 and 4 and of some of the lemmas are omitted from this version.

2 Decomposing the k-Level of an Arrangement of Distance Functions

Preliminaries. Let $\mathcal{K} = \{K_1, \ldots, K_n\}$, where each K_i is a homothetic copy of K represented by a point $K_i^* = (c_i, \rho_i)$ in $\mathbb{R}^4$. For $1 \le i \le n$, set $f_i = f_{c_i,\rho_i}$, and let $\mathcal{F} := \mathcal{F}(\mathcal{K}) = \{f_i \mid 1 \le i \le n\}$. (As noted, distances from points x that lie inside some copy K_i, still measured by $f_i(x)$, are nonpositive.) We will not distinguish between a function of $\mathcal{F}$ and its graph. For a point $x \in \mathbb{R}^3$, K_i is the *nearest neighbor* of x in $\mathcal{K}$ (under the K-distance dist_K), i.e., $i = \arg\min_{1\le j\le n} f_j(x)$, if f_i appears on the lower envelope of $\mathcal{F}$ at x.

The *level* of a point $p \in \mathbb{R}^4$ in $\mathcal{A}(\mathcal{F})$ is the number of functions in $\mathcal{F}$ whose graphs lie below p. For a parameter $k < n$, the *k-level* of $\mathcal{A}(\mathcal{F})$, denoted $\mathcal{A}_k(\mathcal{F})$, is the (closure of the) locus of all points on $\bigcup\mathcal{F}$ whose level is k. We define the *$(\le k)$-level* of $\mathcal{A}(\mathcal{F})$, denoted $\mathcal{A}_{\le k}(\mathcal{F})$, to be the (closure of the) set of all points in $\mathbb{R}^4$ of level at most k, i.e., the set of points that lie on or below $\mathcal{A}_k(\mathcal{F})$ (not necessarily on $\cup\mathcal{F}$). The projection of $\mathcal{A}_k(\mathcal{F})$ onto $\mathbb{R}^3$, denoted by $\mathcal{M}_k = \mathcal{M}_k(\mathcal{K})$, is a subdivision of $\mathbb{R}^3$. If a cell $\tau^\downarrow$ of $\mathcal{M}_k$ is the projection of a cell τ of $\mathcal{A}_k(\mathcal{F})$ that lies on the graph of f_i, then K_i is the k-th nearest neighbor in $\mathcal{K}$ (under the K-distance function) of all points in $\tau^\downarrow$. Note that $A_0(\mathcal{F})$ is the lower envelope of $\mathcal{F}$, and $\mathcal{M}_0(\mathcal{K})$ is the Voronoi diagram of $\mathcal{K}$ (under the K-distance).

The decomposition. The main technical result of this section is that $\mathcal{A}_k(\mathcal{F})$ can be decomposed into $O^*(n^3(k+1))$ *elementary cells*, namely, constant-complexity semi-algebraic regions, each homeomorphic to a ball; see below for a more precise definition.

For $1 \le i \ne j \le n$, let $B_{ij} = \{x \in \mathbb{R}^3 \mid f_i(x) = f_j(x)\}$ be the *bisector* of K_i and K_j under the K-distance. Note that if $K_i \subset K_j$ then, as is easily verified, $f_j(x) < f_i(x)$ for all $x \in \mathbb{R}^3$ and B_{ij} is undefined. The following lemma gives a useful property of bisectors.

▶ **Lemma 5.** *Fix a homothet $K_i \in \mathcal{K}$. Let λ be a ray in $\mathbb{R}^3$ emanating from c_i. For any $1 \le j \ne i \le n$ such that $K_i \not\subset K_j$, λ intersects B_{ij} in at most one point, say, ξ. Furthermore, $f_i(x) \le f_j(x)$ for all $x \in c_i\xi$ (where $c_i\xi$ denotes the segment with endpoints c_i and ξ) and $f_j(x) > f_i(x)$ for all $x \in \lambda \setminus c_i\xi$.*

We are now ready to describe the decomposition of the cells of $\mathcal{A}_k(\mathcal{F})$ into elementary cells. (Our ultimate goal is to decompose the region below $\mathcal{A}_k(\mathcal{F})$ into elementary cells, but we start with this subtask.) For $1 \le i \le n$, let $\mathcal{A}_k^i = \mathcal{A}_k^i(\mathcal{F})$ denote the subset of cells of $\mathcal{A}_k(\mathcal{F})$ that lie on the graph of f_i, and let $\mathcal{M}_k^i$ denote their projections (which are the cells of $\mathcal{M}_k$ for which K_i is the k-th nearest neighbor). We describe the algorithm for decomposing the cells of $\mathcal{M}_k^i$ (or of $\mathcal{A}_k^i$). Lemma 5 implies that, for any $j \ne i$, B_{ij} can be viewed as the graph of a function $g_{ij} : \mathbb{S}^2 \to \mathbb{R}_{\ge 0}$ in spherical coordinates with c_i as the origin. That is, for a given $u \in \mathbb{S}^2$, let $\lambda(u)$ be the ray emanating from c_i in direction u, and let $x_{ij}(u)$ be the intersection point of $\lambda(u)$ with the bisector B_{ij} if such an intersection point exists; otherwise $x_{ij}(u)$ is undefined. We set $g_{ij}(u) = f_j(x_{ij}(u))$ if $x_{ij}(u)$ is defined and $+\infty$ otherwise. Let $G^{(i)} = \{g_{ij} \mid 1 \le j \ne i \le n\}$. We define the *level* of a point $x \in \mathbb{R}^3$ in (the 3D arrangement, within $\mathbb{S}^2 \times \mathbb{R}$) $\mathcal{A}(G^{(i)})$ to be the number of functions in $G^{(i)}$ whose graphs intersect the (relatively open) segment c_ix. The following lemma will be useful in analyzing the complexity of the decomposition of $\mathcal{A}_k^i$.

▶ **Lemma 6.** *For a parameter $k < n$, let τ be a 3D cell of $\mathcal{A}_k^i$. Then $\tau^\downarrow$ is a cell of level at most k in $\mathcal{A}(G^{(i)})$.*

Proof. It is obvious from the definition of $G^{(i)}$ that $\tau^\downarrow$ is a cell in $\mathcal{A}(G^{(i)})$. We now argue that its level in $\mathcal{A}(G^{(i)})$ is at most k. Choose a point $x \in \tau^\downarrow$. Let λ be the ray emanating from c_i and passing through x. Let g_{ij} be a function whose graph intersects the segment

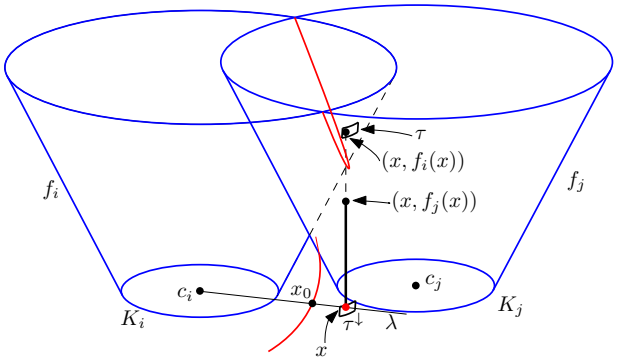


Figure 1 A bisector crossing λ. (The figure is drawn in the 3D c-space.)

$c_i x$, say, at a point $x_0 \in B_{ij}$. (See Fig. 1.) By Lemma 5, $f_i(y) < f_j(y)$ for all points y on λ preceding x_0, and $f_j(y) < f_i(y)$ for all points on λ beyond x_0. In particular, $f_i(x) > f_j(x)$. Since τ is a cell of $\mathcal{A}(\mathcal{F})$, f_j does not intersect τ, so f_j lies below τ in $\mathbb{R}^4$ (in the x_4-, or rather ρ-direction). Furthermore, $\tau \in \mathcal{A}_k(\mathcal{F})$, so there are exactly k functions of $\mathcal{F}$ whose graphs lie below τ, implying that the level of $\tau^\downarrow$ in $\mathcal{A}(G^{(i)})$ is at most k, as claimed. ◀

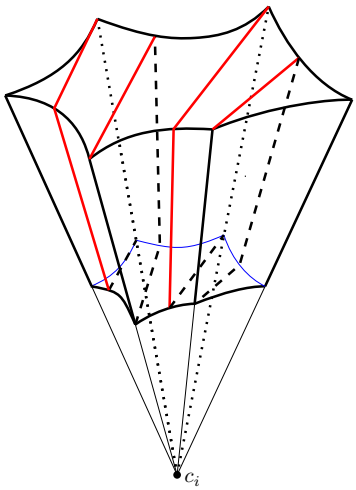


Figure 2 Vertical decomposition of a cell in the spherical coordinate system.

We decompose the cells τ of $\mathcal{A}_k^{(i)}(\mathcal{F})$ by constructing "vertical decompositions" of the corresponding 3D projections $\tau^\downarrow$ in the spherical coordinate system around c_i. As for standard vertical decompositions in 3D, we proceed in two stages. The first stage erects a wall from each edge of τ as follows. Fix an edge e of $\tau^\downarrow$. For a point $x \in e$, let $\lambda(x)$ be the ray emanating from c_i and passing through x, and let $h(x) = \lambda(x) \cap \tau^\downarrow$. By Lemma 5, $h(x)$ is a (connected) segment that touches e at one of its endpoints – it is either the top endpoint, for all points on e, or the bottom endpoint (with respect to distances from c_i). We erect the wall $\hat{e} = \bigcup_{x \in e} h(x)$ in $\tau^\downarrow$; parts of the wall may extend to infinity when $\tau^\downarrow$ is unbounded and e is a bottom edge. We repeat this step for all edges of $\tau^\downarrow$. These walls decompose $\tau^\downarrow$ into *frustums* (truncated cones), each of which, denoted by Δ, has a unique pair of *front* and *back* faces; the other faces of Δ lie on the created walls. (See Fig. 2.) The projections of the front and back faces of Δ on $\mathbb{S}^2$ (with c_i as the origin) are identical, and we denote this identical

projection by $\Delta^{\downarrow}$. The complexity of $\Delta^{\downarrow}$ may be arbitrarily large (a typical situation in the first stage of vertical decompositions in general; see, e.g., [19]). The subcell Δ itself is of the form $\Delta = \{(u, r) \mid u \in \Delta^{\downarrow}, r \in [g^-(u), g^+(u)]\}$, where g^-, g^+ are the functions of $G^{(i)}$ whose graphs contain the front and back faces of Δ, respectively.

The second stage decomposes Δ into constant-complexity frustums, which we refer to as *pseudo-cones*. We decompose $\Delta^{\downarrow}$ into spherical pseudo-trapezoids, by drawing portions of meridians within $\Delta^{\downarrow}$ from each vertex and meridian-tangency point. We then lift each pseudo-trapezoid $\sigma^{\downarrow}$ to form the pseudo-cone $\sigma = \{(u, r) \mid u \in \sigma^{\downarrow}, r \in [g^-(u), g^+(u)]\}$ in $\mathbb{R}^3$.

Repeating this step for all frustums created in the first stage, we obtain a decomposition of $\tau^{\downarrow}$ into pseudo-cones, each being a semi-algebraic set of constant complexity and bounded by up to six facets (compare with standard vertical decompositions in 3D [19]). We refer to these pseudo-cones as *elementary cells*. We note that each pseudo-cone ϕ is *defined* by a subset D_ϕ of at most seven functions of $\mathcal{F}$ (the function f_i and up to six additional functions that define the functions g_{ij} that form the frustum), in the sense that ϕ appears in the vertical decomposition of a 3D cell of $\mathcal{A}(D_\phi)$. By repeating this step for all cells of $\mathcal{M}_k^i$ and for all homothets K_i of $\mathcal{K}$, we obtain a decomposition of $\mathcal{M}_k = \mathcal{M}_k(\mathcal{K})$ into elementary cells. Finally, we lift each elementary cell of this decomposition vertically to $\mathcal{A}_k(\mathcal{F})$ in a straightforward manner, to obtain a corresponding decomposition of $\mathcal{A}_k(\mathcal{F})$.

Kaplan *et al.* [26, Lemma 6.2] have shown that the complexity of the vertical decomposition of the $(\leq k)$-level $\mathcal{A}_{\leq k}(\mathcal{G})$ of a set of $\mathcal{G}$ of m bivariate semi-algebraic functions of constant complexity is $O(k^2\psi(n/k)\lambda_s(k))$, where $\psi(r)$ is the maximum complexity of the lower envelope of a subset of $\mathcal{G}$ of size at most r, $\lambda_s(t)$ is the maximum length of Davenport-Schnizel sequences of order s composed of t symbols [34], and s is a constant depending on the complexity of the functions in $\mathcal{G}$. By applying the argument in [26] to our spherical coordinate system and using the worst-case bound $\psi(m) = O^*(m^2)$ on the complexity of the lower envelope of m constant-complexity bivariate functions [34], we conclude that the cells of $\mathcal{M}_k^i$ can be decomposed into $O^*(n^2k)$ elementary cells. Hence, $\mathcal{A}_k(\mathcal{F})$, or $\mathcal{M}_k(\mathcal{K})$, can be decomposed into $O^*(n^3k)$ elementary cells. By adapting the randomized incremental algorithm described in [26], $\mathcal{M}_k^i$, for each i, can be computed in $O^*(n^2k)$ expected time. We thus obtain:

▶ **Theorem 7.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. The cells of $\mathcal{M}_k(\mathcal{K})$, and the cells of the k-level in the arrangement of their K-distance functions, can be decomposed into $O^*(n^3k)$ elementary cells, in $O^*(n^3k)$ expected time.*

3 Vertical Shallow Cuttings

A key notion that we need for constructing the dynamic NN data structures of Section 4 is that of a vertical shallow cutting of $\mathcal{A}(\mathcal{F})$, as studied in [15, 18, 26] (for simpler scenarios). For a parameter k, a *vertical k-shallow cutting* of $\mathcal{F}$ is a collection of pairwise openly disjoint semi-unbounded *pseudo-prisms* (prisms for short), where each prism consists of all points that lie vertically below some pseudo-cone, of the decomposition (of Section 2) that covers $\mathcal{A}_{\leq k}(\mathcal{F})$, and is thus a semi-algebraic set of constant complexity. Furthermore, the ceilings of these prisms collectively form an x_4-monotone surface that lies between $\mathcal{A}_k(\mathcal{F})$ and $\mathcal{A}_{2k}(\mathcal{F})$, and each prism is crossed by the graphs of at most $O(k)$ functions of $\mathcal{F}$. The *conflict list* of a prism σ is the set of functions whose graphs cross σ.

Following the ideas in [26], for a parameter $t \in [1, n]$, we construct a vertical (n/t)-shallow cutting as follows. We set two parameters: $r = \beta t \log t$, where β is a sufficiently large constant, independent of t, and $q = \beta \log t = r/t$. We choose a random subset $\mathsf{N} \subseteq \mathcal{F}$ consisting of

r functions, construct $\mathcal{A}_q(\mathsf{N})$, and compute the decomposition of $\mathcal{A}_q(\mathsf{N})$ into a family Ξ of pseudo-cones as described in Section 2. For each pseudo-cone $\tau \in \Xi$, we associate a label $\varphi(\tau)$ which is the function of N whose graph contains τ. Finally, we erect a semi-unbounded prism $\tau^{\uparrow}$ from each pseudo-cone τ of Ξ, given by $\tau^{\uparrow} = \bigcup\{\{c\} \times (-\infty, \rho] \mid (c, \rho) \in \tau\}$. Let $\Xi^{\uparrow}$ be the resulting set of pseudo-prisms. By Theorem 7, with large probability, $|\Xi| = O^*(r^3 q) = O^*(t^3)$. We next show that Ξ is a vertical k-shallow cutting of $\mathcal{F}$, where $k = n/t$, with high probability.

Range spaces and shallow ε-nets. Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a (finite) range space, where X is a set of objects and $\mathcal{R} \subseteq 2^{\mathsf{X}}$ a set of ranges. Let $0 < \varepsilon < 1$ be a given parameter. A subset $\mathsf{N} \subseteq \mathsf{X}$ is called a *shallow ε-net* of Σ if it satisfies the following two properties for every range $R \in \mathcal{R}$ and for any parameter $\zeta \geq 0$:

(i) If $|R \cap \mathsf{N}| \leq \zeta \log \frac{1}{\varepsilon}$ then $|R \cap \mathsf{X}| \leq \alpha(\zeta + 1)\varepsilon|\mathsf{X}|$, and

(ii) If $|R \cap \mathsf{X}| \leq \zeta\varepsilon|\mathsf{X}|$ then $|R \cap \mathsf{N}| \leq \alpha(\zeta + 1) \log \frac{1}{\varepsilon}$.

Here α is a suitable constant. Note the difference between shallow and standard ε-nets: Property (i) (with $\zeta = 0$) implies that a shallow ε-net is also a standard ε-net (possibly with a recalibration of ε). Property (ii) has no parallel in the case of standard ε-nets – there is no guarantee how a standard net interacts with small ranges (of the entire set X). The following result by Sharir and Shaul [35] is a generalization of the result on standard ε-nets [24].

▶ **Lemma 8** (Theorem 2.2 of Sharir and Shaul [35]). *Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a range space with VC-dimension d. With a suitable choice of the constant of proportionality, a random sample $\mathsf{N} \subseteq \mathsf{X}$ of $O\left(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon} + \log\frac{1}{\delta}\right)$ is a shallow ε-net with probability at least $1 - \delta$.*

Ξ is a vertical shallow cutting. We apply the above result to the range space $\Sigma = (\mathcal{F}, \mathcal{R})$, so that each range in $\mathcal{R}$ is the subset of the surfaces of $\mathcal{F}$ that cross some region in $\mathbb{R}^4$, taken from some family Γ of regions. Concretely, this includes the following families Γ: (i) the set of pseudo-cones generated by the decomposition of a set of at most seven functions of $\mathcal{F}$ (recall that a pseudo-cone is defined by at most seven functions), (ii) the set of pseudo-prisms erected on these pseudo-cones, (iii) the set of edges in an arrangement of five functions of $\mathcal{F}$ (an edge in the arrangement of a subset of $\mathcal{F}$ is defined by at most five functions), and (iv) the set of rays in the positive x_4-direction. Using standard arguments, it can be shown that the VC-dimension of Σ is finite. Therefore, by applying Lemma 8 with $\varepsilon = 1/t$ and $\delta = 1/3$ and choosing the constant of proportionality appropriately, the random subset $\mathcal{R}$ is a $(1/t)$-shallow net of Σ with probability at least $2/3$, so we assume that N is indeed a $(1/t)$-shallow net of Σ. Recall that $q = \beta \log t$. Assuming $\beta \geq 2\alpha$, where α is the constant in the definition of shallow nets, the converse of property (ii) of shallow nets, with $\zeta = 1$, implies that the level of any point p on $\mathcal{A}_q(\mathsf{N})$ with respect to $\mathcal{F}$ is at least n/t. Also, by property (i), the level of p is at most $O(n/t)$. Finally, the shallow net property also implies that the size of the conflict-list of any prism in $\Xi^{\uparrow}$ is $O(n/t)$. The conflict list of all prisms can be computed in $O^*(nt^3)$ time. Hence, we obtain:

▶ **Theorem 9.** *Let K be a compact, centrally-symmetric strictly convex semi-algebraic set in $\mathbb{R}^3$ of constant complexity. Let $\mathcal{K}$ be a set of n homothetic copies of K. For any parameter $t \in [1, n-1]$, there exists a vertical (n/t)-shallow cutting Ξ of the arrangement of the distance functions of $\mathcal{K}$ of size $O^*(t^3)$. The cutting Ξ, along with the conflict list of each of its prisms, can be computed in $O^*(nt^3)$ expected time.*

4 Dynamic Data Structures for Proximity Queries

In this section we describe a dynamic data structure for answering *intersection-detection* queries amid $\mathcal{K}$. Namely, for a query homothet $K_0 \in \mathbb{K}$, determine whether K_0 intersects some homothet of $\mathcal{K}$, and, if the answer is yes, return such a homothet. In addition to answering queries, the data structure can be updated efficiently as a homothet $K_0 \in \mathbb{K}$ is inserted into $\mathcal{K}$ or deleted from $\mathcal{K}$. The data structure can be extended to answering *intersection-reporting* queries, namely, reporting all homothets of $\mathcal{K}$ that intersect K_0, at an additional cost of $O(k)$.

As already noted, the insertion of a new homothet into $\mathcal{K}$ can be handled using the standard dynamization technique by Bentley and Saxe [13] (see also [9]). That is, assume that we have an intersection-detection data structure that can handle deletions, which can be constructed in $P(n)$ time, so that a query can be answered in $Q(n)$ time and an object can be deleted in $D(n)$ time. Then the amortized insertion time is $O((P(n)/n)\log n)$, the deletion time remains $D(n)$, and the overall query time is $O(Q(n))$, assuming that $Q(n) = \Omega(n^{\varepsilon})$.

We first describe a data structure that answers a query in $O^*(1)$ time using $O^*(n^3)$ space and handles a deletion in $O^*(n^2)$ amortized expected time, and then describe a linear-size data structure that answers a query in $O^*(n^{2/3})$ time and handles a deletion in $O(\log^2 n)$ amortized expected time.

4.1 Fast query-time data structure

We follow the same approach of Agarwal and Matoušek for performing dynamic halfspace range reporting [9] (slightly improved but more involved approaches were proposed in [15, 26], but the one in [9] will do for our setting).

Data structure. Let $\mathcal{F}$ be the collection of distance functions corresponding to the elements of $\mathcal{K}$. Roughly speaking, we build a tree data structure on $\mathcal{F}$ using vertical shallow cuttings. That is, we start with $\mathcal{F}$, choose a parameter $t > 1$, compute an (n/t)-vertical shallow cutting Ξ of $\mathcal{F}$ of size $O^*(t^3)$, in $O^*(nt^3)$ time, create a child for each prism τ of Ξ, recursively construct the data structure for the conflict list $\mathcal{F}_\tau$ of every $\tau \in \Xi$, and attach it as the subtree of the child corresponding to τ. The recursion proceeds until we reach cells τ with $|\mathcal{F}_\tau| = O(1)$, in which case we just store $\mathcal{F}_\tau$ at τ, as a list, say.

A query, with a copy $K_0 = (c_0, \rho_0)$ of K, is answered in a standard way, by traversing the tree in a top-down manner. At each node (prism) τ that we visit, we know that $(c_0, \rho_0) \in \tau$. We check whether (c_0, ρ_0) lies in any child prism of τ, and, if so, recurse at that child. Otherwise, we locate the child prism τ' for which (c_0, ρ_0) lies above τ', and report any element whose function belongs to the conflict list $\mathcal{F}_{\tau'}$.

However, this simple recursive approach runs into a technical difficulty when we delete a function from $\mathcal{F}$. When deleting a function from the root, the deletion has to be propagated to some of its children, to their children, and so on. On average, each function appears in the conflict lists of $O^*(t^2)$ prisms of Ξ. When such a "good" function is deleted, the deletion is propagated to only $O^*(t^2)$ descendants, leading to $O^*(n^2)$ overall deletion time, as can easily be verified. However, some of the functions (in fact, up to n/t of them) may appear in the conflict lists of $\Omega(t^3)$ prisms of Ξ. If these "bad" functions are the first n/t functions to be deleted and then the data structure is reconstructed, the overall cost of the deletion operations will be too high. As in [9, 26], we circumvent this difficulty by maintaining a partition of $\mathcal{F}$ into a few subsets and constructing a cutting for each subset that is good for all functions in that subset (i.e., each function in the subset appears in only $O^*(t^2)$ conflict lists).

We now describe the data structure in detail. We fix a sufficiently large constant $t > 0$. The data structure is periodically reconstructed after performing some deletions. We use m to denote the size of $\mathcal{F}$ when the data structure was previously reconstructed and n to denote its current size. We reconstruct the data structure after deleting $m/2t$ functions from $\mathcal{F}$. Thus $n \geq m(1 - 1/2t)$. Let $P(r)$ denote the maximum time spent in constructing the data structure for a set of r functions. We pay for the reconstruction cost by charging $2tP(m - m/2t)/m \leq 2tP(n)/n$ time to each of the $m/2t$ delete operations that occurred before the reconstruction (the inequality holds because P is assumed to be superlinear). The data structure for $\mathcal{F}$ is a (recursively defined) tree $\Psi(\mathcal{F})$. We describe how a subtree $\Psi(\mathcal{G})$, for a subset $\mathcal{G}$ of ν functions, is constructed.

If $\nu \leq n_0$, for some sufficiently large constant n_0, then $\Psi(\mathcal{G})$ is a single leaf, and we simply store $\mathcal{G}$ at that leaf. So assume that $\nu > n_0$. The root of $\Psi(\mathcal{G})$ stores the following data:

- A partition of $\mathcal{G}$ into subsets $\mathcal{G}_1, \ldots, \mathcal{G}_u$, $u \leq \lceil \log_2 t \rceil$, as described below.
- For each $1 \leq i \leq u$, a vertical (ν/t)-shallow cutting Ξ_i for $\mathcal{G}_i$. For each $\Delta \in \Xi_i$, we store its conflict list $\mathcal{G}_{i,\Delta}$.
- For each $1 \leq i \leq u$ and $\Delta \in \Xi_i$, a pointer to the tree $\Psi(\mathcal{G}_{i,\Delta})$.
- For each function $f \in \mathcal{G}_i$, the set $L_f \subseteq \Xi_i$ of prisms Δ crossed by the graph of f.
- A counter χ that keeps track of how many functions of $\mathcal{G}$ can be deleted before we reconstruct $\Psi(\mathcal{G})$.

After the construction of $\Psi(\mathcal{G})$, before any deletions occur, the following properties hold:

(P1) The counter χ is set to $\nu/2t$.
(P2) For each i, the x_4-monotone surface formed by the pseudo-cones of Ξ_i (i.e., the ceilings of prisms of Ξ_i) lie above $\mathcal{A}_{\nu/t}(\mathcal{G}_i)$.
(P3) For every i and $\Delta \in \Xi_i$, $|\mathcal{G}_{i,\Delta}| \leq cn/t$, for some suitable absolute constant c.
(P4) Each function $f \in \mathcal{G}_i$ appears in the conflict lists of at most $\kappa := 2C_1 t^{2+\delta}$ prisms of Ξ_i, where C_1 and $\delta > 0$ are the constants appearing in (1) below.

We now describe the construction of the partition $\mathcal{G}_1, \ldots, \mathcal{G}_u$, which proceeds iteratively. Suppose we have constructed $\mathcal{G}_1, \ldots, \mathcal{G}_{i-1}$. Set $\mathcal{G}_i^* = \mathcal{G} \setminus \bigcup_{j<i} \mathcal{G}_i$ and put $\nu_i = |\mathcal{G}_i^*|$. If $\nu_i \leq \nu/t$, we stop. Otherwise, set $t_i = t\nu_i/\nu$ and $k = \nu_i/t_i = \nu/t$.

We construct a vertical k-shallow cutting Ξ_i of $\mathcal{G}_i^*$ of size $O(t_i^{3+\delta})$, for an arbitrarily small constant $\delta > 0$. For each $\Delta \in \Xi_i$, $|\mathcal{G}_{i,\Delta}^*| \leq c\nu_i/t_i = c\nu/t$. We note that

$$\sum_{\Delta \in \Xi_i} |\mathcal{G}_{i,\Delta}^*| \leq O(t_i^{3+\delta}) \frac{c\nu}{t} \leq C_1 \left(\frac{t\nu_i}{\nu}\right)^{3+\delta} \frac{\nu}{t} \leq C_1 \nu_i t^{2+\delta}. \tag{1}$$

A function $f \in \mathcal{G}_i^*$ is called *good* if it appears in the conflict list of at most κ prisms of Ξ_i and *bad* otherwise. Let $\mathcal{G}_i$ be the set of good functions of $\mathcal{G}_i^*$; set $\mathcal{G}_{i+1}^* = \mathcal{G}_i^* \setminus \mathcal{G}_i$. By (1), $|\mathcal{G}_{i+1}^*| \leq \nu_i/2$. Hence, $u \leq \lceil \log_2 t \rceil$. It is easily seen that properties (P1)–(P4) hold after the construction. This completes the description of the data structure. The total size and preprocessing time are $O^*(n^3)$.

Query procedure. Let $K(c_0, \rho_0) \in \mathbb{K}$ be a query homothet. Recall that $K(c_0, \rho_0)$ intersects a homothet of $\mathcal{K}$ if the point (c_0, ρ_0) lies above the lower envelope of the corresponding set $\mathcal{F}$ of functions. The query is performed as follows. We search $\Psi(\mathcal{F})$ in a top-down manner with (c_0, ρ_0). Suppose we are at the root of a subtree $\Psi(\mathcal{G})$. If $\Psi(\mathcal{G})$ is a leaf, we check all functions of $\mathcal{G}$ in a brute-force manner and return an answer in $O(n_0) = O(1)$ time. Otherwise, for

each $i \leq u$, we check whether (c_0, ρ_0) lies above the lower envelope of $\mathcal{G}_i$. If the answer is yes for any i, we return yes and also return an intersecting homothet (see below). Otherwise, we return no.

For a fixed $i \leq u$, we proceed as follows. We search with c_0, in a brute force manner, in $O^*(t^3) = O(1)$ time, to determine the pseudo-cone $\tau^{\downarrow}$ in the projection of Ξ_i that contains c_0. Let f_i be the function associated with τ, i.e., the ceiling of Δ lies in the graph of f_i. We test whether $\rho_0 \geq f_i(c_0)$. If so, K_i and $K(c_0, \rho_0)$ intersect, so we return yes and report K_i as an intersection witness and terminate the query. If not, (c_0, ρ_0) lies in the semi-unbounded prism Δ, and we continue the search recursively in the data structure $\Psi(\mathcal{G}_{i,\Delta})$.

Let $Q(\nu)$ be the maximum time spent by the query procedure in the subtree $\Psi(\mathcal{G})$ storing at most ν functions. Then we obtain the following recurrence:

$$Q(\nu) \leq \lceil \log_2 t \rceil Q(c\nu/t) + O(t^{3+\delta}).$$

Its solution is easily seen to be $Q(\nu) = O(n^{\varepsilon})$, for any constant $\varepsilon > 0$, provided that n_0 and t are chosen sufficiently large.

Deletion procedure. Let $f \in \mathcal{F}$ be a function that we wish to delete from $\mathcal{F}$. Again, we visit $\Psi(\mathcal{F})$ in a top-down manner. Suppose we are at the root v of a subtree $\Psi(\mathcal{G})$. If v is a leaf, then we simply delete f from $\mathcal{G}$ and stop. If v is an internal node, we first decrement the counter χ stored at v. If χ becomes 0, we reconstruct $\Psi(\mathcal{G})$ from scratch (for the current $\mathcal{G}$). Otherwise, we find the index i such that $f \in \mathcal{G}_i$. For each $\Delta \in L_f$, we delete f from $\mathcal{G}_{i,\Delta}$ and then recursively delete f from $\Psi(\mathcal{G}_{i,\Delta})$. By construction, $|L_f| \leq \kappa$. The deletion procedure maintains the properties (P3) and (P4), and (P2) is replaced with a slightly weaker property:

(P2') For every i, the x_4-monotone surface formed by the pseudo-cones of Ξ_i (i.e., the ceilings of prisms of Ξ_i) lie above or on $\mathcal{A}_{\nu/2t}(\mathcal{G}_i)$.

The correctness of the query procedure follows from property (P2'). Following the analysis in [9], the amortized deletion time, including the time spent in reconstructing the data structure, is $O^*(n^2)$. Putting everything together, and combining this with the technique of Bentley and Saxe [13] for insertions, we obtain:

▶ **Lemma 10.** *$\mathcal{K}$ can be preprocessed, in $O^*(n^3)$ expected time, into a data structure of size $O^*(n^3)$, so that an intersection-detection query, including the cost of reporting an intersecting member of $\mathcal{K}$ if one exists, can be performed in $O^*(1)$ time. A homothet can be inserted into $\mathcal{K}$ or deleted from $\mathcal{K}$ in $O^*(n^2)$ amortized expected time.*

4.2 Linear-size data structure

Next, we present a linear-size dynamic data structure that supports intersection-detection queries in $O^*(n^{2/3})$ time each, and can handle updates (insertions and deletions) in $O(\log^2 n)$ amortized expected time per update.

For a homothet $K(c_0, \rho_0)$, let $f^+_{c_0,\rho_0} = \{(c, \rho) \in \mathbb{R} \mid \rho \geq f_{c_0,\rho_0}(c)\}$ be the region lying above the graph of the distance function f_{c_0,ρ_0}, which corresponds to the set of homothets that intersect $K(c_0, \rho_0)$. Put $\mathcal{K}^* = \{(c, \rho) \mid K(c, \rho) \in \mathcal{K}\} \subset \mathbb{R}^4$. For a query homothet $K(c_0, \rho_0)$, we wish to determine whether any point of $\mathcal{K}^*$ lies in $f^+_{c_0,\rho_0}$. We construct a linear-size partition tree for answering these queries, following the same approach as in [9, 32, 35]. We need a few definitions. Let $\mathbb{F}$ be the set of distance functions corresponding to the homothets in $\mathbb{K}$. Let $P \subset \mathbb{K}$ be a set of n points (representing homothets of K). For a parameter

$k \geq 1$, we call a semi-algebraic set $\gamma \subset \mathbb{K}$, which is semi-unbounded in the $(-x_4)$-direction, *k-shallow* if $|P \cap \gamma| \leq k$. A major ingredient of the approach in [32, 35] is to construct a so-called *test set* composed of a small number of semi-algebraic sets, which represent well (in the sense made precise below) all distance functions of $\mathbb{F}$ that are (n/r)-shallow, for a given parameter $r > 1$, with respect to P. Formally, a finite collection Φ of semi-algebraic sets of constant complexity (not necessarily a subset of $\mathbb{F}$) is called a *test set* for (n/r)-shallow ranges in $\mathbb{F}$ with respect to P if it satisfies the following properties:

- **(i)** Every set in Φ is (n/r)-shallow with respect to P.
- **(ii)** The complement of the union of any m sets of Φ can be decomposed into at most $\zeta(m)$ "elementary cells" (semi-algebraic sets of constant complexity) for any $m \geq 1$, where $\zeta(m)$ is a suitable monotone increasing superlinear function of m.
- **(iii)** Any (n/r)-shallow set $\sigma \in \mathbb{F}$ can be covered by the union of at most δ ranges of Φ, where δ is a constant (independent of r).

Sharir and Shaul [35] showed that if there exists a test set of size $r^{O(1)}$ for (n/r)-shallow ranges of $\mathbb{F}$, then one can construct a linear-size partition tree on P so that an $\mathbb{F}$-emptiness query can be answered in $O^*(n/\zeta^{-1}(n))$ time, for the corresponding function ζ.

Test-set construction. We describe an algorithm for constructing a test set of size $O(r^4)$ with $\zeta(m) = O^*(m^3)$ and $\delta = 1$. Let $\mathcal{F}$ be the set of distance functions of $\mathcal{K}$ as above. We take a random subset $R \subseteq \mathcal{F}$ of $s = cr \log r$ functions, for some sufficiently large constant r, where $c > 0$ is another sufficiently large absolute constant, independent of, and much smaller than r, and construct the (standard) 4D vertical decomposition $\mathcal{A}^{\shortparallel}(R)$ of the arrangement $\mathcal{A}(R)$, of size $O^*(r^4)$, as described in [3,30,34]. By a standard random-sampling argument [24], with probability at least $1/2$, each cell of $\mathcal{A}^{\shortparallel}(R)$ is crossed by at most n/r functions of $\mathcal{F}$, provided that c is chosen sufficiently large. If this is not the case, we discard R and choose another random subset, until we find, in expected $O(1)$ trials, one with the desired property. We clip $\mathcal{A}^{\shortparallel}(R)$ within the halfspace $x_4 \geq 0$ (to restrict it to within $\mathbb{K}$). We choose a subset Ξ of the cells of $\mathcal{A}^{\shortparallel}(R)$, consisting of those cells that have at most n/r functions of $\mathcal{F}$ passing *fully below* them. By construction, these cells cover $\mathcal{A}_{\leq n/r}(\mathcal{F})$, and are contained in $\mathcal{A}_{\leq 2n/r}(\mathcal{F})$.

Let τ be a cell of Ξ. Set $\Phi_\tau = \bigcup_{(c_0,\rho_0)\in\tau} f^+_{c_0,\rho_0}$. Since τ and f_{c_0,ρ_0} are semi-algebraic sets of constant complexity, Φ_τ is also a semi-algebraic set of constant complexity. For a homothet $K_i \in \mathcal{K}$, the point $K_i^* \in \mathbb{K}$ lies in Φ_τ if and only if there is a point $(c_0, \rho_0) \in \tau$ such that $K_i^* \in f^+_{c_0,\rho_0}$. This happens when $(c_0, \rho_0) \in f_i^+$, i.e., the graph of the function f_i crosses τ or lies below τ. By construction, there are at most $n/r + n/r = 2n/r$ such functions. Consequently, Φ_τ is $(2n/r)$-shallow with respect to the points of $\mathcal{K}^*$ with $\delta = 1$.

Set $\Phi = \{\Phi_\tau \mid \tau \in \Xi\}$. Φ is a family of $O^*(r^4)$ constant-complexity semi-algebraic sets in $\mathbb{K}$, each of which is $(2n/r)$-shallow with respect to $\mathcal{K}^*$. This is the desired test set. Each set Φ_τ is unbounded in the positive x_4-direction, therefore the complement of the union of a subset of m sets is the region lying below their lower envelope. By a result in [5], the complement of their union (i.e., the region below their lower envelope) can be decomposed into $O^*(m^3)$ pseudo-prisms. Hence, we obtain:

▶ **Lemma 11.** *Let $P \subset \mathbb{K}$ be a set of n points and $r \geq 1$ a parameter. A test set Φ of size $O(r^4)$ for the functions in $\mathbb{F}$ that are (n/r)-shallow with respect to P can be computed in $O^*(r^4)$ expected time, so that $\zeta(m) = O^*(m^3)$ and $\delta = 1$.*

Using Lemma 11 and adapting the approach in [35], we can preprocess $\mathcal{K}$ into a linear-size data structure that supports intersection-detection queries in $O^*(n^{2/3})$ time. By (re)constructing portions of the partition tree periodically (as in the previous data structure), deletions can be handled efficiently. Omitting the straightforward details, we obtain:

▶ **Lemma 12.** *Let $\mathcal{K}$ be a set of n homothets of K. $\mathcal{K}$ can be preprocessed, in $O(n \log n)$ expected time, into a data structure of size $O(n)$, so that an intersection-detection query can be answered in $O^*(n^{2/3})$ time. A homothet of K can be inserted into or deleted from the data structure in $O(\log^2 n)$ amortized time.*

By combining this data structure with Lemma 10 in a standard manner [1], we can obtain the following space/query-time trade-off:

▶ **Theorem 13.** *Let $\mathcal{K}$ be a set of n homothets of K, and let $s \in [n, n^3]$ be a storage parameter. $\mathcal{K}$ can be preprocessed, in $O^*(s)$ expected time, into a data structure of size $O^*(s)$, so that an intersection-detection query can be answered in $O^*(n/s^{1/3})$ time. A homothet can be inserted into or deleted from the data structure in $O^*(s/n)$ amortized time.*

Finally, by combining this data structure with the parametric-search technique [8], we establish Theorem 1 (for answering nearest-neighbor queries in $\mathcal{K}$ under the K-distance).

References

1 P. K. Agarwal, B. Aronov, E. Ezra, M. J. Katz, and M. Sharir. Intersection queries for flat semi-algebraic objects in three dimensions and related problems. *ACM Transactions on Algorithms*, 21:25:1–25:59, 2025. doi:10.1145/3721290.

2 P. K. Agarwal, B. Aronov, E. Ezra, and J. Zahl. An efficient algorithm for generalized polynomial partitioning and its applications. *SIAM Journal on Computing*, 50:760–787, 2021. doi:10.1137/19M1268550.

3 P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 1999. doi:10.1137/S0097539795295936.

4 P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999. doi:10.1090/conm/223/03131.

5 P. K. Agarwal, E. Ezra, and M. Sharir. Vertical decomposition in 3D and 4D with applications to line nearest-neighbor searching in 3D. In *35th ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 150–170, 2024. doi:10.1137/1.9781611977912.8.

6 P. K. Agarwal, H. Kaplan, M. J. Katz, and M. Sharir. Segment proximity graphs and nearest neighbor queries amid disjoint segments. In *32nd European Symposium on Algorithms (ESA 2024)*, pages 7:1–7:20, 2024. doi:10.4230/LIPIcs.ESA.2024.7.

7 P. K. Agarwal, M. J. Katz, and M. Sharir. On reverse shortest paths in geometric proximity graphs. *Computational Geometry*, 117:102053, 2024. doi:10.1016/j.comgeo.2023.102053.

8 P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22:794–806, 1993. doi:10.1137/0222051.

9 P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995. doi:10.1007/BF01293483.

10 A. Andoni and P. Indyk. Nearest neighbors in high dimensional spaces. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, 3rd edition, 2017. Chapter 43. doi:10.1201/9781315119601.

11 R. Ben Avraham, O. Filtser, H. Kaplan, M. J. Katz, and M. Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Transactions on Algorithms*, 11(4):29:1–29:29, 2015. doi:10.1145/2700222.

12 S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, Berlin, 2nd edition, 2006.

13 J. L. Bentley and J. B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *Journal of Algorithms*, 1:301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

14 S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48:360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

15 T. M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest-neighbor queries. *Journal of the ACM*, 57:16:1–16:15, 2010. doi:10.1145/1706591.1706596.

16 T. M. Chan. Dynamic generalized closest pair: Revisiting Eppstein's technique. In *3rd Symposium on Simplicity in Algorithms (SOSA 2020)*, pages 33–37, 2020. doi:10.1137/1.9781611976014.6.

17 T. M. Chan and Z. Huang. Faster algorithms for reverse shortest path in unit-disk graphs and related geometric optimization problems: Improving the shrink-and-bifurcate technique. In *41st International Symposium on Computational Geometry (SoCG 2025)*, pages 32:1–32:14, 2025. doi:10.4230/LIPIcs.SoCG.2025.32.

18 T. M. Chan and K. Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete & Computational Geometry*, 56:866–881, 2016. doi:10.1007/S00454-016-9784-4.

19 B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. In *16th International Colloquium on Automata, Languages and Programming (ICALP 1989)*, pages 179–193, 1989. Also in Theoretical Computer Science 84 (1991), 77–105. doi:10.1007/BFB0035760.

20 K. L. Clarkson. Nearest neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods in Learning and Vision*. MIT Press, 2006.

21 K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989. doi:10.1007/BF02187740.

22 M. de Berg and S. Cabello. An $O(n \log n)$ algorithm for single-source shortest paths in disk graphs. In *33rd European Symposium on Algorithms (ESA 2025)*, pages 81:1–81:15, 2025. doi:10.4230/LIPIcs.ESA.2025.81.

23 D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *ACM Journal of Experimental Algorithmics*, 5:1, 2000. doi:10.1145/351827.351829.

24 D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. doi:10.1007/BF02187876.

25 H. Kaplan, M. J. Katz, R. Saban, and M. Sharir. The unweighted and weighted reverse shortest path problem for disk graphs. *ACM Transactions on Algorithms*, 2023. To appear. Also in 31st European Symposium on Algorithms (ESA 2023), 67:1–67:14.

26 H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. *Discrete & Computational Geometry*, 64:838–904, 2020. doi:10.1007/s00454-020-00243-7.

27 M. J. Katz, R. Saban, and M. Sharir. Near-linear algorithms for visibility graphs over a 1.5-dimensional terrain. In *32nd European Symposium on Algorithms (ESA 2024)*, pages 77:1–77:17, 2024. doi:10.4230/LIPICS.ESA.2024.77.

28 M. J. Katz, R. Saban, and M. Sharir. BFS and reverse shortest paths for ball intersection graphs in three and higher dimensions. In *36th International Symposium on Algorithms and Computation (ISAAC 2025)*, pages 45:1–45:15, 2025. doi:10.4230/LIPICS.ISAAC.2025.45.

29 K. Klost. An algorithmic framework for the single source shortest path problem with applications to disk graphs. *Computational Geometry*, 111:101979, 2023. doi:10.1016/j.comgeo.2022.101979.

30 V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51:699–730, 2004. doi:10.1145/1017460.1017461.

31 C.-H. Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM Journal on Computing*, 51(3):723–765, 2022. doi:10.1137/20M1388371.

32 J. Matoušek. Reporting points in halfspaces. *Computational Geometry*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

33 J. Matoušek and Z. Patáková. Multilevel polynomial partitioning and simplified range searching. *Discrete & Computational Geometry*, 54:22–41, 2015. doi:10.1007/s00454-015-9701-2.

34 M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

35 M. Sharir and H. Shaul. Semialgebraic range reporting and emptiness searching with applications. *SIAM Journal on Computing*, 40(4):1045–1074, 2011. doi:10.1137/090765092.

36 H. Wang and Y. Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023. doi:10.20382/JOCG.V14I1A2.