2012 IEEE 53rd Annual Symposium on Foundations of Computer Science

On Range Searching with Semialgebraic Sets II

Pankaj K. Agarwal
Duke University

Jiří Matoušek
Charles University

Micha Sharir
Tel Aviv University and
Courant Institute, NYU

***Abstract*—Let P be a set of n points in $\mathbb{R}^d$. We present a linear-size data structure for answering range queries on P with constant-complexity semialgebraic sets as ranges, in time close to $O(n^{1-1/d})$. It essentially matches the performance of similar structures for simplex range searching, and, for $d \geq 5$, significantly improves earlier solutions by the first two authors obtained in 1994. This almost settles a long-standing open problem in range searching.**

The data structure is based on the polynomial-partitioning technique of Guth and Katz [arXiv:1011.4105], which shows that for a parameter r, $1 < r \leq n$, there exists a d-variate polynomial f of degree $O(r^{1/d})$ such that each connected component of $\mathbb{R}^d \setminus Z(f)$ contains at most n/r points of P, where $Z(f)$ is the zero set of f. We present an efficient randomized algorithm for computing such a polynomial partition, which is of independent interest and is likely to have additional applications.

***Index Terms*—Range searching, semialgebraic sets, polynomial partition, ham-sandwich cuts**

I. Introduction

Range searching. Let P be a set of n points in $\mathbb{R}^d$, where d is a small constant. Let Γ be a family of geometric "regions," called *ranges*, in $\mathbb{R}^d$, each of which can be described algebraically by some fixed number of real parameters. For example, Γ can be the set of all axis-parallel boxes, balls, simplices, or cylinders, or the set of all intersections of pairs of ellipsoids. The Γ-*range searching* problem can be defined as: *Preprocess P into a data structure so that the number of points of P lying in a query range $\gamma \in \Gamma$ can be counted efficiently.* Actually, we consider a more general setting, where one assumes a weight function on the points in P and asks for the cumulative weight of the points in $P \cap \gamma$. The weights are assumed to belong to a semigroup, i.e., subtractions are not allowed. We assume that the semigroup operation can be executed in constant time.

In this paper we consider the case in which Γ is a set of constant-complexity semialgebraic sets. We recall that a *semialgebraic set* is a subset of $\mathbb{R}^d$ obtained from a finite number of sets of the form $\{x \in \mathbb{R}^d \mid g(x) \geq 0\}$, where g is a d-variate polynomial with integer coefficients, by Boolean operations (unions, intersections, and complementations). Specifically, we let $\Gamma_{d,\Delta,s}$ denote the family of all semialgebraic sets in $\mathbb{R}^d$ defined by at most s polynomial inequalities of degree at most Δ each. If d, Δ, s are all regarded as constants, we refer to the sets in $\Gamma_{d,\Delta,s}$ as *constant-complexity semialgebraic sets* (such sets are sometimes also called *Tarski cells*). By *semialgebraic range searching* we mean $\Gamma_{d,\Delta,s}$-range searching for some parameters d, Δ, s (although in most applications the actual collection γ of ranges is only a restricted subset of such a collection $\Gamma_{d,\Delta,s}$). Besides being interesting in its own right, semialgebraic range searching also arises in a wide range of geometric searching problems, such as searching for a point nearest to a query geometric object, counting the number of input objects intersecting a query object, and many others.

This paper focuses on the *low storage* version of range searching with constant-complexity semialgebraic sets—the data structure is allowed to use only linear or near-linear storage, and the goal is to make the query time as small as possible. As it is typical in computational geometry, we will use the *real RAM* model of computation, where we can compute exactly with arbitrary real numbers and each arithmetic operation is executed in constant time.

Previous work. Motivated by a wide range of applications, several variants of range searching have been studied in computational geometry and database systems for more than three decades. See [1], [23] for comprehensive surveys of this topic. The early work focused on the so-called *orthogonal range searching*, where the ranges are axis-parallel boxes. After three decades of extensive work on this particular case, some basic questions still remain open. However, geometry plays almost no role in the known data structures for orthogonal range searching.

The most basic and most studied truly geometric instance of range searching is with *halfspaces*, or more generally *simplices*, as ranges. Studies in the early 1990s have essentially determined the optimal trade-off between the worst-case query time and the storage (and preprocessing time) required by any data structure for simplex range searching. Lower bounds for this trade-off have been given by Chazelle [7] under the semigroup model of computation, where subtraction of the point weights is not allowed. We also refer to [19] and references therein for recent lower bounds for the case where subtractions are also allowed.

The data structures proposed for simplex range searching over the last two decades [21], [22] match the known lower bounds within polylogarithmic factors. The state-of-the-art upper bounds are by (i) Chan [6], who, building on many earlier results, provides a linear-size data structure

Work by Pankaj Agarwal was supported by NSF under grants IIS-07-13498, CCF-09-40671, and CCF-10-12254, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, and by an ARL award W9132V-11-C-0003. Work by Jiří Matoušek has been supported by the ERC Advanced Grant No. 267165. Work by Micha Sharir has been supported by NSF Grant CCF-08-30272, by Grant 338/09 from the Israel Science Fund, by the Israeli Centers for Research Excellence (I-CORE) program (Center No. 4/11) and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

0272-5428/12 $26.00 © 2012 IEEE
DOI 10.1109/FOCS.2012.32

with $O(n \log n)$ expected preprocessing time and $O(n^{1-1/d})$ query time, and (ii) Matoušek [22], who provides a data structure with $O(n^d)$ storage, $O((\log n)^{d+1})$ query time, and $O(n^d(\log n)^\varepsilon)$ preprocessing time.[1] A trade-off between space and query time can be obtained by combining these two data structures [22].

Yao and Yao [32] were perhaps the first to consider range searching in which ranges were delimited by graphs of polynomial functions. Agarwal and Matoušek [2] have introduced a systematic study of semialgebraic range searching. Building on the techniques developed for simplex range searching, they presented a linear-size data structure with $O(n^{1-1/b+\varepsilon})$ query time, where $b = \max(d, 2d-4)$. For $d \leq 4$, this almost matches the performance for the simplex range searching, but for $d \geq 5$ there is a gap in the exponents of the corresponding bounds. See also [28] for some recent developments.

The bottleneck in the performance of the just mentioned range-searching data structure of [2] is a combinatorial geometry problem, known as the *decomposition of arrangements into constant-complexity cells*. Here, we are given a set Σ of r algebraic surfaces in $\mathbb{R}^d$ (i.e., zero sets of d-variate polynomials), with degrees bounded by a constant Δ_0, and we want to decompose each cell of the arrangement $\mathcal{A}(\Sigma)$ into subcells that are constant-complexity semialgebraic sets, i.e., belong to $\Gamma_{d,\Delta,s}$ for some constants Δ (bound on degrees) and s (number of defining polynomials), which may depend on d and Δ_0, but not on r. The crucial quantity is the total number of the resulting subcells over all cells of $\mathcal{A}(\Sigma)$; namely, if one can construct such a decomposition with $O(r^b)$ subcells, with some constant b, for every r and Σ, then the method of [2] yields query time $O(n^{1-1/b+\varepsilon})$. The only known general-purpose technique for producing such a decomposition is the so-called *vertical decomposition* [8], [27], which decomposes $\mathcal{A}(\Sigma)$ into roughly n^{2d-4} Tarski cells, for $d \geq 4$ [18], [27].

An alternative approach, based on *linearization*, was also proposed in [2]. It maps the semialgebraic ranges in $\mathbb{R}^d$ to simplices in some higher-dimensional space and uses simplex range searching there. However, its performance depends on the specific form of the polynomials defining the ranges. In some special cases (e.g. when ranges are balls in $\mathbb{R}^d$), linearization yields better query time than the decomposition-based technique mentioned above but for general constant-complexity semialgebraic ranges, linearization yields worse query time.

Our results. In a recent breakthrough, Guth and Katz [12] have presented a new space decomposition technique, called polynomial partitioning. For a set $P \subset \mathbb{R}^d$ of n points and a real parameter r, $1 < r \leq n$, an *r-partitioning polynomial* for P is a nonzero d-variate polynomial f such that each connected component of $\mathbb{R}^d \setminus Z(f)$ contains at most n/r points of P, where $Z(f) := \{x \in \mathbb{R}^d \mid f(x) = 0\}$ denotes the zero set of f. The decomposition of $\mathbb{R}^d$ into $Z(f)$ and the connected components of $\mathbb{R}^d \setminus Z(f)$ is called a *polynomial partition* (induced by f). Guth and Katz show that an r-partitioning polynomial of degree $O(r^{1/d})$ always exists, but their argument does not lead to an efficient algorithm for constructing such a polynomial, mainly because it relies on ham-sandwich cuts in high-dimensional spaces, for which no efficient construction is known. Our first result is an efficient randomized algorithm for computing an r-partitioning polynomial.

Theorem I.1. *Given a set P of n points in $\mathbb{R}^d$, for some fixed d, and a parameter $r \leq n$, an r-partitioning polynomial for P of degree $O(r^{1/d})$ can be computed in randomized expected time $O(nr + r^3)$.*

Next, we use this algorithm to bypass the arrangement-decomposition problem mentioned above. Namely, we use partitions induced by partitioning polynomials to construct partition trees that answer range queries with constant-complexity semialgebraic sets in near-optimal time, using linear storage. While there have already been several combinatorial applications of the Guth-Katz technique (the most impressive being the original one in [12], which solves the famous Erdős's distinct distances problem, and others presented in [14], [15], [29], [34]), ours seems to be the first *algorithmic* application.

We establish two range-searching results, both based on polynomial partitions. For the first result, we need to introduce the notion of *D-general position*, for an integer $D \geq 1$. We say that a set $P \subset \mathbb{R}^d$ is in D-general position if no k points of P are contained in the zero set of a nonzero d-variate polynomial of degree at most D, where $k := \binom{D+d}{d}$. This is the number one expects for a "generic" point set. Indeed, d-variate polynomials of degree at most D have at most $k-1$ distinct nonconstant monomials, from which it follows that any set of $k-1$ points in $\mathbb{R}^d$ is contained in the zero set of a d-variate polynomial of degree at most D; e.g., see [10], [11].

Theorem I.2. *Let d, Δ, s and $\varepsilon > 0$ be constants. Let $P \subset \mathbb{R}^d$ be an n-point set in D_0-general position, where D_0 is a suitable constant depending on d, Δ, and ε. Then the $\Gamma_{d,\Delta,s}$-range searching problem for P can be solved with $O(n)$ storage, $O(n \log n)$ expected preprocessing time, and $O(n^{1-1/d+\varepsilon})$ query time.*

Of course, we would like to handle arbitrary point sets, not only those in D_0-general position. This can be achieved by an infinitesimal perturbation of the points of P. A general technique known as "simulation of simplicity" (in the version considered by Yap [33]) ensures that the perturbed set P' is in D_0-general position. If a point $p \in P$ lies in the interior of a query range γ, then so does the corresponding perturbed point $p' \in P'$, and similarly for p in the interior of $\mathbb{R}^d \setminus \gamma$. However, for p on the boundary of γ, we cannot be sure if p' ends up inside or outside γ.

Let us say that a *boundary-fuzzy* solution to the $\Gamma_{d,\Delta,s}$-range searching problem is a data structure that, given a query $\gamma \in \Gamma_{d,\Delta,s}$, returns an answer in which all points of P in the interior of γ are counted and none in the interior of $\mathbb{R}^d \setminus \gamma$ is

[1] Here and in the sequel, ε denotes an arbitrarily small positive constant. The implicit constants in the asymptotic notation may depend on it, generally tending to infinity as ε decreases to 0.

counted, while each point $p \in P$ on the boundary of γ may or may not be counted. In some applications, we can think of the points of P being imprecise anyway (e.g., their coordinates come from some imprecise measurement), and then boundary-fuzzy range searching may be adequate.

Corollary I.3. *Let d, Δ, s, and $\varepsilon > 0$ be constants. Then for every n-point set in $\mathbb{R}^d$, there is a boundary-fuzzy $\Gamma_{d,\Delta,s}$-range searching data structure with $O(n)$ storage, $O(n \log n)$ expected preprocessing time, and $O(n^{1-1/d+\varepsilon})$ query time.*

We next present a different data structure that, at a somewhat higher preprocessing cost, not only gets rid of the boundary-fuzziness condition but also has a slightly improved query time (in terms of n). The main idea is that we build an auxiliary recursive data structure to handle the potentially large subset of points that lie in the zero set of the partitioning polynomial.

Theorem I.4. *Let d, Δ, s, and $\varepsilon > 0$ be constants. Then the $\Gamma_{d,\Delta,s}$-range searching problem for an arbitrary n-point set in $\mathbb{R}^d$ can be solved with $O(n)$ storage, $O(n^{1+\varepsilon})$ expected preprocessing time, and $O(n^{1-1/d} \log^B n)$ query time, where B is a constant depending on d, Δ, s and ε.*

We remark that the dependence of B on Δ, s, and ε is reasonable, but its dependence on d is of the form $d^{O(d)}$.

II. Polynomial Partitions

In this section we briefly review the Guth-Katz technique for later use. We begin by stating their result.

Theorem II.1 (Guth-Katz [12])**.** *Given a set P of n points in $\mathbb{R}^d$ and a parameter $r \le n$, there exists an r-partitioning polynomial for P of degree at most $O(r^{1/d})$ (for d fixed).*

The degree in the theorem is asymptotically optimal in the worst case because the number of connected components of $\mathbb{R}^d \setminus Z(f)$ is $O((\deg f)^d)$ for every polynomial f (see, e.g., Warren [31, Theorem 2]).

The Guth-Katz proof uses the following (discrete version of the) *polynomial ham sandwich* theorem of Stone and Tukey [30]: *If $A_1, \ldots, A_k$ are finite sets in $\mathbb{R}^d$ and D is an integer satisfying $\binom{D+d}{d} - 1 \ge k$, then there exists a nonzero polynomial f of degree at most D that simultaneously bisects all the sets A_i.* Here "f bisects A_i" means that $f > 0$ in at most $\lfloor |A_i|/2 \rfloor$ points of A_i and $f < 0$ in at most $\lfloor |A_i|/2 \rfloor$ points of A_i; f might vanish at any number of the points of A_i, possibly even at all of them.

Guth and Katz inductively construct collections $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_m$ of subsets of P. For $j = 0, 1, \ldots, m$, $\mathcal{P}_j$ consists of at most 2^j pairwise-disjoint subsets of P, each of size at most $n/2^j$; $\bigcup \mathcal{P}_j$ may not contain all points of P. Initially, we have $\mathcal{P}_0 = \{P\}$. The algorithm stops when each subset in $\mathcal{P}_m$ has at most n/r points. By the invariant, $m \le \lceil \log_2 r \rceil$. Having constructed $\mathcal{P}_{j-1}$, we use the polynomial ham-sandwich theorem to construct a polynomial f_j that bisects each set of $\mathcal{P}_{j-1}$, with $\deg f_j = O(2^{j/d})$ (this is indeed an asymptotic upper bound for the smallest D satisfying $\binom{D+d}{d} - 1 \ge 2^{j-1}$, assuming d to be a constant). For every subset $Q \in \mathcal{P}_{j-1}$, let $Q^+ = \{q \in Q \mid f_j(q) > 0\}$ and $Q^- = \{q \in Q \mid f_j(q) < 0\}$. We set $\mathcal{P}_j := \{Q^+, Q^- \mid Q \in \mathcal{P}_{j-1}\}$; empty subsets are not included in $\mathcal{P}_j$.

The desired r-partitioning polynomial for P is then $f := f_1 f_2 \cdots f_m$. We have

$$\deg f = \sum_{j=1}^{m} \deg f_j = \sum_{j=1}^{m} O(2^{j/d}) = O(r^{1/d}).$$

By construction, the points of P lying in a single connected component of $\mathbb{R}^d \setminus Z(f)$ belong to a single member of $\mathcal{P}_m$, which implies that each connected component contains at most n/r points of P.

For the sake of completeness, we recall the proof of the Stone-Tukey polynomial ham-sandwich theorem. We begin by observing that $\binom{D+d}{d} - 1$ is the number of all nonconstant monomials of degree at most D in d variables. Thus, we fix a collection $\mathcal{M}$ of $k \le \binom{D+d}{d} - 1$ such monomials. Let $\Phi\colon \mathbb{R}^d \to \mathbb{R}^k$ be the corresponding *Veronese map*, which maps a point $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ to the k-tuple of the values at $(x_1, \ldots, x_d)$ of the monomials from $\mathcal{M}$. For example, for $d = 2$, $D = 3$, and $k = 8 \le \binom{3+2}{2} - 1$, we may use $\Phi(x_1, x_2) = (x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2) \in \mathbb{R}^8$.

Let $B_i := \Phi(A_i) \subset \mathbb{R}^k$, $i = 1, \ldots, k$. By the standard *ham-sandwich theorem* (see, e.g., [24]), there exists a hyperplane h in $\mathbb{R}^k$ that simultaneously bisects all the B_i's, in the sense that each open halfspace bounded by h contains at most half of the points of each of the sets B_i. In a more algebraic language, there is a nonzero k-variate linear polynomial, which we also call h, that bisects all the B_i's, in the sense of being positive on at most half of the points of each B_i, and being negative on at most half of the points of each B_i. Then $f := h \circ \Phi$ is the desired d-variate polynomial of degree at most D bisecting all the A_i's.

III. Constructing a Partitioning Polynomial

In this section we present an efficient randomized algorithm that, given a point set P and a parameter $r < n$, constructs an r-partitioning polynomial. The main difficulty in converting the above Guth-Katz proof into an efficient algorithm is the use of the (standard) ham-sandwich theorem in a possibly high-dimensional space $\mathbb{R}^k$. A straightforward algorithm for computing ham-sandwich cuts in $\mathbb{R}^k$ inspects all possible ways of splitting the input point sets by a hyperplane, and has running time about n^k. Compared to this easy upper bound, the best known ham-sandwich algorithms can save a factor of about n [20], but this is insignificant in higher dimensions. A recent result of Knauer, Tiwari, and Werner [17] shows that a certain incremental variant of computing a ham-sandwich cut is $W[1]$-hard (where the parameter is the dimension), and thus one perhaps should not expect much better exact algorithms.

We observe that the exact bisection of each A_i is not needed in the Guth-Katz construction—it is sufficient to replace the Stone–Tukey polynomial ham-sandwich theorem by a weaker result, as described below.

Constructing a well-dissecting polynomial. We say that a polynomial f is *well-dissecting* for a point set A if $f > 0$ on at most $\frac{7}{8}|A|$ points of A and $f < 0$ on at most $\frac{7}{8}|A|$ points of A. Given point sets $A_1, \ldots, A_k$ in $\mathbb{R}^d$ with n points in total, we present a Las-Vegas algorithm for constructing a polynomial f of degree $O(k^{1/d})$ that is well-dissecting for at least $\lceil k/2 \rceil$ of the A_i's.

As in the above proof of the Stone–Tukey polynomial ham-sandwich theorem, let D be the smallest integer satisfying $\binom{D+d}{d} - 1 \geq k$. We fix a collection $\mathcal{M}$ of k distinct nonconstant monomials of degree at most D, and let Φ be the corresponding Veronese map. For each $i = 1, 2, \ldots, k$, we pick a point $a_i \in A_i$ uniformly at random and compute $b_i := \Phi(a_i)$. Let h be a hyperplane in $\mathbb{R}^k$ passing through $b_1, \ldots, b_k$, which can be found by solving a system of linear equations, in $O(k^3)$ time.

If the points $b_1, \ldots, b_k$ are not affinely independent, then h is not determined uniquely (this is a technical nuisance, which the reader may want to ignore on first reading). In order to handle this case, we prepare in advance, before picking the a_i's, *auxiliary* affinely independent points $q_1, \ldots, q_k$ in $\mathbb{R}^k$, which are in general position with respect to $\Phi(A_1), \ldots, \Phi(A_k)$; here we mean the "ordinary" general position, i.e., no unnecessary affine dependences, that involve some of the q_i's and the other points, arise. The points q_i can be chosen at random, say, uniformly in the unit cube; with high probability, they have the desired general position property. (If we do not want to assume the capability of choosing a random real number, we can pick the q_i's uniformly at random from a sufficiently large discrete set.) If the dimension of the affine hull of $b_1, \ldots, b_k$ is $k' < k-1$, we choose the hyperplane h through $b_1, \ldots, b_k$ and $q_1, \ldots, q_{k-k'-1}$. If h is not unique, i.e., $q_1, \ldots q_{k-k'-1}$ are not affinely independent with respect to $b_1, \ldots b_k$, which we can detect while solving the linear system, we restart the algorithm by choosing $q_1, \ldots, q_k$ anew and then picking new $a_1, \ldots, a_k$. In this way, after a constant expected number of iterations, we obtain the uniquely determined hyperplane h through $b_1, \ldots, b_k$ and $q_1, \ldots, q_{k-k'-1}$ as above, and we let $f = h \circ \Phi$ denote the resulting d-variate polynomial. We refer to these steps as one *trial* of the algorithm. For each A_i, we check whether f is well-dissecting for A_i. If f is well-dissecting for only fewer than $k/2$ sets, then we discard f and perform another trial.

We now analyze the expected running time of the algorithm. The intuition is that f is expected to well-dissect a significant fraction, say, at least half, of the sets A_i. This intuition is reflected in the next lemma. Let X_i be the indicator variable of the event: A_i *is* ***not*** *well-dissected by* f.

Lemma III.1. *For every* $i = 1, 2, \ldots, k$, $\mathbb{E}[X_i] \leq 1/4$.

Proof: Let us fix i and the choices of a_j (and thus of $b_j = \Phi(a_j)$) for all $j \neq i$. Let k_0 be the dimension of F_0, the affine hull of $\{b_j \mid j \neq i\}$. Then the resulting hyperplane h passes through the $(k-2)$-flat F spanned by F_0 and $q_1, \ldots, q_{k-k_0-2}$, irrespective of which point of A_i is chosen. If a_i, the point chosen from A_i, is such that $b_i = \Phi(a_i)$ lies on F_0, then h also passes through q_{k-k_0-1}.

Put $B_i := \Phi(A_i)$, and let us project the configuration orthogonally to a 2-dimensional plane π orthogonal to F. Then F appears as a point $F^* \in \pi$, and B_i projects to a (multi)set B_i^* in π. The random hyperplane h projects to a random line h^* in π, whose choice can be interpreted as follows: pick $b_i^* \in B_i^*$ uniformly at random; if $b_i^* \neq F^*$, then h^* is the unique line through b_i^* and F^*; otherwise, when $b_i^* = F^*$, h^* is the unique line through F^* and $q_{k-k_0-1}^*$; by construction, $q_{k-k_0-1}^* \neq F^*$. The indicator variable X_i is 1 if and only if the resulting h^* has more than $\frac{7}{8}|B_i^*|$ points of B_i^* (strictly) on one side.

The special role of $q_{k-k_0-1}^*$ can be eliminated if we first move the points of B_i^* coinciding with F^* to the point $q_{k-k_0-1}^*$, and then slightly perturb the points so as to ensure that all points of B_i^* are distinct and lie at distinct orientations from F^*; it is easy to see that these transformations cannot decrease the probability of $X_i = 1$. Finally, we note that whether a point $b^* \in B_i^*$ lies below or above h^* only depends on the orientation of the vector $\overrightarrow{F^*b^*}$, so we can also assume the points of B_i^* to lie on the unit circle around F^*.

Using the standard planar ham-sandwich theorem, we partition B_i^* into two subsets L_i^* and R_i^* of equal size by a line through the center F^*. Then we bisect L_i^* by a ray from F^*, and we do the same for R_i^*. It is easily checked (see Figure 1) that there always exist two of the resulting quarters, one of L_i^* and one of R_i^* (the ones whose union forms an angle $\leq \pi$ between the two bisecting rays), such that every line connecting F^* with a point in either quarter contains at least $\frac{1}{4}|B_i^*|$ points of B_i^* on each side. Referring to these quarters as "good", we now take one of the bisecting rays, say that of L_i^*, and rotate it about F^* away from the good quarter of L_i^*. Each of the $\frac{1}{8}|B_i^*|$ points that the ray encounters has the property that the line supporting the ray has at least $\frac{1}{8}|B_i^*|$ points of B_i^* on each side. This implies that, for at least half of the points in each of the two remaining quarters, the line connecting F^* to such a point has at least $\frac{1}{8}|B_i^*|$ points of B_i^* on each side. Hence at most $\frac{1}{4}|B_i|$ points of B_i can lead to a cut that is not well-dissecting for B_i.

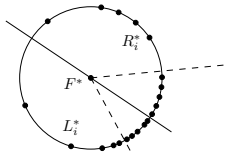

Fig. 1. Illustration to the proof of Lemma III.1.

We conclude that, still conditioned on the choices of a_j, $j \neq i$, the event $X_i = 1$ has probability at most $1/4$. Since this holds for every choice of the a_j, $j \neq i$, the unconditional probability of $X_i = 1$ is also at most $1/4$, and thus $\mathbb{E}[X_i] \leq 1/4$ as claimed. ∎

Hence, the expected number of sets A_i that are not well-

dissected by f is

$$\mathbb{E}\big[\sum_{i=1}^{k} X_i\big] = \sum_{i=1}^{k} \mathbb{E}[X_i] \le k/4.$$

By Markov's inequality, with probability at least $1/2$, at least half of the A_i's are well-dissected by f. We thus obtain a polynomial that is well-dissecting for at least half of the A_i's after an expected constant number of trials.

It remains to estimate the running time of each trial. The points $b_1, \ldots, b_k$ can be chosen in $O(n)$ time. Computing h involves solving a $k \times k$ linear system, which can be done in $O(k^3)$ time using Gaussian elimination, or even faster using fast matrix multiplication. Note that we do *not* need to actually compute the entire sets $\Phi(A_i)$. No computation is needed for passing from h to f—we just re-interpret the coefficients. To check which of $A_1, \ldots A_k$ are well-dissected by f, we evaluate f at each point of $A = \bigcup_i A_i$. First we evaluate each of the k monomials in $\mathcal{M}$ at each point of A. If we proceed incrementally, from lower degrees to higher ones, this can be done with $O(1)$ operations per monomial and point of A, in $O(nk)$ time in total. Then, in additional $O(nk)$ time, we compute the values of $f(q)$, for all $q \in A$, from the values of the monomials. Putting everything together we obtain the following lemma.

Lemma III.2. *Given point sets $A_1, \ldots, A_k$ in $\mathbb{R}^d$ (for fixed d) with n points in total, a polynomial f of degree $O(k^{1/d})$ that is well-dissecting for at least $\lceil k/2 \rceil$ of the A_i's can be constructed in $O(nk + k^3)$ randomized expected time.*

Constructing a partitioning polynomial. We now describe the algorithm for computing an r-partitioning polynomial f. We essentially imitate the Guth–Katz proof, with Lemma III.2 replacing the polynomial ham-sandwich theorem, but with an additional twist.

The algorithm works in phases. At the end of the j-th phase, for $j \ge 1$, we have a family $f_1, \ldots, f_j$ of j polynomials and a family $\mathcal{P}_j$ of at most 2^j pairwise-disjoint subsets of P, each of size at most $(7/8)^j n$. $\mathcal{P}_j$ is not necessarily a partition of P, since the points of $P \cap Z(f_1 f_2 \cdots f_j)$ do not belong to $\bigcup \mathcal{P}_j$. Initially, $\mathcal{P}_0 = \{P\}$. The algorithm stops when each set in $\mathcal{P}_j$ has at most n/r points. In the j-th phase, the algorithm constructs f_j and $\mathcal{P}_j$ as follows.

At the beginning of the j-th phase, let $\mathcal{L}_j = \{Q \in \mathcal{P}_{j-1} \mid |Q| > (7/8)^j n\}$ be the family of the "large" sets in $\mathcal{P}_{j-1}$, and set $\kappa_j = |\mathcal{L}_j| \le (8/7)^j$. We also initialize the collection $\mathcal{P}_j$ to $\mathcal{P}_{j-1} \setminus \mathcal{L}_j$, the family of "small" sets in $\mathcal{P}_{j-1}$. Then we perform at most $\lceil \log_2 \kappa_j \rceil$ dissecting steps, as follows: After s steps, we have a family $g_1, \ldots, g_s$ of polynomials, the current set $\mathcal{P}_j$, and a subfamily $\mathcal{L}_j^{(s)} \subseteq \mathcal{L}_j$ of size at most $\kappa_j/2^s$, consisting of the members of $\mathcal{L}_j$ that were not well-dissected by any of $g_1, \ldots, g_s$. If $\mathcal{L}_j^{(s)} \ne \emptyset$ we choose, using Lemma III.2, a polynomial g_{s+1} of degree at most $c(\kappa_j/2^s)^{1/d}$ (with a suitable constant c that depends only on d) that well-dissects at least half of the members of $\mathcal{L}_j^{(s)}$. For each $Q \in \mathcal{L}_j^{(s)}$, let $Q^+ = \{q \in Q \mid g_{s+1}(q) > 0\}$ and $Q^- = \{q \in Q \mid g_{s+1}(q) < 0\}$. If Q is well-dissected, i.e., $|Q^+|, |Q^-| \le \frac{7}{8}|Q|$, then we add Q^+, Q^- to $\mathcal{P}_j$, and otherwise, we add Q to $\mathcal{L}_j^{(s+1)}$. Note that in the former case the points $q \in Q$ satisfying $g_{s+1}(q) = 0$ are "lost" and do not participate in the subsequent dissections. By Lemma III.2, $|\mathcal{L}_j^{(s+1)}| \le |\mathcal{L}_j^{(s)}|/2 \le \kappa_j/2^{s+1}$.

The j-th phase is completed when $\mathcal{L}_j^{(s)} = \emptyset$, in which case we set $f_j := \prod_{\ell=1}^{s} g_\ell$. By construction, each point set in $\mathcal{P}_j$ has at most $(7/8)^j n$ points, and every point not in $\cup \mathcal{P}_j$ lies in $Z(f_1 \cdots f_j)$. Furthermore,

$$\deg f_j \le \sum_{s \ge 0} c(\kappa_j/2^s)^{1/d} = O(\kappa_j^{1/d}),$$

where again the constant of proportionality depends only on d. Since every set in $\mathcal{P}_{j-1}$ is split into at most two sets before being added to $\mathcal{P}_j$, $|\mathcal{P}_j| \le 2|\mathcal{P}_{j-1}| \le 2^j$.

If $\mathcal{P}_j$ contains subsets with more than n/r points, we begin the $(j+1)$-st phase with the current $\mathcal{P}_j$; otherwise the algorithm stops and returns $f := f_1 f_2 \cdots f_j$. This completes the description of the algorithm.

Clearly, the number m of phases of the algorithm is at most $\lceil \log_{8/7} r \rceil$. Following the same argument as in [12], it can be shown that all points lying in a single connected component of $\mathbb{R}^d \setminus Z(f)$ belong to a single member of $\mathcal{P}_m$, and thus each connected component contains at most n/r points of P. Since the degree of f_j is $O(\kappa_j^{1/d})$, $\kappa_j \le (8/7)^j$, and $m \le \lceil \log_{8/7} r \rceil$, we conclude that

$$\deg f = O\Big(\sum_{j=1}^{m} \kappa_j^{1/d}\Big) = O\Big(\sum_{j=1}^{m} (8/7)^{j/d}\Big) = O(r^{1/d}).$$

As for the expected running time of the algorithm, the s-th step of the j-th phase takes $O(n\kappa_j/2^s + (\kappa_j/2^s)^3)$ expected time, so the j-th phase takes a total of $O(n\kappa_j + \kappa_j^3)$ expected time. Substituting $\kappa_j \le (8/7)^j$ in the above bound and summing over all j, the overall expected running time of the algorithm is $O(nr + r^3)$. This completes the proof of Theorem I.1.

Remark. It is a challenging open problem to improve the expected running time in Theorem I.1 to $O(n \operatorname{polylog}(n))$ when r is a small fractional power of n, say, $r = n^{0.001}$. The bottleneck in the current algorithm is the subproblem of evaluating a given d-variate polynomial f of degree $D = O(r^{1/d})$ at n given points; everything else can be performed in $O(n \operatorname{polylog}(r) + r^{O(1)})$ expected time. Finding the signs of f at those points would actually suffice, but this probably does not make the problem any simpler.

This problem of *multi-evaluation* of multivariate real polynomials has been considered in the literature, and there is a nontrivial improvement over the straightforward $O(nr)$ algorithm, due to Nüsken and Ziegler [25]. However, its running time is still a far cry from what we are aiming at. Let us remark that in a different setting, for polynomials over finite fields (and over certain more general finite rings), there is a remarkable method for multi-evaluation by Kedlaya and

Umans [16] achieving $O(((n+D^d)\log q)^{1+\varepsilon})$ running time, where q is the cardinality of the field.

IV. Crossing a Polynomial Partition with a Range

In this section we define the crossing number of a polynomial partition and describe an algorithm for computing the cells of a polynomial partition that are crossed by a semialgebraic range, both of which will be crucial for our range-searching data structures. We begin by recalling a few results on arrangements of algebraic surfaces.

Let Σ be a set of algebraic surfaces in $\mathbb{R}^d$. The *arrangement* of Σ, denoted by $\mathcal{A}(\Sigma)$, is the partition of $\mathbb{R}^d$ into maximal relatively open connected subsets, called *cells*, such that all points within each cell lie in the the same subset of Σ. If $\mathcal{F}$ is a set of d-variate polynomials, then with a slight abuse of notation, we use $\mathcal{A}(\mathcal{F})$ to denote the arrangement $\mathcal{A}(\{Z(f) \mid f \in \mathcal{F}\})$. We need the following result on arrangements:

Lemma IV.1 (Basu, Pollack and Roy [5, Theorem 16.18])**.** *Let $\mathcal{F} = \{f_1, \ldots, f_s\}$ be a set of d-variate polynomials, each of degree at most Δ. Then the arrangement $\mathcal{A}(\mathcal{F})$ in $\mathbb{R}^d$ has at most $(s\Delta)^{O(d)}$ cells, and it can be computed in time at most $T = s^{d+1}\Delta^{O(d^4)}$. Each cell is described as a semialgebraic set using at most T polynomials of degree bounded by $\Delta^{O(d^3)}$. Moreover, the algorithm supplies adjacency information for the cells, indicating which cells are contained in the boundary of each cell, and it also supplies an explicitly given point in each cell.*

The following recent result of Barone and Basu [4] is a refinement of a series of previous studies; e.g., see [5]:

Lemma IV.2. *Let V be a k-dimensional algebraic surface in $\mathbb{R}^d$ defined by a set $\mathcal{G}$ of d-variate polynomials, each of degree at most Δ, and let $\mathcal{F}$ be a set of s polynomials of degree at most $D \geq \Delta$. Then at most $O(1)^d\Delta^{d-k}(sD)^k$ cells (of all dimensions) of $\mathcal{A}(\mathcal{F}\cup\mathcal{G})$ lie in V.*

The crossing number of polynomial partitions. Let $P \subset \mathbb{R}^d$ be a set of n points in $\mathbb{R}^d$, and let f be an r-partitioning polynomial for P. Recall that the *polynomial partition* $\Omega = \Omega(f)$ induced by f is the partition of $\mathbb{R}^d$ into the zero set $Z(f)$ and the connected components $\omega_1, \omega_2, \ldots, \omega_t$ of $\mathbb{R}^d \setminus Z(f)$. As already noted, Warren's theorem [31] implies that $t = O(r)$. We call $\omega_1, \ldots, \omega_t$ the *cells* of Ω (although they need not be cells in the sense typical, e.g., in topology; they need not even be simply connected). Ω also induces a partition $P^*, P_1, \cdots, P_t$ of P, where $P^* = P \cap Z(f)$ is the *exceptional part*, and $P_i = P \cap \omega_i$, for $i = 1, \ldots, t$, are the *regular parts*. By construction, $|P_i| \leq n/r$ for every $1 \leq i \leq t$, but we have no control over the size of P^*—this will be the source of most of our technical difficulties.

Next, let γ be a range in $\Gamma_{d,\Delta,s}$. We say that γ *crosses* a cell ω_i if neither $\omega_i \subseteq \gamma$ nor $\omega_i \cap \gamma = \emptyset$. The *crossing number* of γ is the number of cells of Ω crossed by γ, and the *crossing number* of Ω (with respect to $\Gamma_{d,\Delta,s}$) is the maximum of the crossing numbers of all $\gamma \in \Gamma_{d,\Delta,s}$. Similar to many previous range-searching algorithms [6], [21], [22], the crossing number of Ω will determine the query time of our range searching algorithms described in Sections V and VII.

Lemma IV.3. *If Ω is a polynomial partition induced by an r-partitioning polynomial of degree at most D, then the crossing number of Ω with respect to $\Gamma_{d,\Delta,s}$, with $\Delta \leq D$, is at most $Cs\Delta D^{d-1}$, where C is a suitable constant depending only on d.*

Proof: Let $\gamma \in \Gamma_{d,\Delta,s}$; then γ is a Boolean combination of up to s sets of the form $\gamma_j := \{x \in \mathbb{R}^d \mid g_j(x) \geq 0\}$, where $g_1, \ldots, g_s$ are polynomials of degree at most Δ. If γ crosses a cell ω_i, then at least one of the ranges γ_j also crosses ω_i, and thus it suffices to establish that the crossing number of any range γ, defined by a single d-variate polynomial inequality $g(x) \geq 0$ of degree at most Δ, is at most $C\Delta D^{d-1}$.

We apply Lemma IV.2 with $V := Z(g)$, which is a surface of dimension $k \leq d-1$, and with $s = 1$ and $\mathcal{F} = \{f\}$, which is the r-partitioning polynomial. Then, for each cell ω_i crossed by γ, $\omega_i \cap Z(g)$ is a nonempty union of some of the cells in $\mathcal{A}(\mathcal{F}\cup\{g\})$ that lie in V. Thus, the crossing number of γ is at most $O(1)^d\Delta D^{d-1}$, and multiplying this bound by s yields the bounded asserted in the lemma. ∎

Algorithmic issues. We need to perform the following algorithmic primitives (for d *fixed* as usual) for the range-searching algorithms:

(A1) Given an r-partitioning polynomial f of degree $D = O(r^{1/d})$, compute (a suitable representation of) the partition Ω and the induced partition of P into $P^*, P_1, \ldots, P_t$. By computing $\mathcal{A}(\{f\})$, using Lemma IV.1, and then testing the membership of each point $p \in P$ in each cell ω_i in time polynomial in r, the above operation can be performed in $O(nr^c)$ time,where c is a constant depending only on d.

(A2) Given (a suitable representation of) Ω as in (A1) and a query range $\gamma \in \Gamma_{d,\Delta,1}$, i.e., a range defined by a single d-variate polynomial g of degree $\Delta \leq D$, compute which of the cells of Ω are crossed by γ and which are completely contained in γ.
By computing the arrangement $\mathcal{A}(\{f,g\})$ and deducing the required classification of the cells ω_i from the combinatorial information about the cells of this arrangement, using Lemma IV.1, the above task can be accomplished in time $O(r^c)$, with c as in (A1).

V. Constant Fan-Out Partition Tree

We are now ready to describe our first data structure for $\Gamma_{d,\Delta,s}$-range searching, which is a constant fan-out (branching degree) partition tree, and works for points in general position. For points not in general position, it provides a boundary-fuzzy output count, as discussed in the Introduction.

Let P be a set of n points in $\mathbb{R}^d$, and let Δ, s be constants. We choose r as a (large) constant depending on d, Δ, and ε. We assume P to be in D_0-general position for some sufficiently large constant $D_0 \gg r^{1/d}$. We construct a partition tree $\mathcal{T}$ of fan-out $O(r)$ as follows. We first construct an

r-partitioning polynomial f for P using Theorem I.1, and compute the partition Ω of $\mathbb{R}^d$ induced by f, as well as the corresponding partition $P = P^* \cup P_1 \cup \cdots \cup P_t$ of P, where $t = O(r)$. Since r is a constant, the (A1) operation, discussed in Section IV, performs this computation in $O(n)$ time. Since P is in D_0-general position, and since we choose D_0 to be at least $\deg f$, the size of $P^* = P \cap Z(f)$ is bounded by D_0.

We set up the root of $\mathcal{T}$, where we store the partition polynomial f, a suitable representation of the partition Ω, a list of the points of the exceptional part P^*, and $w(P_i)$, the sum of weights of all points of P_i, for each $i = 1, 2, \ldots, t$. The regular parts P_i are not stored explicitly at the root. Instead, for each P_i we recursively build a subtree representing it. The recursion terminates, at leaves of $\mathcal{T}$, as soon as we reach point sets of size smaller than a suitable constant n_0. The points of each such set are stored explicitly at the corresponding leaf of $\mathcal{T}$.

Since each node of $\mathcal{T}$ requires only a constant amount of storage and each point of P is stored at only one node of $\mathcal{T}$, the total size of $\mathcal{T}$ is $O(n)$. The preprocessing time is $O(n \log n)$ since $\mathcal{T}$ has depth $O(\log_r n)$ and each level is processed in $O(n)$ time.

To process a query range $\gamma \in \Gamma_{d,\Delta,s}$, we start at the root of $\mathcal{T}$ and maintain a global counter which is initially set to 0. Among the cells $\omega_1, \ldots, \omega_t$ of the partition Ω stored at the root, we find, using the (A2) operation, those cells completely contained in γ, and those crossed by γ. Actually, we compute a superset of the cells that γ crosses, namely, the cells crossed by the zero set of at least one of the (at most s) polynomials defining γ. For each cell $\omega_i \subseteq \gamma$, we add the weight $w(P_i)$ to the global counter. We also add to the global counter the weights of the points in $P^* \cap \gamma$, which we find by inspecting each point of P^* separately. Then we recurse in each subtree corresponding to a cell ω_i crossed by γ. The leaves, with point sets of size $O(1)$, are processed by inspecting their points individually. By Lemma IV.3, the number of cells crossed by any of the polynomials defining γ at any interior node of $\mathcal{T}$ is at most $Cs\Delta D^{d-1} \le C' r^{1-1/d}$, where $C' = C'(d, s, \Delta)$ is a constant independent of r.

The query time $\mathsf{Q}(n)$ obeys the following recurrence:

$$\mathsf{Q}(n) \le \begin{cases} C' r^{1-1/d} \mathsf{Q}(n/r) + O(1) & \text{for } n > n_0, \\ O(n) & \text{for } n \le n_0, \end{cases}$$

It is well known (e.g., see [21]), and easy to check, that the recurrence solves to $\mathsf{Q}(n) = O(n^{1-1/d+\varepsilon})$, for every fixed $\varepsilon > 0$, with an appropriate sufficiently large choice of r as a function of C' and ε, and with an appropriate choice of n_0. This concludes the proof of Theorem I.2.

If the points of P are not in D_0-general position, we perturb them infinitesimally using the general perturbation scheme of Yap [33], so that the perturbed set is in D_0-general position. Then we construct the above data structure on the perturbed point set. By answering the query for this perturbed set, we obtain a boundary-fuzzy answer for the original point set. The preprocessing cost, storage, and query time remain asymptotically the same as in Theorem I.2. This concludes the proof of Corollary I.3.

VI. Decomposing a Surface into Monotone Patches

As mentioned in the Introduction, if we construct an r-partitioning polynomial f for an arbitrary point set P, the exceptional set $P^* = P \cap Z(f)$ may be large, as is schematically indicated in Fig. 2 (left). Since P^* is not partitioned by f in any reasonable sense, it must be handled differently, as described below.

Following the terminology in [13], [26], we call a direction $v \in \mathbb{S}^{d-1}$ *good* for f if, for any $a \in \mathbb{R}^d$, the polynomial $p(t) = f(a + vt)$ does not vanish identically, that is, any line in direction v intersects $Z(f)$ at finitely many points. As argued in [26], a random direction is good for f with high probability. By choosing a good direction and rotating the coordinate system, we assume that the x_d-direction, referred to as the *vertical* direction, is good for f.

In order to deal with P^*, we partition $Z(f)$ into finitely many pieces, called *patches*, in such a way that each of the patches is *monotone* in the vertical direction, meaning that every line parallel to the x_d-axis intersects it at most once (in Fig. 2 (middle), there are five one-dimensional patches $\pi_1, \ldots, \pi_5$). Then we treat each patch π separately: We project the point set $P^* \cap \pi$ orthogonally to the coordinate hyperplane $H := \{x_d = 0\}$, and we preprocess the projected set, denoted P^*_π, for range searching with suitable ranges. These ranges are projections of ranges of the form $\gamma \cap \pi$, where $\gamma \in \Gamma_{d,\Delta,s}$ is one of the original ranges. In Fig. 2 (middle), the patch π_1 is drawn thick, a range γ is indicated as a gray disk, and the projection γ_{π_1} of $\gamma \cap \pi_1$ is shown as a thick segment in H.

The projected range γ_π is typically more complicated than the original range γ (it involves more polynomials of larger degrees), but crucially, it is only $(d-1)$-dimensional, and $(d-1)$-dimensional queries can be processed somewhat more efficiently than d-dimensional ones, which makes the whole scheme work. We will discuss this in more detail in Section VII below, but first we recall the notion of *cylindrical algebraic decomposition* (CAD, or also *Collins decomposition*), which is a tool that allows us to decompose $Z(f)$ into monotone patches, and also to compute the projected ranges γ_π.

Given a finite set $\mathcal{F} = \{f_1, \ldots, f_s\}$ of d-variate polynomials, a *cylindrical algebraic decomposition adapted to* $\mathcal{F}$ is a way of decomposing $\mathbb{R}^d$ into a finite collection of relatively open *cells*, which have a simple shape (in a suitable sense), and which refine the arrangement $\mathcal{A}(\mathcal{F})$. We refer, e.g., to [5, Chap. 5,12] for a definition and construction of the "standard" CAD. Here we will use a simplified variant, which can be regarded as the "first stage" of the standard CAD, and which is captured by [5, Theorem 5.14, Algorithm 12.1]. We also refer to [26, Appendix A] for a concise treatment, which is perhaps more accessible at first encounter.

Let $\mathcal{F}$ consist of polynomials in $\mathbb{R}[x_1, \ldots, x_d]$. To obtain the first-stage CAD, one constructs a suitable collection $\mathcal{E} = \mathcal{E}(\mathcal{F})$ of polynomials in the variables $x_1, \ldots, x_{d-1}$ (denoted by

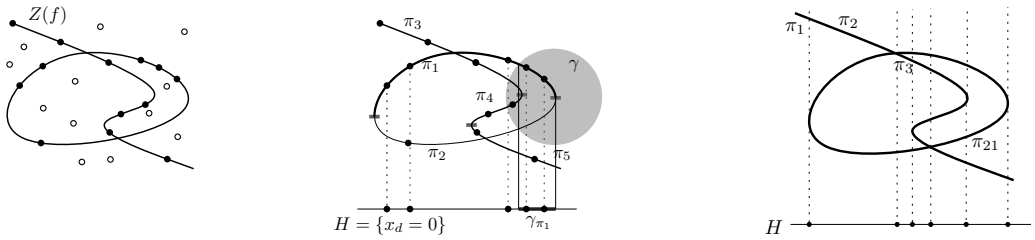

Fig. 2. Left: the zero set of the partitioning polynomial. Middle: a partition of the zero set into monotone patches that project to the hyperplane H bijectively. Right: a schematic illustration of the first-stage cylindrical algebraic decomposition; the zero sets of the polynomials in $\mathcal{E}$ are indicated by the dots in H, and the vertical walls are drawn by dashed lines.

$\mathrm{Elim}_{X_k}(\mathcal{F})$ in [5]). Roughly speaking, the zero sets of the polynomials in $\mathcal{E}$, viewed as subsets of the coordinate hyperplane H (which is identified with $\mathbb{R}^{d-1}$), contain the projection onto H of all intersections $Z(f_i) \cap Z(f_j)$, $1 \le i < j \le s$, as well as the projection of the loci in $Z(f_i)$ where $Z(f_i)$ has a vertical tangent hyperplane, or a singularity of some kind. The actual construction of $\mathcal{E}$ is somewhat more complicated, and we refer to the aforementioned references for more details.

Having constructed $\mathcal{E}$, the single-stage CAD is obtained as the arrangement $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$ in $\mathbb{R}^d$, where the polynomials in $\mathcal{E}$ are now considered as d-variate polynomials (in which the variable x_d is not present). In geometric terms, we erect a "vertical wall" in $\mathbb{R}^d$ over each zero set within H of a $(d-1)$-variate polynomial from $\mathcal{E}$, and the CAD is the arrangement of these vertical walls plus the zero sets of $f_1, \ldots, f_s$. The first-stage CAD is illustrated in Fig. 2 (right), for the same (single) polynomial as in Fig. 2 (left).

In our algorithm, we are interested in the cells of the CAD that are contained in some of the $Z(f_i)$; these are going to be the monotone patches alluded to above. The following lemma summarizes the properties of the first-stage CAD that we will need; we refer to [5, Theorem 5.14, Algorithm 12.1] for a proof.

Lemma VI.1 (Single-stage CAD)**.** *Given a set $\mathcal{F} = \{f_1, \ldots, f_s\} \subset \mathbb{R}[x_1, \ldots, x_d]$, each of degree at most D, there is a set $\mathcal{E} = \mathcal{E}(\mathcal{F})$ of $O(s^2 D^3)$ polynomials in $x_1, \ldots, x_{d-1}$, each of degree $O(D^2)$, which can be computed in time $s^2 D^{O(d)}$, such that the first-stage CAD defined by these polynomials, i.e., the arrangement $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$ in $\mathbb{R}^d$, has the following properties:*

(i) ("Cylindrical" cells) *For each cell σ of $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$, there exists a unique cell τ of the $(d-1)$-dimensional arrangement $\mathcal{A}(\mathcal{E})$ in H, such that one of the following possibilities occur:*
   (a) *$\sigma = \{(x, \xi(x)) \mid x \in \tau\}$, where $\xi \colon \tau \to \mathbb{R}$ is a continuous semialgebraic function (that is, σ is the graph of ξ above τ).*
   (b) *$\sigma = \{(x,t) \mid x \in \tau, t \in (\xi_1(x), \xi_2(x))\}$, where each ξ_i, $i = 1, 2$, is either a continuous semialgebraic function $\tau \to \mathbb{R}$, or the constant function $\tau \to \{\infty\}$, or the constant function $\tau \to \{-\infty\}$, and $\xi_1(x) < \xi_2(x)$ for all $x \in \tau$ (that is, σ is a portion of the "cylinder" $\tau \times \mathbb{R}$ between two consecutive graphs).*

(ii) (Refinement property) *If $\mathcal{F}' \subseteq \mathcal{F}$, then $\mathcal{E}' = \mathcal{E}(\mathcal{F}') \subseteq \mathcal{E}$, and thus each cell of $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$ is fully contained in some cell of $\mathcal{A}(\mathcal{F}' \cup \mathcal{E}')$.*

Returning to the problem of decomposing the zero set of the partitioning polynomial f into monotone patches, we construct the first-stage CAD adapted to $\mathcal{F} = \{f\}$, and the patches are the cells of $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$ contained in $Z(f)$. If the x_d-direction is good for f, every cell of $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$ lying in $Z(f)$ is of type (a), so if any cell of type (b) lies in $Z(f)$, we choose another random direction and construct the first-stage CAD in that direction. Putting everything together and using Lemma IV.1 to bound the complexity of $\mathcal{A}(\mathcal{F} \cup \mathcal{E})$, we obtain the following lemma.

Lemma VI.2. *Let f be a d-variate polynomial of degree D and v a good direction for f. Then f can be decomposed, in $D^{O(d^4)}$ time, into $D^{O(d)}$ patches each of which is monotone in direction v and can be represented semialgebraically by $D^{O(d^4)}$ polynomials of degree $D^{O(d^3)}$.*

The first-stage CAD can also be used to compute the projection of the intersection of a range in $\Gamma_{d,\Delta,s}$ with a monotone patch of f. Essentially, this is done by forming the arrangement of f and the polynomials defining γ, and by collecting the monotone patches in this arrangement that are contained in $Z(f)$; see the full version [3] for more details.

Lemma VI.3. *Let Π be the decomposition of the zero set of a d-variate polynomial f of degree D into monotone patches with respect to a good direction v, as described in Lemma VI.2, and let γ be a semialgebraic set in $\Gamma_{d,\Delta,s}$, with $\Delta \le D$. For any patch $\pi \in \Pi$, the projection of $\gamma \cap \pi$ in direction v can be represented as a member of $\Gamma_{d-1,\Delta_1,s_1}$, where $\Delta_1 = D^{O(d^3)}$ and $s_1 = (Ds)^{O(d^4)}$. The representation can be computed in $(Ds)^{O(d^4)}$ time.*

VII. Large Fan-Out Partition Tree

We now describe our second data structure for $\Gamma_{d,\Delta,s}$-range searching. Compared to the first data structure from Section V, this one works on arbitrary point sets, without the D_0-general position assumption, or, alternatively, without the fuzzy boundary constraint on the output, and has slightly better

performance bounds. The data structure is built recursively, and this time the recursion involves both n and d.

A. The data structure

Let P be a set of n points in $\mathbb{R}^d$, and let Δ and s be parameters (not assumed to be constant). The data structure for $\Gamma_{d,\Delta,s}$-range searching on P is obtained by constructing a partition tree $\mathcal{T}$ on P recursively, as above, except that now the fan-out of each node is larger (and non-constant), and each node also stores an auxiliary data structure for handling the respective exceptional part. If $n \leq n_0$, where $n_0 = n_0(d, \Delta, s)$ is a suitable parameter (again, not necessarily a constant) whose choice will be specified later, $\mathcal{T}$ consists of a single leaf that simply stores the points of P. Otherwise (i.e., $n > n_0$), we set r suitably (typically to a tiny power of n)— the choice of r will also be specified later.

We construct an r-partitioning polynomial f of degree $D = O(r^{1/d})$, the partition Ω of $\mathbb{R}^d$ induced by f, and the partition of P into the exceptional part P^* and regular parts $P_1, \ldots, P_t$, where $t = O(r)$. Set $n^* = |P^*|$ and $n_i = |P_i|$, for $i = 1, \ldots, t$. The root of $\mathcal{T}$ stores f, Ω, and the total weight $w(P_i)$ of each regular part P_i of P, as before. Still in the same way as before, we recursively preprocess each regular part P_i for $\Gamma_{d,\Delta,s}$-range searching (or stop if $|P_i| \leq n_0$), and attach the resulting data structure to the root as a respective subtree.

Handling the exceptional part. A novel feature of the second data structure is that we also preprocess the exceptional set P^* into an auxiliary data structure, which is stored at the root. Here we recurse on the dimension, exploiting the fact that P^* lies on the algebraic surface $Z(f)$ of dimension at most $d-1$.

We choose a random direction v and rotate the coordinate system so that v becomes the direction of the x_d-axis. We construct the first-stage CAD adapted to $\{f\}$, according to Lemmas VI.1 and IV.1. We check whether all the patches are x_d-monotone, i.e., of type (a) in Lemma VI.1(i); if it is not the case, we discard the CAD and repeat the construction, with a different random direction. This yields a decomposition of $Z(f)$ into a set Π of $D^{O(d)}$ monotone patches, and the running time is $D^{O(d^4)}$ with high probability.

Next, we distribute the points of P^* among the patches: for each patch $\pi \in \Pi$, let P^*_π denote the projection of $P^* \cap \pi$ onto the coordinate hyperplane $H: \; x_d = 0$. We preprocess each set P^*_π for $\Gamma_{d-1,\Delta_1,s_1}$-range searching. Here $s_1 = (Ds)^{O(d^4)}$ is the number of polynomials defining a range and $\Delta_1 = D^{O(d^3)}$ is their maximum degree; the constants hidden in the $O(\cdot)$ notation are the same as in Lemma VI.3.

The preprocessing of the sets P^*_π is done recursively, using an r_1-partitioning polynomial in $\mathbb{R}^{d-1}$, for a suitable value of r_1. The exceptional set at each node of the resulting "$(d-1)$-dimensional" tree is handled in a similar manner, constructing an auxiliary data structure in $d-2$ dimensions, based on a single-stage CAD of the above kind, and storing it at the corresponding node. The recursion on d bottoms out at dimension 1, where the structure is simply a standard binary search tree over the resulting set of points on the x_1-axis.

This completes the description of the data structure, except for the choice of r which will be provided later.

Answering a query. Assume that, for a given P, the data structure for $\Gamma_{d,\Delta,s}$-range searching, as described above, has been constructed, and consider a query range $\gamma \in \Gamma_{d,\Delta,s}$. The query is answered in the same way as before, by visiting the nodes of the partition tree $\mathcal{T}$ in a top-down manner, except that, at each node that we visit, we also query with γ the auxiliary data structure constructed on the exceptional set P^* for that node.

Specifically, for each patch $\pi \in \Pi$, we compute w_π, the weight of $P^* \cap (\gamma \cap \pi)$. If $\gamma \cap \pi = \emptyset$ then $w_\pi = 0$, and if $\gamma \cap \pi = \pi$ then w_π is the total weight of $P^* \cap \pi$. Otherwise, i.e., if γ crosses π then w_π is the same as the weight of $P^*_\pi \cap \gamma_\pi$ because π is x_d-monotone; γ_π is the x_d-projection of $\gamma \cap \pi$. By Lemma VI.3, $\gamma_\pi \in \Gamma_{d-1,\Delta_1,s_1}$ and can be constructed in $(Ds)^{O(d^4)}$ time. We can therefore find the weight of $\gamma_\pi \cap P^*_\pi$ by querying the auxiliary data structure for P^*_π with γ_π. We then add w_π to the global count maintained by the query procedure. This completes the description of the query procedure.

B. Performance analysis

A straightforward analysis shows that the size of the data structure is linear and that it can be constructed in time $O(n^{1+\varepsilon})$, for any constant $\varepsilon > 0$, by choosing r sufficiently large, so we focus on analyzing the query time.

Let $\mathsf{Q}_d(n, \Delta, s)$ denote the maximum overall query time for $\Gamma_{d,\Delta,s}$-range searching on a set of n points in $\mathbb{R}^d$. For $d = 1$, $\mathsf{Q}_1(n, \Delta, s) = O(\Delta s \log n)$ because any range in $\Gamma_{1,\Delta,s}$ is the union of at most Δs intervals. For $n \leq n_0$, $\mathsf{Q}_d(n, \Delta, s) = O(n)$. For $d > 1$ and $n > n_0$, an analysis similar to the one in Section V gives the following recurrence for $\mathsf{Q}_d(n, \Delta, s)$:

$$\begin{aligned} \mathsf{Q}_d(n, \Delta, s) \;\leq\; & C\Delta s r^{1-1/d} \mathsf{Q}_d(n/r, \Delta, s) \\ & + \sum_{\pi \in \Pi} \mathsf{Q}_{d-1}(n_\pi, \Delta_1, s_1) + r^c, \end{aligned} \tag{1}$$

where the constants C and c depend on d, $\sum_\pi n_\pi \leq n$, $D = O(r^{1/d})$, and $|\Pi|, \Delta_1 s_1 \leq (Ds)^{a_d}$ with $a_d = O(d^4)$ (these are rather crude estimates, but we prefer simplicity). The leading term of the recurrence relies on the crossing-number bound given in Lemma IV.3. In order to apply that lemma, we need that $r \geq \Delta^d$, which will be ensured when choosing r. The second term corresponds to querying the auxiliary data structures for the exceptional set P^*, and the last term corresponds to the time spent in computing the cells of the polynomial partition crossed by the query range γ and for computing the projections γ_π for every $\pi \in \Pi$; here we assume that the choice of r will be such that $r \geq (\Delta s)^d$.

Ultimately, we want to derive that if Δ, s are constants, the recurrence (1) implies

$$\mathsf{Q}_d(n, \Delta, s) \leq n^{1-1/d} \log^{B(d,\Delta,s)} n, \tag{2}$$

where $B(d, \Delta, s)$ is a constant depending on d, Δ, s.

However, as was already mentioned, even if Δ, s are constants initially, they are chosen as tiny powers of n in the

recursion for the exceptional parts, and this makes it hard to obtain a direct inductive proof of (2). Instead, we proceed in two stages. First we derive a weaker bound for $\mathsf{Q}_d(n, \Delta, s)$ without assuming Δ, s to be constants. Namely, we prove that for every constant $\nu > 0$ there exists a constant $A_{d,\nu}$ such that, with a suitable choice of r and n_0,

$$\mathsf{Q}_d(n, \Delta, s) \leq (\Delta s)^{A_{d,\nu}} n^{1-1/d+\nu} \tag{3}$$

for all d, n, Δ, s (with $\Delta s \geq 2$, say). We can assume that $\nu \leq 1/d$ because otherwise the query time is trivially $O(n)$. We choose $r = (2C\Delta s)^{1/\nu}$, which ensures that $r \geq (\Delta s)^d$.

Next, we derive the stronger bound (2) for constant values of Δ, s by using this weaker bound for the $(d-1)$-dimensional queries on the projected exceptional parts, i.e., for the second term in the recurrence (1). In this stage, we choose $r = n^\delta$ for a sufficiently small constant $\delta > 0$. Our choice of δ and $A_{d,\nu}$ implies that $B = d^{O(d)}$. Additional details can be found in the full version [3].

This concludes the proof of Theorem I.4.

VIII. Open Problems

We conclude this paper by mentioning a few open problems: A very interesting and challenging problem is, in our opinion, the fast-query case of range searching with constant-complexity semialgebraic sets, where the goal is to answer a query in $O(\log n)$ time using roughly n^d space.

The range-searching data structure for arbitrary point sets—the one with large fan-out—is so complex and has very high exponent in the polylogarithmic factor because we have difficulty with handling highly degenerate point sets, where many points lie on low-degree algebraic surfaces. It would be nice to find a construction of suitable "multilevel polynomial partitions", as touched upon in [15], [34].

Another open problem, related to the construction of polynomial partitions, is the fast evaluation of a multivariate polynomial at many points, as briefly discussed at the end of Section III.

References

[1] P. K. Agarwal and J. Erickson, Geometric range searching and its relatives, in: *Advances in Discrete and Computational Geometry* (B. Chazelle, J. E. Goodman and R. Pollack, eds.), AMS Press, Providence, RI, 1998, pp. 1–56.

[2] P. K. Agarwal and J. Matoušek, On range searching with semialgebraic sets, *Discrete Comput. Geom.* 11 (1994), 393–418.

[3] P. K. Agarwal, J. Matoušek, M. Sharir, On range searching with semialgebraic sets II, in arXiv.

[4] S. Barone and S. Basu, Refined bounds on the number of connected components of sign conditions on a variety, *Discrete Comput. Geom.* 47 (2012), 577–597.

[5] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*, Algorithms and Computation in Mathematics 10, Springer-Verlag, Berlin, 2003.

[6] T. M. Chan, Optimal partition trees, *Discrete Comput. Geom.* 47 (2012), 661–690.

[7] B. Chazelle, Lower bounds on the complexity of polytope range searching, *J. Amer. Math. Soc.* 2 (1989), 637–666.

[8] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir, A singly exponential stratification scheme for real semi-algebraic varieties and its applications, *Theoret. Comput. Sci.*, 84 (1991), 77–105.

[9] D. Cox, J. Little and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer Verlag, Heidelberg, 1997.

[10] Gy. Elekes, H. Kaplan and M. Sharir, On lines, joints, and incidences in three dimensions, *J. Combinat. Theory, Ser. A* 118 (2011), 962–977.

[11] L. Guth and N. H. Katz, Algebraic methods in discrete analogs of the Kakeya problem, *Advances Math.* 225 (2010), 2828–2839.

[12] L. Guth and N. H. Katz, On the Erdős distinct distances problem in the plane, in arXiv:1011.4105.

[13] H. Hironaka, Triangulations of algebraic sets, in *Proceedings of Symposia Pure Math* (Robin Hartshorne, ed.), Vol. 29, Amer. Math. Soc., Providence, R.I., 1975, 165-185.

[14] H. Kaplan, J. Matoušek and M. Sharir, Simple proofs of classical theorems in discrete geometry via the Guth–Katz polynomial partitioning technique, *Discrete Comput. Geom.* 48 (2012), 499–517.

[15] H. Kaplan, J. Matoušek, Z. Safernová and M. Sharir, Unit distances in three dimensions, *Combinat. Probab. Comput.*, in press. Also in arXiv:1107.1077.

[16] K. Kedlaya and Ch. Umans, Fast modular composition in any characteristic, *Proc. 49th Annu. IEEE Sympos. Foundat. Comput. Sci.* (2008), 146–155.

[17] Ch. Knauer, H. R. Tiwari and D. Werner, On the computational complexity of Ham-Sandwich cuts, Helly sets, and related problems, *Proc. 28th Annu. Sympos. Theoret. Aspects Comput. Sci.* (2011), 649–660.

[18] V. Koltun, Almost tight upper bounds for vertical decompositions in four dimensions, *J. ACM* 51(5) (2004), 699–730.

[19] K. G. Larsen, On range searching in the group model and combinatorial discrepancy, *Proc. 52nd Annu. IEEE Sympos. Found. Comp. Sci.*, 2011, 542–549.

[20] C.-Y. Lo, J. Matoušek and W. Steiger, Ham-sandwich cuts in $\mathbb{R}^d$, *Proc. 24th Annu. ACM Sympos. Theory Comput. Sci.* (1992), 539–545.

[21] J. Matoušek, Efficient partition trees, *Discrete Comput. Geom.* 8 (1992), 315–334.

[22] J. Matoušek, Range searching with efficient hierarchical cuttings, *Discrete Comput. Geom.* 10 (1993), 157–182.

[23] J. Matoušek, Geometric range searching, *ACM Comput. Surv.* 26(4) (1994), 421–461.

[24] J. Matoušek, *Using the Borsuk-Ulam Theorem*, Lectures on Topological Methods in Combinatorics and Geometry Series, Springer Verlag, Heidelberg, 2003.

[25] M. Nüsken and M. Ziegler, Fast multipoint evaluation of bivariate polynomials, *Proc. 12th Annu. European Sympos. Algorithms* (2004), 544–555.

[26] J. T. Schwartz and M. Sharir, On the Piano Movers' problem: II. General techniques for computing topological properties of real algebraic manifolds, *Advances Appl. Math.* 4 (1983), 298–351.

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

[28] M. Sharir and H. Shaul, Semi-algebraic range reporting and emptiness searching with applications, *SIAM J. Comput.* 40 (2011), 1045–1074.

[29] J. Solymosi and T. Tao, An incidence theorem in higher dimensions, *Discrete Comput. Geom.* 48 (2012), 255–280.

[30] A. H. Stone and J. W. Tukey, Generalized sandwich theorems, *Duke Math. J.* 9 (1942), 356–359.

[31] H. E. Warren, Lower bound for approximation by nonlinear manifolds, *Trans. Amer. Math. Soc.* 133 (1968), 167–178.

[32] A. C. Yao and F. F. Yao, A general approach to d-dimensional geometric queries, *Proc. 17th Annu. ACM Sympos. Theory Comput.*, 163–168.

[33] C. K. Yap, A geometric consistency theorem for a symbolic perturbation scheme, *J. Comput. Syst. Sci.* 40 (1990), 2–18.

[34] J. Zahl, An improved bound on the number of point-surface incidences in three dimensions, in arXiv:1104.4987.