Further Results on Colored Range Searching

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
tmc@illinois.edu

Qizheng He
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
qizheng6@illinois.edu

Yakov Nekrich
Department of Computer Science, Michigan Technological University, Houghton, MI, USA
yakov.nekrich@googlemail.com

Abstract

We present a number of new results about range searching for colored (or "categorical") data:

1. For a set of n colored points in three dimensions, we describe randomized data structures with $O(n \operatorname{polylog} n)$ space that can report the distinct colors in any query orthogonal range (axis-aligned box) in $O(k \operatorname{polyloglog} n)$ expected time, where k is the number of distinct colors in the range, assuming that coordinates are in $\{1, \ldots, n\}$. Previous data structures require $O(\frac{\log n}{\log\log n} + k)$ query time. Our result also implies improvements in higher constant dimensions.
2. Our data structures can be adapted to halfspace ranges in three dimensions (or circular ranges in two dimensions), achieving $O(k \log n)$ expected query time. Previous data structures require $O(k \log^2 n)$ query time.
3. For a set of n colored points in two dimensions, we describe a data structure with $O(n \operatorname{polylog} n)$ space that can answer colored "type-2" range counting queries: report the number of occurrences of every distinct color in a query orthogonal range. The query time is $O(\frac{\log n}{\log\log n} + k \log\log n)$, where k is the number of distinct colors in the range. Naively performing k uncolored range counting queries would require $O(k \frac{\log n}{\log\log n})$ time.

Our data structures are designed using a variety of techniques, including colored variants of randomized incremental construction (which may be of independent interest), colored variants of shallow cuttings, and bit-packing tricks.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Range searching, geometric data structures, randomized incremental construction, random sampling, word RAM

Digital Object Identifier 10.4230/LIPIcs.SoCG.2020.28

Related Version A full version of this paper is available at `http://arxiv.org/abs/2003.11604`.

Funding *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

1 Introduction

Colored range searching (also known as "categorical range searching", or "generalized range searching") have been extensively studied in computational geometry since the 1990s. For example, see the papers [5, 11, 18, 19, 20, 21, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34, 36, 37, 38, 40, 46] and the survey by Gupta et al. [22]. Given a set of n colored data points (where the color of a point represents its "category"), the objective is to build data structures that can provide statistics or some kind of summary about the colors of the points inside a query range. The most basic types of queries include:

- *colored range reporting*: report all the distinct colors in the query range.
- *colored "type-1" range counting*: find the number of distinct colors in the query range.
- *colored "type-2" range counting*: report the number of points of color χ in the query range, for *every* color χ in the range.

In this paper, we focus on colored range reporting and type-2 colored range counting. Note that the output size in both instances is equal to the number k of distinct colors in the range, and we aim for query time bounds that depend linearly on k, of the form $O(f(n) + kg(n))$. Naively using an uncolored range reporting data structure and looping through all points in the range would be too costly, since the number of points in the range can be significantly larger than k.

1.1 Colored orthogonal range reporting

The most basic version of the problem is perhaps *colored orthogonal range reporting*: report the k distinct colors inside an orthogonal range (an axis-aligned box). It is not difficult to obtain an $O(n \operatorname{polylog} n)$-space data structure with $O(k \operatorname{polylog} n)$ query time [22] for any constant dimension d: one approach is to directly modify the d-dimensional range tree [15, 43], and another approach is to reduce colored range reporting to uncolored range emptiness [26] (by building a one-dimensional range tree over the colors and storing a range emptiness structure at each node). Both approaches require $O(k \operatorname{polylog} n)$ query time rather than $O(\operatorname{polylog} n + k)$ as in traditional (uncolored) orthogonal range searching: the reason is that in the first approach, each color may be discovered polylogarithmically many times, whereas in the second approach, each discovered color costs us $O(\log n)$ range emptiness queries, each of which requires polylogarithmic time.

Even the 2D case remains open, if one is interested in optimizing logarithmic factors. For example, Larsen and van Walderveen [32] and Nekrich [37] independently presented data structures with $O(n \log n)$ space and $O(\log\log U + k)$ query time in the standard word-RAM model, assuming that coordinates are integers bounded by U. The query bound is optimal, but the space bound is not. Recently, Chan and Nekrich [11] have improved the space bound to $O(n \log^{3/4+\varepsilon} n)$ for an arbitrarily small constant $\varepsilon > 0$, while keeping $O(\log\log U + k)$ query time.

In 3D, the best result to date is by Chan and Nekrich [11], who obtained a data structure with $O(n \log^{9/5+\varepsilon} n)$ space and $O(\frac{\log n}{\log\log n} + k)$ query time. The first step is a data structure for the case of 3D dominance (i.e., 3-sided) ranges: as they noted, this case can be solved in $O(n)$ space and $O(\frac{\log n}{\log\log n} + k)$ time by a known reduction [44, Section 3.1] to 3D 5-sided box stabbing [12]. For 3D 5-sided box stabbing (or more simply, 2D 4-sided rectangle stabbing), a matching lower bound of $\Omega(\frac{\log n}{\log\log n} + k)$ is known for $O(n \operatorname{polylog} n)$-space structures, due to Pătraşcu [42]. A natural question then arises: is $O(\frac{\log n}{\log\log n} + k)$ query time also tight for 3D colored dominance range reporting?

We show that the answer is no – the $O(\frac{\log n}{\log\log n})$ term can in fact be improved when k is small. Specifically, we present a randomized data structure for 3D colored dominance range reporting with $O(n \log n)$ space and $O(\log\log U + k \log\log n)$ expected time in the standard word-RAM model. (We use only Las Vegas randomization, i.e., the query algorithm is always correct; an oblivious adversary is assumed, i.e., the query range should be independent of the random choices made by the preprocessing algorithm.) Combining with Chan and Nekrich's method [11], we can then obtain a data structure for 3D colored orthogonal range reporting with $O(n \log^{2+\varepsilon} n)$ space and $O(\log\log U + k \log\log n)$ expected query time.

An improved solution in 3D automatically implies improvements in any constant dimension $d > 3$, by using standard range trees [15, 43] to reduce the dimension, at a cost of about one logarithmic factor (ignoring $\log\log$ factors) per dimension. This way, we obtain a data structure in d dimensions with $O(n \log^{d-1+\varepsilon} n)$ space and $O(k(\frac{\log n}{\log\log n})^{d-3} \log\log n)$ query time.[1] (Note that $O((\frac{\log n}{\log\log n})^{d-3} \log\log n)$ is the current best query time bound for standard (uncolored) range emptiness [9] for $O(n \operatorname{polylog} n)$-space structures on the word RAM.)

1.2 Colored 3D halfspace range reporting

An equally fundamental problem is *colored halfspace range reporting*. In 2D, an $O(n)$-space data structure with $O(\log n + k)$ query time is known [2, 22]. In 3D, the current best result is obtained by applying a general reduction of colored range reporting to uncolored range emptiness [26], which yields $O(n \log n)$ space and $O(k \log^2 n)$ query time [22]. (An alternative solution with $O(n)$ space and $O(n^{2/3+\varepsilon} + k)$ time is also known, by reduction to simplex range searching.) The 3D case is especially important, as 2D colored circular range reporting reduces to 3D colored halfspace range reporting by the standard lifting transformation.

We describe a randomized data structure with $O(n \log n)$ space and $O(k \log n)$ expected query time for 3D halfspace ranges (and thus 2D circular ranges). This is a logarithmic-factor improvement over the previous query time bound.

1.3 Colored 2D orthogonal type-2 range counting

Finally, we consider colored orthogonal "type-2" range counting: compute the number of occurrences of every color in a given orthogonal range. Despite the nondescript name, colored type-2 counting is quite natural, providing more information than colored reporting, as we are generating an entire histogram. The problem was introduced by Gupta et al. [23] (and more recently revisited by Ganguly et al. [20] in external memory). An old paper by Bozanis et al. [5] gave a solution in the 1D case with $O(n)$ space and $O(\log n + k)$ query time, which implies a solution in 2D with $O(n \log n)$ space and $O(\log^2 n + k \log n)$ query time. Alternatively, to answer a colored type-2 counting query, we can first answer a colored range reporting query, followed by k standard (uncolored) range counting queries, if we store each color class in a standard range counting data structure; by known results on colored range reporting [11] and standard range counting [27], this then yields $O(n \log^{3/4+\varepsilon} n)$ space and $O(k \frac{\log n}{\log\log n})$ query time. Thus, in some sense, a type-2 counting query corresponds to "simultaneous" range counting queries on multiple point sets.[2]

We present a data structure for the problem in 2D with $O(n \log^{1+\varepsilon} n)$ space and $O(\frac{\log n}{\log\log n} + k \log\log n)$ query time in the standard word-RAM model. As 2D standard (uncolored) range counting has an $\Omega(\frac{\log n}{\log\log n})$ time lower bound for $O(n \operatorname{polylog} n)$-space structures [41], our result shows, surprisingly, that answering multiple range counting queries "simultaneously" are cheaper than answering one by one – we only have to pay $O(\log\log n)$ cost per color!

1.4 Techniques

Our solutions for colored 3D dominance range reporting and 3D halfspace range reporting are based on similar ideas. We in fact propose two different methods.

[1] In all reported bounds, we implicitly assume $k > 0$. The $k = 0$ case can be handled by answering one initial uncolored range emptiness query.

[2] See [1] for a different notion of "concurrent" range reporting.

In the first method (Section 2), we solve the $k = 1$ case (testing whether a range contains only one color) by introducing a colored variant of *randomized incremental construction*; we then extend the solution to the general case by a randomized one-dimensional range tree over the colors. Along the way, we prove a combinatorial lemma which may be of independent interest: in a colored point set in 3D, if we randomly permute the color classes and randomly permute the points within each color classes, and if we insert the points in the resulting order, then the convex hull undergoes $O(n \log n)$ structural changes in expectation. (It is a well known fact, by Clarkson and Shor [14], that in the uncolored setting, if we insert points in a random order, the 3D convex hull undergoes $O(n)$ structural changes in expectation.)

In the second method (Section 3), which is slightly more efficient, we solve the $k = 1$ case differently, by adapting known techniques for uncolored 3D halfspace range reporting [6] based on *random sampling* (namely, conflict lists of lower envelopes of random subsets). The approach guarantees only $\Omega(1)$ success probability per query (in the uncolored setting, *shallow cuttings* can fix the problem, but they do not seem easily generalizable to the 3D colored setting). Fortunately, we show that a solution for the $k = 1$ case with constant success probability is sufficient to complete the solution for the general case.

Our method for colored 2D type-2 orthogonal range counting (Section 5) is technically the most involved. It is obtained by a nontrivial combination of several techniques, including the *recursive grid* approach of Alstrup, Brodal, and Rauhe [3], bit packing tricks, and 2D shallow cuttings. Our work demonstrates yet again the power of the recursive grid approach (see [9, 11, 12] for other recent examples).

2 Colored 3D Halfspace Range Reporting: First Method

In this and the next section, we describe our two methods for 3D halfspace ranges. The case of 3D dominance ranges is similar and will be addressed later in Section 4.

2.1 Combinatorial lemmas on colored randomized incremental construction

Our first method relies on a simple combinatorial lemma related to a colored version of randomized incremental construction of 3D convex hulls (the uncolored version of the lemma, where all points are assigned different colors, is well known in computational geometry, from the seminal work by Clarkson and Shor [14]):

▶ **Lemma 1.** *Given a set S of n colored points in $\mathbb{R}^3$, if we first randomly permute the color classes, then for each color according to this order we simultaneously insert all points with that color, then the expected total number of structural changes to the convex hull is $O(n)$.*

Proof. Consider a random permutation of the colors. Let C_i be the i-th color class, i.e., the set of all points with the i-th color in the permutation. Let m be the number of color classes. Let $V_i = \bigcup_{j=1}^{i} C_j$ contain all points with the first i colors. Let $\mathrm{CH}(V_i)$ denote the convex hull of V_i. Let Δ_i^+ be the set of all facets in $\mathrm{CH}(V_i)$ that are not in $\mathrm{CH}(V_{i-1})$, i.e., all hull facets created when we insert the i-th color class C_i.

For each i, we have $\mathbb{E}[|C_i|] = \frac{n}{m}$ and $\mathbb{E}[|V_i|] = \frac{in}{m}$. We use backwards analysis [45]. Observe that $|\Delta_i^+|$ is bounded by the total degree of all points of C_i in $\mathrm{CH}(V_i)$. The total degree over all points in $\mathrm{CH}(V_i)$ is $O(|V_i|)$. Conditioned on a fixed V_i, we have $\mathbb{E}[|\Delta_i^+|] = O(\frac{|V_i|}{i})$. So, unconditionally, $\mathbb{E}[|\Delta_i^+|] = O(\frac{n}{m})$. Therefore, the expected total number of hull facets created is $\mathbb{E}[\sum_{i=1}^{m} |\Delta_i^+|] = O(n)$. ◀

The following refinement of the lemma further bounds the total amount of changes to the convex hull when we additionally insert points one by one in a random order within each color class. The proof is slightly trickier. (The first lemma is already sufficient to bound the space of our new data structure, but the refined lemma will be useful in bounding the preprocessing time.)

▶ **Lemma 2.** *Given a set S of n colored points in $\mathbb{R}^3$, if we first randomly permute the color classes, then randomly permute the points in each color class, and insert the points one by one according to this order, then the expected total number of structural changes of the convex hull is $O(n \log n)$.*

Proof. Continuing the earlier proof, let Δ_i^- be the set of all facets in $\mathrm{CH}(V_{i-1})$ that are not in $\mathrm{CH}(V_i)$, i.e., all hull facets destroyed when we insert the i-th color class C_i. Since the total number of facets destroyed is at most the total number of facets created, $\mathbb{E}[\sum_{i=1}^m |\Delta_i^-|] \le \mathbb{E}[\sum_{i=1}^m |\Delta_i^+|] = O(n)$.

Now, consider a random permutation of the points in C_i. Let $V_{i,j}$ contain all points in V_{i-1} and also the first j points of C_i. Let $G_{i,j}$ be the subgraph formed by all edges of $\mathrm{CH}(V_{i,j})$ that are incident to the vertices of C_i. Then every vertex v in $G_{i,j}$ is either in C_i or is incident to a facet of Δ_i^- (because if $v \notin C_i$, then v must be a vertex of $\mathrm{CH}(V_{i-1})$, and at least one of its incident facets in $\mathrm{CH}(V_{i-1})$ will be destroyed when C_i is inserted). Thus, $G_{i,j}$ has $O(|C_i| + |\Delta_i^-|)$ vertices, and since $G_{i,j}$ is a planar graph, it has $O(|C_i| + |\Delta_i^-|)$ edges.

Let $\Delta_{i,j}^+$ be the set of all facets in $\mathrm{CH}(V_{i,j})$ that are not in $\mathrm{CH}(V_{i,j-1})$, i.e., all hull facets created when we insert the j-th point in C_i. We use backwards analysis again. Observe that $|\Delta_{i,j}^+|$ is bounded by the degree of the j-th point in C_i in $\mathrm{CH}(V_{i,j})$. The total degree over all points of C_i in $\mathrm{CH}(V_{i,j})$ is at most twice the number of edges in $G_{i,j}$. Conditioned on a fixed C_i and a fixed $V_{i,j}$, we thus have $\mathbb{E}[|\Delta_{i,j}^+|] = O\left(\frac{|C_i|+|\Delta_i^-|}{j}\right)$. As the right-hand side does not depend on the local permutation of the color class C_i, the expectation holds conditioned only on the global permutation of the colors. Unconditionally, the expected total number of hull facets created is

$$O\left(\mathbb{E}\left[\sum_{i=1}^{m}\sum_{j=1}^{|C_i|}\frac{|C_i|+|\Delta_i^-|}{j}\right]\right) = O\left(\mathbb{E}\left[\sum_{i=1}^{m}(|C_i|+|\Delta_i^-|)\log n\right]\right) = O(n\log n). \qquad ◀$$

Remarks.

1. The $O(n \log n)$ bound in the refined lemma is tight: Consider $\frac{n}{2}$ points lying on the xy-plane in convex position, each assigned a different color. In addition, add $\frac{n}{2}$ points on the z-axis above the xy-plane, all with a common color χ_0. When we insert the color class for χ_0, there are already $\Omega(n)$ points on the xy-plane with probability $\Omega(1)$. In an iteration where the next point we insert with color χ_0 has larger z-coordinate than all previous points, the insertion would create $\Omega(n)$ new hull edges in expectation. By a well known analysis, the expected number of such iterations is given by the Harmonic number, which is $\Theta(\log n)$. This shows an $\Omega(n \log n)$ lower bound.
2. The same argument holds for other geometric structures besides 3D convex hulls, e.g., Voronoi diagrams of 2D points and trapezoidal decompositions of 2D disjoint line segments.
3. We can generalize the refined lemma to the setting when we have a hierarchy of color classes with ℓ levels, and we randomly permute the child subclasses of each color class. (The refined lemma corresponds to the $\ell = 2$ case.) The bound becomes $O(n \log^{\ell-1} n)$. This result seems potentially relevant to implementing randomized incremental constructions in a hierarchical external-memory model.

2.2 The $k = 1$ case

We now reveal how colored randomized incremental construction can help solve the colored range reporting problem. We start with the case $k = 1$, i.e., we want to test whether there is only one color in the query range. By an uncolored range search, we can find one point in the range (in $O(\log n)$ time for 3D halfspace ranges) and identify its color χ. Thus, the problem is to verify that all points in the range have the same color χ.

Fix a total ordering of the colors. It is easy to see that the problem reduces to two subproblems: for a given query color χ, (i) decide whether there exists a point in the range with color $< \chi$, and (ii) decide whether there exists a point in the range with color $> \chi$. By symmetry, it suffices to solve subproblem (i). To this end, we imagine inserting the points in increasing order of color, and maintaining a data structure for (uncolored) range emptiness for the points. We can make this semi-dynamic data structure (which supports insertions only) *persistent*. Then we can solve subproblem (i) by querying a past version of the range emptiness data structure, right after all points with color $< \chi$ were inserted.

In the case of 3D upper halfspaces (lower halfspaces can be handled symmetrically), a range emptiness query reduces to finding an extreme point on the upper hull along a query direction, or equivalently, intersecting the lower envelope of the dual planes at a query vertical line. By projection, this reduces to a planar point location query, answerable in $O(\log n)$ time by a linear-space data structure [15, 43]. However, we need a data structure that supports insertions, and in general this increases the query time (by an extra logarithmic factor via the standard "logarithmic method" [4]).

The key is to observe that the above approach works regardless of which total ordering of the colors we use. Our idea is simply to use a *random ordering* of the colors! (For (ii), note that the reverse of a uniformly random ordering is still uniformly random.) By Lemma 1, the upper hull undergoes $O(n)$ expected number of structural changes. So is the dual lower envelope. We can then apply a known dynamic planar point location method; for example, the method by Chan and Nekrich [10] achieves $O(\log n(\log\log n)^2)$ query time and $O(\log n \log\log n)$ amortized update time per change to the envelope. The data structure can be made persistent, for example, by applying Dietz's technique [17], with a $\log\log n$ factor penalty (the space usage is related to the total update time). The final data structure supports queries in $O(\log n(\log\log n)^3)$ (worst-case) time and uses $O(n\log n(\log\log n)^2)$ expected space. (Note that the space bound can be made worst-case, by repeating $O(1)$ expected number of times until a "good" ordering is found.)

Remark on preprocessing time. It isn't obvious how to efficiently insert an entire color class to the 3D convex hull, even knowing that the total number of structural changes is small. To get good preprocessing time, we propose inserting points one by one within each color class, since Lemma 2 ensures that the number of changes to the convex hull is still near linear ($O(n\log n)$). Several implementation options can then yield $O(n \operatorname{polylog} n)$ expected preprocessing time: (i) we can use a general-purpose dynamic convex hull data structure [7] (in the insertion-only case, the cost per update is $O(f\log^2 n)$ where f is the amount of structural changes); (ii) we can adapt standard randomized incremental algorithms, e.g., handling the point location steps by using history DAGs [35] (this requires further randomized analysis); or (iii) we can adapt standard randomized incremental algorithms, but handling the point location steps by using a known dynamic planar point location method [10].

▶ **Theorem 3.** *For n colored points in $\mathbb{R}^3$, there is a data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n\log n(\log\log n)^2)$ space that can test whether the number of colors in a query halfspace is exactly 1 in $O(\log n(\log\log n)^3)$ time.*

2.3 The general case

Previous papers [22, 26] (see also [13] in the uncolored case) have noted a straightforward black-box reduction of colored range reporting to the $k = 0$ case (range emptiness), essentially by using a one-dimensional range tree over the colors: More precisely, we split the color classes into two halves. We build a data structure for $k = 0$, and recursively build a data structure for the two halves. Space usage increases by a logarithmic factor. If the $k = 0$ structure has $Q_0(n)$ query time, the overall query time is $O(kQ_0(n) \log n)$, since at each of the $O(\log n)$ levels of recursion tree, $O(k)$ nodes are examined.

We present a new black-box reduction of colored range reporting to the $k \leq 1$ case, which saves a logarithmic factor, by using a similar idea but with randomization.

▶ **Theorem 4.** *Suppose that for n colored points, there is a data structure with $P(n)$ (expected) preprocessing time and $S(n)$ space that can decide whether the number of colors in a query range is exactly 1 in $Q_1(n)$ time. In addition, the data structure can decide whether the range is empty, and if not, report one point, in $Q_0(n)$ time. Then there is a randomized Las Vegas data structure with $O(P(n) \log n)$ expected preprocessing time and $O(S(n) \log n)$ space that can report all k distinct colors in a query range in $O(k(Q_0(n) + Q_1(n)))$ expected time, assuming that $P(n)/n$ and $S(n)/n$ are nondecreasing.*

Proof. We split the color classes into two parts, where each color is *randomly* assigned to one of the two parts. We build the given $k = 1$ structure and range emptiness structure, and recursively build a data structure for the two parts. Space usage increases by a logarithmic factor (with high probability).

To answer a query, we test whether the range is empty or whether $k = 1$. If so, we are done. Otherwise, we recursively query both parts.

Consider a query range that is independent of the random choices made by the data structure. At the i-th level of the recursion tree, how many nodes are examined (in expectation)? This question is analogous to the following: place k balls randomly (independently) into 2^i bins; how many bins contain two or more balls? The number is upper-bounded by the number of pairs of balls that are in the same bin. Since the probability that a fixed pair of balls are placed in the same bin is $1/2^i$, the expected number of pairs is at most $k^2/2^i$.

Thus, the expected number of nodes examined at the i-th level is at most $\min\{2^i, k^2/2^i\}$. The overall expected number of nodes examined is

$$O\left(\sum_i \min\{2^i, k^2/2^i\}\right) = O\left(\sum_{i:\, 2^i \leq k} 2^i + \sum_{i:\, 2^i > k} k^2/2^i\right) = O(k). \qquad ◀$$

Combining Theorems 3 and 4 yields:

▶ **Theorem 5.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log^2 n (\log\log n)^2)$ space that can report all k distinct colors in a query halfspace in $O(k \log n (\log\log n)^3)$ expected time.*

3 Colored 3D Halfspace Range Reporting: Second Method

We next describe a slightly better (and simpler) method for colored 3D halfspace range reporting.

3.1 The $k = 1$ case

The idea is to relax the $k = 1$ subproblem and allow the query algorithm to occasionally be wrong (since we will be using randomization anyways for the general case). The algorithm has constant error probability and can only make one-sided errors: if it returns "yes", we must have $k = 1$. We work in dual space: given a set of colored planes in $\mathbb{R}^3$, we want to decide whether the number of colors among the planes below a query point is exactly 1.

Preprocessing. Take a random sample R of the planes, where each color class is included independently with probability $\frac{1}{2}$. Take the lower envelope $\mathrm{LE}(R)$ of R, and consider the vertical decomposition $\mathrm{VD}(R)$ of the region underneath $\mathrm{LE}(R)$. (The vertical decomposition is defined as follows: we triangulate each face of $\mathrm{LE}(R)$ by joining each vertex to the bottom vertex of the face; for each triangle, we form the unbounded prism containing all points underneath the triangle.) For each cell $\Delta \in \mathrm{VD}(R)$, let L_Δ denote the set of distinct colors among all planes intersecting Δ (the "color conflict list" of Δ). We store the list L_Δ if $|L_\Delta| \leq c$ for a sufficiently large constant c; otherwise, we mark Δ as "bad".

Clearly, the space usage is $O(n)$, since there are $O(n)$ cells in $\mathrm{VD}(R)$ and each list stored has constant size. To bound the preprocessing time, we can generate (up to c elements of) each list L_Δ by answering colored range reporting queries at the three vertices of Δ, since a plane intersects Δ iff it is below at least one of the vertices of Δ. By previous results, these $O(n)$ colored range reporting queries take $O(n \operatorname{polylog} n)$ time.

In addition, for each color class, we store an (uncolored) range emptiness structure (i.e., a planar point location structure for the xy-projection of the lower envelope of the color class). This takes $O(n)$ space in total.

Querying. Given a query point q, we find the cell $\Delta(q)$ of $\mathrm{VD}(R)$ containing q in $O(\log n)$ time by planar point location (on the xy-projection of $\mathrm{VD}(R)$). If the cell does not exist (i.e., q lies above $\mathrm{LE}(R)$), or if the cell is bad, we return "no". Otherwise, for each of the at most c colors in the conflict list $L_{\Delta(q)}$, we test whether any plane below q has that color by querying the corresponding range emptiness structure in $O(\log n)$ time. We return "yes" iff exactly one color passes the test. The overall query time is $O(\log n)$.

The algorithm is clearly correct if it returns "yes". Consider a fixed query point q, such that there is just one color χ among all planes below q. The algorithm would erroneously return "no" in two scenarios: (i) when q lies above $\mathrm{LE}(R)$, or (ii) when $|L_{\Delta(q)}| > c$. The probability of (i) is the probability that the color χ is chosen in the random sample R, which is $\frac{1}{2}$. By the following lemma, and Markov's inequality, the probability of (ii) is at most 0.1 (say) for a sufficiently large constant c. This lemma directly follows from Clarkson and Shor's technique [14] (see the full paper for a quick proof).

▶ **Lemma 6.** *For a fixed point q, we have $\mathbb{E}[|L_{\Delta(q)}|] = O(1)$.*

We conclude:

▶ **Theorem 7.** *For n colored points in $\mathbb{R}^3$, there is a randomized Monte Carlo data structure with $O(n \operatorname{polylog} n)$ preprocessing time and $O(n)$ space that decides whether the number of colors in a query halfspace is exactly 1 in $O(\log n)$ time; if the actual answer is true, the algorithm returns "yes" with probability $\Omega(1)$, else it always returns "no".*

Remarks. The method can be viewed as a variant of Chan's random-sampling-based method for uncolored 3D halfspace range reporting [6]. In the uncolored setting, errors can be completely avoided by replacing lower envelopes of samples with *shallow cuttings* [33], but it is unclear how to do so in the colored setting.

3.2 The general case

Finally, to solve the general problem, we use a variant of Theorem 4 that tolerates one-sided errors in the given $k = 1$ data structure.

▶ **Theorem 8.** *Suppose that for n colored points, there is a randomized Monte Carlo data structure with $P(n)$ (expected) preprocessing time and $S(n)$ space that decides whether the number of colors in a query range is exactly 1 in $Q_1(n)$ time; if the actual answer is true, the algorithm returns "yes" with probability $\Omega(1)$, else it always returns "no". In addition, the data structure can decide whether the range is empty, and if not, report one point, in $Q_0(n)$ time (without errors). Then there is a randomized Las Vegas data structure with $O(P(n)\log n)$ expected preprocessing time and $O(S(n)\log n)$ space that can report all k distinct colors in a query range in $O(k(Q_0(n) + Q_1(n)))$ expected time, assuming that $P(n)/n$ and $S(n)/n$ are nondecreasing.*

Proof. We use the same approach as in the proof of Theorem 4. In the query algorithm, if the range is empty or the $k = 1$ structure returns "yes", we are done; otherwise, we recursively query both parts.

To analyze the query time, we say that a node in the recursion tree is *bad* if the number of colors in the query range at the node is exactly 1. Our earlier analysis shows that the expected total number of non-bad nodes visited is $O(k)$. However, because of the possibility of one-sided errors, the query algorithm may examine some bad nodes. For each bad node v visited by the query algorithm, we charge v to its lowest ancestor u that is not bad. Then for a fixed node u, we may have up to two paths of nodes charged to u. The expected number of nodes charged to a fixed node u is at most $O(\sum_i (1 - \Omega(1))^i) = O(1)$. We conclude that the expected total number of nodes visited is $O(k)$. ◀

Combining Theorems 7 and 8 yields:

▶ **Theorem 9.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log n)$ space that can report all k distinct colors in a query halfspace in $O(k \log n)$ expected time.*

4 Colored 3D Dominance Range Reporting

Both methods can be adapted to solve the colored 3D dominance range reporting problem: here, we want to report the distinct colors of all points inside a 3-sided range of the form $(-\infty, q_1] \times (-\infty, q_2] \times (-\infty, q_3]$. Equivalently, we can map input points (p_1, p_2, p_3) to orthants $[p_1, \infty) \times [p_2, \infty) \times [p_3, \infty)$, and the problem becomes reporting the distinct colors among all orthants containing a query point $q = (q_1, q_2, q_3)$. By replacing values with their ranks, we may assume that all coordinates are in $\{1, \ldots, n\}$ (in a query, an initial predecessor search to reduce to rank space requires an additional $O(\log\log U)$ cost by van Emde Boas trees). We assume the standard word-RAM model.

In the first method, the combinatorial lemmas on colored randomized incremental constructions can be extended to the union of the orthants (a "staircase polyhedron"). In fact, by a known transformation involving an exponentially spaced grid [9, 39], orthants can be mapped to halfspaces and a union of orthants can be mapped to a halfspace intersection, or in the dual, a 3D convex hull. For the $k = 1$ structure, we not only randomly permute the color classes but also randomly permute the points inside each color class, and maintain the union of the orthants as points are inserted one by one. Instead of using persistence, we reduce to static 3D point location: we insert in reverse order, and as a new orthant is inserted, we create a region for the newly added portion of the union (i.e., the new orthant minus the old union). Identifying the smallest color of the orthants containing q (to solve subproblem (i)) reduces to locating the region containing q. The expected total size of these regions is $O(n \log n)$ by Lemma 2; we can further subdivide each of these regions into boxes (by taking a vertical decomposition), without asymptotically increasing the total size. Known results on orthogonal point location in a 3D subdivision of (space-filling) boxes [16, 12] then give $O((\log \log n)^2)$ query time and space linear in the size of the subdivision. Thus, the final data structure for the general case has $O(n \log^2 n)$ space and $O(\log \log U + k(\log \log n)^2)$ expected query time.

In the second method, we replace lower envelopes with unions of orthants. The only main change is that planar point location queries for orthogonal subdivisions now cost $O(\log \log n)$ time by Chan's result [8] instead of $O(\log n)$. Thus, the final data structure has $O(n \log n)$ space and $O(\log \log U + k \log \log n)$ expected time.

▶ **Theorem 10.** *For n colored points in $\mathbb{R}^3$, there is a randomized Las Vegas data structure with $O(n \operatorname{polylog} n)$ expected preprocessing time and $O(n \log n)$ space that can report all k distinct colors in a query dominance range in $O(\log \log U + k \log \log n)$ expected time.*

We can extend the result of Theorem 10 to orthogonal ranges with more sides or to $d > 3$ dimensions. See the full paper for more details.

5 Colored 2D Orthogonal Type-2 Range Counting

Our solution for orthogonal type-2 range counting is described in stages. First we consider the *capped* variant of type-2 range counting. A capped query returns the correct answer if the number of colors k in the query range does not exceed $\log^3 n$. If $k > \log^3 n$, the answer to the capped query is *NULL*. Capped queries in the case when the query range is bounded on 2 sides are considered in Section 5.1. We extend the solution to 3-sided and 4-sided queries, as well as for the case when the number of colors can be arbitrarily large, in the full paper.

5.1 Capped 2-Sided Queries

With foresight, we will solve the more general weighted version of this problem. Each point in S is also assigned a positive integer *weight*. For a 2-sided query range Q, we want to identify all colors that occur in Q; for each color we report the total weight of all its occurrences in Q.

We will denote by n the total weight of all points in S; we will denote by m the total number of points in S. We prove the following result:

▶ **Lemma 11.** *Let S be the set of m points in $\mathbb{R}^2$ with total weight $n \geq m$. There exists a data structure that uses $O(m(\log \log n)^2)$ words of space and supports 2-sided capped type-2 counting queries in $O(\log n / \log \log n + k \log \log n)$ time.*

Our data structure is based on the recursive grid approach [3]. The set of points is recursively sub-divided into vertical slabs (or columns) and horizontal slabs (or rows).

Data Structure. Let $\tau = \log^3 n_0$ where n_0 is the total weight of all points in the global data set (thus τ remains unchanged on all recursion levels). We divide the set of points into $\sqrt{n/\tau}$ columns so that either the total weight of all points in a column is bounded by $O(\sqrt{n\tau})$ or a column contains only one point. This division can be obtained by scanning the set of points in the left-to-right order. We add points to a column C_i for $i = 1, 2, \ldots$ by repeating the following steps: (1) if the weight of the next point p exceeds $\sqrt{n\tau}$, we increment i, (2) we add p to C_i, and (3) if the total weight of C_i exceeds $\sqrt{n\tau}$, we increment i.

Thus either the total weight of a column exceeds $\sqrt{n\tau}$ or the next column contains a point of weight at least $\sqrt{n\tau}$. Hence the number of columns is $O(\sqrt{n/\tau})$. We also divide the set of points into rows satisfying the same conditions. Let $p_{ij} = (x_i, y_j)$ denote the point where the upper boundary of the j-th row intersects the right boundary of the i-th column. Let $Dom(i,j) = [0, x_i] \times [0, y_j]$, denote the range dominated by p_{ij}. If $Dom(i,j)$ contains at most τ distinct colors, we store the list L_{ij} of colors that occur in $Dom(i,j)$. For every color in L_{ij} we also keep the number of its occurrences in $Dom(i,j)$. If the range $Dom(i,j)$ contains more than τ different colors, we set $L_{ij} = NULL$. Thus L_{ij} provides the answer to a capped type-2 counting query on $[0, x_i] \times [0, y_j]$.

Every row/column of weight at least τ^2 that contains more than one point is recursively divided in the same way as explained above. If the total weight of all points is smaller than τ^2, we can answer a type-2 range counting query in $O(k)$ time.

Slow Queries. A query $[0, a] \times [0, b]$ is answered as follows. We identify the column C_{i+1} containing a and the row R_{j+1} containing b. The query is then divided into the middle part $[0, x_i] \times [0, y_j]$, the upper part $[0, a] \times [y_j, b]$ and the right part $[x_i, a] \times [0, y_j]$. The answer to the middle query is stored in the pre-computed list L_{ij}. The upper query is contained in the row R_{j+1} and the right query is contained in the column C_{i+1}. Hence we can answer the upper and the right query using data structures on R_{j+1} and C_{i+1} respectively. If $L_{ij} = NULL$, we return $NULL$ because the number of colors in the query range exceeds $\log^2 n$; if the answer to a query on C_{i+1} or R_{j+1} is $NULL$, we also return $NULL$. Otherwise, we merge the answers to the three queries. The resulting list L can contain up to three items of the same color because the same color can occur in the left, right, and middle query. Since the items in L are sorted by color, we can scan L and compute the total number of occurrences for each color in time proportional to the length of L.

The total query time is given by the formula $Q(n,k) = O(k) + Q(\sqrt{n\tau}, k_1) + Q(\sqrt{n\tau}, k_2)$ where k is the number of colors in the query range and n is the total weight of all points. We denote by k_1 (resp. k_2) the total number of colors reported by the query on R_{j+1} (resp. C_{i+1}). There are at most 2^i recursive calls at level i of recursion. The total weight of points at recursion level i is bounded by $n^{1/2^i} \log^{3(1-1/2^i)} n$. Hence the number of recursion levels is bounded by $\ell = \log\log n - 2\log\log\log n$ and the total query time is $\sum_{i=1}^{\ell} 2^i \cdot k = O(k \cdot (\log n / \log\log n))$.

Fast Queries. We can significantly speed-up queries using the following approach. We keep colors of all points in a column/row in the rank space. Thus each point column or row on the l-th level of recursion contains $O(n^{1/2^l})$ points. Hence for any list L_{ij} on the l-th recursion level we can keep each color and the number of its occurrences in $Dom(i,j)$ using $O((1/2^l)\log n)$ bits.

As explained above, the query on recursion level l is answered by merging three lists: the list L_{ij} that contains the pre-computed answer to the middle query, the list of colors that occur in the right query, and the list of colors that occur in the upper query. Every list occupies $O(k/2^l)$ words of $\log n$ bits. Hence we can merge these lists in $O(k/2^l)$ time using table look-ups. Hence the total query time is $\sum_{i=1}^{\ell} 2^i \cdot \lceil k/2^i \rceil = O(\log n/\log\log n + k \cdot \log\log n)$.

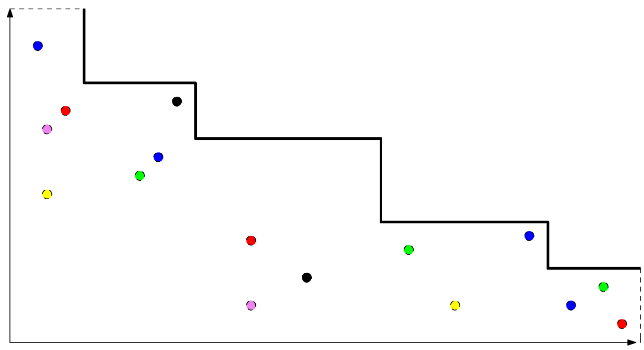

Figure 1 Example of colored t-shallow cutting for $t = 3$.

Color Encoding. In order to merge lists efficiently we must be able to convert the color encoding for the slab V^l into color encoding for the slab V^{l-1} that contains V^l. Moreover the conversion should be performed in $O(k/2^l)$ time, i.e., in sub-constant time per color. For this purpose we introduce the concept of *colored t-shallow cutting* that adapts the concept of shallow cutting to the muti-color scenario. A colored t-shallow cutting for a set of points S is the set of $O(|S|/t)$ cells. Each cell is a rectangle with one corner in the point $(0, 0)$. Each cell contains points of at most $2t$ different colors. If some point q is not contained in any cell of the t-shallow cutting, then q dominates points of at least t different colors.

A colored t-shallow cutting can be constructed using the staircase approach, see e.g., [47]. We start in the point $(0, x_{\max} + 1)$ where $x_{\max}$ is the largest x-coordinate of a point in S. We move p in the $+y$ direction until p dominates $2t$ different colors. Then we move p in the $-x$ direction until p dominates t different colors. We alternatingly move p in $+y$ and $-x$ directions until the x-coordinate of p is 0 or the y-coordinate of p is $y_{\max} + 1$ where $y_{\max}$ is the largest y-coordinate of any point in S. Each point where we stopped moving p in $+y$ direction and started moving p in the $-x$ direction is the upper right corner of some cell. We can show that the number of cell does not exceed $O(|S|/t)$: Let $c_i = (x_i, y_i)$ and $c_{i+1} = (x_{i+1}, y_{i+1})$ denote two consecutive corners (in the left-to-right order) of a t-shallow cutting. Consider all points $p = (x_p, y_p)$ such that $x_i \le x_p \le x_{i+1}$ and $y_p \le y_{i+1}$. By our construction, points p that satisfy these conditions have t different colors. Hence there are at least t such points p and we can assign t unique points to every corner of a colored t-shallow cutting. Hence the number of corners is $O(n/t)$.

For each slab V^l on any recursion level l, we construct colored t-shallow cuttings for $t = 2, 4, \ldots, \tau$. For every cell c_j of each shallow cutting we create the list $\mathrm{clist}(c_j)$ of colors that occur in c_j. Colors in $\mathrm{clist}(c_j)$ are stored in increasing order. For each color we store its rank in V^l and its rank in the slab V^{l-1} that contains V^l. Consider a 2-sided query to a slab V^l on recursion level l. The answer to this query is a sorted list $LIST(q)$ of t colors in the rank space of V^l. If $t < \log^2 n$, then the 2-sided query range is contained in some cell c_j of the colored $2^{\lceil \log t \rceil}$ shallow cutting. Using $\mathrm{clist}(c_j)$ we can convert colors in $LIST(q)$ into the rank space of V^{l-1} where V^{l-1} is the slab that contains V^l. The conversion is based on a universal look-up table and takes $O(t/2^l)$ time.

This result can be extended to 3-sided and 4-sided capped queries using divide-and-conquer on range trees and the recursive grid approach. The space usage of the data structure for capped 4-sided queries is $O(m \log^{\varepsilon} n)$. Finally the range tree on colors enables us to get rid of the restriction on the number of colors, but the space usage of the data structure is increased by $O(\log n)$ factor. The complete description can be found in the full paper.

▶ **Theorem 12.** *Let S be the set of m points in $\mathbb{R}^2$ with total weight $n \geq m$. There exists a data structure that uses $O(m \log m \log^{\varepsilon} n)$ words of space and supports 4-sided type-2 range counting queries in $O(\log n / \log\log n + k \log\log n)$ time.*

References

1 Peyman Afshani, Cheng Sheng, Yufei Tao, and Bryan T. Wilkinson. Concurrent range reporting in two-dimensional space. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 983–994, 2014. doi:10.1137/1.9781611973402.73.

2 Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. Indexing uncertain data. In *Proc. 28th ACM Symposium on Principles of Database Systems (PODS)*, pages 137–146, 2009. doi:10.1145/1559795.1559816.

3 Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proc. 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2000. doi:10.1109/SFCS.2000.892088.

4 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

5 Panayiotis Bozanis, Nectarios Kitsios, Christos Makris, and Athanasios K. Tsakalidis. New upper bounds for generalized intersection searching problems. In *Proc. 22nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 464–474, 1995. doi:10.1007/3-540-60084-1_97.

6 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of ($\leq k$)-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000. doi:10.1137/S0097539798349188.

7 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

8 Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word RAM. *ACM Transactions on Algorithms*, 9(3):22, 2013.

9 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

10 Timothy M. Chan and Yakov Nekrich. Towards an optimal method for dynamic planar point location. *SIAM Journal on Computing*, 47(6):2337–2361, 2018.

11 Timothy M. Chan and Yakov Nekrich. Better data structures for colored orthogonal range reporting. In *Proc. 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 627–636, 2020.

12 Timothy M. Chan, Yakov Nekrich, Saladi Rahul, and Konstantinos Tsakalidis. Orthogonal point location and rectangle stabbing queries in 3-d. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 31:1–31:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.31.

13 Bernard Chazelle, Richard Cole, Franco P. Preparata, and Chee-Keng Yap. New upper bounds for neighbor searching. *Information and Control*, 68(1-3):105–124, 1986. doi:10.1016/S0019-9958(86)80030-4.

14 Kenneth L Clarkson and Peter W Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

15 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

16 Mark de Berg, Marc van Kreveld, and Jack Snoeyink. Two-dimensional and three-dimensional point location in rectangular subdivisions. *Journal of Algorithms*, 18(2):256–277, 1995.

17 Paul F. Dietz. Fully persistent arrays. In *Proc. 1st Workshop on Algorithms and Data Structures (WADS)*, pages 67–74, 1989. doi:10.1007/3-540-51542-9_8.

18 Hicham El-Zein, J. Ian Munro, and Yakov Nekrich. Succinct color searching in one dimension. In *Proc. 28th International Symposium on Algorithms and Computation (ISAAC)*, pages 30:1–30:11, 2017. doi:10.4230/LIPIcs.ISAAC.2017.30.

19 Travis Gagie, Juha Kärkkäinen, Gonzalo Navarro, and Simon J. Puglisi. Colored range queries and document retrieval. *Theor. Comput. Sci.*, 483:36–50, 2013. doi:10.1016/j.tcs.2012.08.004.

20 Arnab Ganguly, J. Ian Munro, Yakov Nekrich, Rahul Shah, and Sharma V. Thankachan. Categorical range reporting with frequencies. In *Proc. 22nd International Conference on Database Theory (ICDT)*, pages 9:1–9:19, 2019. doi:10.4230/LIPIcs.ICDT.2019.9.

21 Roberto Grossi and Søren Vind. Colored range searching in linear space. In *Proc. 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 229–240, 2014. doi:10.1007/978-3-319-08404-6_20.

22 Prosenjit Gupta, Ravi Janardan, Saladi Rahul, and Michiel H. M. Smid. Computational geometry: Generalized (or colored) intersection searching. In *Handbook of Data Structures and Applications*, chapter 67, pages 1042–1057. CRC Press, 2nd edition, 2018. URL: https://www.csa.iisc.ac.in/~saladi/Papers/ds2-handbook.pdf.

23 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *J. Algorithms*, 19(2):282–317, 1995. doi:10.1006/jagm.1995.1038.

24 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Algorithms for generalized halfspace range searching and other intersection searching problems. *Comput. Geom.*, 6:1–19, 1996. doi:10.1016/0925-7721(95)00012-7.

25 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. A technique for adding range restrictions to generalized searching problems. *Inf. Process. Lett.*, 64(5):263–269, 1997. doi:10.1016/S0020-0190(97)00183-X.

26 Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. Algorithms for some intersection searching problems involving circular objects. *International Journal of Mathematical Algorithms*, 1:35–52, 1999.

27 Joseph JáJá, Christian Worm Mortensen, and Qingmin Shi. Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In *Proc. 15th International Symposium on Algorithms and Computation (ISAAC)*, pages 558–568, 2004. doi:10.1007/978-3-540-30551-4_49.

28 Ravi Janardan and Mario A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

29 Haim Kaplan, Natan Rubin, Micha Sharir, and Elad Verbin. Efficient colored orthogonal range counting. *SIAM J. Comput.*, 38(3):982–1011, 2008. doi:10.1137/070684483.

30 Haim Kaplan, Micha Sharir, and Elad Verbin. Colored intersection searching via sparse rectangular matrix multiplication. In *Proc. 22nd ACM Symposium on Computational Geometry (SoCG)*, pages 52–60, 2006. doi:10.1145/1137856.1137866.

31 Kasper Green Larsen and Rasmus Pagh. I/O-efficient data structures for colored range and prefix reporting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 583–592, 2012. doi:10.1137/1.9781611973099.49.

32 Kasper Green Larsen and Freek van Walderveen. Near-optimal range reporting structures for categorical data. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 256–276, 2013.

33 Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom.*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.

34 Christian Worm Mortensen. Generalized static orthogonal range searching in less space. Technical report, IT University Technical Report Series 2003-33, 2003.

35 Ketan Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice-Hall, 1994.

36 S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002. `doi:10.1145/545381.545469`.

37 Yakov Nekrich. Efficient range searching for categorical and plain data. *ACM Trans. Database Syst.*, 39(1):9, 2014. `doi:10.1145/2543924`.

38 Yakov Nekrich and Jeffrey Scott Vitter. Optimal color range reporting in one dimension. In *Proc. 21st European Symposium on Algorithms (ESA)*, pages 743–754, 2013. `doi:10.1007/978-3-642-40450-4_63`.

39 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proc. 27th ACM Symposium on Computational Geometry (SoCG)*, pages 458–463, 2011. `doi:10.1145/1998196.1998271`.

40 Manish Patil, Sharma V. Thankachan, Rahul Shah, Yakov Nekrich, and Jeffrey Scott Vitter. Categorical range maxima queries. In *Proc. 33rd ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014. `doi:10.1145/2594538.2594557`.

41 Mihai Patrascu. Lower bounds for 2-dimensional range counting. In *Proc. 39th ACM Symposium on Theory of Computing (STOC)*, pages 40–46, 2007. `doi:10.1145/1250790.1250797`.

42 Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011. `doi:10.1137/09075336X`.

43 F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer–Verlag, 1985.

44 Saladi Rahul. Approximate range counting revisited. In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 55:1–55:15, 2017. `doi:10.4230/LIPIcs.SoCG.2017.55`.

45 Raimund Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, pages 37–67. Springer-Verlag, 1993.

46 Qingmin Shi and Joseph JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Inf. Process. Lett.*, 95(3):382–388, 2005. `doi:10.1016/j.ipl.2005.04.008`.

47 Darren Erik Vengroff and Jeffrey Scott Vitter. Efficient 3-d range searching in external memory. In *Proc. 28th ACM Symposium on Theory of Computing (STOC)*, pages 192–201, 1996.