Better Data Structures for Colored Orthogonal Range Reporting

Timothy M. Chan[*] Yakov Nekrich[†]

Abstract
Range searching on categorical, or "colored", data has been studied extensively for over two decades. In this paper, we obtain the current best results for perhaps the most basic, and most often studied, version of the geometric problem: colored orthogonal range reporting.

Given n colored points in two-dimensional space $[U]^2$, we present a data structure with $O(n \log^{3/4+\varepsilon} n)$ space, for an arbitrarily small constant $\varepsilon > 0$, so that all k distinct colors in any axis-aligned query rectangle can be reported in (optimal) $O(\log\log U + k)$ time; this is the first method to break the $O(n \log n)$ space barrier.

In three dimensions, we present a data structure with $O(n \log^{9/5+\varepsilon} n)$ space and $O(\log n / \log\log n + k)$ time; this improves the previous space bound of $O(n \log^4 n)$.

1 Introduction

Range searching [1, 12, 34] is among the most fundamental classes of problems studied in computational geometry. A generalization known as *colored range searching* (or "categorical range searching", or just "generalized range searching") [17] has received considerable attention, motivated by applications from databases and information retrieval, among other things. In the generalized problem, each data object has a color (representing its "category"), and the objective is to obtain information about the colors of the objects inside a query range, for example, to report or count all distinct colors.

Although numerous papers have appeared on colored range searching [5, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 30, 31, 32, 33, 37] (for example, exploring different types of geometric ranges, range vs. intersection searching, reporting vs. counting, static vs. dynamic data structures, online vs. offline queries, internal vs. external memory, etc.), we still do not have complete understanding on arguably the most basic version of the problem: static *colored orthogonal range reporting*, in the standard (internal-memory) word-RAM model of computation. This problem is the main focus of the present paper.

More precisely, given n points in a constant dimension d, where each point is assigned a color, we want to build a data structure so that for any axis-aligned query box (i.e., hyperrectangle) q, we can quickly report the distinct colors among the points inside q. Ideally, the query time should be proportional to k, the number of distinct colors in q. The naive solution of answering a standard (uncolored) range reporting query and then extracting the distinct colors is insufficient, since the query time would be proportional to the number of points in q, which could be much larger than k.

Colored 2D Orthogonal Range Reporting. The complexity of colored orthogonal range reporting is open even in the 2D case. Table 1 gives the long list of previous results in the standard word-RAM model, assuming that coordinates are integers in $[U] = \{1, \ldots, U\}$, and space is measured in words. The best to date were obtained independently a few years ago by Larsen and van Walderveen [27] (SODA 2013) and Nekrich [31] (TODS 2014), and achieved $O(n \log n)$ space and optimal $O(\log\log U + k)$ query time. This query time bound is optimal among $O(n \log^{O(1)} n)$-space data structures, but the space bound may not be.

Virtually all previous methods require at least $n \log n$ space,[1] even when $O(\log^{O(1)} n + k)$ query time is allowed. The reason is that all previous methods first solve the subproblem when the query rectangle is 3-sided (unbounded from below, say); Larsen and van Walderveen's and Nekrich's 3-sided solutions achieved optimal $O(n)$ space and $O(\log\log U + k)$ query time. The general 4-sided problem is then solved using a standard black-box reduction via range trees, which increases space by a logarithmic factor.

Intuitively, one could explain the $n \log n$ space barrier by analogy to standard (uncolored) 3D orthogonal range reporting. Colored 2D range searching is similar to uncolored 3D range searching by viewing colors as z-coordinates, but the best current result for general 3D 6-sided range reporting queries has $O(n \log^{1+\varepsilon} n)$ space and $O(\log\log U + k)$ query time [6], which is obtained via a black-box reduction to the 5-sided case, similarly at a loss of a logarithmic factor in space. (Through-

[*]Department of Computer Science, University of Illinois at Urbana-Champaign (tmc@illinois.edu). Work supported in part by NSF Grant CCF-1814026.

[†]Department of Computer Science, Michigan Technological University (yakov.nekrich@googlemail.com).

[1]One exception is the linear-space data structure by Grossi and Vind [16], but their query time is very large (close to linear).

	space	query time
Janardan & Lopez (1993) [22]	$O(n \log n)$	$O(\log^2 n + k)$
	$O(n \log^2 n)$	$O(\log n + k)$
Gupta, Janardan, & Smid (1995) [18]	$O(n \log^2 n)$	$O(\log n + k)$
Agarwal, Govindarajan, & Muthukrishnan (2002) [2]	$O(n \log^2 n)$	$O(\log \log U + k)$
Mortensen (2003) [28]	$O(n \log n \log \log n)$	$O(\log^2 \log U + k)$
	$O(n \log n)$	$O(\log n \log^2 \log n + k)$
Shi & JaJa (2005) [37]	$O(n \log n)$	$O(\log n + k)$
Larsen & van Walderveen (2013) [27]	$O(n \log n)$	$O(\log \log U + k)$
Nekrich (2014) [31]	$O(n \log n)$	$O(\log \log U + k)$

Table 1: Previous data structures for colored 2D orthogonal range reporting.

out the paper, ε denotes an arbitrarily small positive constant.) This analogy turns out to be faulty.

Our new result is the first data structure that breaks the $O(n \log n)$ space barrier. Specifically, the structure requires $O(n \log^{3/4+\varepsilon} n)$ space while maintaining optimal $O(\log \log U + k)$ query time.

Colored 3D Orthogonal Range Reporting. The problem in 3D appears even tougher. The best result that we can find explicitly stated in the literature has $O(n \log^4 n)$ space and $O(\log^2 n + k)$ query time [17], although by using more recent geometric data structuring techniques [7], it is not difficult to obtain a solution with $O(n \log^3 n)$ space and $O(\log n / \log \log n + k)$ query time (see the remarks after Lemma 3.1).

Our method in 2D (unlike many previous methods) is powerful enough that it can be extended to solve the 5-sided problem in 3D. We obtain a new data structure for colored 3D 5-sided orthogonal range reporting with $O(n \log^{4/5+\varepsilon} n)$ space and $O(\log n / \log \log n + k)$ query time. As a consequence, this implies a data structure for the colored 3D general (6-sided) orthogonal range reporting with $O(n \log^{9/5+\varepsilon} n)$ space and $O(\log n / \log \log n + k)$ query time. This is an even greater improvement than in 2D, by *more than one* logarithmic factor compared to the previous space bounds.

Note that improvements in low-dimensional case are important, as they automatically lead to improvements in higher dimensions via the standard range-tree reduction, which loses one logarithmic factor per dimension in space and query time. Consequently, we can obtain a data structure in dimension $d \geq 4$ with $O(n \log^{d-6/5+\varepsilon} n)$ space and $O(\log^{d-2} n / \log \log n + k \log^{d-3} n)$ query time. However, the result here is not as strong, as we now have to pay a nonconstant $\log^{d-3} n$ penalty per output color.

Related Work: The "Fractional-Power-of-Log" Phenomenon. What make our 2D and 3D results striking are the unusual space bounds involving fractional powers of $\log n$. We are not aware of similar space bounds in the literature on static word-RAM geometric data structures, except for one: Chan and Tsakalidis [10] noted a static data structure for vertical ray shooting among horizontal rectangles in 3D with $O(n \log^{1/2+\varepsilon} n)$ space and $O(\log n)$ expected query time. Their result was actually obtained, via persistence, from a *dynamic* data structure for vertical ray shooting among horizontal line segments in 2D with $O(\log^{1/2+\varepsilon} n)$ update time. There have been several past occurrences of the fractional-power-of-log phenomenon in dynamic data structures, notably, for 1D rank queries (also with near $\sqrt{\log n}$ update time) and 2D orthogonal range reporting [9, 29, 39] (with near $\log^{2/3} n$ update time), as well as in algorithms for offline orthogonal range counting (including inversion counting) [8]. However, there does not appear to be a direct connection with our techniques here, other than some commonalities in the usage of bit packing. In particular, our method is more intricate than the "micro- vs. macro-structures" framework described by Chan and Tsakalidis [9, 10].

Our Techniques. Our data structures are obtained by a nontrivial combination of a series of ideas:

0. bit packing;
1. certain sublinear-space structures, obtained by dividing the input into blocks;
2. a variant of the recursive grid technique by Alstrup, Brodal, and Rauhe [4] (and also Chan, Larsen, and Pătraşcu [6]);
3. range trees over the colors, with a large fan-out.

Although some of these ideas in isolation have been used in prior work (for example, range trees on colors), what is original is how we assemble the pieces together. We hope that our approach will find further applications to other geometric data structuring problems.

A rough roadmap to our 2D solution can be described as follows. Colored range reporting is harder than uncolored version because we need to report each color once. Straightforward application of many standard techniques could result in reporting the same color $\Omega(\log n)$ times. We address this problem by applying bit packing to the output: if the number of points n is small, we can pack many colors into one machine word. First we apply this idea to uncolored range reporting (Idea 0). Next we obtain a data structure supporting colored reporting queries in $O(2^{\log^{1-\varepsilon} n} + (k \log^4 n)/w + k)$ time, where k is the number of output colors (Idea 1). Thus we can report each color in $O(1)$ time provided that n is sufficiently small. Both data structures use small space, but have high query time. As a next step, we reduce the query time and simultaneously get rid of the restriction on n. We achieve this goal by using the recursive grid approach and storing pre-computed answers for selected query ranges (Idea 2). This data structure uses $O(n \log^{\varepsilon} n)$ space and supports colored reporting queries in $O(\log \log U + k)$ time, under the assumption that k is sufficiently small ($k \ll 2^{\log^{1/4} n}$). We get rid of this assumption on k by using a range tree on colors with high fan-out (Idea 3). Our final data structure uses $O(n \log^{3/4+\varepsilon} n)$ space. Interestingly, the query time remains $O(\log \log U + k)$.

Preliminaries. In the rest of the paper, we assume a w-bit word RAM model with $w = \Omega(\log n)$, which supports (possibly nonstandard) operations on words. No "cheating" is involved when we set $w = \varepsilon \log n$ at the end, because nonstandard operations can be simulated by table lookup in constant time, after an initial preprocessing in $2^{O(w)} = n^{O(\varepsilon)}$ time.

Unless noted otherwise, space usage is measured in words rather than in bits.

By default, a 2D 3-sided range refers to one that is unbounded from below in y, and a 3D 5-sided range refers to one that is unbounded from below in z.

2 The 2D Solution

Our data structure for colored 2D orthogonal range reporting is obtained by combining a series of different ideas.

Idea 0: Bit-Packing the Output. An initial idea is *bit packing*—the fact that multiple elements can be packed in a single word, if $\log n \ll w$. We start by observing that a standard data structure for (uncolored) orthogonal range searching allows the output of a query to be packed, potentially allowing for query time that is slightly sublinear in k.

LEMMA 2.1. *Given a set of n points in $\mathbb{R}^2$ and any constant $\varepsilon \in (0,1)$, we can build an $O(n \log^{2\varepsilon} n)$-space data structure that can answer an (uncolored) orthogonal range reporting query in $O(2^{\log^{1-\varepsilon} n} + (k \log n)/w)$ time with k output points. The output is a list of the labels of the points, assuming that each label is $O(\log n)$-bit long.*

Proof. This follows from a standard range tree [1, 12, 34] with fan-out f, except that each canonical subset S is stored as a list of $O((|S| \log n)/w)$ words (since a word can store $\Omega(w/\log n)$ points by bit packing). The data structure uses $O(n(\log_f n)^2)$ space, and the answer to a query is a union of $O((f \log_f n)^2)$ such lists, occupying a total of $O((k \log n)/w)$ words. We can set $f = 2^{(\log^{1-\varepsilon} n)/3}$. ∎

Idea 1: Sublinear Space by Blocking. Our main new idea is showing that it is actually possible to obtain nontrivial data structures with *sublinear space* for colored range reporting, under the assumption that the point set has already been preprocessed for uncolored range searching, specifically, in the data structure from Lemma 2.1. We start with a *capped* version of the problem, where the output size is promised to be at most a fixed value K_0. The basic idea is to divide data into blocks of size B (like in external-memory data structures), and remember only a small number of points per block. Note that the space bound $O(K_0 n/B)$ below is indeed sublinear if the parameter B is chosen to be slightly bigger than K_0. There are extra $\log n$ factors in the query cost, but at the end these factors will be offset by w in the denominator.

LEMMA 2.2. *Suppose that we are given a set S of n colored points in $\mathbb{R}^2$, and we have already built the data structure in Lemma 2.1 for S, where the label of each point stores the ranks of its color, its x-coordinate, and its y-coordinate.*

Given parameters K_0 and B, we can build an additional data structure with $O(K_0 n/B)$ space that can answer a colored 3-sided range reporting query with k output points in $O(2^{\log^{1-\varepsilon} n} + (B \log^2 n)/w + k)$ time, provided that $k \le K_0$. Failure is reported if $k > K_0$.

Proof. Divide the plane into n/B vertical slabs each containing B points of S. In each slab, consider the (y-)lowest point of each color, and put the $K_0 + 1$ lowest of these lowest points (from each slab) into a set M. Then $|M| = O(K_0 \cdot n/B)$. Store M in a known data structure for colored 3-sided range reporting structure with $O(|M|)$ space and $O(\log^{O(1)}|M|+k)$ query time (e.g., by Janardan and Lopez [22]).[2]

To answer a query for a 3-sided range q, we do the following:

1. Find all distinct colors in $M\cap q$ by querying the structure for M in $O(\log^{O(1)} n+k)$ time.

2. Let σ_L and σ_R be the slabs containing the left and right edges of q. Colors from $S\cap\sigma_L\cap q$ and $S\cap\sigma_R\cap q$ may not necessarily be found in $M\cap q$, and so these two slabs need to be dealt with differently: Generate the list of all $O(B)$ points in $S\cap\sigma_L$ and $S\cap\sigma_R$ by querying the data structure from Lemma 2.1 in $O(2^{\log^{1-\varepsilon} n}+(B\log n)/w)$ time. Sort the combined list by color, which occupies $O((B\log n)/w)$ words, using a bit-packed version of mergesort in $O(((B\log n)/w)\cdot\log n)$ time [3]. Then perform a linear scan in $O((B\log n)/w)$ time to filter out points not in q and remove duplicate colors; this gives the list of all distinct colors in $S\cap(\sigma_L\cup\sigma_R)\cap q$.

We output the union of the lists from steps 1 and 2, with duplicates removed, in $O(k)$ additional time. We can detect duplicates by using a bit vector of size $O(n)$. If more than K_0 colors are found, failure is reported.

Correctness is easy to see: Consider a point $p\in S\cap q$. If p is in σ_L or σ_R, then the color of p is found in step 2. Otherwise, p is in some slab σ that q completely cuts across, and the color of p is the color of some point in $M\cap\sigma\cap q$ and is found in step 1, unless $M\cap\sigma\cap q$ contains more than K_0 distinct colors, in which case failure is reported. ∎

Lemma 2.3. *Under the same setting as Lemma 2.2, we can build an additional data structure with $O((K_0 n/B)\log n)$ space that can answer a colored 4-sided range reporting query with k output points in $O(2^{\log^{1-\varepsilon} n}+(B\log^2 n)/w+k)$ time, provided that $k\le K_0$. Failure is reported if $k>K_0$.*

Proof. We can reduce 4-sided queries to 3-sided queries by using a standard range-tree divide-and-conquer (as was also done in previous work [2, 31]). More precisely, we store the point set in our 3-sided structure from Lemma 2.2, divide the point set into the top and bottom halves by the median horizontal line, and recursively build a data structure for each half. Then a 4-sided query can be decomposed into two 3-sided queries, by jumping to the node at which the 4-sided range is split into two by the dividing horizontal line. Space usage increases by a logarithmic factor, but the query time increases only by a constant factor. Note that the structure for Lemma 2.1 is built only once for the global point set S. ∎

We now remove the cap assumption, by working with logarithmically many values of K_0:

Theorem 2.1. *Given a set of n colored points in $\mathbb{R}^2$ and any constant $\varepsilon\in(0,1)$, we can build a data structure with $O(n\log^{2\varepsilon} n)$ space that can answer a colored 4-sided range reporting query with k output points in $O(2^{\log^{1-\varepsilon} n}+(k\log^4 n)/w+k)$ time.*

Proof. In Lemma 2.3, we set $B=K_0\log^2 n$; then space is $O(n/\log n)$ (which is sublinear!). We build this structure for every K_0 that is a power of 2. The total space is $O((n/\log n)\cdot\log n)=O(n)$, excluding the structure from Lemma 2.1, which is built just once and requires $O(n\log^{2\varepsilon} n)$ space.

To answer a query, we query the structure from Lemma 2.3, with a time limit of $O(2^{\log^{1-\varepsilon} n}+(K_0\log^4 n)/w+K_0)$, for $K_0=2^{j_0},2^{j_0+1},\ldots$ with $j_0=\lfloor\log^{1-\varepsilon} n\rfloor$, until success is reached. The total query time forms a geometric series. ∎

The data structure in the above theorem works well only when k is large ($k\gg 2^{\log^{1-\varepsilon} n}$). We would like to reduce the overhead term $2^{\log^{1-\varepsilon} n}$ to $\log\log U$ somehow…

Idea 2: Grid Recursion. We now adopt a powerful *recursive grid* method, which was first proposed by Alstrup, Brodal, and Rauhe [4] for standard (uncolored) 2D orthogonal range reporting, and has since found several applications in geometric data structures (e.g., [6, 7, 25, 35]). More precisely, we use a "lop-sided" variant due to Chan, Larsen, and Pătraşcu [6, Section 3], originally for (uncolored) 3D 4-sided and 5-sided orthogonal range reporting; in a lop-sided grid, there will be more columns than rows. In adapting the method for colored reporting, we run into problems when the query output size k is large, because we need to store more information per grid cell. Luckily the method from Theorem 2.1 works well for large k and can be combined perfectly with the grid recursion:

Theorem 2.2. *Given a set S of n colored points in $[U]^2$ and any constant $\varepsilon\in(0,1)$, we can build a data structure with $O(n\log^{O(\varepsilon)} n)$ space that can answer a colored 4-sided range reporting query with k output points in $O(\log\log U+(k\log^{4/(1-\varepsilon)} k)/w+k)$ time.*

Proof. Suppose that there is already a colored j-sided range reporting structure $\mathcal{DS}_j$ ($j\in\{2,3,4\}$)

[2] If a more self-contained solution is desired, we could instead use bootstrapping here; see Section 3.

with $O(n \log^{1+\alpha_j} n)$ *bits* of space and $O(\log\log U + (k \log^{4/(1-\varepsilon)} k)/w + k)$ query time. For example, known linear-space data structures for colored 2-sided (dominance) range reporting [27, 31] imply $\alpha_2 = 0$, and by using a standard range tree construction to add sides, we can easily obtain $\alpha_3 = 1$ and $\alpha_4 = 2$. (Known data structures on colored 3-sided range reporting [27, 31] actually give $\alpha_3 = 0$ and $\alpha_4 = 1$, but to be more self-contained, we will not use such structures.)

Form an $(n/A^2) \times A$ grid, where each row contains A^2 points of S and each column contains n/A points of S, for some parameter A to be set later. For $j \in \{3, 4\}$, our j-sided data structure for S is built as follows:

1. For each column σ, recursively build a data structure for $S \cap \sigma$.

2. For each row σ, build the $\mathcal{DS}_j$ structure for $S \cap \sigma$; the total space in bits is $O(n \log^{1+\alpha_j} A)$. Here, we assume that coordinates and colors are replaced by their ranks in $S \cap \sigma$, which require $O(\log A)$ bits each (this is known as *rank space reduction*).

   For each column σ, build the $\mathcal{DS}_{j-1}$ structure for $S \cap \sigma$ (possibly with x- and y-axes swapped); the total space in bits is $O(n \log^{1+\alpha_{j-1}} n)$.

3. For each grid cell γ, take up to $2^{\log^{1-\varepsilon} n} + 1$ points of distinct colors in γ, and put them in a set M. Then $|M| = O(2^{\log^{1-\varepsilon} n} \cdot A \cdot n/A^2)$. Store M in the $\mathcal{DS}_j$ structure with $O(|M| \log^{1+\alpha_j} |M|) = O((n/A) 2^{\log^{1-\varepsilon} n} \log^{1+\alpha_j} n)$ bits of space.

4. Build the data structure from Theorem 2.1 for S with $O(n \log^{1+2\varepsilon} n)$ bits of space.

(In the base case when $n \le A^2$, we can just build the structure $\mathcal{DS}_j$ directly, with $O(n \log^{1+\alpha_j} A)$ bits of space.)

To answer a query for a j-sided range q, we do the following:

1. If q is completely inside a column σ, then recursively solve the problem in σ.

2. Otherwise, let σ_T and σ_B be the rows containing the top and bottom edges of q, and let σ_L and σ_R be the columns containing the left and right edges of q. Find all distinct colors in $S \cap \sigma_T \cap q$, $S \cap \sigma_B \cap q$, $S \cap \sigma_L \cap q$, and $S \cap \sigma_R \cap q$ by querying the $\mathcal{DS}_j$ or $\mathcal{DS}_{j-1}$ structures at the rows/columns (depending on whether the range is j- or $(j-1)$-sided in the row/column), in $O(\log\log U + (k \log^{4/(1-\varepsilon)} k)/w + k)$ time. We may need an additional $O(k)$ time to undo rank space reduction for the output colors.

3. Find all distinct colors in $M \cap q$ by querying the $\mathcal{DS}_j$ structure for M in $O(\log\log U + (k \log^{4/(1-\varepsilon)} k)/w + k)$ time.

4. If more than $2^{\log^{1-\varepsilon} n}$ colors have been found in the previous step, then query the data structure from Theorem 2.1 in time $O(2^{\log^{1-\varepsilon} n} + (k \log^4 n)/w + k)$, which is bounded by $O((k \log^{4/(1-\varepsilon)} k)/w + k)$, since $k > 2^{\log^{1-\varepsilon} n}$.

We return the union of the lists of colors found in steps 2–4, with duplicates removed, in $O(k)$ additional time.

Correctness is easy to see: Consider a point $p \in S \cap q$. If p is in σ_T, σ_B, σ_L, or σ_R, then the color of p is found in step 2 (or in step 1 if $\sigma_L = \sigma_R$). Otherwise, p is in some grid cell γ completely inside q, and the color of p is the color of some point in $M \cap \gamma \cap q$ and is found in step 3, unless $M \cap \gamma \cap q$ contains more than $2^{\log^{1-\varepsilon} n}$ distinct colors, in which case the color of p is found in step 4.

To analyze the query time, note that after an initial predecessor search we can jump directly to the first node of the recursion at which q is not completely contained in a column. The predecessor search takes $O(\log\log U)$ time [38]. Afterwards, steps 2–4 take $O(\log\log U + (k \log^{4/(1-\varepsilon)} k)/w + k)$ time. This bounds the overall query time.

Since there are $O(\log_A n)$ levels of recursion, the total space in bits is $O((n \log^{1+\alpha_j} A + n \log^{1+\alpha_{j-1}} n + (n/A) 2^{\log^{1-\varepsilon} n} \log^{1+\alpha_j} n + n \log^{1+2\varepsilon} n) \cdot \log_A n)$. Setting $A = 4^{\log^{1-\varepsilon} n}$ gives the bound $O(n \log^{1+\alpha_j(1-\varepsilon)} n + n \log^{1+\alpha_{j-1}+\varepsilon} n + n \log^{1+3\varepsilon} n)$. Thus, we can effectively replace α_j with

$$\max\{\alpha_j(1-\varepsilon),\ \alpha_{j-1}+\varepsilon,\ 3\varepsilon\}.$$

As $\alpha_2 = 0$, after $O((1/\varepsilon)\log(1/\varepsilon))$ rounds of bootstrapping, we obtain $\alpha_3 = O(\varepsilon)$. After $O((1/\varepsilon)\log(1/\varepsilon))$ further rounds of bootstrapping, we obtain $\alpha_4 = O(\varepsilon)$. This proves the theorem. $\square$

The above theorem works well when the output size k is sufficiently small, up to around $2^{w^{1/4}}$. We need one final ingredient to obtain an efficient solution in general…

Final Idea: Range Tree on Colors. The final idea is an old one: use a range tree over the colors. If a query's output size at a tree node is smaller than a cap of around $K_0 \approx 2^{w^{1/4}}$, we can apply an existing solution for small k (Theorem 2.2); otherwise, we recurse in the children. This idea dates back at least to old work on (uncolored) circular range reporting by Chazelle et al. in the 1980s [11] (where higher-order Voronoi diagrams were used to solve the capped problem); in the context of colored range searching, the usual application sets the cap to $K_0 = 0$, i.e., it provides a reduction of the colored problem to uncolored range emptiness, at the expense of extra logarithmic factors (e.g., see [21] or [17, Section 3.4]). What makes our application more interesting is that we will use a tree of a fairly large fan-out $f \approx 2^{w^{1/4}}$, to avoid losing a full logarithmic factor in space.

One technical issue needs to be addressed first: how can we decide whether a query's output size at a tree node is smaller than the cap K_0? The following lemma solves this subproblem with a polylogarithmic approximation factor, which is sufficient for our purposes (a solution by random sampling is straightforward, but we give a deterministic method).

Lemma 2.4. *Given a set S of n colored points in $\mathbb{R}^2$ and a parameter K_0, we can build a data structure with $O((n/K_0)\log^{O(1)} n)$ space so that for a query 4-sided axis-aligned rectangle, we can conclude that the number of distinct colors is at most $K_0 \log^{O(1)} n$ or at least K_0 in $O(\log^{O(1)} n)$ time.*

Proof. By the standard range tree construction, we can form a collection R of canonical rectangles, with total size $\sum_{r\in R} |S \cap r| = O(n \log^2 n)$, such that any query axis-aligned rectangle can be expressed as a disjoint union of at most $O(\log^2 n)$ canonical rectangles.

Let R' be the subcollection of all canonical rectangles $r \in R$ that contain at least K_0 distinct colors. Then $|R'| = O((n/K_0)\log^2 n)$. We store R' in a data structure with $O(|R'| \log^{O(1)} |R'|)$ space that supports rectangle enclosure queries (finding a rectangle of R' contained in a given query axis-aligned rectangle) in $O(\log^{O(1)} |R'|)$ time; note that rectangle enclosure queries in 2D can be reduced to orthogonal range queries in 4D [34].

Consider a query rectangle q. If some rectangle of R' is contained in q, then q contains at least K_0 distinct colors. If no rectangle of R' is contained in q, then the $O(\log^2 n)$ canonical rectangles associated with q all have at most K_0 distinct colors, so q contains at most $O(K_0 \log^2 n)$ distinct colors. □

Theorem 2.3. *Given a set of n colored points in $[U]^2$ and any constant $\varepsilon \in (0, 1)$, we can build a data structure with $O(n \log^{3/4+O(\varepsilon)} n)$ space that can answer a colored 4-sided range reporting query with k output points in $O(\log \log U + k)$ time.*

Proof. We build a range tree over the colors with fan-out f, where each node stores the data structures from Theorem 2.2 and Lemma 2.4. More precisely, we store the point set in the structures from Lemma 2.4 for some choice of K_0 and from Theorem 2.2, divide the set of all colors into f subsets $C_1, \dots, C_f$ of (roughly) equal size, and for each $i = 1, \dots, f$, recursively build a data structure for the subset of points with colors from C_i. Since there are $O(\log_f n)$ levels of recursion, the total space is $O((n \log^{O(\varepsilon)} n + (n/K_0)\log^{O(1)} n) \cdot \log_f n)$.

To answer a query, we first query the structure from Lemma 2.4. If the number of distinct colors is at most $K_0 \log^{O(1)} n$, then we query the structure from Theorem 2.2. If the number of distinct colors exceeds K_0, we recursively query the data structure for C_i for every $i = 1, \dots, f$. An exception is at the root: here, we skip the query by Lemma 2.4 and instead directly query the structure from Theorem 2.2 with a time limit of $O(\log \log U + (K_0 \log^{4/(1-\varepsilon)} K_0)/w + K_0)$, and if success is not reached, we recursively query the data structure for each C_i.

We now analyze the query time. We visit a node of the range tree only if its parent contains at least K_0 different colors in the query range. Thus the total number of visited nodes per level is $O(fk/K_0)$ (we can identify at least K_0 unique colors per f visited nodes). The total number of nodes visited in the recursion is $O((fk/K_0)\log_f n)$ over all levels. All queries to data structures from Theorem 2.2 take $O((fk/K_0)\log_f n \cdot \log\log U + (k \log^{4/(1-\varepsilon)}(K_0 \log^{O(1)} n))/w + k)$ time. All calls to data structures from Lemma 2.4 take $O((fk/K_0)\log_f n \cdot \log^{O(1)} n)$ time. Additionally, we spend $O(\log \log U + (k \log^{4/(1-\varepsilon)} K_0)/w + k)$ time at the root node.

We set $f = 2^{w^{(1-\varepsilon)/4}}$ and $K_0 = f^2$. For this choice of K_0 and f, we have $\log^{4/(1-\varepsilon)}(K_0 \log^{O(1)} n) = O(w)$, and $(fk/K_0)\log_f n \cdot (\log\log U + \log^{O(1)} n) = o(k)$. Hence, the total query cost is $O(\log \log U + k)$. The space usage is $O(n \log^{3/4+O(\varepsilon)} n)$. □

3 The 3D Solution

In this section, we extend our 2D solution to solve the colored 3D 5-sided range reporting problem. We first note that a solution to the 3D 3-sided case follows from previous work:

Lemma 3.1. *Given a set of n points in $[U]^3$, we can build a data structure with $O(n)$ space that can answer a colored 3-sided (i.e., dominance) range reporting query with k output points in $O(\log_w n + k)$ time.*

Proof. It is not difficult to reduce colored 3D dominance to uncolored *3D 5-sided box stabbing* (given a set of 5-sided boxes in 3D, reporting the rectangles that contain a query point). For example, see [36, Section 3.1] for a description of the reduction. The 3D 5-sided box stabbing problem was recently solved optimally by Chan et al. [7] with $O(n)$ space and $O(\log_w n+k)$ query time. □

Note that by using Lemma 3.1 and a standard range tree construction to add sides, we can immediately obtain a data structure with $O(n\log^2 n)$ space and $O(\log_w n+k)$ query time for colored 3D 5-sided range reporting, and $O(n\log^3 n)$ space and $O(\log_w n+k)$ query time for colored 3D general 6-sided range reporting. We will improve these space bounds.

Idea 1: Sublinear Space by Blocking. We follow the same overall plan as in our 2D solution. The main difference lies in a new version of Lemmas 2.2–2.3, which requires more logarithmic factors in the space and query time bounds (this will explain why the fraction in the final space bound is increased from 3/4 to 4/5). The key idea is again to obtain sublinear-space structures by dividing into blocks and retaining a small number of points (based on the cap) per block. We assume the availability of the following data structure (the remark after Lemma 3.1 implies a structure with $\alpha = 2$ and $\beta = 0$, for example):

Hypothesis 3.2. *Given a set of n colored points in $[U]^3$ and a parameter K_0, we can build a data structure with $O(n\log^\alpha n)$ space that can answer a colored 5-sided range reporting query with k output points in $O(\log_w n + (K_0\log^\beta n)/w + k)$ time, provided that $k \le K_0$. If $k > K_0$, failure is reported.*

Lemma 3.3. *Suppose that we are given a set S of n colored points in $\mathbb{R}^3$, and we have already built the data structure in Lemma 2.1 for the xy-projection of S, where the label of each point stores the ranks of its x-coordinate, y-coordinate, and color.*

Given parameters K_0 and B and any constant $\varepsilon \in (0,1)$, assuming Hypothesis 3.2, we can build an additional data structure with $O((K_0 n/B)\log^{1+\alpha} n)$ space that can answer a colored 5-sided range reporting query with k output points in $O(2^{\log^{1-\varepsilon} n} + (B\log^3 n)/w + (K_0\log^\beta n)/w + k)$ time, provided that $k \le K_0$. If $k > K_0$, failure is reported.

Proof. Let $\sigma_\downarrow$ denote the xy-projection of an object (or a set of objects) σ from 3D to 2D, and let $\sigma^\uparrow$ denote the lifting of an object σ from 2D to 3D.

We build a standard 2D range tree for $S_\downarrow$, except that a node is made a leaf when the size of its corresponding point set drops below B. This gives a collection of $O((n/B)\log n)$ canonical rectangles and leaf rectangles (of total size $O((n/B)\log^2 n)$), such that each leaf rectangle contains $O(B)$ points of $S_\downarrow$, and any axis-aligned rectangle q in 2D can be covered by $O(\log^2 n)$ canonical rectangles completely inside q, and $O(\log n)$ extra leaf rectangles crossing the boundary of q.

For each canonical rectangle r, consider the (z-)lowest point in $r^\uparrow$ of each color, and put the K_0+1 lowest among these points in a set M. Then $|M| = O(K_0 \cdot (n/B)\log n)$. Store M in the colored 5-sided range reporting structure from Hypothesis 3.2 with $O(|M|\log^\alpha |M|) = O((K_0 n/B)\log^{1+\alpha} n)$ space.

To answer a query for a 5-sided range q, we do the following:

1. Find all distinct colors in $M\cap q$ by querying the structure for M in $O(\log_w n + (K_0\log^\beta n)/w + k)$ time.
2. Cover $q_\downarrow$ by $O(\log^2 n)$ canonical rectangles and $O(\log n)$ leaf rectangles. For each of these leaf rectangles r, generate the list of all $O(B)$ points in $S\cap r^\uparrow$ by querying the data structure from Lemma 2.1 in $O(2^{\log^{1-\varepsilon} n} + (B\log n)/w)$ time. Sort the combined list over the $O(\log n)$ leaf rectangles by color (which occupies $O((B\log^2 n)/w)$ words) using a packed mergesort in $O((B\log^2 n)/w \cdot \log n)$ time. Then perform a linear scan in $O((B\log^2 n)/w)$ time to filter out points not in q and remove duplicate colors; this gives the list of colors of points in $S\cap r^\uparrow\cap q$ over all of these leaf rectangles r.

We return the union of the lists from steps 1 and 2, with duplicates removed, in $O(k)$ additional time. If more than K_0 colors are found, failure is reported.

Correctness is easy to see: Consider a point $p \in S\cap q$. If $p_\downarrow$ is in a leaf rectangle associated with $q_\downarrow$, then the color of p is found in step 2. Otherwise, $p_\downarrow$ is in a canonical rectangle r completely inside $q_\downarrow$, and the color of p is the color of some point in $M\cap r^\uparrow\cap q$ and is found in step 1, unless $M\cap r^\uparrow\cap q$ contains more than k_0 distinct colors, in which case failure is reported. □

Bootstrapping helps reduce the number of logarithmic factors:

Corollary 3.1. *Hypothesis 3.2 is true for $\alpha = 2\varepsilon$ and $\beta = 5$.*

Proof. Apply Lemma 3.3 with $B = K_0\log^2 n$. Then the new data structure has $O(n\log^{2\varepsilon} n + n\log^{\alpha-1} n)$ space and $O(2^{\log^{1-\varepsilon} n} + (K_0\log^5 n)/w + (K_0\log^\beta n)/w + k)$ query time. Thus, α and β can effectively be replaced by $\max\{\alpha-1, 2\varepsilon\}$ and $\max\{\beta, 5\}$. A constant number of rounds of bootstrapping makes $\alpha = 2\varepsilon$ and $\beta = 5$. □

We can now obtain the following theorem from Lemma 3.3 (with $\alpha = 2\varepsilon$ and $\beta = 5$), in exactly the same way that we obtain Theorem 2.1 from Lemma 2.3:

THEOREM 3.1. *Given a set of n colored points in $\mathbb{R}^3$ and any constant $\varepsilon \in (0,1)$, we can build a data structure with $O(n \log^{2\varepsilon} n)$ space that can answer a colored 5-sided range reporting query with k output points in $O(2^{\log^{1-\varepsilon} n} + (k \log^5 n)/w + k)$ time.*

Idea 2: Grid Recursion. The next step is to use the recursive grid method to obtain the following theorem from Theorem 3.1, in the same way that we obtain Theorem 2.2 from Theorem 2.1:

THEOREM 3.2. *Given a set S of n colored points in $[U]^3$, and any constant $\varepsilon \in (0,1)$, we can build a data structure with $O(n \log^{O(\varepsilon)} n)$ space that can answer a colored 5-sided range reporting query with k output points in $O(\log_w n + (k \log^{5+O(\varepsilon)} k)/w + k)$ time.*

The grid decomposition is applied to the xy-projection $S_\downarrow$. Note that 3D 4-sided/5-sided ranges are projected to 2D 3-sided/4-sided ranges. The method is almost identical to that in Theorem 2.1, so there is no need to redescribe the proof. The only main difference is in step 3 of the construction: in each grid cell γ, we consider the (z-)lowest point of $S \cap \gamma^\uparrow$ of each color, and put the $2^{\log^{1-\varepsilon} n} + 1$ lowest of these lowest points in the set M. Another difference is that we start with the colored 3D dominance structure from Lemma 3.1 instead of colored 2D dominance, and so all $\log\log U$ terms are replaced by $\log_w n$.

Final Idea: Range Tree on Colors. The proof of Lemma 2.4 easily extends to 3D, with larger polylogarithmic factors. We can thus obtain the following theorem from Theorem 3.2, in exactly the same way that we obtain Theorem 2.3 from Theorem 2.2:

THEOREM 3.3. *Given a set of n colored points in $[U]^3$ and any constant $\varepsilon \in (0,1)$, we can build a data structure with $O(n \log^{4/5+O(\varepsilon)} n)$ space that can answer a colored 5-sided range reporting query with k output points in $O(\log_w n + k)$ time.*

The general 6-sided case can be reduced to the 5-sided case by a standard range tree construction, which increases space by a logarithmic factor but does not increase the query time. This leads to our final result in 3D:

COROLLARY 3.2. *Given a set of n colored points in $[U]^3$ and any constant $\varepsilon \in (0,1)$, we can build a data structure with $O(n \log^{9/5+O(\varepsilon)} n)$ space that can answer a colored 6-sided range reporting query with k output points in $O(\log_w n + k)$ time.*

4 Final Remarks

An obvious open problem is to improve the exponent in the logarithmic factors in our space bounds. With this type of techniques, it seems difficult to get below $n\sqrt{\log n}$ space, however.

In applications where the output size k is expected to be small, Theorems 2.2 and 3.2 may already be good enough and have near-linear ($O(n \log^\varepsilon n)$) space. If all queries have approximately the same fixed value of k, then we can again get near-linear space, for example, by randomly partitioning the colors into $O(k/\log n)$ subclasses, and building the structure for Theorems 2.2 and 3.2 for the points from each subclass (the query range would have about $O(\log n)$ colors in each subclass with high probability by the Chernoff bound, so these theorems would work well in such a case).

Can the $\log_w n$ term in the query time bound be improved for colored 3D orthogonal range reporting? Note that if the query time could be improved for colored 3D dominance range reporting (Lemma 3.1), we would automatically get better query time for our data structures for colored 5-sided and 6-sided range reporting. (In the proof of Lemma 3.1, colored 3D dominance is reduced to the 3D 5-sided box stabbing problem, for which $\log_w n$ is known to be tight; however, reduction in the opposite direction is unclear.)

References

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

[2] P. K. Agarwal, S. Govindarajan, and S. Muthukrishnan. Range searching in categorical data: Colored range searching on grid. In *Proc. 10th Annual European Symposium on Algorithms (ESA)*, pages 17–28, 2002.

[3] S. Albers and T. Hagerup. Improved parallel integer sorting without concurrent writing. *Inf. Comput.*, 136(1):25–51, 1997.

[4] S. Alstrup, G. S. Brodal, and T. Rauhe. New data structures for orthogonal range searching. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2000.

[5] P. Bozanis, N. Kitsios, C. Makris, and A. 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.

[6] T. M. Chan, K. G. Larsen, and M. Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

[7] T. M. Chan, Y. Nekrich, S. Rahul, and K. 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.

[8] T. M. Chan and M. Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 161–173, 2010.

[9] T. M. Chan and K. Tsakalidis. Dynamic orthogonal range searching on the RAM, revisited. In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 28:1–28:13, 2017.

[10] T. M. Chan and K. Tsakalidis. Dynamic planar orthogonal point location in sublogarithmic time. In *Proc. 34th International Symposium on Computational Geometry (SoCG)*, pages 25:1–25:15, 2018.

[11] B. Chazelle, R. Cole, F. P. Preparata, and C. Yap. New upper bounds for neighbor searching. *Information and Control*, 68(1-3):105–124, 1986.

[12] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 3rd edition, 2008.

[13] H. El-Zein, J. I. Munro, and Y. Nekrich. Succinct color searching in one dimension. In *Proc. 28th International Symposium on Algorithms and Computation (ISAAC)*, pages 30:1–30:11, 2017.

[14] T. Gagie, J. Kärkkäinen, G. Navarro, and S. J. Puglisi. Colored range queries and document retrieval. *Theor. Comput. Sci.*, 483:36–50, 2013.

[15] A. Ganguly, J. I. Munro, Y. Nekrich, R. Shah, and S. V. Thankachan. Categorical range reporting with frequencies. In *Proc. 22nd International Conference on Database Theory (ICDT)*, pages 9:1–9:19, 2019.

[16] R. Grossi and S. Vind. Colored range searching in linear space. In *Proc. 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 229–240, 2014.

[17] P. Gupta, R. Janardan, S. Rahul, and M. 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.

[18] P. Gupta, R. Janardan, and M. H. M. Smid. Further results on generalized intersection searching problems: Counting, reporting, and dynamization. *J. Algorithms*, 19(2):282–317, 1995.

[19] P. Gupta, R. Janardan, and M. H. M. Smid. Algorithms for generalized halfspace range searching and other intersection searching problems. *Comput. Geom.*, 6:1–19, 1996.

[20] P. Gupta, R. Janardan, and M. H. M. Smid. A technique for adding range restrictions to generalized searching problems. *Inf. Process. Lett.*, 64(5):263–269, 1997.

[21] P. Gupta, R. Janardan, and M. H. M. Smid. Algorithms for some intersection searching problems involving circular objects. *International Journal of Mathematical Algorithms*, 1:35–52, 1999.

[22] R. Janardan and M. A. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry and Applications*, 3(1):39–69, 1993.

[23] H. Kaplan, N. Rubin, M. Sharir, and E. Verbin. Efficient colored orthogonal range counting. *SIAM J. Comput.*, 38(3):982–1011, 2008.

[24] H. Kaplan, M. Sharir, and E. Verbin. Colored intersection searching via sparse rectangular matrix multiplication. In *Proc. 22nd ACM Symposium on Computational Geometry (SoCG)*, pages 52–60, 2006.

[25] M. Karpinski and Y. Nekrich. Space efficient multidimensional range reporting. In *Proc. 15th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 215–224, 2009.

[26] K. G. Larsen and R. Pagh. I/O-efficient data structures for colored range and prefix reporting. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 583–592, 2012.

[27] K. G. Larsen and F. van Walderveen. Near-optimal range reporting structures for categorical data. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 256–276, 2013.

[28] C. W. Mortensen. Generalized static orthogonal range searching in less space. Technical report, IT University Technical Report Series 2003-33, 2003.

[29] C. W. Mortensen. Fully dynamic orthogonal range reporting on RAM. *SIAM J. Comput.*, 35(6):1494–1525, 2006.

[30] S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

[31] Y. Nekrich. Efficient range searching for categorical and plain data. *ACM Trans. Database Syst.*, 39(1):9, 2014.

[32] Y. Nekrich and J. S. Vitter. Optimal color range reporting in one dimension. In *Proc. 21st Annual European Symposium Algorithms (ESA)*, pages 743–754, 2013.

[33] M. Patil, S. V. Thankachan, R. Shah, Y. Nekrich, and J. S. Vitter. Categorical range maxima queries. In *Proc. 33rd ACM Symposium on Principles of Database Systems (PODS)*, pages 266–277, 2014.

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

[35] S. Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015.

[36] S. Rahul. Approximate range counting revisited. In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 55:1–55:15, 2017.

[37] Q. Shi and J. JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Inf. Process. Lett.*, 95(3):382–388, 2005.

[38] P. van Emde Boas, R. Kaas, and E. Zijlstra. Design and

implementation of an efficient priority queue. *Math. Sys. Theory*, 10:99–127, 1977.

[39] B. T. Wilkinson. Amortized bounds for dynamic orthogonal range reporting. In *Proc. 22th Annual European Symposium on Algorithms (ESA)*, pages 842–856, 2014.