Efficient Algorithms for Generalized Intersection Searching on Non-Iso-Oriented Objects

Prosenjit Gupta* Ravi Janardan* Michiel Smid†

Abstract

Generalized intersection searching problems are a class of geometric query-retrieval problems where the questions of interest concern the intersection of a query object with aggregates of geometric objects (rather than with individual objects.) This class contains, as a special case, the well-studied class of standard intersection searching problems and is rich in applications. Unfortunately, the solutions known for the standard problems do not yield efficient solutions to the generalized problems. Recently, efficient solutions have been given for generalized problems where the input and query objects are iso-oriented (i.e., axes-parallel) or where the aggregates satisfy additional properties (e.g., connectedness). In this paper, efficient algorithms are given for several generalized problems involving non-iso-oriented objects. These problems include: generalized halfspace range searching, segment intersection searching, triangle stabbing, and triangle range searching. The techniques used include: computing suitable sparse representations of the input, persistent data structures, and filtering search.

Keywords: Computational geometry, data structures, filtering search, geometric duality, intersection searching, persistence.

1 Introduction

Consider the following generic searching problem: Suppose that we are given a set S of n geometric objects in $\mathcal{R}^d$. Moreover assume that the objects come aggregated in disjoint groups, where the grouping is dictated by the underlying application. (The number of groups can range from 1 to $\Theta(n)$.) Our goal is to preprocess S into a data structure so that given any query object q, we can report or count efficiently the groups that are intersected by q. (We say that q *intersects a group* iff q intersects some object in the group.) Notice that we are not interested in reporting or counting the individual objects intersected by q as is the case in a *standard intersection searching* problem. Indeed the standard problem is a special case of the above formulation, where each group has cardinality 1. For this reason, we call our version a *generalized intersection searching* problem.

For our purposes, it will be convenient to associate with each group a different color and imagine that all the objects in the group have that color. Suppose that q intersects i groups. Then, we can restate our problem as: "*Preprocess a set S of n colored geometric objects so that for any query object q, the i distinct colors of the objects that are intersected by q can be reported or counted effi-*

*Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A. Email: {pgupta, janardan}@cs.umn.edu. The research of these authors was supported in part by NSF grant CCR–92–00270. Portions of this work were done while RJ was visiting MS at the Max-Planck-Institut für Informatik, in Saarbrücken, Germany, in July 1993. RJ would like to thank the MPI for its generous support.

†Max-Planck-Institut für Informatik, D-66123 Saarbrücken, Germany. Email: michiel@mpi-sb.mpg.de. This author was supported by the ESPRIT Basic Research Actions Program, under contract No. 7141 (project ALCOM II).

10th Computational Geometry 94-6/94 Stony Brook, NY, USA
© 1994 ACM 0-89791-648-4/94/0006..$3.50

ciently." This is the version that we will consider throughout the paper.

Before going further, let us illustrate the usefulness of our generalized formulation with some applications: **(1)** In designing a VLSI chip, the wires (line segments) can be grouped naturally according to the circuits they belong to. A problem of interest to the designer is determining which circuits (rather than wires) become connected when a new wire is to be added. This is an instance of the *generalized segment intersection searching* problem. **(2)** Consider a collection of supply points (e.g., warehouses) of different types in $\mathcal{R}^2$. We would like to preprocess these points so that given a demand point q and a radius r, we can determine the types of supply points that are within distance r from q. By using a well-known "lifting" transformation, this problem can be transformed to an instance of *generalized halfspace range searching* in $\mathcal{R}^3$.

1.1 Potential approaches

One approach to solving a generalized problem is to take advantage of known solutions for the corresponding standard problem. For example, to solve a generalized reporting problem, we can determine all the objects intersected by q (a standard problem) and then read off the distinct colors among these. However, with this approach the query time can be very high since q could intersect $\Omega(n)$ objects but only $O(1)$ distinct colors. Thus, the challenge in the generalized reporting problem is to attain a query time that is sensitive to the output size i, typically of the form $O(f(n) + i)$ or $O(f(n)+i\cdot\text{polylog}(n))$, where $f(n)$ is "small" (e.g., $\text{polylog}(n)$ or n^p, where $0 < p < 1$). For a generalized counting problem, it is not even clear how one can use the solution for the corresponding standard problem (a mere count) to determine how many distinct colors are intersected. Nevertheless, we seek here query times of the form $O(f(n))$. Of course, in both cases, we seek solutions that are also as space-efficient as possible.

1.2 Previous work

While the standard problems have been investigated extensively, their generalized counterparts have been less studied. The generalized problems were first considered in [JL93], where efficient solutions were given for several problems defined on *iso-oriented objects* (i.e., the input and the query objects are axes-parallel). In [GJS93a], efficient solutions were given for the counting, reporting, and dynamic versions of several iso-oriented problems. In [GJS93b], solutions were given for generalized problems involving circular and circle-like objects (among other results). In [AvK93], Agarwal and van Kreveld consider the problem of reporting the intersections of a query line segment with color classes consisting of line segments and satisfying the property that each color class is a simple polygon or a connected component.

1.3 Summary of results and techniques

In this paper, we present efficient solutions to several generalized intersection searching problems that are defined on non-iso-oriented objects. Specifically, we consider the following problems: generalized halfspace range searching in $\mathcal{R}^d$, for any fixed $d \geq 2$, generalized segment intersection searching, triangle stabbing, and triangle range searching in $\mathcal{R}^2$. Our main results are summarized in Table 1. No results were known previously for any of these problems, with the exception of generalized segment intersection searching: For this problem, the following results were known: (a) $O(n^{1+\epsilon})$ space and $O(n^{1/2+\epsilon} + i)$ query time for color classes consisting of simple polygons or connected components [AvK93]; (b) $O((n + \chi)^2 \log n)$ (resp. $O((n + \chi)^2)$) space and $O(\log n + i)$ (resp. $O(\log^2 n + i)$) query time for general color classes [JL93]; and (c) $O(n^{1+\epsilon})$ space and $O((i + 1)\sqrt{n}\log^{O(1)} n)$ query time for general color classes [AvK93]. (χ is the number of pairwise-intersecting segments.)

Our results are based on a combination of several techniques: (1) Computing for each color class a sparse representation which captures essential information about the color class and allows us to reduce the generalized problem at hand to a standard problem. (2) The persistence-addition technique of Driscoll et al. [DSST89], which allows us to reduce a generalized problem to a generalized dynamic problem one dimension lower. (3) A version of filtering search which, in combination with

#	*Input objects*	*Query object*	*Space*	*Query time*
1	Points in $\mathcal{R}^d$	Halfspace in $\mathcal{R}^d$	$d=2$ $n\log n$	$\log^2 n + i$ $n^{1/2}$
			$d=3$ $n\log^2 n$ $n^{2+\epsilon}$ $n\log n$	$n^{1/2+\epsilon}+i$ $\log^2 n + i$ $n^{2/3+\epsilon}$
			$d \geq 2$ $n^{\lfloor d/2 \rfloor + \epsilon}$	$\log n + i\log^2 n$
2	Line segs. in $\mathcal{R}^2$	Halfplane	$n\log n$	$\log^2 n + i$ $n^{1/2}$
		Vertical ray	$n\alpha(n)\log n$	$\log^2 n + i$ $(n\alpha(n))^{1/2}$
3	Line segs. of length $\geq$ a constant in unit square	Line	$n\log n$	$\log^2 n + i$
4	Lines in $\mathcal{R}^2$	Vertical line segment	$n^{2.5-\mu}\log n$ $n^2\log n$	$n^\mu + i$ $\log n + i$
5	Line segs. in $\mathcal{R}^2$	Vertical line segment	$(n+\chi)\log n$	$\log n + i$
		Arbitrary line segment		$\log^2 n + i\log n$
6	Triangles	Point	$n^{3/2}\log n$	$\log^2 n + i$
	Fat-Wedges		$n\log n$	$\log^2 n + i$
7	Points	Fat-Triangle	$n\log^3 n$	$\log^4 n + i\log^2 n$

Table 1: Summary of main results for generalized intersection searching problems; additional results can be found in the text. Wherever the output size, i, is missing in a query time bound, it is a counting problem. A *fat-wedge* or *fat-triangle* is one where each interior angle is greater than or equal to a fixed constant. Here:

- n : input size
- i : output size (number of distinct colors intersected)
- ϵ : arbitrarily small constant > 0
- μ : tunable parameter, $0.5 < \mu < 1$
- χ : number of pairwise-intersecting segments, $0 \leq \chi \leq \binom{n}{2}$
- $\alpha(n)$: slow-growing inverse of Ackermann's function

persistence, yields a space-query time tradeoff. Moreover, when the input objects or query objects satisfy certain reasonable conditions (e.g., fatness), then we use further ideas to obtain very efficient solutions. Due to space limitations, we will describe only a subset of our results here and will omit all proofs and many details. The full paper is available as [GJS93c].

2 Generalized halfspace range searching in $\mathcal{R}^d$

Let S be a set of n colored points in $\mathcal{R}^d$, for any fixed $d \geq 2$. We show how to preprocess S so that for any query hyperplane Q, the i distinct colors of the points lying in the halfspace Q^- (i.e., below Q) can be reported or counted efficiently. We

first give solutions for $\mathcal{R}^2$ and $\mathcal{R}^3$. Let $\mathcal{F}$ denote the well-known point-hyperplane duality transform [Ede87]. Using $\mathcal{F}$ we map S to a set S' of hyperplanes (lines in $\mathcal{R}^2$ and planes in $\mathcal{R}^3$) and map Q to the point $q = \mathcal{F}(Q)$. Our problem is now equivalent to: "Report or count the i distinct colors of the hyperplanes lying on or above q, i.e., the hyperplanes that are intersected by the vertical ray r emanating upwards from q."

Let S_c be the set of hyperplanes of color c. For each color c, we compute the *upper envelope* E_c of the hyperplanes in S_c. In $\mathcal{R}^2$, E_c is an unbounded convex chain, and in $\mathcal{R}^3$, E_c is an unbounded convex polytope whose facets are convex polygons. It is clear that (i) r intersects a c-colored hyperplane iff r intersects E_c and, moreover, (ii) if r intersects E_c, then r intersects the interior of a unique facet of E_c. (For this version of the paper, we assume that r does not intersect two or more facets of E_c at a common boundary; in the full paper [GJS93c], we show how to remove this assumption.) Let $\mathcal{E}$ be the collection of the envelopes of the different colors. By the above discussion, our problem is equivalent to: "Report or count the facets of $\mathcal{E}$ that are intersected by r", which is a standard intersection searching problem! Since we use different techniques to solve this problem in $\mathcal{R}^2$ and in $\mathcal{R}^3$, we will discuss each case separately.

2.1 Solving the ray–envelope intersection problem in $\mathcal{R}^2$

We store the x-projections of the line segments of $\mathcal{E}$ in a *segment tree* T. Let v be any node of T. Associated with v is an x-interval $I(v)$. Let $Strip(v)$ be the vertical strip defined by $I(v)$. We say that a segment $s \in \mathcal{E}$ is *allocated* to a node $v \in T$ iff $I(v) \neq \emptyset$ and s crosses $Strip(v)$ but not $Strip(parent(v))$. Let $\mathcal{E}(v)$ be the set of segments allocated to v. Within $Strip(v)$, the segments of $\mathcal{E}(v)$ can be viewed as lines since they cross $Strip(v)$ completely. Let $\mathcal{E}'(v)$ be the set of points dual to these lines. We store $\mathcal{E}'(v)$ in an instance $H(v)$ of the standard halfplane reporting (resp. counting) structure for $\mathcal{R}^2$ given in [CGL85] (resp. [Mat92b]). This structure has size $O(m)$ and a query time of $O(\log m + k_v)$ (resp. $O(m^{1/2})$), where $m = |\mathcal{E}(v)|$ and k_v is the output size at v.

To answer a query, we search in T using q's x-coordinate. At each node v visited, we need to report or count the lines intersected by r. But, by duality, this is equivalent to answering, in $\mathcal{R}^2$, a halfplane query at v using the query $\mathcal{F}(q)^- = Q^-$, which we do using $H(v)$.

Theorem 2.1 *A set S of n colored points in $\mathcal{R}^2$ can be stored in a data structure of size $O(n \log n)$ so that the i distinct colors of the points lying in any query halfplane can be reported (resp. counted) in time $O(\log^2 n + i)$ (resp. $O(n^{1/2})$).* □

Using a similar approach, we can also solve Problem 2 in Table 1.

2.2 Solving the ray–envelope intersection problem in $\mathcal{R}^3$

We triangulate the facets of E_c for all colors c. For any triangle t, let $h(t)$ be its supporting plane. Let t' and q' be the projections of t and q (the origin of ray r) on the xy-plane respectively. Clearly, t is intersected by r iff (a) t''s interior contains q' and (b) $h(t)$ is on or above q.

Wlog assume that t' has a vertical side; otherwise decompose it into two such triangles. To find the triangles satisfying condition (a), we store each t' in a segment tree T according to its x-span. Let v be any node of T and let $A(v)$ be the set of triangles allocated to v. Let $m = |A(v)|$. Note that if $t' \in A(v)$ then both its non-vertical sides, call the upper one t'_u and the lower one t'_l, cross $Strip(v)$. If $q' \in Strip(v)$, then q' is in t''s interior iff q' is above t'_l and below t'_u. Since t'_l and t'_u behave like lines within $Strip(v)$, by duality we have that $q' \in t'$ iff in $\mathcal{R}^2$ one endpoint of $\overline{\mathcal{F}(t'_l)\mathcal{F}(t'_u)}$ lies in the open halfplane $\mathcal{F}(q')^-$ and the other lies in the open halfplane $\mathcal{F}(q')^+$. Next, consider condition (b). By duality, $h(t)$ is on or above q iff in $\mathcal{R}^3$ the point $\mathcal{F}(h(t))$ is in the halfspace $\mathcal{F}(q)^-$.

So, our problem at v is to report or count the i_v triangles of $A(v)$ that satisfy the above halfplane and halfspace queries. We can do this by augmenting v with a 3-level data structure $\mathcal{D}(v)$, using partition trees [Mat92a] in the outer two levels of $\mathcal{D}(v)$ and the data structure of [AHL90] for halfspace range reporting at the innermost level. For the counting problem, we use partition trees at all three levels. $\mathcal{D}(v)$ can be built in space

$O(m \log m)$ (resp. $O(m)$) so that all the desired triangles can be reported (resp. counted) in time $O(m^{1/2+\epsilon} + i_v)$ (resp. $O(m^{2/3+\epsilon})$). For the reporting problem, instead of partition trees, we can also use 2-dimensional cutting trees [Mat91a] for the two outer levels of $\mathcal{D}(v)$. Then $\mathcal{D}(v)$ has size $O(m^{2+\epsilon})$ and query time $O(\log m + i_v)$. Plugging these bounds into the segment tree, we get:

Theorem 2.2 *The reporting version of the generalized halfspace range searching problem for a set of n colored points in $\mathcal{R}^3$ can be solved in $O(n \log^2 n)$ (resp. $O(n^{2+\epsilon})$) space and $O(n^{1/2+\epsilon} + i)$ (resp. $O(\log^2 n + i)$) query time, where i is the output size and $\epsilon > 0$ is an arbitrarily small constant. The counting version is solvable in $O(n \log n)$ space and $O(n^{2/3+\epsilon})$ query time.* □

2.3 Generalized halfspace range reporting in $d \geq 2$ dimensions

The preceding approach extends to $\mathcal{R}^d$, $d > 3$, but the space bound is high, namely, $O(n^{d\lfloor d/2 \rfloor + \epsilon})$. We now give an approach for the *reporting* problem in $\mathcal{R}^d$ ($d \geq 2$) that needs much less space. In preprocessing, we store the distinct colors in the input point-set S at the leaves of a balanced binary tree CT (in no particular order). For any node v of CT, let $C(v)$ be the colors stored in the leaves of v's subtree and let $S(v)$ be the points of S colored with the colors in $C(v)$. At v, we store a data structure $HSE(v)$ to solve the *halfspace emptiness* problem on $S(v)$, i.e., "does a query halfspace contain any points of $S(v)$?" $HSE(v)$ returns "true" iff the query halfspace is empty. If $|S_v| = n_v$, then $HSE(v)$ uses $O(n_v^{\lfloor d/2 \rfloor + \epsilon})$ space and has query time $O(\log n_v)$ [Mul93, page 290].

To answer a generalized halfspace reporting query we do a depth-first search in CT and query $HSE(v)$ at each node v visited. If v is a non-leaf then we continue searching below v iff the query returns "false"; if v is a leaf, then we output the color stored there iff the query returns "false".

Theorem 2.3 *For any fixed $d \geq 2$, a set S of n colored points in $\mathcal{R}^d$ can be stored in a structure of size $O(n^{\lfloor d/2 \rfloor + \epsilon})$ such that the i distinct colors of the points contained in a query halfspace Q^- can be reported in time $O(\log n + i \log^2 n)$. Here $\epsilon > 0$ is an arbitrarily small constant.* □

We note that the reporting (resp. counting) problem can also be solved in $O(n^{d+\epsilon})$ space and $O(\log n + i)$ (resp. $O(\log n)$) time, using constant-depth cutting trees.

3 Generalized intersection reporting on lines and line segments

We consider several versions of the generalized intersection searching problem for a set S of n colored lines or line segments; the query q is a line or a line segment.

3.1 Querying colored line segments with a line

We dualize the colored line segments of S to a set S' of colored doublewedges and map the query line q to a point q'. Thus, our problem reduces to reporting the i distinct colors that are stabbed by q'. In Section 4, we solve this problem in $O(n^{3/2} \log n)$ space with a query time of $O(\log^2 n + i)$. In the rest of this section, we consider the case where the segments of S all lie in the unit square $\mathcal{U}$ and each segment has length at least λ, where $\lambda > 0$ is a constant. These assumptions are reasonable for practical applications and they allow a very efficient solution.

For now assume that all the segments intersect the y-axis Y. Thus, one endpoint of s has negative x-coordinate and the other has positive x-coordinate. Thus in the corresponding dual doublewedge one of the bounding lines has positive slope and the other has negative slope. We split each doublewedge into a *left-wedge* and a *right-wedge*. Note that each wedge is y-monotone. Consider the right-wedges. Because of y-monotonicity, q' is contained in a right-wedge w iff the horizontal, leftward-directed ray r emanating from q' intersects the boundary of w. Thus for each color c, we compute the *left-envelope* of the boundaries of all c-colored right wedges. If there are n_c c-colored right wedges, then the c-colored left-envelope has size $O(n_c)$ (see [Ede87, page 357, Problem 15.6]). In this way, we obtain a collection S'' of colored line segments. Note that (i) r intersects the boundary of a c-colored right-wedge iff r intersects a c-colored

left-envelope and (ii) if r intersects a c-colored left-envelope then it intersects a unique line segment of this envelope. Thus we have a standard problem which we can solve in $O(n \log n)$ space and $O(\log^2 n + i)$ query time by using a segment tree as in Section 2.1.

What if the segments of S do not all intersect Y? Suppose that there is a constant K such that each segment intersects one of K fixed lines $Y_1, \ldots, Y_K$. Let $S_i \subseteq S$ be the set of segments intersecting Y_i. (If a segment intersects more than one Y_i, we put it in any one of the S_i's; thus the S_i's partition S.) For $1 \leq i \leq K$, we create a coordinate system $\mathcal{C}_i$, where Y_i is the y-axis and any line perpendicular to Y_i is taken as the x-axis. We give the segments of S_i coordinates in $\mathcal{C}_i$ and store them in the data structure described above. To answer a query, we query each of the K structures separately. The space and query time are as above.

We can now solve the problem stated at the beginning. Wlog assume that the origin is at the bottom-left corner of $\mathcal{U}$. Consider the $K = 2 + 2\lceil\sqrt{2}/\lambda\rceil$ lines $x = i \cdot \lambda/\sqrt{2}$ and $y = i \cdot \lambda/\sqrt{2}$, where $0 \leq i \leq \lceil\sqrt{2}/\lambda\rceil$. Since each segment has length at least λ, either its x-span or its y-span is at least $\lambda/\sqrt{2}$. Thus each segment intersects one of the K lines. We now use the above structure.

Theorem 3.1 *Let $\lambda > 0$ be a constant and let $\mathcal{U}$ be the unit square. A set S of n colored line segments in $\mathcal{R}^2$, where each segment has length at least λ and all segments lie in $\mathcal{U}$, can be stored in a structure of size $O(n \log n)$ such that the i distinct colors of the segments that are intersected by a query line q can be reported in time $O(\log^2 n + i)$.* □

3.2 Querying colored lines with a vertical line segment

We give a simple approach based on persistence [DSST89], and then show how to incorporate filtering search to get a space-query time tradeoff.

We divide the plane into $t = O(n^2)$ strips by drawing vertical lines through the intersection points of the n lines. Within any strip, V_k, the lines are totally ordered from top to bottom, as $E_k : \ell_1, \ell_2, \ldots, \ell_n$. Suppose that the vertical query segment q is in V_k and let ℓ_a and ℓ_b be the highest and lowest lines of E_k intersected by q. Our problem is now equivalent to the following *generalized 1-dimensional range searching* problem: Given colored integers $1, 2, \ldots, n$ (where integer j gets the color of ℓ_j), report the distinct colors in the query interval $[a, b]$. In [GJS93a], this problem is solved using a structure D_k of size $O(n)$ and a query time $O(\log n + i)$. D_k supports updates in $O(\log n)$ time with $O(\log n)$ memory modifications. By sweeping over the strips V_k, $1 \leq k \leq t$, and making the associated D_k's partially persistent, we get a solution with space $O(n^2 \log n)$ and query time $O(\log n + i)$.

In a nutshell, the idea for the space-query-time tradeoff is as follows: We extract from the sequence $\mathcal{E} = (E_1, \ldots, E_{t+1})$ a smaller subsequence $\mathcal{E}' = (E'_1, \ldots, E'_m)$ such that (i) E'_j and E'_{j+1} differ in just two (not necessarily adjacent) positions, (ii) for each $E_k \in \mathcal{E}$ there is an $E'_j \in \mathcal{E}'$ which "approximates" E_i in a sense that we will elaborate upon later, and (iii) $m = O(n^{2.5-\mu})$, where $0.5 < \mu < 1$ is a tunable parameter. Properties (i) and (iii) suggest that we can apply persistence to $\mathcal{E}'$ and get a scheme with space bound $o(n^2)$; property (ii) suggests that instead of querying E_k, as we might in the simple scheme, we can query E'_j in the new scheme and still be assured of correctness.

Formally, we define a sequence $b_1, b_2, \ldots, b_B$ of distinguished list positions called *borders*, where $b_i = (i-1)\lfloor n^\mu + 1 \rfloor + 1$ and $B = \Theta(n^{1-\mu})$. We construct $\mathcal{E}'$ by scanning $\mathcal{E}$. Let E_i be the currently scanned list of $\mathcal{E}$. Let E'_j be the most recently constructed list of $\mathcal{E}'$ and suppose that we constructed E'_j when we reached $E_{\delta_j} \in \mathcal{E}$. If E_{i+1} is obtained from E_i by swapping lines across some border b_k, i.e. by swapping the b_kth line α with either the $(b_k - 1)$th line β or the $(b_k + 1)$th line γ, then we create E'_{j+1} from E'_j by swapping α with β or γ, as appropriate; we also set $\delta_{j+1} = i + 1$. It can be shown that E'_j approximates any list $E_k \in \{E_{\delta_j}, E_{\delta_j+1}, \ldots, E_{\delta_{j+1}-1}\}$ in the following sense: for any two successive borders b_l and b_{l+1}, the (unordered) set of lines at positions $b_l + 1, \ldots, b_{l+1}$ in E'_j is the same as the (unordered) set of lines in E_k at these same positions. Moreover, it can be shown that the border-lines in E'_j and E_k are the same; this implies that the border-lines in $\{E_{\delta_j}, E_{\delta_j+1}, \ldots, E_{\delta_{j+1}-1}\}$ are the same and can be totally ordered. Finally, using k-set theory, it can be proved that $|\mathcal{E}'| = O(n^{2.5-\mu})$.

The data structure consists of (1) a red-black

tree T'_j storing the total order of the border lines of E'_j, (2) an instance D'_j of the generalized 1-dimensional range searching structure built on the colored integers $I_j : (1, 2, \ldots, n)$, where integer p gets the color of the pth line of E'_j, and (3) a red-black tree I'_j storing I_j. We make all these structures persistent.

Given q, suppose we need to query E'_j. We find the smallest border b_s on or above q's upper endpoint and, symmetrically the greatest border b_g. We query the structure D'_j with $[b_s + 1, b_g]$. Then we scan all the integers $b_s, b_s - 1, \ldots, b_{s-1} + 1$ in I_j and report the distinct colors of the lines of E'_j at these positions that are intersected by q; symmetrically for b_g.

Theorem 3.2 *A set S of n colored lines in $\mathcal{R}^2$ can be stored in a data structure of size $O(n^{2.5-\mu} \log n)$ such that the i distinct colors of the lines that are intersected by a vertical query line segment can be reported in $O(n^\mu + i)$ time. Here μ is a tunable parameter in the range $0.5 < \mu < 1$. The problem is also solvable in $O(n^2 \log n)$ space and $O(\log n + i)$ query time.* □

We remark that our method is based on an idea used in [AvKO93] for a different (standard) problem. However, there are several crucial differences: (i) because our problem is a generalized one, our algorithm for constructing $\mathcal{E}'$ uses a different swap criterion, (ii) because our query is a vertical line segment rather than a ray (as in [AvKO93]), our choice of borders needs to be different in order to get a sublinear query time. Further discussion of this appears in [GJS93c].

3.3 Querying colored line segments with a line segment

3.3.1 Vertical query line segments

We can use an approach similar to the one at the beginning of Section 3.2. However, since we are now dealing with line segments rather than lines, we must overcome a subtle problem that can arise. We discuss this later. We draw vertical lines through the endpoints and the intersection points of the segments of S. Within any strip, the segments that cross it can be totally ordered. We sweep over the strips starting at the leftmost non-empty strip. Let $s_1, s_2, \ldots, s_m$ be the segments that cross this strip, sorted from bottom to top. For $1 \leq i \leq m$, we give s_i a label $l(s_i) = i$ and give this label the color of s_i. We store the segments $s_1, \ldots, s_m$ in this order in a partially persistent red-black tree T_S. We also store the colored labels $l(s_i)$, $1 \leq i \leq m$, in a partially persistent version T_l of the data structure of [GJS93a] for the generalized 1-dimensional range reporting problem.

Suppose we sweep from the ith to the $(i+1)$th strip. The cases where we encounter a right endpoint of a segment or the intersection point of two segments are easy to handle [GJS93c]. If we encounter the left endpoint of segment s, then in the current version of T_S, we locate s. Let t and u be the segments that are immediately below and above s in the $(i+1)$th strip. We insert s into the current version of T_S and store with it a label $l(s)$ that lies between $l(t)$ and $l(u)$. Moreover, we give the label $l(s)$ the same color as s and insert this colored number into the current version of T_l.

In this scheme, we need to assign special labels to the segments as we encounter them because all the segments are not present in each strip. However, we must be careful in choosing the labels since otherwise we may end up getting labels consisting of $\Theta(n)$ bits. Towards this end, we use a labeling scheme due to Dietz and Sleator [DS87]. Using their approach we take integer labels in the range $[0..O(n^2)]$, i.e., labels consisting of only $O(\log n)$ bits. We need to give segment s a label that lies in between $l(t)$ and $l(u)$. Using the scheme of [DS87], this may result in the relabeling of other segments. Dietz and Sleator show how to choose the labels such that only $O(1)$ amortized relabelings are necessary per update. If we relabel segment s from $l(s)$ to $l'(s)$, then we just delete the colored number $l(s)$ from T_l and insert the number $l'(s)$, having the same color as $l(s)$, into it.

Now let q be a vertical query segment. We locate the strip containing q and then search in the version of T_S corresponding to this strip for the lowest and highest segments s and t that intersect q. Finally, we search in the version of T_l corresponding to this strip for the distinct colors of all labels that are contained in the interval $[l(s), l(t)]$.

Theorem 3.3 *A set S of n colored line segments*

in the plane can be preprocessed into a data structure of size $O((n+\chi)\log n)$ such that the i distinct colors of the segments intersected by a vertical query line segment q can be reported in $O(\log n + i)$ time. Here χ, $0 \leq \chi \leq \binom{n}{2}$, is the number of pairwise intersections among the segments in S. □

3.3.2 Arbitrary query line segments

If q is not vertical, then we use a different approach which we sketch briefly. We break up the input segments at their χ intersection points and store them in a segment tree T. For any node $v \in T$, the set $S(v)$ of segments allocated to v can be totally ordered within $Strip(v)$ as $s_1, \ldots, s_m$, from bottom to top. Suppose $q = \overline{ab}$ is such that $a, b \in Strip(v)$, with a below b. Let s_u be the lowest segment of $S(v)$ that is above a; define s_w symmetrically w.r.t. b. Clearly it suffices to report the distinct colors in $\{s_u, s_{u+1}, \ldots, s_w\}$, which we can solve using a structure for generalized 1-dimensional range searching. If $q = [a, b]$ is not as above, then we can identify a set V of $O(\log n)$ nodes in T such that for each $v \in V$, the segment $q_v = q \cap Strip(v)$ is as above. We simply query at each such v with q_v.

Theorem 3.4 *A set S of n colored line segments in $\mathcal{R}^2$ can be preprocessed into a data structure of size $O((n+\chi)\log n)$ so that the i distinct colors of the segments that are intersected by a query segment q can be reported in $O(\log^2 n + i \log n)$ time. Here χ, $0 \leq \chi \leq \binom{n}{2}$ is the number of pairwise intersections between segments in S.* □

4 Generalized triangle stabbing

We consider the following problem: "Preprocess a set S of n colored triangles in $\mathcal{R}^2$, so that the i distinct colors of the triangles stabbed by any query point q can be reported efficiently."

Wlog assume that each triangle $t \in S$ has a horizontal side. We group the triangle vertices into $\Theta(n^{1/2})$ vertical strips each of size $\Theta(n^{1/2})$. The triangles that intersect any strip V_i form two disjoint subsets T_i and T_i', where T_i (resp. T_i') consists of triangles having no (resp. at least one) vertex inside V_i. We further subdivide V_i into vertical strips by taking each triangle in T_i' and drawing a vertical line through each of its vertices that lies in V_i. Let W_{ij} be any such substrip within V_i and let T_{ij} consist of the triangles of T_i' that cross W_{ij}. Note that, by construction, no triangle of T_{ij} can have a vertex inside W_{ij}. (This partitioning technique is reminiscent of a method used in [OY88] for computing the measure of the union of iso-oriented boxes in $\mathcal{R}^d$.)

Given the query point q, suppose that $q \in V_i$ and $q \in W_{ij}$. Then we need to only report the distinct colors of the triangles of T_i and of T_{ij} that are stabbed by q. We discuss how to do this for V_i and T_i. (The discussion for W_{ij} and T_{ij} is similar.)

For any triangle $t \in T_i$, let $p(t)$ be the vertex of t shared by the horizontal side $h(t)$ of t and the slanted side $s(t)$ of t which crosses V_i. Let $w(t)$ be the wedge defined by $p(t)$ and (the extensions of) $h(t)$ and $s(t)$. Consider the set $R_i \subseteq T_i$ that yields right-facing wedges. Clearly, q stabs $t \in R_i$ iff the horizontal, leftward directed ray from q intersects $w(t)$. We can now solve the resulting generalized ray-rightwedge intersection problem by computing left envelopes and using a segment tree (as in Section 3.1). Symmetrically for left-facing wedges.

Theorem 4.1 *A set S of n colored triangles in $\mathcal{R}^2$ can be stored in a data structure of size $O(n^{3/2}\log n)$ such that the i distinct colors of the triangles that are stabbed by a query point q can be reported in $O(\log^2 n + i)$ time.* □

5 Generalized triangle range searching

We wish to preprocess a set S of n colored points in $\mathcal{R}^2$ so that the i distinct colors of the points contained in a query triangle q can be reported efficiently. For arbitrary q we note that the problem can be solved in $O(n^{2+\epsilon})$ space and $O(\log n + i)$ query time using constant-depth cutting trees. In the rest of this section, we give an efficient solution for query triangles that are fat. We define a *fat-triangle* to be a triangle in which each internal angle is at least γ for some constant $\gamma > 0$.

We begin with a solution for a query *fat-wedge* q. i.e., a wedge where the internal angle at the vertex v_q of q is at least γ. For now assume that q is y-monotone. We store the points of S at the leaves of a balanced binary search tree T by non-decreasing

y-coordinates from left to right. We augment each node v of T with an instance $HP(v)$ of the structure of Theorem 2.1 for generalized halfplane range reporting; $HP(v)$ is built on the points in v's descendant leaves.

Given q, we divide it into an upper wedge q_a and a lower wedge q_b, by drawing a horizontal line L through v_q. Consider q_a. Let l_a be the line containing the slanted side of q_a. We search in T using the y-coordinate of v_q and determine a set V_a of nodes that lie to the right of the search path. We query $HP(v)$ at each $v \in V_a$ with the halfplane l_a^-. Symmetrically for q_b. It can be shown that the space used is $O(n \log^2 n)$ and the query time is $O(\log^3 n + i \log n)$.

What if q is not y-monotone? In preprocessing, we select $t = \lceil 2\pi/\gamma \rceil$ coordinate systems $\mathcal{C}_i = (x_i y_i)$, where all the $\mathcal{C}_i$ share the same origin and $\mathcal{C}_{i+1}$ is offset from $\mathcal{C}_i$ by an angle γ, $0 \leq i \leq t-1$ (indices are modulo t.) Within each $\mathcal{C}_i$ we build an instance of the data structure for y_i-monotone fat-wedges. Given a query fat-wedge q, we locate a $\mathcal{C}_i$ such that q is y_i-monotone and then query the associated structure. (It is easily seen that such a $\mathcal{C}_i$ exists.)

Now consider the case of a query fat-triangle q. We store the points by non-decreasing x-coordinates from left to right in a balanced search tree T' and augment each node v with an instance $FW(v)$ of the structure given above for fat-wedges, which is built on the points in v's descendant leaves. Given q, we divide it into two triangles q_l and q_r, each with a vertical side s, with q_l to the left of s and q_r to the right. We search in T' with the x-coordinate of s and identify a set V_l of nodes that lie to the left of the search path. For each node $v \in V_l$, we query $FW(v)$ with the wedge supporting q_l, which is a fat-wedge. Symmetrically for q_r.

Theorem 5.1 *Let $\gamma > 0$ be a constant. A set S of n colored points in $\mathcal{R}^2$ can be stored in a data structure of size $O(n \log^3 n)$ such that the i distinct colors of the points that are contained in a query triangle q each of whose internal angles is at least γ can be reported in time $O(\log^4 n + i \log^2 n)$.* □

6 Conclusions and further work

We have presented efficient solutions to several generalized intersection problems involving non-iso-oriented objects. Our methods have included sparse representations, persistence, and filtering search.

Besides improving our bounds, three problems are of particular interest: (i) obtaining dynamic data structures for the generalized problems considered here, (ii) obtaining linear-space or near linear-space solutions with output-sensitive query times (of the form $O(n^p+i)$ or $O(n^p+i \cdot polylog(n))$, $0 < p < 1$) for the generalized halfspace range searching problem in $d \geq 4$ dimensions and for the generalized simplex range searching problem in $d \geq 2$ dimensions, and (iii) solving the counting versions of Problems 3–7 in Table 1.

References

[AHL90] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 331–340, 1990.

[AvK93] P.K. Agarwal and M. van Kreveld. Connected component and simple polygon intersection searching. In *Proceedings of the 1993 Workshop on Algorithms and Data Structures*, pages 36–47, August 1993.

[AvKO93] P.K. Agarwal, M. van Kreveld, and M. Overmars. Intersection queries for curved objects. *Journal of Algorithms*, 15:229–266, 1993.

[CGL85] B.M. Chazelle, L.J. Guibas, and D.T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[DS87] P.F. Dietz and D.D. Sleator. Two algorithms for maintaining order in a list. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 365–372, 1987.

[DSST89] J.R. Driscoll, N. Sarnak, D.D. Sleator, and R.E. Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38:86–124, 1989.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer–Verlag, 1987.

[GJS93a] P. Gupta, R. Janardan, and M. Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. In *Proceedings of the 1993 Workshop on Algorithms and Data Structures*, pages 361–372, August 1993.

[GJS93b] P. Gupta, R. Janardan, and M. Smid. On intersection searching problems involving curved objects. Technical Report TR–93–42, Dept. of Computer Science, University of Minnesota, 1993. Submitted.

[GJS93c] P. Gupta, R. Janardan, and M. Smid. Efficient algorithms for generalized intersection searching with non-iso-oriented objects. Technical Report TR–93–73, Dept. of Computer Science, University of Minnesota, 1993. Submitted.

[JL93] R. Janardan and M. Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry & Applications*, 3:39–69, 1993.

[Mat91a] J. Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6:385–406, 1991.

[Mat92a] J. Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[Mat92b] J. Matoušek. Range searching with efficient hierarchical cuttings. In *Proceedings of the 8th Annual ACM Symposium on Computational Geometry*, pages 276–285, 1992.

[Mul93] K. Mulmuley. *Computational Geometry: An introduction through randomized algorithms.* Prentice–Hall, 1993.

[OY88] M.H. Overmars and C.K. Yap. New upper bounds in Klee's measure problem. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 550–556, 1988.