Computing Faces in Segment and Simplex Arrangements

(Preliminary Version)

NANCY M. AMATO[*]
Texas A&M Univ.
amato@cs.tamu.edu

MICHAEL T. GOODRICH[†]
Johns Hopkins Univ.
goodrich@cs.jhu.edu

EDGAR A. RAMOS[‡]
Univ. of Illinois
ramos@cs.uiuc.edu

Abstract

For a set S of n line segments in the plane, we give the first work-optimal deterministic parallel algorithm for constructing their arrangement. It runs in $O(\log^2 n)$ time using $O(n \log n + k)$ work in the EREW PRAM model, where k is the number of intersecting line segment pairs, and provides a fairly simple divide-and-conquer alternative to the optimal sequential "plane-sweep" algorithm of Chazelle and Edelsbrunner. Moreover, our method can be used to output all k intersecting pairs while using only $O(n)$ working space, which solves an open problem posed by Chazelle and Edelsbrunner. We also describe a sequential algorithm for computing a single face in an arrangement of n line segments that runs in $O(n\alpha^2(n) \log n)$ time, which improves on a previous $O(n \log^2 n)$ time algorithm.

For collections of simplices in $\mathbb{R}^d$, we give methods for constructing a set of $m = O(n^{d-1} \log^c n + k)$ cells of constant descriptive complexity that covers their arrangement, where $c > 1$ is a constant and k is the number of faces in the arrangement. The construction is performed sequentially in $O(m)$ time, or in $O(\log n)$ time using $O(m)$ work in the EREW PRAM model. The covering can be augmented to answer point location queries in $O(\log n)$ time. In addition to supplying the first parallel methods for these problems, we improve on the previous best sequential methods by reducing the query times (from $O(\log^2 n)$ in $\mathbb{R}^3$ and $O(\log^3 n)$ in $\mathbb{R}^d$, $d > 3$), and also the size and construction cost of the covering (from $O(n^{d-1+\epsilon} + k)$).

1 Introduction

Geometric sampling and its geometric counterparts have proven to be very powerful tools in computational geometry for designing efficient sequential algorithms and data structures. In this paper we use geometric sampling techniques to obtain improved solutions (sequential and parallel) to some important problems in computational geometry[1] An attractive feature of our algorithms, and in fact, of many geometric sampling algorithms, is that they are very simple. This is even true of our deterministic algorithms (with the exception of the derandomization step itself), in spite of the fact that they must deal with a problem often encountered in geometric sampling algorithms: namely, that the total size of the subproblems may grow by a constant factor with each recursive application of the sampling. In our line segment algorithms we use "pruning" computations to keep the sizes of the subproblems within certain bounds.

Segment arrangements. Let S be a set of n line segments in the plane, and let k be their number of pairwise intersections. The *segment intersection* problem is to report all k intersecting segment pairs in S, with a slightly more difficult variant being that of constructing a triangulation of the arrangement of the segments in S. This problem has been studied extensively in the computational geometry literature [6, 8]. Chazelle and Edelsbrunner [11] gave an optimal method for computing segment intersections and constructing their arrangement that runs in $O(n \log n + k)$ time and uses a number of beautiful techniques, including plane sweeping and topological sweeping. A number of researchers [12, 17, 33, 34], have given elegant randomized methods that run in $O(n \log n + k)$ expected time. In fact, if $k \geq n \log^{1+\delta} n$ for some constant $\delta > 0$, then these methods run in this bound with high probability[2] [31]. In the parallel domain, Clarkson, Cole, and Tarjan [16, 15] show that one can construct a segment arrangement in parallel in $O(\log n)$ time and $O(n \log n + k)$ expected work in the CRCW PRAM model.[3]

There is no previous deterministic optimal-work parallel algorithm for the general segment intersection problem, however. The best previous methods for the general problem are a method of Goodrich [20], which runs in $O(\log n)$ time and $O(n \log^2 n + k \log n)$ work in the CREW PRAM model and a method of Rüb, which runs in $O(\log n \log\log n)$ time using $O((n + k) \log n \log\log n)$ work in the same parallel model. One can achieve an optimal $O(n \log n + k)$ work bound, however, for some special cases [20, 22, 23, 39].

We show how to solve the problem of computing a segment arrangement in $O(\log^2 n)$ time and $O(n \log n + k)$ work

[*]This research supported in part by an AT&T Bell Laboratories Graduate Fellowship and by NSF Grant CCR-9315696. A portion of this effort was done while this author was visiting the International Computer Science Institute, Berkeley, CA.

[†]This research supported by the NSF under Grants IRI-9116843 and CCR-9300079.

[‡]This research supported in part by NSF Grant CCR-9118874.

[1]For background material on geometric sampling, see [3, 14, 28, 34].

[2]We say that an event parameterized by n holds "with high probability" if its probability is at least $1 - 1/n^\delta$ for some constant $\delta > 0$.

[3]The CRCW PRAM is the synchronous shared-memory model that allows for concurrent reads and concurrent writes—which in this instance can be resolved arbitrarily. The CREW PRAM allows for concurrent reads but requires writes to be exclusive, and the EREW PRAM requires both reads and writes to be exclusive.

in the EREW PRAM model (augmented with processor allocation calls [20]). Our method also provides an optimal sequential divide-and-conquer alternative to the plane-sweep method of Chazelle and Edelsbrunner [11], and uses $O(n)$ space if only required to output the intersections, which solves an open problem of Chazelle and Edelsbrunner[4].

A single face in a segment arrangement. Again, let S be a set of n line segments in the plane and let p be a point. In the single face problem, we are interested in computing the face of the arrangement of S containing p, that is, the connected component of $\mathbb{R}^2 - \bigcup_{s \in S} s$ that contains p. It is known that the boundary complexity of a single face is $O(n\alpha(n))$ where $\alpha(n)$ is the very slowly-growing inverse of Ackerman's function. The best previously known deterministic algorithm is an $O(n \log^2 n)$ method due to Mitchell [32], yet there are randomized ones that run in expected time $O(n\alpha(n) \log n)$ [12, 18]. The known lower bound is $\Omega(n \log n)$.

We describe an almost-optimal deterministic algorithm, in that it runs in time $O(n\alpha^2(n) \log n)$. The algorithm uses a divide-and-conquer approach based on deterministic geometric sampling, together with a pruning mechanism to avoid a blow-up in the total size of the subproblems as the computation progresses.

Simplices in higher dimensions. Let S be a set of n $(d-1)$-simplices in $\mathbb{R}^d$, for any fixed $d \geq 3$, e.g., triangles in $\mathbb{R}^3$, or tetrahedra in $\mathbb{R}^4$. When the simplices in S may intersect, the complexity k of $\mathcal{A}(S)$ can vary between n and n^d. For $d = 3$, de Berg, Guibas, and Halperin [19] build a *vertical decomposition* D of $\mathcal{A}(S)$ of size $O(n^{2+\alpha} + k)$ in $O(n^{2+\alpha} + |D| \log n)$ time, for any constant $\alpha > 0$; it supports point location queries in $O(\log^2 n)$ time. For $d \geq 3$, Pellegrini [35] constructs a *covering*[5] for $\mathcal{A}(S)$ of size $O(n^{d-1+\alpha} + k)$ in $O(n^{d-1+\alpha} + k)$ time, for any constant $\alpha > 0$. Using the same space, but with $O(n^{d-1+\alpha} + k \log n)$ work, the covering can be augmented to support point location queries in $O(\log^3 n)$ time. We know of no parallel methods for processing $\mathcal{A}(S)$ when $d \geq 3$.

We describe methods for constructing a covering for $\mathcal{A}(S)$ of size $m = O(n^{d-1} \log^{O(1)} n + k)$, which can be augmented to support point location queries in $O(\log n)$ time using the same storage. In the EREW PRAM model, the covering is constructed in $O(\log n)$ time using $O(m)$ work, and the point location structure in $O(\log^2 n)$ time using $O(m + k \log n)$ work. Thus, in addition to supplying the first parallel methods, we improve on the best known sequential results [19, 35] by reducing the query time, and also the size and construction cost of the covering. When the simplices in S are interior disjoint, e.g., non-simple polyhedra, a triangulation of $\mathcal{A}(S)$ of size $O(n^{d-1})$ can be built in $O(\log n)$ time using $O(n^{d-1})$ work in the EREW PRAM model, matching the sequential result of Pellegrini [35].

[4]Recently, this has also been solved by I. Balaban [5] with an entirely different approach. His algorithm, however, does not seem to parallelize nor to be adaptable to computing the arrangement.

[5]A *covering* is a set of cells of constant descriptive complexity whose union contains $\mathcal{A}(S)$. Unlike decompositions composed of interior disjoint cells, such as triangulations, the cells in a covering may overlap.

Contents of the paper. Section 2 contains the geometric sampling results needed in our algorithms for segments. Sections 3 and 4 contain the algorithms for all faces and single face in an arrangement of segments respectively. Section 5 contains the results for simplices in higher dimensions.

2 Geometric sampling

Let $\mathcal{X}$ be a class of geometric objects in $\mathbb{R}^d$ (for example all line segments in the plane), and let $X \subseteq \mathcal{X}$ be of size n. The *arrangement* of X is the collection of connected components of $\mathbb{R}^d - \bigcup_{x \in X} x$, together with the arrangements on each $x \in X$ of $X - \{x\}$ restricted to x. In general, we are interested in a particular subset $\mathcal{A}(X)$ of the arrangement, and its decomposition $\mathcal{T}(X)$ into a collection of cells of constant complexity. Let $R \subseteq X \subseteq \mathcal{X}$. For a cell $\sigma \in \mathcal{T}(R)$, let the *triggers* of σ, denoted as $\Delta(\sigma)$, be the set of objects in X that determine σ, and let the *killers* of σ, denoted as X_σ, be the set of objects in X that intersect σ, which we also call the *conflict list* of σ. We restrict our attention to classes where $|\Delta(\sigma)| \leq D$ for some constant $D \geq 1$.

In some applications $(\mathcal{X}, \mathcal{T})$ satisfies a property called *locality*: for $R \subseteq X \subseteq \mathcal{X}$, $\sigma \in \mathcal{T}(R)$ iff $\Delta(\sigma) \subset R$ and $X_\sigma \cap R = \emptyset$. This is the case, for example, for the complete arrangement. In some applications where one is interested only in a subset of the arrangement, for example a single face in an arrangement, locality fails. However, it has been noted [1, 18] that in this case a property that we call *monotonicity* holds: for $R, R' \subseteq X$, if $\sigma \in \mathcal{T}(R)$ then $\Delta(R) \subseteq R$ and $X_\sigma \cap R = \emptyset$; and if $\sigma \in \mathcal{T}(R)$ and $R' \subset R$ with $\Delta(\sigma) \subset R'$, then $\sigma \in \mathcal{T}(R')$. Note that locality implies monotonicity.

Polynomial construction

The basis for our divide-and-conquer approach is the geometric sampling theorem below. We state it in some generality, and then specialize it to our particular problem. For completeness, we give a detailed proof. It uses standard techniques in geometric sampling (see e.g. [1, 4, 17, 21, 29]).

For sequential computation, the r^δ factor in property (ii) of the theorem can be improved to $\log r$ using derandomization by the method of conditional probabilities. However, we prefer our form for three reasons: the method of limited independence is better suited for parallelization; even sequentially there is no apparent way to derandomize using the method of conditional probability when locality does not apply; and finally the method of limited independence is computationally simpler.

We need the following tail estimate for random variables with limited independence (see [38, 34]).

Lemma 2.1 *Let $I = \sum_{j=1}^{m} I_m$ be the sum of m $2K$-wise independent (K fixed), identical 0-1 random variables (i.e. each variable is 1 with probability $p > 0$ and 0 otherwise). Let $\mu = \mathbf{E}[I] = mp$. Then, for some constant $C > 0$, Prob$\{|I - \mu| \geq \mu/t\} \leq C(t^{2K}/\mu^K)$. In particular Prob$\{I = 0\} \leq C(1/\mu^K)$.*

A p-sample of X is a sample R obtained by n identical independent 0-1 (characteristic) random variables, each of which is 1 with probability p. Let $f(X,r) = \mathbf{E}[|\mathcal{T}(R)|]$ for a p-sample with $p = r/n$. A K-wise independent p-sample and $f_K(X,r)$ are defined similarly for K-wise independent 0-1 random variables. In the cases we consider, K is sufficiently large so that $f_K(X,r) = O(f(X,r))$. Our way of dealing with monotonicity follows the analysis in [1].

Theorem 2.2 *Let $(\mathcal{X}, \mathcal{T})$ be such that it satisfies the monotonicity property, and that, for $X \subseteq \mathcal{X}$, $f(X, r/t) = O(f(X,r))$ for $t \geq 1$. For $0 < \delta < 1$, and $c \geq 1$, there are constants c', r_0 such that for $X \subseteq \mathcal{X}$ with $|X| = n$ and $r_0 \leq r \leq n$, a sample $R \subseteq X$ with the following properties can be computed in polynomial time sequentially, more precisely in time $O(n^{2K+D+1})$ where $K = \lceil \max(c+2+3D/2, (D+1)/\delta + D/2) \rceil$: (i) $||R| - 3r/2| \leq r$, (ii) $\max_{\sigma \in \mathcal{T}(R)} |X_\sigma| \leq (n/r)r^\delta$, (iii) $\sum_{\sigma \in \mathcal{T}(R)} |X_\sigma|^c \leq c'(n/r)^c f(X,r)$. In the EREW model the computation can be performed in time $O(\log n)$ and work proportional to the sequential time.*

Proof: Let $R \subseteq X$ be a $2K$-wise independent p-sample with $r/n < p \leq 2r/n$. In the argument for (iii) when monotonicity but not locality holds, we need to make use of a particular construction of the sample space for the 0-1 random variables $I_1, \ldots, I_n$ that determine R. We use the construction of Joffe [26]: Let η be a prime number with $n \leq \eta < 2n$; the sample space is $\Omega(\eta, 2K) = Z_\eta^{2K}$. For $1 \leq i \leq \eta$ and $(a_0, \ldots, a_{2K-1}) \in \Omega(\eta, 2K)$, let $X_i = \sum_{j=0}^{2K-1} a_j i^j \bmod \eta$ and let $I_i = 1$ if $0 \leq X_i < 2r$. The 0-1 random variables $I_1, \ldots, I_\eta$ defined by giving each vector of $\Omega(\eta, 2K)$ probability $1/\eta^{2K}$, are $2K$-wise independent and $p :=$ Prob$\{I_i = 1\}$ satisfies $r/n < p \leq 2r/n$.

We claim that R satisfies (i)-(iii) with probability at least $1/2$. To show this, we verify that each of (i)-(iii) fails with probability at most $1/6$. For (i), this follows for r_0 sufficiently large by Lemma 2.1 (the expected value is between r and $2r$ depending on the number η in sample space construction). The argument for (ii) and (iii) is as follows.

Let $\mathbf{T}(X) = \{\sigma \in \mathcal{T}(R) : R \subseteq X\}$. Let A_σ be the event that $\Delta(\sigma) \subseteq R$ and $X_\sigma \cap R = \emptyset$. For $\sigma \in \mathbf{T}(X)$, let $t_\sigma = |X_\sigma|(r/n)$. $p_\sigma :=$ Prob$\{A_\sigma\}$ is equal to $p_{\sigma,1} \cdot p_{\sigma,2}$ where $p_{\sigma,1} =$ Prob$\{X_\sigma \cap R = \emptyset | \Delta(\sigma) \subseteq R\}$ and $p_{\sigma,2} =$ Prob$\{\Delta(\sigma) \subseteq R\}$. By independence $p_{\sigma,2} = p^{|\Delta(\sigma)|}$, since $2K \geq D$. By Lemma 2.1, $p_{\sigma,1} \leq C'(1/t_\sigma^{K-D/2})$ (after fixing at most D variables, the remaining ones are $(2K-D)$-wise independent). Thus, since Prob$\{\sigma \in \mathcal{T}(R)\} \leq p_\sigma$ (equality holds if there is locality),

$$
\begin{aligned}
1 - \text{Prob}\{\max_{\sigma \in \mathcal{T}(R)} |X_\sigma| \leq (n/r)r^\delta\} &= \\
= \sum_{\sigma \in \mathbf{T}(X)} \text{Prob}\{\sigma \in \mathcal{T}(R)\} \cdot [|X_\sigma| > (n/r)r^\delta] \\
\leq C'(1/r^{\delta(K-D/2)}) \sum_{\sigma \in \mathbf{T}(X)} p_{\sigma,2} \\
< C''(1/r^{\delta(K-D/2)}) r^D < 1/6,
\end{aligned}
$$

if $K > (D+1)/\delta + D/2$ and $r \geq r_0$ is sufficiently large (we have used $\sum_{\sigma \in \mathbf{T}(X)} p_{\sigma,2} = O(r^D)$). This verifies the claim for (ii).

For $t = 1, \ldots, r$, let $\mathbf{T}_t(X) = \{\sigma \in \mathbf{T}(X) : (t-1)(n/r) \leq |X_\sigma| < t(n/r)\}$. Let R_t be a $2K$-wise independent p_t-sample with $r/4nt < p_t \leq r/2nt$. Using the sample space described above, the corresponding 0-1 random variables $I_1^{(t)}, \ldots, I_n^{(t)}$ are defined by $I_i^{(t)} = 1$ if $0 \leq X_i < \lfloor r/4t \rfloor$. For $\sigma \in \mathbf{T}(X)$, let $A_\sigma^{(t)}$, $p_\sigma^{(t)}$, $p_{\sigma,1}^{(t)}$ and $p_{\sigma,2}^{(t)}$ be defined for R_t analogously to the same terms for R above. We want to show that for $\sigma \in \mathbf{T}_t(X)$, Prob$\{\sigma \in \mathcal{T}(R)\}$/Prob$\{\sigma \in \mathcal{T}(R_t)\} < C'/(t-1)^{K-3D/2}$ for $t > 1$. For this we first argue that Prob$\{\sigma \in \mathcal{T}(R)\}$/Prob$\{\sigma \in \mathcal{T}(R_t)\} \leq p_\sigma/p_\sigma^{(t)}$: this follows by monotonicity since our specific construction of 0-1 variables implies that Prob$\{\sigma \in \mathcal{T}(R) | A_\sigma\} \leq$ Prob$\{\sigma \in \mathcal{T}(R_t) | A_\sigma^{(t)}\}$ and hence Prob$\{\sigma \in \mathcal{T}(R)\}/p_\sigma \leq$ Prob$\{\sigma \in \mathcal{T}(R_t)\}/p_\sigma^{(t)}$. Then, we verify that $p_\sigma/p_\sigma^{(t)} \leq C'/(t-1)^{K-3D/2}$: this follows because $p_{\sigma,1} < C''/(t-1)^{K-D/2}$ for $t > 1$ by Lemma 2.1, $p_{\sigma,1}^{(t)} \geq 1/2$ because $\mathbf{E}[|R_t \cap X_\sigma| \,|\Delta(\sigma) \subseteq R_t] \leq 1/2$, and $p_{\sigma,2}/p_{\sigma,2}^{(t)} \leq (8t)^{|\Delta(\sigma)|} \leq (8t)^D$. Thus, $\mathbf{E}[\sum_{\sigma \in \mathcal{T}(R)} |X_\sigma|^c]$ equals

$$
\begin{aligned}
& \sum_{\sigma \in \mathbf{T}(X)} \text{Prob}\{\sigma \in \mathcal{T}(R)\} \cdot |X_\sigma|^c \\
\leq\ & \left(\frac{n}{r}\right)^c \sum_{\sigma \in \mathbf{T}_1(X)} \text{Prob}\{\sigma \in \mathcal{T}(R)\} + \\
& \sum_{t=2}^{r} \left(\frac{tn}{r}\right)^c \frac{C'}{(t-1)^{K-3D/2}} \sum_{\sigma \in \mathbf{T}_t(X)} \text{Prob}\{\sigma \in \mathcal{T}(R_t)\} \\
\leq\ & \left(\frac{n}{r}\right)^c f(X, 2r) + \\
& C'' \left(\frac{n}{r}\right)^c \sum_{t=2}^{r} \frac{1}{(t-1)^{K-c-3D/2}} f(X, r/2t) \\
<\ & C'''(n/r)^c f(X, 2r),
\end{aligned}
$$

if $K \geq c + 3D/2 + 2$, where we have used $\sum_{\sigma \in \mathbf{T}(X)}$ Prob$\{\sigma \in \mathcal{T}(R_t)\} = f(X, r/t) = O(f(X,r))$. By Markov's inequality, it follows that there is a c' so that (iii) holds with probability at most $1/6$.

The polynomial time claim follows because the probability space has size $O(n^{2K})$, and each point in it can be tested in time $O(n^{D+1})$ with a not so efficient algorithm (it is guaranteed that a sample satisfying (i)-(iii) will be found). The parallelization is straightforward, in particular, each of the candidate samples can be tested in parallel. ■

For definiteness, we will use Theorem 2.2 with $\delta = 1/2$.

Efficient construction

For our purposes, we need a faster construction of a sample R. This can be achieved using ϵ-approximations which can be constructed efficiently using a technique of Matoušek [30].

Approximations. A pair $(X, \mathcal{F})$, where $X \subseteq \mathcal{X}$ is as before and $\mathcal{F}$ is a family of subsets of $\mathbb{R}^d$ defines a *range space* $(X, \mathcal{R}(X, \mathcal{F}))$ where $\mathcal{R}(X, \mathcal{F}) = \{T \subseteq X : T = \{x \in X : \gamma \cap x \neq \emptyset\}$ for $\gamma \in \mathcal{F}\}$. $A \subseteq X$ is a $(1/r)$-approximation for $(X, \mathcal{R}(X, \mathcal{F}))$ if for each $T \in \mathcal{R}(X, \mathcal{F})$, $||A \cap T|/|A| - |T|/|X|| \leq 1/r$. Matoušek's construction is originally for the range spaces defined by the pairs $(P, \mathcal{H}(\mathbb{R}^d))$, where P is a finite point set in $\mathbb{R}^d$ and $\mathcal{H}(\mathbb{R}^d)$ is the collection of closed half-spaces in $\mathbb{R}^d$, and is easily extended to the pairs $(P, \mathcal{L}(\mathbb{R}^d))$ where $\mathcal{L}(\mathbb{R}^d)$ is the collection of *linear cells* in $\mathbb{R}^d$. A linear cell is the union of $O(1)$ intersections of each of $O(1)$ half-spaces. A parallelization is due to Goodrich [21].

Lemma 2.3 *There is an $0 < \epsilon < 1$, such that for a given $P \subseteq \mathbb{R}^d$ with $|P| = n$ and $r \leq n^\epsilon$, a $(1/r)$-approximation of size $O(r^2 \log r)$ for $(P, \mathcal{R}(P, \mathcal{L}(\mathbb{R}^d)))$ can be constructed in time $O(n \log r)$. In the EREW PRAM model the construction takes time $O(\log n \log r)$ using work $O(n \log r)$.*

Here, we are interested in the pairs $(X, \mathbf{T}(X))$ and $(X, \mathbf{T}(\mathcal{X}))$, where $\mathbf{T}(\mathcal{X}) = \{\sigma \in \mathcal{T}(X) : X \subseteq \mathcal{X},\ |X| \leq D\}$. A standard trick to obtain a good r-sample for X efficiently consists in first computing a $(1/r)$-approximation A for $(X, \mathcal{R}(X, \mathbf{T}(X)))$ in time $O(n \log r)$ and then compute a good $(r/|A|)$-sample for A in time polynomial in $|A|$. With $r \leq n^\epsilon$ and ϵ sufficiently small, the total time is still $O(n \log r)$.

Linearization. To apply Matoušek's construction for approximations, and in order to compute the conflict lists of $\mathcal{T}(R)$ efficiently, we make use of *linearization* for $(\mathcal{X}, \mathbf{T}(\mathcal{X}))$ (first described by Yao and Yao [41], and introduced in geometric sampling by Matoušek [27, 29]). The pair $(\mathcal{X}, \mathcal{F})$ is linearizable if there are maps φ from $\mathcal{X}$ into $\mathbb{R}^d$, and ψ from $\mathcal{F}$ into $\mathcal{L}(\mathbb{R}^d)$ such that for $x \in \mathcal{X}$ and $\sigma \in \mathcal{F}$, $x \cap \sigma \neq \emptyset$ iff $\varphi(x) \in \psi(\sigma)$. The mapping φ is given by bounded degree polynomials in the parameters defining an object $x \in X$; the functions describing the coefficients in the equations of the hyperplanes defining $\psi(\sigma)$ are bounded degree polynomials in the parameters defining σ.

After linearization, an approximation can be constructed using directly the technique of Matoušek in time $O(n \log r)$ for $r \leq n^\epsilon$.

Computing conflict lists. Linearization also simplifies the problem of computing the conflict lists X_σ for $\sigma \in \mathcal{T}(R)$: this is translated into a problem of point location queries in an arrangement of hyperplanes in $\mathbb{R}^d$ (see for example [29]).

Let H be the collection of all the hyperplanes bounding the linear cells $\psi(\sigma)$ for $\sigma \in \mathcal{T}(R)$. $|H|$ is $O(r^D)$. We can construct a point location data structure $\mathcal{D}$ for H to achieve an $O(\log n)$ query time with size $O(|H|^D)$ [9], but we can also use the following less-efficient construction: For each leaf l of $\mathcal{D}$ (its number is $O(|H|^D)$), determine the list $L(l)$ of each cell $\sigma \in \mathcal{T}(R)$ for which $\psi(\sigma)$ contains the cell in the arrangement of H corresponding to l. (The time (and space) used in this construction can be made $O(n)$ by choosing ϵ appropriately.) To compute the conflict lists, first, for each object x, perform a search for $\varphi(x)$ in $\mathcal{D}$ and let $l(x)$ be the leaf where this search "lands". Then, we collect , for each cell $\sigma \in \mathcal{T}(R)$, the objects x for which $L(l(x))$ contains σ. This can be done in time linear in the total size of the conflict lists. Parallel versions of these point location data structures are also known [4].

To summarize, then, we have the following:

Theorem 2.4 *Let $(\mathcal{X}, \mathcal{T})$ be such that it satisfies the monotonicity property and $(\mathcal{X}, \mathbf{T}(\mathcal{X}))$ is linearizable. For $0 < \delta < 1$, there are constants c', r_0, ϵ such that for $X \subseteq \mathcal{X}$ with $|X| = n$ and $r_0 \leq r \leq n^\epsilon$, a sample $R \subseteq X$ can be constructed in time $O(n \log r)$ by first obtaining a $(1/r)$-approximation A for $(X, \mathbf{T}(X))$ (or $(X, \mathbf{T}(\mathcal{X}))$) and then an $(r/|A|)$-sample for A according to Theorem 2.2, so that it satisfies the properties: (i) $||R| - 3r/2| \leq r$, (ii) $\max_{\sigma \in \mathcal{T}(R)} |X_\sigma| \leq (n/r) r^\delta$, (iii) $|\mathcal{T}(R)| \leq c' f(A, r)$, (iv) $\sum_{\sigma \in \mathcal{T}(R)} |X_\sigma| \leq c'(n/r) f(A, r)$. The triangulation $\mathcal{T}(R)$ and conflict lists can be computed in time $O(n \log r + (n/r) f(A, r))$. The construction takes time $O(\log n \log r)$ and optimal $O(n \log r)$ work in the EREW PRAM model.*

Segments and trapezoids

In our applications $\mathcal{X} = \mathcal{S}$ is the class of segments in the plane. The arrangement $\mathcal{A}(S)$ for $S \subseteq \mathcal{S}$ consists of the faces, edges and vertices determined by S. The *visibility diagram* of S is obtained by extending each endpoint and intersection point vertically until it hits another segment, each connected component of the complement of the visibility diagram is called a *trapezoid*, and the *trapezoidal decomposition* $\mathcal{T}(S)$ of S is the collection of all such trapezoids. The size of $\mathcal{A}(S)$ is $O(n + k)$ where k is the number of intersections between the segments; the same bound applies to $\mathcal{T}(S)$.

We elaborate on linearization for the pair $(\mathcal{S}, \mathbf{T}(\mathcal{S}))$. For a segment s, let (l_1, l_2) and (r_1, r_2) be the endpoints of s, the map φ is into $\mathbb{R}^8$ and is defined so $\varphi(s)$ equals the tuple

$$(l_1, l_2, r_1, r_2, l_1 l_2, l_1 r_2, r_1 l_2, r_1 r_2).$$

The map $\psi(\sigma)$ is determined by writing a complete case analysis of the conditions under which a segment s intersects σ: s intersects σ iff either (i) σ contains one endpoint of s, or (ii) s intersects one of the vertical edges of σ, or (iii) s intersect both the top and bottom edges of σ. The result is of the form $c_1 \vee \cdots \vee c_m$ where each c_i is of the form $c_{i,1} \wedge \cdots \wedge c_{i,n_i}$ where each $c_{i,j}$ is an inequality. When these inequalities are written so as to avoid division, one finds that the variables appearing in them are those above defining $\varphi(s)$. Thus, these inequalities determine half-spaces in $\mathbb{R}^8$. Since m and n_i are $O(1)$, the inequalities determine a linear cell.

Estimation of the number of intersections. For the complete arrangement of segments $f(X, r) = O(r + k(r/n)^2)$, so to have the right parameters in (iii) and (iv) of Theorem 2.4, we need to verify that $f(A, r) = O(r + k(r/n)^2)$.

A $(1/r)$-approximation A for $(S, \mathcal{R}(S, \mathbf{T}(\mathcal{S})))$ is also a $(1/r)$-approximation for $(S, \mathcal{R}(S, \mathcal{S}))$. Then, by a result of Brönnimann et al [7] on product range spaces, A can also be used to estimate the number of intersections in $\mathcal{A}(S)$ inside any convex region. More precisely, for a set of segments X, let $v(X,\sigma)$ denote the number of intersections between segments in X that lie in σ, then

Lemma 2.5 *Let A be a $(1/r)$-approximation for S. For any simplex σ, $|v(S,\sigma)/|S|^2 - v(A,\sigma)/|A|^2| < 1/r$.*

Thus, we have that $f(A,r) \leq C((r/|A|)^2 v(A,\sigma)+r) \leq C((r/n)^2 v(S,\sigma)+r)$, as desired. To summarize, we obtain the following construction:

Theorem 2.6 *For $0 < \delta < 1$, there are constants C, r_0, ϵ such that for $r_0 \leq r \leq n^\epsilon$, in time $O(n\log r + (r/n)k)$ one can compute a sample $R \subset S$, its triangulation $\mathcal{T}(R)$ and conflict lists, with the following properties: (i) $||R|-3r/2| \leq r$, (ii) $|\mathcal{T}(R)| \leq C(r+(r/n)^2 k)$, (iii) $\max_{\sigma\in\mathcal{T}(R)} |S_\sigma| \leq (n/r)r^\delta$, (iv) $\sum_{\sigma\in\mathcal{T}(R)} |S_\sigma| \leq C(n+(r/n)k)$. In the EREW model the computation can be performed in time $O(\log n \log r)$ and work proportional to the sequential time.*

In our second application, we are interested in computing the face $f_p(S)$ of $\mathcal{A}(S)$ that contains a given point p. The *complexity* of a face f is the number of vertices and edges on its boundary. Let $\mathcal{T}_p(R)$ be the trapezoidal diagram $\mathcal{T}(R)$ restricted to $f_p(R)$. The complexity of $f_p(S)$ is $O(n\alpha(n))$ where α is the very slowly-growing inverse of Ackerman's function [24, 36]; this bound can be achieved [40]. The same bound applies to $\mathcal{T}_p(S)$. For $\mathcal{T}_p(R)$ monotonicity but not locality holds. Theorem 2.4 directly implies the following.

Theorem 2.7 *For $0 < \delta < 1$, there are constants C, r_0, ϵ such that for $r_0 \leq r \leq n^\epsilon$, in time $O(n\log r)$ one can compute a sample $R \subset S$, its triangulation $\mathcal{T}_p(R)$ and conflict lists with the following properties: (i) $||R|-3r/2| \leq r$, (ii) $|\mathcal{T}(R)| \leq Cr\alpha(r)$, (iii) $\max_{\sigma\in\mathcal{T}(R)} |S_\sigma| \leq (n/r)r^\delta$, (iv) $\sum_{\sigma\in\mathcal{T}(R)} |S_\sigma| \leq Cn\alpha(r)$.*

Randomized algorithms

Efficient randomized algorithms are known for both problems on segments we are considering. Here, we point out that for both, algorithms with equal asymptotic performance can be obtained using the approach used by Chazelle [10] and Brönnimman et al [7] for half-space intersection.

The sampling is *global*, that is, at a given stage a single global sample determines the sample in each subproblem, rather than an independent sample for each subproblem. As a result, one maintains $\mathcal{T}(R_i)$ for a single sample R_i of S at each stage. This has the advantage of bounding globally the sum of the sizes of all subproblems at each step. This approach is related to the lazy randomized incremental construction of [1, 18].

The algorithm inserts the segments in rounds. For each segment s and $i = 0,1,2,3,\ldots,k$, where $k = \Theta(\log n)$, let $I_{s,i}$ be a 0-1 random variable with $\text{Prob}\{I_{s,i}=1\} = p_i$ where $p_0 = c/n$ and $p_i = 2^{i-1}p_0/(1-2^{i-1}p_0)$, for some constant c. $R'_i = \{s : I_{s,i}=1, I_{s,j}=0 \text{ for } j<i\}$ is the set of segments inserted in the i-th round. The set $R_i = \bigcup_{j=0}^{i} R'_j$, those segments inserted up to the i-th round, is a q_i-sample where $q_i = 2^i p_0$ (thus, the expected size of R_i doubles in every round); and R'_i is a p_i-sample of $S - R'_{i-1}$.

For computing the segment arrangement, in the i-th round, for each $\sigma \in \mathcal{T}(R_{i-1})$, compute $\mathcal{T}((R'_i)_\sigma)$ restricted to σ and the corresponding conflict lists. A not very efficient algorithm can be used for these local computations as the expected size of $(R'_i)_\sigma$ is $O(1)$. Then a global clean-up puts together pieces of the same trapezoid $\tau \in \mathcal{T}(R_i)$ which are in different trapezoids $\sigma \in \mathcal{T}(R_{i-1})$, and their corresponding conflict lists. The final result is $\mathcal{T}(R_i)$.

For computing the face containing a point p, the clean-up includes determining through a graph search those trapezoids reachable from p, thus obtaining $\mathcal{T}_p(R_i)$ at stage i.

An analysis similar to that in [7] shows that these algorithms have expected running times $O(n\log n + k)$ and $O(n\alpha(n)\log n)$ respectively.

Unfortunately, it is not clear how to apply the techniques in [7] to derandomize these algorithms. Because of the lack of locality, this seems especially hard for the single face problem. Thus, for derandomization, in the next two sections we turn to *local sampling*.

3 Arrangement of segments

An inefficient algorithm

We first describe a simple, inefficient algorithm that will be useful in the optimal algorithm to deal with subproblems of small size: the algorithm computes a sample R according to Theorem 2.4 for $r = n^\epsilon$ and some constant $0 < \epsilon < 1$, computes the conflict lists of the trapezoids in $\mathcal{T}(R)$, and then it recurses on each of the trapezoids whose conflict list is larger than a certain constant and finishes the smaller ones with some "brute" force method.

The sample and the corresponding conflict lists are computed in time $O(n\log n + (r/n)k)$ and results in $O(r + k(r/n)^2)$ trapezoids. Thus, the total running time for the root problem is $O(n\log n + kr/n)$. An inductive proof shows that the total running time is $O(n\log^c n + k)$ for some constant $c > 1$. The algorithm is readily parallelized with a running time $O(\log^2 n)$ and work $O(n\log^c n + k)$.

Lemma 3.1 *Let S be a set of n line segments that have k pairwise intersections. Then one can compute the arrangement of the segments in S (deterministically and sequentially) in time $O(n\log^c n + k)$. In the EREW PRAM model it takes time $O(\log^2 n)$ using work $O(n\log^c n + k)$.*

An efficient algorithm

To avoid the total size blow-up that results in the undesired polylog term, we use an extra cutting step in each stage. Let $\mathbf{T}_i$ denote the collection of *active* trapezoids at the beginning of stage i (those on which the algorithm recurses); $\mathbf{T}_1$ consists of a single infinite trapezoid.

For a trapezoid τ, let S_τ denote those segments in S that intersect τ, and let S_τ^c and S_τ^{nc} denote those segments in S_τ that cross and those that do not cross (that is, with an endpoint inside) τ respectively. Also, let $S_\tau^{c,\text{lite}}$ (respectively, $S_\tau^{c,\text{heavy}}$) denote those segments in S_τ^c with less than (resp., at least) λ intersections in τ. Let $n_\tau = |S_\tau|$ and let k_τ denote the number of segment intersections inside τ.

During each level of recursion the algorithm will maintain the following invariant:

Invariant: The total size of all active subproblems $\sum_{\sigma\in\mathbf{T}_i} n_\sigma$ is $O(n + k/\lambda)$, where $\lambda = \log n$.

The following steps are performed at stage i:

```
For each $\sigma \in \mathbf{T}_i$:
1.  Compute a sample $R(\sigma) \subseteq S_\sigma$, $\mathbf{T}(\sigma) := \mathcal{T}(R(\sigma))$ and
        conflict lists according to Theorem 2.6 with $r_\sigma = n_\sigma^\epsilon$
2.  For each $\tau \in \mathbf{T}(\sigma)$:
2.1.     Compute $S_\tau^c$ and $S_\tau^{nc}$
2.2.     Compute $S_\tau^{c,\text{lite}}$ and $S_\tau^{c,\text{heavy}}$
2.3.     Compute cutting $\mathcal{T}^\lambda(\tau)$ and conflict lists
2.4.     For each $\mu \in \mathcal{T}^\lambda(\tau)$
2.4.1.       if $|S_\mu| > \lambda^2$ then put $\mu$ in $\mathbf{T}_{i+1}$
             else compute the arrangement of $S_\mu$ restricted to
                 $\mu$ using the inefficient algorithm
```

Cutting $\mathcal{T}^\lambda(\tau)$. The cutting $\mathcal{T}^\lambda(\tau)$ has the following properties: (i) $|\mathcal{T}^\lambda(\tau)| = O(|S_\tau^{c,\text{lite}}|/\lambda)$; (ii) each $\mu \in \mathcal{T}^\lambda(\tau)$ intersects $O(\lambda)$ lite segments; and (iii) a segment s intersects $O(k_{\tau,s}/\lambda + 1)$ trapezoids in $\mathcal{T}^\lambda(\tau)$, where $k_{\tau,s}$ is the number of intersections on s in τ. From this it follows that $\sum_{\mu\in\mathcal{T}^\lambda(\tau)} n_\mu$ is $O(n_\tau + k_\tau/\lambda)$.

Now, we describe how to obtain a cutting with those properties. Let $Q \subseteq S(\sigma)_\tau^c$ such that Q is not non-intersecting inside τ. Q partitions τ into a set of polygons. Let μ be one of them; it has edges on the boundary of τ which we call B-edges, and edges on segments in Q which we call Q-edges. For μ let $b(\mu)$ (resp. $q(\mu)$) be the number of intersections between lite segments and its B-edges (resp. Q-edges). Except for $O(1)$ of them, these polygons are quadrilaterals with two B-edges and two Q-edges, which we call *good* polygons (the others are *bad*). See Figure 3.

Let Q be so that for each polygon μ, $D\lambda \leq b(\mu) \leq E\lambda$, for some constants A and B. Clearly, the number of polygons is $O(|S_\tau^{c,\text{lite}}|/\lambda)$. Since a polygon has at most four Q-edges, then it intersects $O(\lambda)$ lite segments. Let s be a segment. If s intersects the two Q-edges of a good polygon μ, then s is intersected by at least $D/2 - 2\lambda$ (lite) segments in μ, which is at least λ if $D \geq 6$; thus, since s intersects $O(1)$ bad polygons, and the B-edges of $O(1)$ good polygons, it follows that s intersects $O(k_{\tau,s}/\lambda + 1)$ polygons. Properties (i)-(iii) above apply to the collection of trapezoids resulting from the trapezoidal decomposition of the polygons, which is then our cutting $\mathcal{T}^\lambda(\tau)$.

For E sufficiently large, such set Q can be constructed by adding segments incrementally, and it can be done in time $O(n_\tau)$ assuming that the endpoints of the segments on the boundary of τ are sorted. The algorithm works by scanning the ordered lists of intersections of lite segments with the boundary of τ. The conflict lists can be computed in time linear with the output size plus $O(n_\tau \log n_\tau)$ overhead. Thus, the total time for constructing the special cutting is $O(n_\tau \log n_\tau + k_\tau/\lambda)$. We elaborate further in the full version.

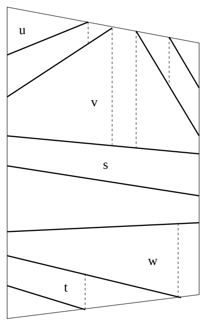

Figure 1: Cutting $\mathcal{T}^\lambda(\tau)$: s, t are good, u, v, w are bad.

Invariant. A segment in a nonterminal trapezoid $\mu \in \mathcal{T}^\lambda(\tau)$: (i) has a real endpoint in τ, (ii) is heavy (in τ), or (iii) is lite (in τ). By the properties of $\mathcal{T}^\lambda(\tau)$, cases (i) and (ii) can be credited to the n and k/λ terms respectively. Case (iii) can be credited to the n term or the k/λ term since while there are only $O(\lambda)$ lite segments in μ, there are at least $\lambda^2 - O(\lambda)$ other segments in μ which can *lend* credit. Thus the total subproblem size is $O(n + k/\lambda)$ and the invariant is enforced in each stage.

Total work. The sample, triangulation and conflict lists in step 1 are computed in time $O(n_\sigma \log n_\sigma + k_\sigma r_\sigma/n_\sigma)$ by Theorem 2.4. By construction $\sum_{\tau\in\mathbf{T}(\sigma)} n_\tau = O(n_\sigma + k_\sigma(r_\sigma/n_\sigma))$. For each τ, step 2.1 is trivially performed in time $O(n_\tau)$, step 2.2 is performed in time $O(n_\tau \log n_\tau)$ using well-known techniques (see e.g. [2, 20]), step 2.3 takes time as indicated above.

Thus, steps 1-2 for σ take time $O((n_\sigma + k_\sigma(r_\sigma/n_\sigma)) \log r_\sigma + k_\sigma/\lambda)$. Using the invariant in the previous stage, this is $O(((n + k/\lambda) + k/\nu) \log r_i)$ where $r_i = \max_{\sigma\in\mathbf{T}_i} r_\sigma$ and $\nu = \min_{\sigma\in\mathbf{T}_i} n_\sigma/r_\sigma$. Assuming $\epsilon \leq 1/2$, since $n_\sigma > \lambda^2$ for non-terminal subproblems, then $\nu \geq \lambda$, and the total time for the stage is $O((n + k/\lambda) \log r_i)$, except for the small *terminal* subproblems that are terminated with the inefficient algorithm. Since r_i decreases as $n^{(1-\epsilon')^i \epsilon'}$, for some $0 < \epsilon' < 1$, then $\sum_i \log r_i = O(\log n)$. So adding this time over all stages results in time $O(n \log n + k)$.

Now we consider the terminal subproblems. They are of size $O(\lambda^2)$, so the arrangement inside μ can be computed in time $O(n_\mu \log^c \log n + k_\mu)$ using the inefficient algorithm. Since $\sum_{\mu\in\mathcal{T}^\lambda(\tau)} n_\mu$ is $O(n_\tau + k_\tau/\lambda)$, and $\sum_{\tau\in\mathbf{T}(\sigma)} (n_\tau + k_\tau/\lambda)$ is $O(n_\sigma + k_\sigma/\lambda)$ (although μ is terminal, $n_\sigma > \lambda^2$), then the total amount of terminal work in one stage is $O((n + k/\lambda) \log^c \log n + k_i)$, where k_i is the number of intersections found in these terminal subproblems. This added over $O(\log \log n)$ stages is $O(n \log n + k)$. Thus, the total time is $O(n \log n + k)$.

The previous discussion is summarized in the following.

Theorem 3.2 *The arrangement of a set of n line segments in the plane with k intersections can be computed in time $O(n \log n + k)$ sequentially and deterministically.*

This result was first obtained by Chazelle and Edelsbrunner [11] using a sweep algorithm. Our algorithm provides a divide-and-conquer sequential alternative, and has the advantage of being parallelizable and of using $O(n)$ space when only required to output the k intersections.

Parallel algorithm

An important characteristic of our algorithm is that it can be readily parallelized using simple techniques.

Theorem 3.3 *The arrangement of a set of n line segments in the plane with k intersections can be computed in the EREW model deterministically in time $O(\log^2 n)$ and work $O(n \log n + k)$. Using randomization, the time is $O(\log n \log\log n)$ with n-polynomial probability.*

All steps in stage i can be parallelized in a straight forward manner using known parallel techniques (see, e.g., [4, 37]). In particular, stage i runs in $O(\log^2 r_i)$ time using optimal work. Also, at each round it suffices to perform one single global processor allocation call. If a model that allows non-global processor allocation calls is used then the time required by the randomized algorithm is reduced to $O(\log n)$.

Linear space

Chazelle and Edelsbrunner [11] posed the problem of whether the k intersections of n given segments can be output in time $O(n \log n + k)$ using space $O(n)$. Our algorithm can be adapted for this task.

Theorem 3.4 *The k intersections of a set S of n segments in the plane can be reported using space $O(n)$ and time $O(n \log n + k)$.*

Proof: At any given time, it is sufficient to keep only one path of the computation tree from the root down to a leaf (or just down to subproblems of size $O(\sqrt{n})$ which can then be solved in $O(n)$ space). At each node on this path one cannot afford to compute all conflict lists at once. Instead, they are computed in batches of size $O(n_\sigma)$. This is possible for $\sigma \in \mathbf{T}_i$: first compute the sizes of the conflict lists, then output them in batches of size $\Theta(n_\sigma)$. Similarly, for each $\tau \in \mathbf{T}(\sigma)$. This is done within the same optimal time. ■

4 Single face

For $Q \subseteq S$ let $\mathcal{A}_\sigma(Q)$ denote the arrangement $\mathcal{A}(Q)$ restricted to the trapezoid σ (it includes the two segments on the top and the bottom). The set of faces in $\mathcal{A}_\sigma(Q)$ that can be reached from p (when the portions of segments outside σ are clipped away) is denoted by $f_{p,\sigma}(Q)$. Clearly, $f_{p,\sigma}(Q)$ contains $f_p(S) \cap \sigma$.

The algorithm keeps at every stage a collection of *active* trapezoids, whose union is guaranteed to contain $f_p(S)$ (with the exception of some pieces that will be explained later). Let $\mathbf{T}_i$ be the collection of active trapezoids at the beginning of stage i. In general, $\mathbf{T}_i$ is not $\mathcal{T}_p(R)$ for some R because sampling is independent in each subproblem. For $\sigma \in \mathbf{T}_i$, we have a conflict list $S(\sigma)$, which is in general different from S_σ because of pruning. Let $n_\sigma = |S(\sigma)|$. For a trapezoid τ contained in σ, following convention, $S(\sigma)_\tau$ denotes the list of conflicts of τ in $S(\sigma)$. Let $S(\sigma)^c_\tau$ be those segments in $S(\sigma)_\tau$ that cross (have no endpoint inside) τ, and let $S(\sigma)^{nc}_\tau$ be those segments that do not cross (have at least one endpoint in) τ. For each trapezoid σ, we maintain for its left (resp. right) vertical edge $l(\sigma)$ (resp. $r(\sigma)$) the sorted list $L_l(\sigma)$ (resp. $L_r(\sigma)$) of intersections between segments in $S(\sigma)$ and $l(\sigma)$ (resp. $r(\sigma)$). These lists also contain connectivity information to be described later. The following steps are performed at stage i:

```
1 For each $\sigma \in \mathbf{T}_i$:
1.1   If $n_\sigma$ is smaller than a constant then compute $f_{p,\sigma}(S_\sigma)$
        by some brute force method.
1.2   Compute a sample $R(\sigma) \subseteq S(\sigma)$, $\mathbf{T}(\sigma) := \mathcal{T}(f_{p,\sigma}(R(\sigma)))$
        and conflict lists according to Theorem 2.7 with $r_\sigma = n_\sigma^\epsilon$
1.3   For each $\tau \in \mathbf{T}(\sigma)$:
1.3.1   Compute $S(\sigma)^c_\tau$ and $S(\sigma)^{nc}_\tau$
1.3.2   Compute $f_{p,\tau}(S(\sigma)^c_\tau)$
2 Use a graph search to determine all faces in $f_{p,\tau}(S(\sigma)^c_\tau)$
      for some $\tau \in \mathbf{T}(\sigma)$ and $\sigma \in \mathbf{T}_i$ that can be reached from
      $p$, and mark them as reachable
3 For each $\sigma \in \mathbf{T}_i$:
3.1   Put each reachable $\tau \in \mathbf{T}(\sigma)$ in $\mathbf{T}_{i+1}$
3.2   For each $\tau \in \mathbf{T}_{i+1}$ prune $S(\sigma)_\tau$ to obtain $S(\tau)$ and
      update connectivity accordingly
```

Running time. Step 1.2 is performed in time $O(n_\sigma \log n_\sigma)$ by Theorem 2.7. Steps 1.3.1 and 1.3.2 are performed in time $O(n_\tau \log n_\tau)$ for each τ and hence time $O(n_\sigma \alpha(r_\sigma) \log n_\sigma)$ for all τ in a given σ (note that 1.3.2 is not a recursive computation, this is a computation of the zone of a line in an arrangement of lines or the intersection of some halfplanes, both of which are simpler problems). Thus, the total time for step 1 is $O(\sum_{\sigma \in \mathbf{T}_i} n_\sigma \alpha(r_\sigma) \log r_\sigma)$. We will see that for each i pruning enforces the global bound $\sum_{\sigma \in \mathbf{T}_i} n_\sigma = O(n\alpha(n))$. Thus, the total time performed in step 1 at each stage is $O(n\alpha(n)\alpha(r) \log r)$ where $r = \max_{\sigma \in \mathbf{T}_i} r_\sigma$. The graph search performed in step 2 takes time $O(n\alpha(n)\alpha(r))$ because this is a bound on the number of trapezoids that can be reached. It will be clear after the explanation of pruning below that step 3 can be performed within the time bound

$O(n\alpha(n)\alpha(r)\log r)$. Since $\sum_i \log r_i = O(\log n)$, the total running time is $O(n\alpha^2(n)\log n)$.

Arrangement of clipped segments. The first observation needed to obtain the global bound on the subproblems is that the graph search of step 2 determines the single face of p in an arrangement of n segments S_{clip}. Indeed, for a segment s, if its endpoint is in the interior of a trapezoid τ, then its portion inside τ is effectively clipped by removing $S(\sigma)^{nc}_\tau$ from consideration. Thus, the segments in S_{clip} are the original segments in S each one possibly with pieces containing its endpoints clipped away. This gives a global bound $O(n\alpha(n))$ on the number of vertices (real vertices, not intersections of segments with vertical boundary segments) in the face computed by the graph search of step 2.

Actually, the previous argument is not quite complete as there are additional pieces of segments removed by pruning. But it will be clear from the explanation of pruning that the claim above remains true.

Pruning. For a trapezoid τ, we want to prune its conflict list $S(\sigma)_\tau$ to obtain $S(\tau)$. First, we put all the noncrossing segments $S(\sigma)^{nc}_\tau$ in $S(\tau)$. Let $f_{\text{reach}}(\tau)$ be those faces in $f_{p,\tau}(S(\sigma)^c_\tau)$ that were marked by the graph search of step 2 as reachable. All those $s \in S(\sigma)^c_\tau$ not bounding a face in $f_{\text{reach}}(\tau)$ are eliminated from consideration for $S(\tau)$. For each vertex of a face in $f_{\text{reach}}(\tau)$, we include in $S(\tau)$ the two segments intersecting at that vertex. The only segments bounding $f_{\text{reach}}(\tau)$ that are not yet considered are those corresponding to pairs of segments in $S(\sigma)^c_\tau$ bounding a face of $f_{\text{reach}}(\tau)$ that cuts through and has no vertices inside τ (these two edges go from one side of τ to the other, do not intersect, are not intersected by any other segment in $S(\sigma)^c_\tau$, and have no segment in $S(\sigma)^c_\tau$ between them). We call them *tunnel pieces*, and they form *tunnels* when all the pieces bounded by the same segments and in neighbor trapezoids τ are put together. We cannot put the two segments bounding a tunnel T in $S(\tau)$ for each τ that contains a tunnel piece of T, because a tunnel may go through many trapezoids τ. Thus, we need to consider in detail these tunnel pieces.

Tunnel pieces. We distinguish three types of tunnel pieces: A tunnel piece is *unobstructed* if it is not intersected by any noncrossing segment; it is *completely obstructed* if it is intersected and crossed by some noncrossing segment; or it is *potentially obstructed* if it is intersected by noncrossing segments but not crossed by any of them. See Figure 4. Tunnel pieces are easily identified: to classify them we perform a point location of the endpoints of the segments in $S(\sigma)^{nc}_\tau$ among the strips determined by all the tunnel pieces in τ, and then we identify strips that are not crossed. This takes time $O(n_\tau \log n_\tau)$ for each τ, and $O(n\alpha(n)\alpha(r)\log r)$ globally.

For a tunnel T, let $s_1(T), s_2(T) \in S$ be the two segments bounding it. Similarly $s_1(t), s_2(t)$ for a tunnel piece t. The troublesome tunnel pieces are the unobstructed and the completely obstructed ones because there can be many of them. On the other hand, a partially obstructed tunnel piece t in τ is easy to handle: put $s_1(t), s_2(t)$ in $S(\tau)$; these two segments can be charged to some segment s with an endpoint in t (a given segment s cannot get more than four charges in this manner). The other two types are handled as follows.

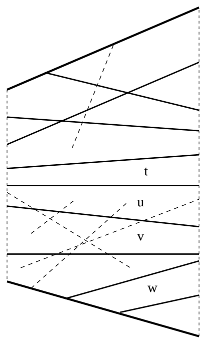

Figure 2: Tunnel pieces: t, u, and v are unobstructed, potentially obstructed and completely obstructed respectively. w is not a tunnel piece.

Let t be an unobstructed tunnel piece in a trapezoid τ. Then t does not put $s_1(t)$ or $s_2(t)$ in $S(\tau)$ (still the neighbor faces in τ may put $s_1(t)$ or $s_2(t)$ in $S(\tau)$), but we need to store the connectivity information that would be lost otherwise as follows. Consider a maximal chain $\mathbf{t}$ of unobstructed tunnel pieces and let $l(\mathbf{t})$ and $r(\mathbf{t})$ be its vertical edges on the left and on the right. Let τ_l (resp. τ_r) be the trapezoid whose right (resp. left) vertical edge contains $l(\mathbf{t})$ (resp. $r(\mathbf{t})$). τ_l and τ_r will store in their lists $L_r(\tau_l)$ and $L_l(\tau_r)$ pointers that allow to jump directly between τ_l and τ_r in both directions; previous connectivity information may need to be updated (to connect pieces of tunnel previously found).

Finally, let us consider the completely obstructed tunnel pieces. Only two of them in a particular tunnel T, contained in trapezoids τ_l and τ_r respectively, can possibly intersect $f_p(S)$ (for example, if p is outside the tunnel, then they are the leftmost and the rightmost completely obstructed tunnel pieces in the tunnel). Pruning is handled as follows: put $s_1(T), s_2(T)$ in $S(\tau_l)$ and $S(\tau_r)$ (any other completely obstructed piece t in T does not put $s_1(T), s_2(T)$ in its trapezoid τ, but again the neighbor faces may do it). Those segments that are kept (at most four) can be charged to a vertex of the single face of S_{clip}.

Note that after the pruning is performed, the single face of p in the resulting arrangement of segments (since some of them had pieces taken away, the resulting set is different) is the same as the single face of p in the original arrangement: no less can be reached because we are only taking away pieces of segments (the connectivity information of unobstructed tunnel pieces is not lost); and no more can be reached because a segment piece is removed only if it does not bound $f_p(S)$ (except for the bounding segments of an unobstructed tunnel piece, but in that case the tunnel piece itself is removed and the connectivity information prevents unreachable points from becoming reachable).

This completes the argument that gives the global bound $\sum_{\sigma \in \mathbf{T}_i} n_\sigma = O(n\alpha(n))$.

5 Simplices in higher dimensions

In this section we consider a set S of n possibly intersecting $(d-1)$-simplices in $\mathbb{R}^d$, for any fixed $d \geq 3$, e.g., triangles in $\mathbb{R}^3$, or tetrahedra in $\mathbb{R}^4$. When the simplices in S may intersect, the complexity k of $\mathcal{A}(S)$ can vary between n and n^d. We give methods for constructing a covering for $\mathcal{A}(S)$ of size $m = O(n^{d-1}\log^{O(1)} n + k)$. We also describe how the hierarchical description of the covering can be used to answer point location, vertical ray shooting, and incidence queries in $O(\log n)$ time using the same storage. The constructions are performed sequentially in $O(m)$ time, except for the point location structure which requires $O(m + k\log n)$ time. In parallel, the constructions take $O(\log n)$ time using $O(m)$ work in the EREW PRAM model, and the point location structure is built in $O(\log^2 n)$ time using $O(m + k\log n)$ work. When the simplices in S are interior disjoint, e.g., non-simple polyhedra, the construction can be optimized to build a a triangulation of $\mathcal{A}(S)$ of size $O(n^{d-1})$ using as much work.

Our construction is inspired by Pellegrini [35], who gives a recursive construction that has $O(\log n)$ stages, each using constant size cuttings[6] of the simplices in S. An augmented version of the resulting hierarchical structure answers incidence and vertical ray-shooting queries in $O(\log^2 n)$ and $O(\log^3 n)$ time, respectively. Point location queries take $O(\log^3 n)$ time as they are reduced to d vertical ray shooting queries in $\mathcal{A}(S)$. The essential change we make to reduce the size of the covering and obtain faster query times is to decrease the depth of the recursion by increasing the size of the cuttings. Of course, to obtain query times of $O(\log n)$, all secondary structures must follow this principle as well. Finally, increasing the size of the cuttings implies changes for the queries, e.g., a point can be located in $O(1)$ simplices by brute force, but a more efficient method must be used if there are $O(n^\epsilon)$ simplices. We need the following for the construction.

Definition 5.1 Let τ be a $(d-1)$-simplex and σ be a d-simplex or vertical d-cylinder. τ *crosses* σ if τ intersects σ and no $(d-2)$-face of τ intersects σ, and τ *partially crosses* σ if some $(d-2)$-face of τ intersects σ. If τ partially crosses σ, then the vertical projections to $\mathbb{R}^{d-1}$ of τ and σ intersect.

Lemma 5.2 ([4]) *Given a simplex σ and a set H of n hyperplanes intersecting σ in $\mathbb{R}^d$, a $(1/r)$–cutting for H (restricted to σ), including conflict lists, with $m = O(r^{d-1} + (r/n)^d v(H,\sigma))$ simplices can be constructed in $O(\log n)$ time using $O(nr^c + nm)$ work in the* EREW PRAM *model, where $c \geq 1$ is some constant and $v(H,\sigma)$ is the number of vertices of $\mathcal{A}(H)$ in σ.*

Constructing a covering of $\mathcal{A}(S)$. We build a sequence $C_0, C_1, \ldots, C_h$ of sets of d-simplices and vertical d-cylinders. For each $\sigma \in C_i$, we partition its *conflict list* S_σ into two sets S^{pc}_σ and S^c_σ, where S^{pc}_σ contains the $(d-1)$-simplices that partially cross σ, and S^c_σ contains the $(d-1)$-simplices that cross σ. The basis C_0 is a large simplex σ_0 containing S with $S^{pc}_{\sigma_0} = S$ and $S^c_{\sigma_0} = \emptyset$. Inductively, we have a set C_{i-1} that covers σ_0, such that, for every $\sigma \in C_{i-1}$, $|S^{pc}_\sigma| \leq n_{i-1}$ and $|S^c_\sigma| \leq n_{i-2}$, where $n_{i-1} = n_{i-2}/r_{i-1}$, $r_{i-1} = n^\epsilon_{i-2}$, for some constant $0 < \epsilon < 1$, and $n_{-1} = n_0 = n$. C_i is obtained from C_{i-1} by refining $\sigma \in C_{i-1}$ if $|S^{pc}_\sigma| > n_i$ or $|S^c_\sigma| > n_{i-1}$.

If $|S^c_\sigma| > n_{i-1}$, we construct a $(1/\rho_1)$-cutting (restricted to σ) for the hyperplanes spanning the simplices in S^c_σ, where $\rho_1 = r_{i-1}|S^c_\sigma|/n_{i-2} \leq r_{i-1}$. Thus, each new simplex τ in C_i has $|S^c_\tau| \leq |S^c_\sigma|/\rho_1 = n_{i-2}/r_{i-1} = n_{i-1}$ (the set S^{pc}_τ is empty since every simplex in S^c_σ crosses τ). By Lemma 5.2, there are $O({\rho_1}^{d-1} + (\frac{\rho_1}{|S^c_\sigma|})^d v(S,\sigma))$ simplices in the cutting.

If $|S^{pc}_\sigma| > n_i$, we vertically project σ and the simplices in S^{pc}_σ to $\mathbb{R}^{d-1}$, obtaining a $(d-1)$–simplex and a set S'^{pc}_σ of $(d-1)$–simplices, respectively. We construct a $(1/\rho_2)$-cutting for the hyperplanes spanning the $(d-2)$-faces of simplices in S'^{pc}_σ, where $\rho_2 = r_i|S^{pc}_\sigma|/n_{i-1} \leq r_i$. Next, the $(d-1)$-simplices in the resulting cutting are vertically projected obtaining d–cylinders which are restricted to σ. The choice of ρ_2 guarantees that each new d-cylinder τ in C_i has $|S^{pc}_\tau| \leq |S^{pc}_\sigma|/\rho_2 = n_{i-1}/r_i = n_i$ and $|S^c_\tau| \leq |S^{pc}_\sigma| \leq n_{i-1}$. There are $O({\rho_2}^{d-1})$ d-cylinders resulting from the cutting.

There are $h = O(\log\log n)$ levels in the recursion. The size of C_i is given by the recurrence

$$\begin{aligned}|C_i| &\leq \sum_{\sigma\in C_{i-1}} c_1\left\{{\rho_1}^{d-1} + \left(\frac{\rho_1}{|S^c_\sigma|}\right)^d v(S,\sigma) + {\rho_2}^{d-1}\right\}\\ &\leq Ar^{d-1}_{i-1}|C_{i-1}| + B\left(\frac{r_{i-1}}{n_{i-2}}\right)^d k\end{aligned}$$

for some constants $A, B > 1$. This recurrence's solution is $|C_i| \leq C^i(n/n_{i-1})^{d-1} + D(1/n_{i-1})^d k$ for some constants $C, D \geq 1$. Verifying this inductively we need $C \geq A$ and $D \geq ADr^{-1}_{i-1} + B$, which are satisfied for appropriate choices of C and D, e.g., $C \geq A$, $D = A + B$, and $D < r_{i-1}$. Thus, the total number of simplices generated is

$$\sum_{i=0}^{h} C^i\left(\frac{n}{n_{i-1}}\right)^{d-1} + D\left(\frac{1}{n_{i-1}}\right)^d k = O(n^{d-1}\log^{O(1)} n + k).$$

We now examine the time and work bounds of the algorithm. By Lemma 5.2, for each $\sigma \in C_i$ we spend $O(\log n_{i-1})$ time and $O(n_{i-1}r^E_i)$ work for some constant $E \geq d$. Thus, the total time is $\sum_i \log n_{i-1} = O(\log n)$, and the total work is

$$\begin{aligned}\sum_{i=0}^{h} n_{i-1}r^E_i|C_i| &\leq n^{d-1}\sum_{i=0}^{h} C^i n^{2-d}_{i-1} r^E_i + Dk\sum_{i=0}^{h} n^{1-d}_{i-1}r^E_i\\ &= n^{d-1}\sum_{i=0}^{h} C^i n^{2-d+\epsilon E}_{i-1} + Dk\sum_{i=0}^{h} n^{1-d+\epsilon E}_{i-1}\\ &= O(n^{d-1}\log^{O(1)} n + k),\end{aligned}$$

for ϵ chosen sufficiently small so that $\epsilon E < d - 2$.

[6]In a $(1/r)$–cutting, the conflict list of each simplex in the cutting has size at most n/r.

Below we describe how the hierarchical structure built when constructing the covering of $\mathcal{A}(S)$ can be augmented to answer incidence and vertical ray shooting queries in $O(\log n)$ time. Then we briefly discuss how point location queries can be performed in $O(\log n)$ time using a sequence of d vertical ray shooting queries.

Incidence queries. As in [35], the basic data structure is a two–level search tree. The primary search tree T^{pc} (a *PC-tree*) is built from the cylinders generated from the sets S^{pc}_σ, and each secondary search tree T^c (a *C-tree*) is built from the d-simplices generated from the sets S^c_σ. For example, for $\sigma \in C_i$ the cylinders of C_{i+1} resulting from the cutting of S^{pc}_σ are stored in the node v_σ associated with σ in T^{pc}, and the simplices of C_{i+1} resulting from the cutting of S^c_σ are contained in the root of v_σ's secondary structure T^c.

During an incidence query, the query point p is first located in the set of d-cylinders contained at the root of T^{pc}. Since there are $O(r^{d-1}_{i+1})$ d-cylinders contained in each node at level i in T^{pc}, the search cannot be performed by brute force (as was done in [35]). However, we can build a point location structure for the $(d-1)$-dimensional arrangement of the hyperplanes spanning the $(d-2)$-faces of the simplices in S'^{pc}_σ. This can be done in $O(\log r_{i+1})$ time using $O(r^{d-1}_{i+1})$ work, and the point location query can be answered in $O(\log r_{i+1})$ time [4]. If p is contained on the boundary of a d-cylinder $\sigma \in T^{pc}$, then a simple check determines whether p is incident to the $(d-1)$-simplex in S defining σ. If not, then p is located recursively in the secondary search tree T^c (for the current node of T^{pc}) and in the indicated subtree of T^{pc}. We also build a point location structure for each node in a C-tree, the only difference being that we build a d-dimensional structure using $O(r^d_i)$ work. Thus, an incidence query takes time $\sum_i \log r_i = O(\log n)$. For each $\sigma \in C_i$ we spend $O(r^d_i)$ work so the point location structures can be built within the same resource bounds as the basic structure, i.e., the work is folded into the r^E_i factor.

Vertical ray shooting queries. Consider a d-cylinder $\sigma \in T^{pc}$ containing the query point p: the $(d-1)$-simplex immediately below p in $S^{pc}_\sigma \cup S^c_\sigma$ is either the $(d-1)$-simplex immediately below p in S^{pc}_σ or the $(d-1)$-simplex immediately below p in S^c_σ. Since T^{pc} is composed of vertical cylinders, queries can be answered recursively on PC-trees. Next consider simplices $\sigma_1 \in C_i$ and $\sigma_2 \in C_{i+1}$ where $p \in \sigma_2 \subset \sigma_1$ and both σ_1 and σ_2 are contained in a C-tree. Let $b(\sigma_2)$ be the subset of the $(d-1)$-simplices in $S^c_{\sigma_1} \setminus S^c_{\sigma_2}$ that lie below σ_2. Queries in C-trees can be answered recursively since the $(d-1)$-simplex immediately below p in $S^c_{\sigma_1}$ is either the $(d-1)$-simplex immediately below p in $b(\sigma_2)$ or in $S^c_{\sigma_2}$. Thus, we explain how to answer a query in $b(\sigma_2)$. As in [35], we use an auxiliary structure to answer the query in $b(\sigma_2)$. However, we will use a different (and arguably simpler) method. Using the algorithm of Amato *et al.* [4], we compute the intersection (in σ_1) of the halfspaces containing σ_2 whose bounding hyperplanes span the $(d-1)$-simplices in $b(\sigma_2)$. Next, we vertically project the resulting intersection $b(\sigma_2)^\cap$ to obtain a $(d-1)$–dimensional arrangement of size $O(n^{\lfloor d/2 \rfloor}_{i-2})$, and build a point location structure for it. This point location structure is used to find the simplex below p in $b(\sigma_2)$ (see Figure 3).

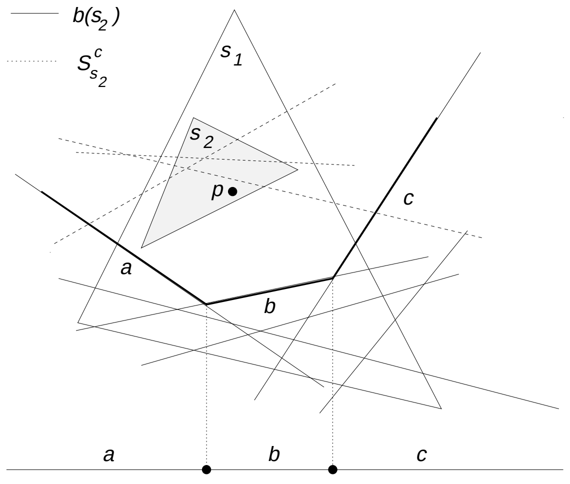

Figure 3:

Since the intersection is constructed hierarchically, a hierarchical $(d-1)$-dimensional point location structure with query time $O(\log n_{i-2})$ can be constructed when the intersection is computed. The point location structures are built in $O(\log n_{i-2})$ time using $O(n^{\lfloor d/2 \rfloor}_{i-2} \log^{O(1)} n_{i-2})$ work (adding $O(r^{\lfloor d/2 \rfloor (d-1)}_{i-1})$ work to each stage of the intersection construction, which does not affect the complexity bounds of the intersection algorithm [4]). Given these auxiliary structures, a vertical ray shooting query for S is answered in $\sum_i \log n_{i-2} = O(\log n)$ time. Using $n^{\lfloor d/2 \rfloor + \delta}_{i-2}$, $0 < \delta < 1$, as an upper bound on the work for $\sigma \in C_i$, the total work of building the auxiliary point location structures is bounded by

$$\begin{aligned}
&\sum_{i=1}^{h} n^{\lfloor d/2 \rfloor + \delta}_{i-2} |C_i| \\
&\leq \sum_{i=1}^{h} n^{\lfloor d/2 \rfloor + \delta}_{i-2} \left\{ C^i \left(\frac{n}{n_{i-1}} \right)^{d-1} + D \left(\frac{1}{n_{i-1}} \right)^{d} k \right\} \\
&= n^{d-1} \sum_{i=1}^{h} C^i n^{1+\delta-\lceil d/2 \rceil + \epsilon(d-1)}_{i-2} + Dk \sum_{i=1}^{h} n^{\delta - \lceil d/2 \rceil + \epsilon d}_{i-2} \\
&= O(n^{d-1} \log^{O(1)} n + k),
\end{aligned}$$

for ϵ chosen sufficiently small so that $\epsilon d < \lceil d/2 \rceil - \delta$.

Point location queries. In [35], a point location query is reduced to a series of d vertical ray shooting queries in faces of $\mathcal{A}(S)$ of decreasing dimension. Using our vertical ray shooting data structures in the point location algorithm of [35] we obtain a query time of $O(\log n)$, and storage and construction costs of $m = O(n^{d-1} \log^{O(1)} n + k)$ and $O(m + k \log n)$, respectively. To obtain the parallel result we note that Pellegrini's [35] construction of the connectivity graph of $\mathcal{A}(S)$ (used in the search) can be parallelized in a straightforward manner yielding the point location structure in $O(\log^2 n)$

time in the EREW PRAM model. The construction uses $O(m + k \log n)$ work. The time is $O(\log^2 n)$ because we construct planar line segment arrangements (Theorem 3.2). The construction also uses a parallel algorithm for finding the connected components of a graph (see, e.g., [13, 25]) (details will be provided in the full paper).

Triangulating non-intersecting $(d-1)$–simplices in $\mathbb{R}^d$. If the simplices in S are interior disjoint, then Pellegrini [35] notes that a slight modification of the method for building the incidence query data structure can be used to construct a triangulation of $\mathcal{A}(S)$ of size $O(n^{d-1})$ in time $O(n^{d-1})$. The hierarchical representation of the triangulation supports point location queries in $O(\log n)$ time. This idea can be used in parallel to construct a triangulation of $\mathcal{A}(S)$ of size $O(n^{d-1})$ in $O(\log n)$ time using $O(n^{d-1})$ work in the EREW PRAM model (details will be provided in the full paper).

References

[1] P. Agarwal, J. Matoušek, and O. Schwarzkopf. Computing many faces in arrangements of lines and segments. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994.

[2] P. K. Agarwal. Partitioning arrangements of lines: II. Applications. *Discrete Comput. Geom.*, 5:533–573, 1990.

[3] P. K. Agarwal. Geometric partitioning and its applications. In J. E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the DIMACS special year*. Amer. Math. Soc., 1991.

[4] N.M. Amato, M.T. Goodrich, and E.A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93)*, pages 683–694, 1994.

[5] I.J. Balaban. An optimal algorithm for finding segment intersections. In *Proc. 11th Annu. ACM Sympos. Computational Geometry*, 1995.

[6] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[7] Hervé Brönnimann, Bernard Chazelle, and Jiří Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93)*, pages 400–409, 1993.

[8] B. Chazelle. Reporting and counting segment intersections. *J. Comput. Syst. Sci.*, 32:156–182, 1986.

[9] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[10] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[11] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[12] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments. *SIAM J. Comput.*, 22:1286–1302, 1993.

[13] K. W. Chong and T. W. Lam. Finding connected components in $o(\log n \log\log n)$ time on the EREW PRAM. In *Proc. 4th Annu. ACM-SIAM Symp. Discrete Algorithms*, pages 11–20, 1993.

[14] K. L. Clarkson. Randomized geometric algorithms. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 117–162. World Scientific, Singapore, 1992.

[15] K. L. Clarkson, R. Cole, and R. E. Tarjan. Erratum: Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.*, 2(3):341–343, 1992.

[16] K. L. Clarkson, R. Cole, and R. E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. *Internat. J. Comput. Geom. Appl.*, 2(2):117–133, 1992.

[17] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[18] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 105–114, 1994.

[19] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 1–10, 1994.

[20] M. T. Goodrich. Intersecting line segments in parallel with an output-sensitive number of processors. *SIAM J. Comput.*, 20:737–755, 1991.

[21] M. T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 73–82, 1993.

[22] M. T. Goodrich, S. Shauck, and S. Guha. Parallel methods for visibility and shortest path problems in simple polygons. *Algorithmica*, 8:461–486, 1992.

[23] M. T. Goodrich, S. Shauck, and S. Guha. Addendum to "parallel methods for visibility and shortest path problems in simple polygons". *Algorithmica*, 9:515–516, 1993.

[24] L. J. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.

[25] S. Halperin and U. Zwick. An optimal randomized logarithmic time connectivity algorithm for the erew pram. In *Proc. ACM Symp. Parallel Algorithms and Architectures*, pages 1–10, 1994.

[26] A. Joffe. On a set of almost deterministic k-independent random variables. *Annals of Probability*, 2:161–162, 1974.

[27] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

[28] J. Matoušek. Epsilon-nets and computational geometry. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 69–89. Springer-Verlag, 1993.

[29] J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 478–484, 1993.

[30] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[31] K. Mehlhorn, M. Sharir, and E. Welzl. Tail estimates for the efficiency of randomized incremental algorithms for line segment intersection. *Comput. Geom. Theory Appl.*, 3:235–246, 1993.

[32] J. S. B. Mitchell. On computing a single face in an arrangement of line segments. Manuscript, School Oper. Res. Indust. Engrg., Cornell Univ., Ithaca, NY, July 1990.

[33] K. Mulmuley. A fast planar partition algorithm, I. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 580–589, 1988.

[34] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, New York, 1993.

[35] Marco Pellegrini. On point location and motion planning among simplices. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, 1994.

[36] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete Comput. Geom.*, 3:123–136, 1987.

[37] J. H. Reif, editor. *Synthesis of Parallel Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.

[38] John H. Reif and Sandeep Sen. Optimal randomized parallel algorithms for computational geometry. *Algorithmica*, 7(1):91–117, 1992.

[39] C. Rüb. Computing intersections and arrangements for red-blue curve segments in parallel. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 115–120, 1992.

[40] A. Wiernik and M. Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete Comput. Geom.*, 3:15–47, 1988.

[41] A. C. Yao and F. F. Yao. A general approach to D-dimensional geometric queries. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, pages 163–168, 1985.