Chapter 1

Introduction

Computational geometry emerged as a scientific discipline about fifteen years ago. In its initial stages, it evolved around planar searching and sorting problems. Typical problems were convex hull, point location, closest point, Voronoi diagram, and orthogonal range searching, all of which can be easily cast into generalized two-dimensional searching and sorting. Gradually, the field expanded to include problems whose successful solution requires the application of sophisticated mathematical tools drawn from algebra, probability, topology, differential geometry, extremal graph theory and many other fields. Now a days the difference between combinatorial and computational geometry is becoming obscure. In many problems the combinatorial and algorithmic questions are so interwined that it is nearly impossible to separate one from the other, and progress in one of them leads to a satisfactory solution of the other. One such area in computational geometry, where algorithmic questions rely heavily on various mathematical properties, is the study of *arrangements* of lines or curves in the plane, or of hyperplanes or more general surfaces in higher dimensions. Many fundamental geometric problems can be formulated in terms of arrangements.

In this book, we study several problems involving arrangements of lines, segments, or curves in the plane. The book has two aims: to answer some combinatorial as well as algorithmic questions related to arrangements and to obtain fast algorithms for several other problems by formulating them in terms of arrangements.

We begin by defining arrangements and discussing some of the known results on arrangements. The remainder of the introduction gives a brief overview of the results described in the later chapters and of the tools used to obtain these results — Davenport-Schinzel sequences, random sampling and deterministic partitioning, and spanning trees with low stabbing number.

1.1 Arrangements

Let $\mathcal{L}$ be a finite set of lines in the Euclidean plane $\mathbb{R}^2$. The *arrangement* $\mathcal{A}(\mathcal{L})$ of $\mathcal{L}$ is the partition of the plane formed by $\mathcal{L}$, whose *vertices* are intersection points of lines in $\mathcal{L}$, *edges* are maximal connected (open) portions of lines in $\mathcal{L}$ not containing any vertex, and *faces* are maximal connected (open) portions of the plane not meeting any vertex or edge (see figure 1.1). The notion of arrangement can be easily generalized to curves in the plane, or to hyperplanes or hypersurfaces in higher dimensions. In $\mathbb{R}^d$, the arrangement $\mathcal{A}(\mathcal{H})$ of a finite family $\mathcal{H}$ of $(d-1)$-dimensional hyperplanes consists of open convex d-dimensional polyhedra (also referred to as *cells*) and various relatively open convex k-dimensional polyhedral faces bounding them, for $0 \leq k \leq d-1$.

The *combinatorial complexity* of an arrangement in the plane is the number of its vertices, edges and faces, and in higher dimensions it is the total number of faces of all dimensions. It is easy to see that an arrangement of n lines in $\mathbb{R}^2$ lying in general position has $\Theta(n^2)$ vertices, edges and faces. By generalizing this result to higher dimensions, it is shown that the combinatorial complexity of an arrangement of n hyperplanes in $\mathbb{R}^d$ is $\Theta(n^d)$. If Γ is a set of n hypersurfaces in $\mathbb{R}^d$, such that every d of them intersect in at most $O(1)$ points, e.g. graphs of $(d-1)$–variate polynomials of bounded degree, then the combinatorial complexity of $\mathcal{A}(\Gamma)$ is also $\Theta(n^d)$.

Arrangements of lines and "pseudo-lines" have been studied from various mathematical points of view for a very long time (see Steiner [123], von Staudt [122], Sylvester [126], Levi [85], Melchior [97] for some early works). The first systematic exposition of arrangements is due to Grünbaum [69], [71]. Arrangements in higher dimensions have received considerably less attention, and not much is known about them except for some basic results.

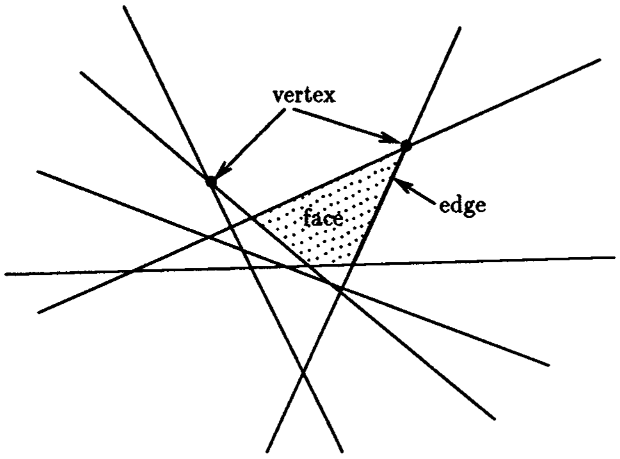

Figure 1.1: An arrangement of lines

Interested readers can find details on d-dimensional arrangements in [69], [70]. The recent book of Edelsbrunner [52] also provides an extensive survey of known results in this area.

The algorithmic study of arrangements, however, has started only recently, primarily because of their significance in many fundamental geometric problems. The arrangement of a set of n lines in the plane can be computed in time $O(n^2 \log n)$ using a line sweep approach (see [115]); this has been improved to $O(n^2)$ by Edelsbrunner et al. [62] (see also [30]). In fact the algorithm of [62] computes, in time $O(n^d)$, the arrangement of a given set of n hyperplanes in $\mathbb{R}^d$. Recently, Edelsbrunner and Guibas [54] presented another $O(n^2)$ algorithm to construct the arrangement of a given set of n lines using "topological sweep", which requires only $O(n)$ working storage. Constructing arrangements of curves is much more difficult; Edelsbrunner et al. [56] have generalized the approach of [62] to construct arrangements of arcs of some simple shape in the plane (in particular, it is required that any pair of these arcs intersect in at most some constant number of points). Under the assumption that certain basic operations on arcs, such as computing intersection points of a pair of arcs and points of vertical tangency of a given arc, can be performed in $O(1)$ time, the running time of their algorithm is almost (slightly superlinear–) quadratic. But unlike the algorithm of [62], this approach does not generalize to constructing arrangements of surfaces in higher dimensions.

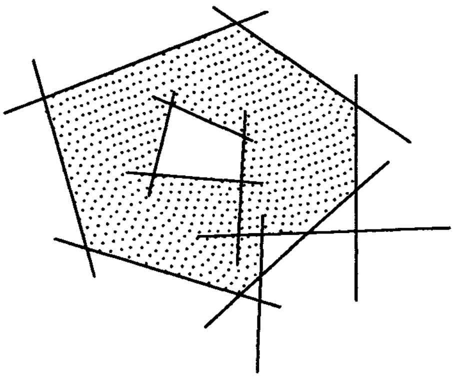

Figure 1.2: A non-convex face in an arrangement of segments

In several applications it suffices to compute a single face or few faces in an arrangement. For arrangements of lines it is obvious that a single face can have at most n edges, because each face is convex, and a single face can be easily computed in $O(n \log n)$ time using a divide and conquer algorithm [114]. But faces in arrangements of segments or of general arcs are not necessarily convex (see figure 1.1), and in fact the result of Wiernik and Sharir [133] shows that a single face in an arrangement of n segments can have $\Omega(n\alpha(n))$ edges in the worst case (see also [121]), where $\alpha(n)$ is a functional inverse of Ackermann's function (see Chapter 2 for details on Ackermann's function). A matching upper bound has been obtained in Pollack et al. [105] using Davenport-Schinzel sequences, which has been extended to general arcs by Guibas et al. [75]. Guibas et al. have also given an efficient algorithm to compute a single face in arrangements of arcs under an appropriate model of computation.

In some other applications, we are required to compute $m > 1$ faces; one such application is described in Chapter 3. If $K(m, n)$ denotes the maximum number of edges in m distinct faces of arrangements of n lines, then obviously $K(m, n) = O(mn)$. But a somewhat surprising result of Canham [16] shows that $K(m, n) = O(m^2 + n)$, implying that $K(\sqrt{n}, n)$ is not larger than $K(1, m)$ by more than a constant factor. Since then a great deal of effort has been made to obtain tight bounds on $K(m, n)$ (see [52]). Edelsbrunner and Welzl [66] proved that $K(m, n) = \Omega(m^{2/3}n^{2/3} + n)$, and very recently Clarkson et al. [38] have shown that indeed $K(m, n) = \Theta(m^{2/3}n^{2/3} + n)$. Similar, but not fully tight, bounds on the complexity of many faces in arrangements of segments have been obtained in [59], [4]. Edelsbrunner et al. [59] have given randomized algorithms to compute m distinct faces in arrangements of lines or segments (see also [55]).

Chapter 5 describes an $O(m^{2/3}n^{2/3} \log^{5/3} n \log^{\omega/3} \frac{m}{\sqrt{n}} + (m+n) \log n)$ deterministic algorithm for computing the faces in arrangements of n lines in the plane, containing m given points (cf. Section 5.3), where ω is a constant < 3.33. We also present an $O(m^{2/3}n^{2/3} \log n \log^{\omega/3+1} \frac{n}{\sqrt{m}} + n \log^3 n + m \log n)$ algorithm to compute the faces in arrangements of n segments in the plane, containing m given points (cf. Section 5.4). These algorithms are deterministic, faster than previously known (randomized) algorithms, and optimal up to a polylog factor.

These results use Davenport-Schinzel sequences, random sampling and other techniques, which we will discuss in Section 1.2.

A very simple example of a geometric problem that can be formulated in terms of arrangements is reporting or counting intersections in a given collection $\mathcal{G}$ of segments, because this problem is equivalent to asking for reporting/counting the vertices of $\mathcal{A}(\mathcal{G})$. A more interesting application area is *motion planning*, in which, given a set of obstacles in the plane, or in space, and a moving object B, we wish to plan a collision-free path for B from a given initial placement to a specified final placement. A well known technique for solving this problem is to represent the placements of B at which it makes contact with obstacles as a collection Γ of surfaces in an appropriate parametric "configuration space", so that B can be moved from the initial placement to the final placement without colliding with the obstacles if and only if the points of the configuration space representing the initial and final placements of B lie in the same cell of $\mathcal{A}(\Gamma)$. Thus the problem can be reduced to computing a single cell in $\mathcal{A}(\Gamma)$, and locating a point in it [105], [75]. Using this observation and efficient algorithms to compute a single cell in arrangements of arcs, a number of fast algorithms have been developed for various motion planning problems (see e.g. [75], [5]).

Many problems in computational geometry can be reduced to computing the *lower envelope* of the arrangement of a given collection Γ of surfaces or surface-patches in $\mathbb{R}^{d+1}$. We can view each surface $\gamma_i \in \Gamma$ as the graph of a partially defined function $x_{d+1} = f_i(x_1, \ldots, x_d)$, whose domain is the projection of γ_i on the hyperplane $x_{d+1} = 0$. The lower envelope $\mathcal{M}_\Gamma(x_1, \ldots, x_d)$ of $\mathcal{A}(\Gamma)$ is defined as the pointwise minimum of these functions, that is

$$\mathcal{M}_\Gamma(x_1, \ldots, x_d) = \min_{1 \le i \le n} \{f_i(x_1, \ldots, x_n)\}.$$

The combinatorial structure of $\mathcal{M}_\Gamma$ is a cell complex in $\mathbb{R}^d$ such that within each cell the lower envelope is attained by a single function f_i (see figure 1.1). This is basically the cell decomposition of the orthogonal projection of $\mathcal{M}_\Gamma$ on $x_{d+1} = 0$. Similarly, we can define the *upper envelope* of the arrangement of a collection of surfaces.

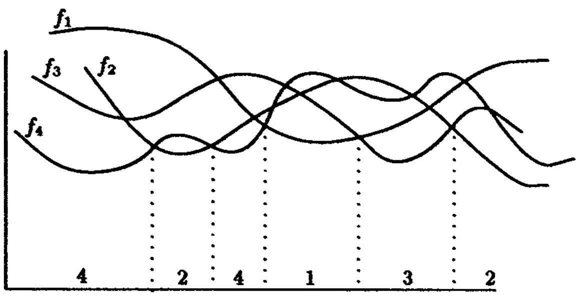

Figure 1.3: Lower envelope of a set of curves

It has been shown in [63] that Voronoi diagrams and all of its generalizations in $\mathbb{R}^d$ can be viewed as the lower envelope of arrangements of a collection of surfaces in $\mathbb{R}^{d+1}$, where the surfaces depend on the distance function used to define the Voronoi diagram, and on the shape of sites for which we want to compute the diagram. Other problems that can be reduced to computing the lower envelope of arrangements include transversals [64], [79], visibility [44], motion planning [83], and various questions in dynamic computational geometry [6]. Expressing problems in terms of arrangements also lead to efficient algorithms for computing stabbing lines for a set of segments in the plane [61], computing minimum volume simplices for a set of points in $\mathbb{R}^d$ [62], half-plane range searching, and many others.

Another application studied in this thesis is *red-blue intersection detection* problem, which is defined as follows: Given a collection Γ of n "red" Jordan arcs and another collection Γ' of m "blue" Jordan arcs, determine whether any arc of Γ intersects any arc of Γ'. Chapter 3 describes several efficient algorithms for detecting a red-blue intersection, by formulating it in terms of arrangements of arcs. For the general case, we reduce the problem to that of computing up to $2(m+n)$ faces of $\mathcal{A}(\Gamma)$ and $\mathcal{A}(\Gamma')$. In some special cases the running time can be significantly improved, e.g. if the arcs in Γ' form the boundary of a simply connected region, then the problem can be reduced to detecting an intersection between Γ' and a single cell of $\mathcal{A}(\Gamma)$. These algorithms rely heavily on Davenport-Schinzel sequences, which we describe and analyze in the next section.

Recently several researchers have considered the following decomposition problem involving arrangements of hyperplanes:

> *Given a set $\mathcal{H}$ of n hyperplanes in $\mathbb{R}^d$, and a parameter $1 < r < n$, partition the space into $O(r^d)$ simplices, each of which intersects $O(\frac{n}{r})$ hyperplanes of $\mathcal{H}$.*

This decomposition algorithm has turned out to be useful in a variety of geometric problems involving arrangements of lines in the plane, or hyperplanes in higher dimensions. The existence of a decomposition with somewhat weaker properties was originally proved in [78], [36], using probabilistic arguments. Later Chazelle and Friedman [26] gave a deterministic algorithm that runs in polynomial time. For higher dimensions, this is still the best known algorithm, but in the case of two-dimensions, faster algorithms have been developed by exploiting several properties of arrangements of lines in the plane. One such algorithm with a time complexity of $O(nr^2 \log^2 r)$ has been developed by Matoušek [89].

In Chapter 4 of this book we present a deterministic algorithm which runs in roughly nr time, and thus is almost an order of magnitude faster than that of Matoušek in terms of r. This improvement is quite crucial in many geometric problems (see Chapter 4 for details). We demonstrate the versatility and significance of our partitioning algorithm by describing fast solutions for numerous geometric problems, based on it. These applications include computing many faces in arrangements of lines or segments, counting intersections in a set of segments, computing incidences between points and lines, and computing spanning trees of low stabbing number.

In Chapter 5 we present fast algorithms for a variety of problems involving arrangements of segments, using a data structure called *spanning tree with low stabbing number*. The most interesting problem considered there is *ray shooting in arrangements of segments*: given an arrangement of segments in the plane, preprocess it so that the first intersection of a query ray with one of the given segments can be quickly reported. Ray shooting has applications in many other problems, including hidden surface removal [102], and computation a single cell in arrangements of triangles in $\mathbb{R}^3$ [5]. We will describe ray shooting and other related problems in greater detail in Section 1.4.

1.2 Davenport-Schinzel Sequences

Davenport-Schinzel sequences have a strong combinatorial structure that arises in many geometric problems. They were originally introduced by Davenport and Schinzel [48] in connection with solutions to linear differential equations. A sequence, $U = (u_1, u_2, \ldots, u_m)$, composed of n distinct symbols is a *Davenport-Schinzel* sequence of order s, for $s > 0$ (an (n, s)–DS sequence for short), if it satisfies the following two conditions:

(i) $\forall i < m$, $u_i \neq u_{i+1}$.

(ii) There do not exist $s+2$ indices $1 \le i_1 < i_2 < \cdots < i_{s+2} \le m$ such that

$$\begin{aligned} u_{i_1} &= u_{i_3} = \cdots = a \\ u_{i_2} &= u_{i_4} = \cdots = b \end{aligned}$$

and $a \neq b$.

These sequences characterize the combinatorial structure of the lower envelope of arrangements of n x-monotone curves in the plane (or n univariate functions), in the sense that the "lower-envelope sequence" of indices of curves as they appear along the envelope is a (n, s)–DS-sequence, where s is the maximum number of intersections between any pair of curves. Moreover, Atallah [6] proved that, for any given (n, s)–DS sequence, there exists a collection of n x-monotone curves, whose lower envelope sequence is the same as the given Davenport-Schinzel sequence.

Let $\lambda_s(n)$ denote the maximum length of a (n, s)–DS sequence. It is easy to see that $\lambda_1(n) = n$ and $\lambda_2(n) = 2n - 1$. Hart and Sharir [77] proved that $\lambda_3(n) = \Theta(n\alpha(n))$, which implies that, for $s \ge 3$, $\lambda_s(n)$ is indeed non-linear. Our results in Chapter 2 prove that $\lambda_4(n) = \Theta(n2^{\alpha(n)})$, and

$$\begin{aligned} \lambda_{2s+2}(n) &= O\left(n \cdot 2^{(\alpha(n))^{s(1+o(1))}}\right) \\ \lambda_{2s+3}(n) &= O\left(n \cdot 2^{(\alpha(n))^{s(1+o(1))} \cdot \log \alpha(n)}\right) \\ \lambda_{2s+2}(n) &= \Omega\left(n \cdot 2^{K_s(\alpha(n))^s}\right) \end{aligned}$$

where, $K_s = \dfrac{1}{(s-1)!}$. Thus our lower and upper bounds are almost tight. For even values of s they are almost identical except for the constant K_s and lower order terms, but for odd values of s, the gap is slightly larger. Our upper bounds imply that $\lambda_s(n)$, for any fixed value of s, is almost linear, because $\alpha(n) \le 5$ for all practical purposes (see Chapter 2). The proofs presented in this chapter are fairly complex, and exploit several properties of Ackermann's function and related functions.

This close relationship between Davenport-Schinzel sequences and lower envelopes of arrangements has made them a tool of of central significance in computational and combinatorial geometry, and numerous applications of

DS–sequences have been obtained, in diverse areas including arrangements [75], [103], [53], [57], [95], motion planning [101], [83], [81], shortest path [9], [35], visibility [44], transversals [64], [57], Voronoi diagrams [57] and many more. Guibas et al. [75] showed that the maximum number of edges in a single cell of an arrangement of n arcs is $O(\lambda_{s+2}(n))$, where s is the maximum number of intersections between any pair of arcs; if all arcs are closed curves or unbounded arcs, the complexity reduces to $O(\lambda_s(n))$. Using Davenport-Schinzel sequences, Edelsbrunner et al. [56] were able to prove that the m distinct faces in arrangements of n arcs have at most $O(m\sqrt{\lambda_{s+2}(n)})$ edges. Guibas et al. have described an $O(\lambda_{s+2}(n)\log^2 n)$ algorithm to compute a single cell in arrangements of n arcs, where s is as above; the algorithm can be generalized for computing more than one face (see [59]). Based on the result of Hart and Sharir [77], Pach and Sharir [103] have shown that the maximum number of facets in the the lower envelope of arrangements of n d-simplices in $\mathbb{R}^{d+1}$ is $\Theta(n^d\alpha(n))$ (see also [53]). An optimal algorithm for computing the lower envelope of arrangements of n triangles in $\mathbb{R}^3$ is described in [57].

Applications of DS-sequences in motion planning include an algorithm to plan a collision free path (allowing rotation) for a convex k-gon in presence of polygonal obstacles (cf. [81]), an efficient algorithm to separate a simple m-gon from another simple n-gon (cf. [105]), and a fast procedure to compute a shortest path in $\mathbb{R}^3$ in the presence of two convex polyhedral obstacles.

In Chapter 3, we describe efficient algorithms, using Davenport-Schinzel sequences, for the red-blue intersection detection problem defined in the previous section. We start with an almost linear algorithm for a special case, where the arcs in Γ' form the boundary of a simply connected region. Then we show that such an algorithm provides a fast procedure for the collision detection problem: given a set O of obstacles, a moving object B and a path Π, determine whether B collides with any obstacle of O while moving along the path Π. Some of the motion planning problems can be reduced to the collision detection problem by showing that there exists a canonical motion such that B can be moved from its initial position to a given final position, if and only if the canonical motion is collision free. One such example is given in Maddila and Yap [87], who considered the problem of moving a simple n-gon in an L-shaped corridor. They gave an $O(n^2)$ algorithm for this problem,