Geometric Structures in Computational Geometry[1]

Herbert Edelsbrunner[2]

1 Introduction

Computational geometry is a flourishing field within computer science whose rapid growth makes it exceedingly difficult – but all the more important – to provide a representative overview of the area that is coherent and still gives an impression of its richness. In this paper we attempt a uniform approach to computational geometry, but one that allows the discussion of different issues such as underlying combinatorial extremum problems and implementation issues unique to geometric programming. Due to space limitations we cannot seriously attempt to approximate a comprehensive survey of the area – instead, we describe a selection of fundamental problems and new results.

The central notion in our presentation is that of an *arrangement* defined by a collection of geometric objects in some space. Traditionally, the term "arrangement" was mainly used for "finite set of lines in the plane" or "planes in three dimensions", or "hyperplanes[3] in $d \geq 4$ dimensions" (see [Gru67], [Zas75]). We will adopt a broader use of the term and also talk about arrangements of circles and line segments in the plane, of triangles in three dimensions, etc. In addition, we will find it convenient to use "arrangement" as a synonym for the cell complex that is the dissection of space defined by the geometric objects[4] (see citeEdelsbrunner).

Section 2 will discuss arrangements of lines, planes, and hyperplanes in appropriate detail. Due to limitations in space, we will restrict ourselves to problems and results that

[1]Research on this paper was supported by the National Science Foundation under grant CCR–8714565.

[2]Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, Illinois 61801, USA.

[3]A *hyperplane* in d–dimensional Euclidean space is an affine subspace whose dimension is $d-1$, one less than the dimension of the embedding space.

[4]E.g. a set of n circles cuts the plane into at most $2\binom{n}{2}+2$ regions, $4\binom{n}{2}$ edges, and $2\binom{n}{2}$ vertices.

are relevant to later sections of this paper. In Section 3 we make an attempt to illustrate the generality of this kind of arrangement by explicating their relationship to well–studied geometric concepts such as point configurations and Voronoi diagrams. The relations are realized by geometric transforms that map different geometric objects to each other.

When one implements a geometric algorithm one has to worry first about the representation of the objects and the treatment of primitive operations. On the lowest level, a geometric object is always represented by a conglomerate of numbers. For example, a plane in three dimensions can be represented by a sequence (a, b, c, d) with the understanding that the plane is the set of points (x, y, z) that satisfy

$$ax + by + cz + d = 0.$$

A primitive operation takes as input a constant number of such represented objects and computes some constant size output that depends on the input objects and their relative position. Thus, primitive operations are inherently numerical with all the traditional problems that come with numerical computations. Section 4 will argue that the numerical parts of many geometric algorithms are amenable to symbolic computations that can help in coping with robustness problems common to purely numerical implementations of geometric primitives.

Section 5 will return to a higher level discussion – the subjects are arrangements defined by line segments. Arrangements of line segments are more intricate structures than arrangements of lines since the boundedness of the line segments creates non–convex and even non–simply connected regions. Various interesting combinatorial and algorithmic results as well as applications of line segment arrangements will be addressed. A natural generalization of line segment arrangements to three dimensions is the notion of an arrangement of triangles in three–dimensional space. Combinatorial and algorithmic issues of such arrangements are considered in Section 6.

Underlying the computational complexity of every geometric problem is what we call its *combinatorial complexity*. For example, the amount of time is takes to construct the view of a scene of (possibly intersecting) triangles[5] is lower bounded by the number of vertices, edges, and regions of the view. This illustrates the importance of a combinatorial analysis of the structure but fails to reflect the full significance of such an analysis. Section 7 argues that the understanding of the combinatorial properties of a structure is absolutely essential in the design of a good algorithm that constructs the structure. Indeed, this relates to the issue of the affinity between proof and algorithm that can be traced to virtually all parts of modern computer science.

Today, computational geometry is rich in results and broad in the collection of topics it considers. Still, there are a number of directions that deserve more attention than they receive today. The exploration of such directions will decide whether computational geometry can live up to the promise of bridging the world of theoretical computer science and the

[5]The *view* is defined as the subdivision of a plane normal to the viewing direction that is obtained by projecting the parts of the triangles visible from the viewpoint.

world of applications. One of those directions is the generalization of the current methods overwhelmingly designed for piecewise linear objects to multivariate polynomials of degree higher than one.

2 Arrangements of Lines, Planes, and Hyperplanes

If we draw n lines h_i in the plane we create a dissection of the plane into a finite collection of regions bounded by edges and vertices. The *vertices* are the intersections of the lines, the *edges* are the connected components of the lines after removing the vertices, and the *regions* are the connected components of $E^2 - \bigcup_{i=1}^{n} h_i$, the plane minus the n lines (see Figure 1). This dissection is called the *arrangement* of the h_i, for $1 \leq i \leq n$. More specifically, the

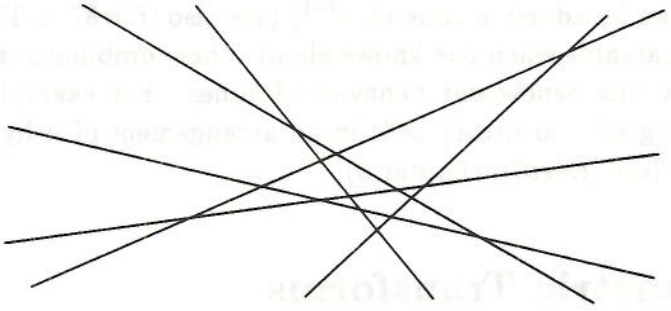

Figure 1: A simple arrangement of six lines decomposing the plane into 22 regions, 36 edges, and 15 vertices.

arrangement of the h_i dissects E^2 into at most $\binom{n}{2} + n + 1$ regions, $2\binom{n}{2} + n$ edges, and $\binom{n}{2}$ vertices. Those bounds on the number of regions, edges, and vertices are tight and realized by all so–called "simple" arrangements: an arrangement of lines is *simple* if any two lines intersect in a point and no point belongs to more than two lines.

In $d \geq 3$ dimensions, the asymptotic order of faces[6] in an arrangement of n hyperplanes in E^d is $\Theta(n^d)$ which is therefore a lower bound on the amount of time required to construct it. An algorithm whose time–complexity matches the lower bound constructs the arrangement of the hyperplanes $h_1, h_2, \ldots, h_n$ incrementally, that is, adds h_i to the arrangement defined by h_1 through h_{i-1}.

Algorithm Line arrangement.
Construct the arrangement of $h_1, h_2, \ldots, h_d$.
for $i := d+1$ **to** n **do**
    add h_i to the arrangement of $h_1, h_2, \ldots, h_{i-1}$.
endfor.

[6] "Face" is a generic term for the elements of an arrangement. There are faces of all dimensions $0 \leq k \leq d$; 0–dimensional faces are called *vertices*, 1–dimensional faces are *edges*, and d–dimensional faces are termed *cells* (or *regions* if $d = 2$).

In the first step, the algorithm builds the arrangement of d hyperplanes which is (combinatorially) unique and dual to the graph defined by the hypercube in d dimensions. This step guarantees that the general step does not need to add a hyperplane to a vertex–free arrangement. We refer to Section 4 for a brief account of a method that avoids all other kinds of special cases in the construction too. Details about the data structure used to represent the arrangement and the different cases that occur can be found in [EOS86] and [Ede87], chapter 7.

The reason for the efficiency of the above algorithm (as mentioned, it is asymptotically optimal) is a combinatorial property of so–called zones in arrangements. A *zone* in an arrangement consists of all cells intersecting a new hyperplane (e.g. the one one that is about to be added). We define the *combinatorial complexity* of a zone as the number of faces bounding its cells. Using a sweep argument, [EOS86] proves that a zone in an arrangement of n hyperplanes in E^d has combinatorial complexity $O(n^{d-1})$ and thus a new hyperplane can be added in time $O(n^{d-1})$ (see also [Ede87]). This combinatorial result is all the more remarkable when one knows about other combinatorial properties of arrangements that contrast this benevolent behavior of zones. For example, the maximum number of faces bounding n^{d-1} arbitrary cells in an arrangement of n hyperplanes in E^d is known to be $\Omega(n^{d-2/3})$ (see [Ede87], chapter 6).

3 Geometric Transforms

Perhaps the best known and also most useful geometric transform is the dual correspondence between points and hyperplanes in E^d. We illustrate this correspondence in two dimensions. Consider the following problem defined for n points $p_1, p_2, \ldots, p_n$ in the plane:

for each p_i sort the other points in angular order around p_i.

For convenience, we assume that no three points are collinear and, for technical reasons, that no two points lie on a common vertical line. The output of this problem consists of n lists of length $n-1$ each. We need $\Omega(n^2)$ time just to produce the lists – so is it possible to do the entire computation in $O(n^2)$ time? This seems unlikely when one remembers that $\Omega(n \log n)$ is a lower bound for the amount of time needed to construct one such list. On the other hand, there might be relations between the n lists that help us saving some time.

Consider the following transform that maps a point to a line and vice versa, that is, a point $p = (\pi_1, \pi_2)$ is mapped to the line $p^* : y = \pi_1 x - \pi_2$, and a non–vertical line ℓ is mapped to a point ℓ^* so that $(\ell^*)^* = \ell$. We see that the transform is an involution by definition. The two important properties of the transform are *incidence preservance*:

$p \in \ell$ if and only if $\ell^* \in p^*$,

and *order preservance*:

p is vertically above ℓ if and only if ℓ^* is vertically above p^*.

Using those two properties one can show that the order of the points around p_i is essentially the same as the sequence[7] of dual lines intersecting p_i^*. Thus, the n sorted sequences can be derived from the arrangement of the dual lines in time $O(n^2)$. Since this arrangement can be constructed in time $O(n^2)$ we have a quadratic algorithm for producing the n sorted lists.

Another example that illustrates the power of geometric transforms and the fundamental role of arrangements involves so–called Voronoi diagrams. Let $s_1, s_2, \ldots, s_n$ be n points in E^2. The *Voronoi diagram* of the s_i is a subdivision of E^2 into n open regions R_i so that $a \in R_i$ if s_i is closer to point a than any s_j, $j \neq i$ (see Figure 2). The Voronoi diagram of the s_i can be derived from an arrangement of n planes obtained via a lifting map from the s_i. For $s_i = (\sigma_{i,1}, \sigma_{i,2})$ let

$$s_i^* : z = 2\sigma_{i,1}x + 2\sigma_{i,2}y - (\sigma_{i,1}^2 + \sigma_{i,2}^2)$$

be the corresponding plane. It is easy to show that if the distance of a point $a = (\alpha_1, \alpha_2)$ from s_i is smaller than from s_j then

$$2\sigma_{i,1}\alpha_1 + 2\sigma_{i,2}\alpha_2 - (\sigma_{i,1}^2 + \sigma_{i,2}^2) > 2\sigma_{j,1}\alpha_1 + 2\sigma_{j,2}\alpha_2 - (\sigma_{j,1}^2 + \sigma_{j,2}^2).$$

In words, the vertical projection of a onto the plane s_i^* lies vertically above the vertical projection of a onto s_j^*. Consequently, the vertical projection onto the xy–plane of the so–called topmost level[8] of the arrangement of the s_i^* equals the Voronoi diagram of the s_i. This is equivalent to saying that we see the Voronoi diagram of the s_i if we look at the s_i^* from the direction of the positive z–axis.

Now, it is true that the Voronoi diagram of the s_i consists only of n regions, and by Euler's relation for planar graphs, it contains at most $3n - 6$ edges and $2n - 4$ vertices, if

[7] In fact, we need to take the sequence of lines that are steeper than p_i^* and the sequence of lines less steep than p_i^*, both sorted by their intersections with p_i^*, and concatenate them to get the angular sequence of the points around p_i.

[8] A face belongs to the *topmost level* if it intersects a vertical line in the topmost intersection of the line with an s_i^*.

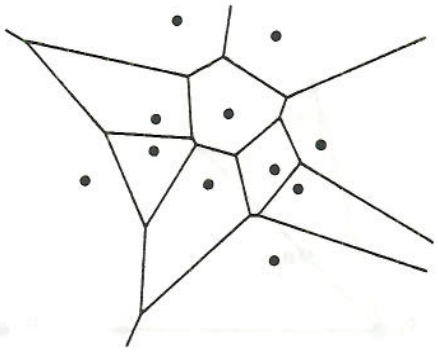

Figure 2: The Voronoi diagram of eleven points in the plane.

$n \geq 3$. Thus, it is apparently wasteful to construct a three–dimensional arrangement (whose combinatorial complexity is $\Theta(n^3)$) just to get the Voronoi diagram. While this point is well taken, the above shows that arrangements are richer in structure than Voronoi diagrams and may therefore be helpful in understanding those diagrams. Indeed, the arrangement must be of cubic combinatorial complexity since it encodes all $n-1$ higher–order Voronoi diagrams of the s_i, and the total number of regions, edges, and vertices of those $n-1$ diagrams is $\Theta(n^3)$, as expected. For details we refer to [Ede87], chapter 13.

4 Geometric Primitives

By a geometric primitive we mean a constant size computation, an elementary operation, an atom of a geometric algorithm. Take for example the sorting problem of Section 3:

> given n points $p_i = (\pi_i, \pi_2)$, for each p_i construct the list of the other points sorted in angular order around p_i.

In order to sort the points we need to be able to perform a comparison, that is, for two points p_j and p_k we need to decide which one comes first in the ordering. One way to solve this problem is to compute for each p_j the angle α_j that is the positive angle between the horizontal half–line emanating from p_i towards the right and the half–line emanating from p_i that goes through p_j (see Figure 3). Now, p_j precedes p_k if $\alpha_j < \alpha_k$.

The problem with the calculation of the α_j is that it is very time–consuming (sure each α_j can be computed in constant time, but the constant is big) and that those computations involve relatively large round–off errors due to the use of trigonometric functions. A way out of this dilemma is the introduction of a primitive that takes three points (p_i, p_j, and p_k) and decides whether they define a left–turn, a right–turn, or whether the three points are collinear (see Figure 3). This can be done by computing the determinant of the matrix

$$\begin{pmatrix} \pi_{i,1} & \pi_{i,2} & 1 \\ \pi_{j,1} & \pi_{j,2} & 1 \\ \pi_{k,1} & \pi_{k,2} & 1 \end{pmatrix}.$$

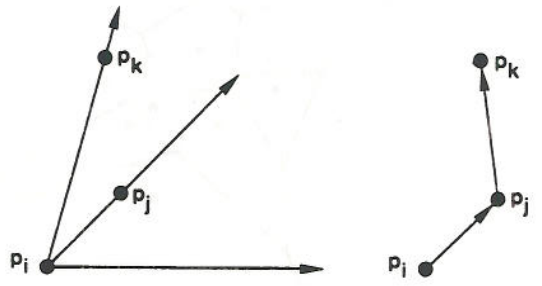

Figure 3: The angular order of points around p_i.

The determinant of this matrix is positive if p_i, p_j, p_k is a left–turn, it is negative if p_i, p_j, p_k is a right–turn, and it is zero if the three points are collinear. We can now sort the points around p_i by, first, separating the points whose y–coordinates are greater than $\pi_{i,2}$ from those whose y–coordinates are smaller than $\pi_{i,2}$, and second, sorting each set by having p_j before p_k if p_i, p_j, p_k is a left–turn.

Now that we have eliminated the need for any trigonometric function, we might have the breath to complain about the special case that occurs when p_i, p_j, and p_k are collinear. How do we break this tie? This question is relatively uncritical for the sorting problem at hands but deserves careful consideration in many other geometric computations. Here is a general purpose technique that eliminates special cases:

$$\text{replace each } p_j \text{ by } p_j(\varepsilon) = (\pi_{j,1} + \varepsilon^{2^{2j-1}}, \pi_{j,2} + \varepsilon^{2^{2j-2}}),$$

for $\varepsilon > 0$ a sufficiently small real number. We treat ε as an indeterminant and solve the determinant problem using that ε is positive and arbitrarily small. The determinant of three points is now a polynomial in ε. Assuming $i < j < k$ we can compute the terms in sequence of decreasing significance (which is by increasing exponent). The first non–zero coefficient of an ε–term decides the sign of the determinant since ε can be chosen small enough so that all later terms are astronomically smaller than the first non–zero term and thus are unable to influence the sign.

$$\det \begin{pmatrix} \pi_{i,1} + \varepsilon^{2^{2i-1}} & \pi_{i,2} + \varepsilon^{2^{2i-2}} & 1 \\ \pi_{j,1} + \varepsilon^{2^{2j-1}} & \pi_{j,2} + \varepsilon^{2^{2j-2}} & 1 \\ \pi_{k,1} + \varepsilon^{2^{2k-1}} & \pi_{k,2} + \varepsilon^{2^{2k-2}} & 1 \end{pmatrix} = \det \begin{pmatrix} \pi_{i,1} & \pi_{i,2} & 1 \\ \pi_{j,1} & \pi_{j,2} & 1 \\ \pi_{k,1} & \pi_{k,2} & 1 \end{pmatrix} -$$

$$-\varepsilon^{2^{2i-2}} \det \begin{pmatrix} \pi_{j,1} & 1 \\ \pi_{k,1} & 1 \end{pmatrix} + \varepsilon^{2^{2i-1}} \det \begin{pmatrix} \pi_{j,2} & 1 \\ \pi_{k,2} & 1 \end{pmatrix} + \varepsilon^{2^{2j-2}} \det \begin{pmatrix} \pi_{i,1} & 1 \\ \pi_{k,1} & 1 \end{pmatrix} + \varepsilon^{2^{2j-2}} \varepsilon^{2^{2i-1}} + \cdots$$

The "perturbation" technique just outlined has the appearance of being expensive for little gain. We disagree. The gain is that the high–level program does not have to bother about special cases at all. The overhead is nominal since in all non–degenerate cases the evaluation of the first three–by–three determinant suffices. In those cases, the new method is no different from the computations carried out without perturbation. If the first determinant is zero, then the tie is broken by computing one or more determinants of sizes at most two–by–two. It is thus fair to say that this case is at most twice as expensive than the first case. Details of this perturbation technique can be found in [EM88].

5 Arrangements of Line Segments

Constructing and reporting all intersecting pairs of a set of n line segments in the plane is one of the most fundamental problems in computational geometry, and it has a host of applications e.g. in computer graphics. Consider, for example, a three–dimensional scene

given by a collection of non–intersecting polytopes and the problem of computing the view from a given eye–position. The view is a projection of the polytopes (or rather their surfaces) onto a plane normal to the viewing direction – the difficulty in this problem is to distinguish between parts that are visible and parts that are invisible to the eye. This is called the *hidden line/surface removal problem.* Notice that the edges of the polytopes give rise to line segments in the projection; for any intersecting pair of line segments we can decide which of the two line segments is closer to the eye–position and which parts of the segments in the neighborhood of the intersection point are hidden.

Due to its importance the intersecting line segments problem was among the earliest problems studied in computational geometry (see [SH76] and [BO79]). Still, it took about 10 years to discover an optimal algorithm for the problem, that is, one that takes $O(n \log n + k)$ time for n line segments with k of the $\binom{n}{2}$ pairs intersecting (see [CE88]). This algorithm not only reports the intersections, it constructs the entire arrangement defined by the line segments plus a few auxiliary vertical edges that guarantee that all regions of the arrangement are convex (see Figure 4). The global structure of the algorithm that constructs the arrangement in optimal time is incremental, that is, it adds a line segment at a time. Life is not as easy as for complete lines though. In order to achieve optimal time, the line segments have to be ordered into a particular sequence, and the insertion of a given line segment is suspended at certain points in time only to be resumed later when it seems appropriate to do so. Still, the algorithm gets by without any fancy data structures supporting its operations.

The boundedness of the line segments makes arrangements of line segments a much richer class of subdivisions than arrangements of lines. For example, a single region in a line arrangement is convex and bounded by at most n edges. In a line segment arrangement, a single region is not necessarily convex and, in general, not even simply connected. The maximum number of edges bounding a single region is $\Theta(n\alpha(n))$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackermann's function (see [PSS88]). This result is based on intricate studies of so–called Davenport–Schinzel sequences (see [HS86]) and their realizability as envelopes of line segments (see [WS88]). The currently best algorithm for constructing a single region in an arrangement of n line segments takes $O(n\alpha(n)\log^2 n)$ time (see [EGS88] which also provides a combinatorial and computational study of collections of regions in line

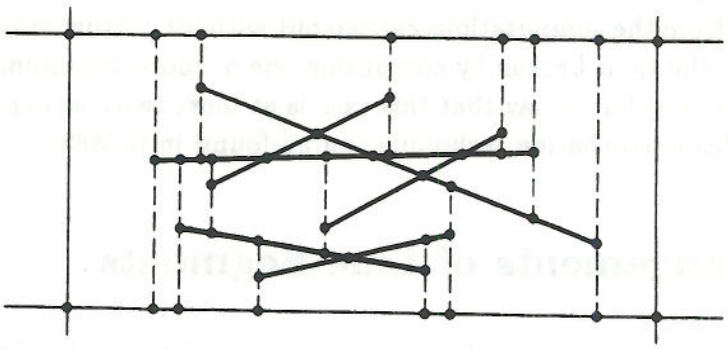

Figure 4: The map defined by six line segments. For convenience, the line segments are enclosed in a rectangular box and vertical edges are extended from their endpoints.

segment arrangements).

Another interesting problem for n line segments is to count the number of intersecting pairs. The currently best method uses partition tree schemes and is described in [GOS88].

6 Envelopes and Hidden Line/Surface Removal

The hidden line/surface removal problem can also be defined for possibly intersecting polytopes in three dimensions. The intersections are not part of the input and have to be computed. As a result, we can expect to get more compact input, or more complicated output when measured in the length of the input. Let us formally define the problem.

> Given a set of n possibly intersecting triangles in three dimensions, construct the view from the direction of the positive z–axis.

The formulation of the problem is such that we can think of each triangle as a partially defined bivariate function[9], $t_i(x,y)$, and the view is the vertical projection onto the xy–plane of the upper envelope of the n functions. Here, the upper envelope of the t_i is defined as $f(x,y) = \max\{t_i(x,y) | 1 \leq i \leq n\}$; if all t_i are undefined for a pair (x,y) then we set $f(x,y) = -\infty$ (see Figure 5).

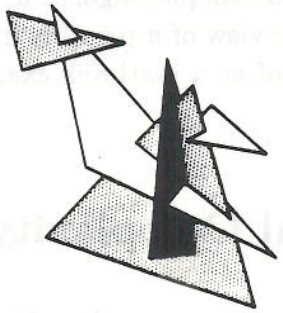

Figure 5: View of five intersecting triangles.

It might help in understanding the problem if we assume for a second that the triangles are pairwise disjoint. In this case, the view is a subgraph of the two–dimensional arrangement of the $3n$ lines obtained by projecting and extending the edges of the n triangles. We see that the combinatorial complexity of the view is thus $O(n^2)$ which is tight as can be seen from the usual checkerboard example.

Now, intersecting triangles intersect along new lines (there are up to $\binom{n}{2}$ such new lines) and it is conceivable that these new lines create a much more complex view. Here is an argument that the view cannot contain many more than a quadratic number of faces, more precisely, the number of faces is $O(n^2\alpha(n))$.

[9] We assume that no triangle is contained in a plane parallel to the z–axis.

- Partition the set of triangles into two sets of $n/2$ triangles each.
- Construct the upper envelope of both sets and project them onto the xy–plane to get the views of the two sets.
- Refine each view (which is a subdivision of the xy–plane) by projecting and extending the $3n$ edges of the triangles. The effect of this step is that the arrangement of $3n$ lines is superimposed onto the view and thus creates a subdivision into convex regions. The number of regions is $O(n^2\alpha(n))$ since a line intersects the view in at most $O(n\alpha(n))$ points (by the result on envelopes of line segments in the plane).
- Get the view of the set of n triangles by merging the two subviews. The merge step does not increase the number of regions since the view within a region of the arrangement is the projection of a convex polytope, and each facet corresponds to a facet in one of the two subviews.

If $R(n)$ is the number of regions of the view of n triangles, we thus get

$$R(n) = 2R(n/2) + O(n^2\alpha(n)) = O(n^2\alpha(n)).$$

By Euler's relation, the number of edges and vertices is of the same order of magnitude as the number of regions, and therefore $O(n^2\alpha(n))$. For further details we refer to [PS88].

We would like to point out a curious aspect of the above upper bound argument: it has the structure of a divide–and–conquer algorithm, and indeed, there is an implementation of the proof that constructs the view of n possibly intersecting triangles in time $O(n^2\alpha(n))$ (see [EGS87]). We view the proof as a textbook example for the affinity between the concepts of proof and algorithm.

7 Combinatorial Complexity

We believe that the earlier sections provide sufficient evidence to convince the reader of the importance of a combinatorial analysis preceding the design of any algorithm. We use this section to highlight a combinatorial problem for arrangements of circles that comes up in the design of an algorithm that constructs some but not all regions of the arrangement. The problem is to determine the maximum number of edges[10] bounding m regions in an arrangement of n circles. We conclude the section by pointing out the close relationship of this problem with some classic extremum problems in combinatorial geometry.

We do the analysis in two steps. For the first step it is convenient to replace the n circles by $2n$ (generalized) disks, where the two disks of a circle c are the two connected components of $E^2 - c$. The intersection of three disks is either connected or disconnected (see Figure 6). Using a planarity argument it is not difficult to show that the intersection of three disks

[10] An *edge* in such an arrangement is a connected component of a circle after removing all points where the circles intersect.

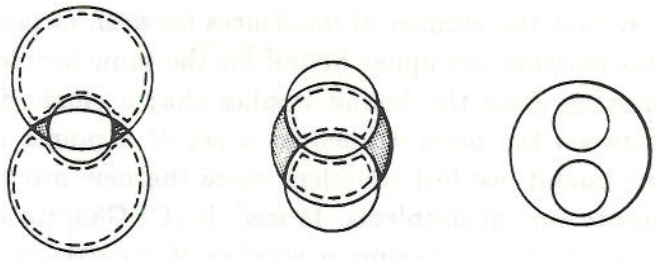

Figure 6: Sample intersections of three disks. If it is connected and bounded by four edges then it is simply connected. If it is disconnected then it consists of two simply connected components bounded by three edges each. If the intersection is connected but not simply connected then the number of bounding edges is at most three.

cannot contain three pairwise disjoint connected sets each touching the boundary circle of each disk. Let us consider the bipartite graph whose nodes are the m regions and the $2n$ disks, and with an arc connecting a region R and a disk d if $R \subseteq d$ and R touches the boundary of d. By the above topological argument we know that this graph contains no $K_{3,3}$. This implies that the number of arcs of the bipartite graph is at most $O(mn^{2/3} + n)$. We next use that a region touching k circles is bounded by at most $6k$ edges and thus obtain $O(mn^{2/3} + n)$ as an upper bound on the total number of edges bounding m regions. As a consequence, the number of bounding edges is $O(n)$ as long as m does not exceed the cube–root of n.

To prepare the second step of the argument, we mark each one of the m regions by a point. We choose a random sample of $f(m,n)$ of the n circles and decompose the cells of the sample arrangement by drawing vertical line segments so that each cell is bounded by two circular and two vertical edges. A probabilistic counting argument (see [Cla87], [Cla88], [HW87], [CEG88]) can be used to show that each one of the $O(f(m,n)^2)$ such trapezoidal cells intersects about $n/f(m,n)$ of the n original circles. We choose $f(m,n) = m^{3/5}/n^{1/5}$. Thus, by our earlier result, the average number of edges bounding each marked region in each cell is about

$$(\frac{n}{f(m,n)})^{2/3} = \frac{n^{4/5}}{m^{2/5}}.$$

Ignoring a couple of interesting technical problems and we thus have a total of $O(m^{3/5}n^{4/5})$ edges.

We remind the reader that we ignored too many details and were too generous in the calculations to produce a serious proof. Nevertheless, the argument can be made to work and the final result is roughly the same as the one shown above. A feature of the sketched proof technique interesting to the combinatorialist is that it can also be used to prove an upper bound on the total number of incidences between m points and n circles in the plane. Now we are getting very close to some old combinatorial problems due to Erdös (see [Erd46], [Erd60]).

Suppose that all circles have the same radius. In this case, the analysis can be tightened

somewhat to show that the number of incidences between m points and n circles is at most $O(m^{2/3}n^{2/3})$. This matches the upper bound for the same problem derived in [SST84]. The problem is interesting since the bound implies that a single distance can occur at most $O(n^{4/3})$ times between the pairs defined by a set of n points in the plane. Although we do not get a new bound, we feel confident since the new proof is simple enough to allow for generalizations to similar problems. Indeed, in [CEG88] we succeed to improve the best previous upper bound on the maximum number of occurrences of a distance in a set of n points in three dimensions. The new bound is roughly $O(n^{3/2})$.

References

[BO79] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.* **C–28** (1979), 643–647.

[CE88] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. Manuscript, 1988.

[CEG88] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir and E. Welzl. Combinatorial complexity bounds for arrangements. Manuscript, 1988.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.* **2** (1987), 195–222.

[Cla88] K. L. Clarkson. Applications of random sampling in computational geometry, II. *In* "Proc. 4th Ann. ACM Sympos. Comput. Geom. 1988", to appear.

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

[EGS87] H. Edelsbrunner, L. J. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discrete Comput. Geom.*, to appear.

[EGS88] H. Edelsbrunner, L. J. Guibas, and M. Sharir. The complexity of many faces in arrangements of lines and of segments. *In* "Proc. 4th Ann. ACM Sympos. Comput. Geom. 1988", to appear.

[EM88] H. Edelsbrunner and E. P. Mücke. Simulation of Simplicity: a technique to cope with degenerate cases in geometric algorithms. *In* "Proc. 4th Ann. ACM Sympos. Comput. Geom. 1988", to appear.

[EOS86] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.* **15** (1986), 341–363.

[Erd46] P. Erdős. On sets of distances of n points. *Amer. Math. Monthly* **53** (1946), 248–250.

[Erd60] P. Erdős. On sets of distances of n points in Euclidean space. *Magyar Tud. Akad. Mat. Kutaló Int. Kozl.* **5**, (1960), 165-169.

[Gru67] B. Grünbaum. *Convex Polytopes.* John Wiley & Sons, London, England, 1967.

[GOS88] L. J. Guibas, M. H. Overmars, and M. Sharir. Intersections, connectivity, and related problems for arrangements of line segments. In preparation.

[HS86] S. Hart and M. Sharir. Nonlinearity of Davenport–Schinzel sequences and of generalized path compression schemes. *Combinatorica* **6** (1986), 151–177.

[HW87] D. Haussler and E. Welzl. ϵ–nets and simplex range queries. *Discrete Comput. Geom.* **2** (1987), 127–151.

[PS88] J. Pach and M. Sharir. The upper envelope on piecewise linear functions and the boundary of a region enclosed by convex plates: combinatorial analysis. *Discrete Comput. Geom.*, to appear.

[PSS88] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete Comput. Geom.* **3** (1988), 123–136.

[SH76] M. I. Shamos and D. Hoey. Geometric intersection problems. *In* "Proc. 17th Ann. ACM Sympos. Found. Comput. Sci. 1976", 208–215.

[SST84] J. Spencer, E. Szemerédi and W. T. Trotter, Jr. Unit distances in the Euclidean plane. In *Graph Theory and Combinatorics*, 293–303, Academic Press, London, 1984.

[WS88] A. Wiernik and M. Sharir. Planar realization of nonlinear Davenport–Schinzel sequences by segments. *Discrete Comput. Geom.* **3** (1988), 15–47.

[Zas75] Th. Zaslavsky. *Facing up to Arrangements: Face-count Formulas for Partitions of Space by Hyperplanes.* Memoirs Amer. Math. Soc. 154, 1975.