Controlled Perturbation for Arrangements of Circles[*]

Extended Abstract

Dan Halperin Eran Leiserowitz

School of Computer Science
Tel Aviv University
{danha,leiserow}@tau.ac.il

Abstract

Given a collection $\mathcal{C}$ of circles in the plane, we wish to construct the arrangement $\mathcal{A}(\mathcal{C})$ (namely the subdivision of the plane into vertices, edges and faces induced by $\mathcal{C}$) using floating point arithmetic. We present an efficient scheme, controlled perturbation, that perturbs the circles in $\mathcal{C}$ slightly to form a collection $\mathcal{C}'$, so that all the predicates that arise in the construction of $\mathcal{A}(\mathcal{C}')$ are computed accurately and $\mathcal{A}(\mathcal{C}')$ is degeneracy free.

We introduced controlled perturbation several years ago, and already applied it to certain types of arrangements. The major contribution of the current work is the derivation of a good (small) resolution bound, that is, a bound on the minimum separation of features of the arrangement that is required to guarantee that the predicates involved in the construction can be safely computed with the given (limited) precision arithmetic. A smaller resolution bound leads to smaller perturbation of the original input.

We implemented the perturbation scheme and the construction of the arrangement and we report on experimental results.

1 Introduction

Computational geometry algorithms often assume general position of the input and the "real RAM" computation model. In the case of an arrangement of circles, general position of the input means that there is no outer or inner tangency between two circles, and that no three circles intersect at a common point (see Figure 1 for a degenerate arrangement). If one wishes to use floating-point arithmetic (to achieve fast running time), then even if the input *is* in general position, round-off errors may cause the algorithm to fail.

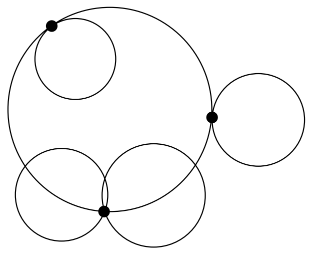
Figure 1: Arrangement of circles with several degeneracies.

Thus, while building the arrangement in an incremental fashion (that is, adding one circle at a time), we will check if there is a potential degeneracy induced by the newly added circle, and if so, we will move that circle, so no degeneracies will occur. The main idea is to carefully relocate the circle — move the circle enough to avoid the degeneracies, but not too much. Depending on the precision of the machine floating-point representation, and some properties of the arrangement to be handled, we determine a bound δ on the magnitude of the perturbation, namely, we guarantee that any input circle will not be moved by a distance greater than δ.

Such a perturbation scheme, as was described above, could be useful for the following reasons: (i) floating-point arithmetic is usually supported by hardware, making computations very fast, (ii) degeneracies are eliminated, thus an algorithm is made easier to analyze and implement, (iii) implementations using exact arithmetic with floating-point filtering, can be sped up, since the perturbation will cause the predicates to be evaluated using the floating-point filters, thus avoiding the use of exact computation.

In many situations, the original input data is inac-

[*]Work reported in this paper has been supported in part by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

curate to begin with (due to, for example, measuring errors or approximate modeling), so the damage incurred by perturbing slightly is negligible.

The predicates that arise in the construction of arrangements of circles include expressions that contain division and square-root operations. Those operation are usually more difficult to handle robustly than addition, subtraction and multiplication.

The perturbation scheme that we follow, controlled perturbation, was first presented in [6] as a method to speed up molecular surface computation. The use of exact computation turned out to be too slow for real time manipulation, so a finite precision method was needed. Controlled perturbation was devised to handle the robustness issues caused by the use of finite precision arithmetic, and to remove all the degeneracies. It was extended in [9], where it was applied to arrangements of polyhedral surfaces. Those arrangements require complex calculations in order to achieve a good perturbation bound.

In [9] (as in [6]), the *resolution bound* (defined in the next section) is assumed to be given. *The resolution bound is a key element in the scheme.* In this work we describe a method for obtaining good resolution bounds, which we anticipate will lead to a better understanding of the method and will open the way to applying the method in other settings.

Related work

Robustness and precision issues have been intensively studied in Computational Geometry in recent years [10].

A prevailing approach to overcoming robustness issues in computational geometry is to use exact computation [7, 13]. Such a strategy gives accurate results, and sometimes even allows the input to be degenerate. When applied naively, exact computation can considerably slow down the performance of a program. One of the possible solutions is to use *filtering* [2, 4, 11]. Typically, the filtering is done at the level of the number type. That is, a predicate is evaluated using exact computation *only* if it cannot be correctly evaluated using finite precision arithmetic. In [12], high-level filtering is done on arrangements of conic arcs; a different approach for computing arrangements of conic arcs is given in [1].

An alternative approach aims to compute robustly with limited precision arithmetic, often by approximating or perturbing the geometric objects [3, 5, 8]. A variety of methods for handling imprecise geometric computations are surveyed in [10]. Controlled perturbation is a method of this type.

2 Overview of the Perturbation Scheme

For an input circle C_i, our algorithm will output a copy C'_i with the same radius but with its center possibly perturbed. We denote by $\mathcal{C}_j$ the collection of circles $\{C_1, \ldots, C_j\}$, and by $\mathcal{C}'_j$ the collection of circles $\{C'_1, \ldots, C'_j\}$.

The input to our algorithm is the collection $\mathcal{C} = \mathcal{C}_n$ of n circles. Each circle C_i is given by the coordinates of its center X_i, Y_i and its radius R_i (we assume that all the input parameters are representable as floating-point numbers with the given precision). The input consists of two additional parameters: (i) the machine precision p, namely the length of the mantissa in the floating-point representation, and (ii) an upper bound on the absolute value of each input number X_i, Y_i and R_i. The perturbation scheme transforms the set $\mathcal{C}$ into the set $\mathcal{C}' = \mathcal{C}'_n$.

We will build the arrangement in an incremental fashion (that is, adding one circle at a time), and if there is a potential degeneracy while adding the current circle, we will perturb it, so no degeneracies will occur. We next describe the two key parameters that govern the perturbation scheme, the resolution bound and the perturbation bound.

Resolution bound

A degeneracy occurs when a predicate evaluates to zero. The goal of the perturbation is to cause all the values of all the predicate expressions (that arise during the construction of the arrangement of the circles) to become significantly non-zero, namely to be sufficiently far away from zero so that our limited precision arithmetic could enable us to safely determine whether they are positive or negative.

The degeneracies that arise in arrangement of circles have a natural geometric characterization as *incidences*. For example, in *outer tangency*, two circles intersect in a single point. In our scheme we transform the requirement that the predicates will evaluate to sufficiently-far-from-zero values into a geometric distance requirement.

This is a crucial aspect of the scheme: the transformation of the non-degeneracy requirement into a separation distance. We will call the bound on the minimum required separation distance, the *resolution bound* and denote it by ε. Deriving a good resolution bound is a central innovation in this work. Previously (e.g., [6]) we assumed that these bounds were given, and in our experiments we used crude (high) bounds. The bound on ε depends on the size of the input numbers (center coordinates and radii) and the machine precision. It is independent of the number

n of input circles.

Perturbation bound

Suppose indeed that ε is the resolution bound for all the possible degeneracies in the case of an arrangement of circles for a given machine precision. When we consider the current circle C_i to be added, it could induce many degeneracies with the circles in $\mathcal{C}'_{i-1}$. Just moving it by ε away from one degeneracy may cause it to come closer to other degeneracies. This is why we use a second bound δ, the *perturbation bound.* The bound δ depends on ε, on the maximum radius of a circle in $\mathcal{C}$, and on a *density* parameter k of the input which bounds the number of circles that are in the neighborhood of any given circle and may effect it during the process, $k \leq n$ (a formal definition of k is given in the full version of the paper; in the worst case $k = n$).

We say that a point q is a valid placement for the center of the currently handled circle C_i, if when moved to q this circle will not induce any degeneracy with any of the circles in $\mathcal{C}'_{i-1}$. The bound δ is computed such that inside the disc D of radius δ centered at the original center of C_i, at least half the points (constituting half of the area of the disc) will be valid placements for the circle. This means that if we choose a point uniformly at random inside D to relocate the center of the current circle, it will be a valid placement with probability at least $\frac{1}{2}$.

After the perturbation, the arrangement $\mathcal{A}(\mathcal{C}')$ is degeneracy free. Moreover, $\mathcal{A}(\mathcal{C}')$ can be robustly constructed with the given machine precision.[1]

An alternative view of our perturbation scheme is as follows. We look to move the centers of the input circles slightly from their original placement such that when constructing the arrangement $\mathcal{A}(\mathcal{C}')$ while using a fixed precision (floating-point) filter, the filter will always succeed and we will never need to resort to higher precision or exact computation.

The details of how to compute the resolution bound and the perturbation bound are given in the full version of this paper.

We quote the result summarizing the resources required by the algorithm.

Theorem 1 *Given a collection $\mathcal{C}$ of n circles, the perturbation algorithm which allows the construction of the arrangement $\mathcal{A}(\mathcal{C}')$ runs in total expected $O(n^2 \log n)$ time.*

[1]The perturbation algorithm should not be confused with the actual construction of the arrangement. It is only a preprocessing stage. However, it is convenient to combine the perturbation with an incremental construction of the arrangement.

3 Defining the Predicates and Determining a Worst Case ε

As was already stated, the main contribution of this work is in computing the resolution bound. To do so, we examine the possible degeneracies, and find the ε required to remove them once we are given the precision of the underlying arithmetic. In other words, we determine for each degeneracy a distance ε such that if a pair of features related to this degeneracy are at least ε apart, then we can safely evaluate the corresponding predicate with the given precision. For each degeneracy we present the appropriate predicate and also compute the worst case ε. Using this ε we then compute the value of δ, the maximum distance of a perturbed circle C_i from its original position, as described in the previous section.

Denote a predicate which takes m arguments and determines the sign of an expression by $Pr_s = sign(E(x_1, \ldots, x_m))$. Denote by Pr_p the predicate which takes m arguments and returns *true* iff $E(x_1, \ldots, x_m) > 0$. We define a degeneracy when $E = 0$.

Since we are using floating-point arithmetic, we cannot compute E exactly. Instead, we are only computing an approximation $\tilde{E}$ of E. We also compute a bound $B > 0$ on the maximum difference between $\tilde{E}$ and the exact value E, namely, $|E - \tilde{E}| \leq B$ or $\tilde{E} - B \leq E \leq \tilde{E} + B$. Thus, if $\tilde{E} > B$ then $E > 0$, and if $\tilde{E} < -B$ then $E < 0$. The bound B is computed according to the method given in [2].

When we add C_i to the collection $\mathcal{C}'_{i-1}$, if *for all* the predicates involving C_i (regarding all the circles that were already inserted), $|\tilde{E}| > B$, then C_i is in a valid place, and there is no need to perturb it. If there *exists* a predicate P, for which $|\tilde{E}| \leq B$, we define such a configuration as a *potential degeneracy*, and we need to perturb C_i. *For each predicate, we need to understand the geometric meaning, of $|\tilde{E}| > B$, so it will be reflected in ε and then in δ.* The details are given in the full version of the paper.

4 Experimental Results

In this section we report on experimental results with our implementation of the perturbation scheme that was described above. We implemented the perturbation scheme as a set of C++ classes. We also implemented the DCEL (Doubly Connected Edge List) construction with a simple point-location mechanism.

We have tested our program on four input sets (see Figure 2): `grid`, `flower`, `rand_sparse`, and `rand_dense`. For `rand_sparse` and `rand_dense`, all the input parameters are given as integers (to "promote" degeneracies). The properties of each input set are given in Table 1. The results of the perturbation and running times for those inputs are give in Table 2 (with the IEEE double number type). The tests have been performed on an Intel Pentium III 1 GHz machine with 2 GB RAM, operating on a Redhat 7.1 using gcc 3.03.

Name	n	R	M
grid	320	10	140
flower	40	100	100
rand_sparse	40	20	100
rand_dense	100	49	100

Table 1: n denotes the number of circles, R denotes the maximum radius and M is the maximum input size (center coordinates).

name	avg.	max.	p_time	t_time
grid	0.1319	0.7275	0.386	0.404
flower	0.9783	3.5470	0.274	0.28
rand_sparse	0.0158	0.0172	0.004	0.006
rand_dense	0.0382	0.3860	0.22	0.23

Table 2: Avg. denotes the average perturbation size, max. denotes the maximum perturbation size, p_time denotes the time of the perturbation (in seconds) and t_time denotes the total (perturbation and DCEL construction) time (in seconds). The given results are from averaging the results of 5 tests for each input.

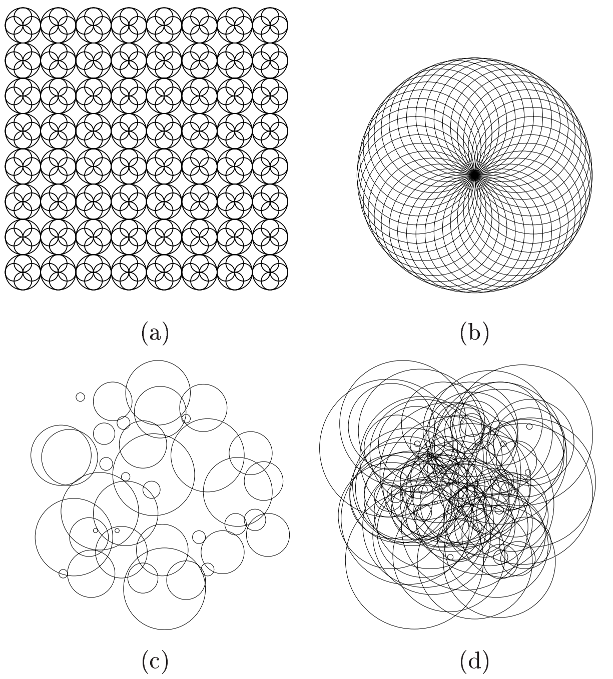

Figure 2: (a) A grid of 320 circles, which involves many inner and outer tangencies. (b) A "flower" composed of 40 circles, all intersecting in a common point. (c) A collection of 40 random circles. (d) A collection of 100 random circles.

References

[1] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer. A computational basis for conic arcs and Boolean operations on conic polygons. In *Proc. ESA 2002*, pages 174–186. Springer-Verlag, 2002.

[2] C. Burnikel, S. Funke, and M. Seel. Exact geometric computation using cascading. *International Journal of Computational Geometry and Applications (IJCGA)*, 11(3):245–266, 2001.

[3] S. Fortune and V. Milenkovic. Numerical stability of algorithms for line arrangements. In *Proc. 7th ACM Sympos. Comput. Geom.*, pages 334–341, June 1991.

[4] S. Fortune and C. J. Van Wyk. Static analysis yields efficient exact integer arithmetic for computational geometry. *ACM Trans. Graph.*, 15(3):223–248, July 1996.

[5] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *Proc. 5th ACM Sympos. Comput. Geom.*, pages 208–217, 1989.

[6] D. Halperin and C. R. Shelton. A perturbation scheme for spherical arrangements with application to molecular modeling. *Comput. Geom. Theory Appl.*, 10:273–287, 1998.

[7] K. Melhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.

[8] V. J. Milenkovic. Verifiable implementations of geometric algorithms using finite, precision arithmetic. *Artif. Intell.*, 37:377–401, 1988.

[9] S. Raab. Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes. In *Proc. 15th ACM Symposium on Computational Geometry*, pages 163–172, 1999.

[10] S. Schirra. Robustness and precision issues in geometric computation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 597–632. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[11] J. Shewchuk. Adaptive robust floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18:305–363, 1997.

[12] R. Wein. High level filtering for arrangements of conic arcs. In *Proc. ESA 2002*, pages 884–895. Springer-Verlag, 2002.

[13] C. K. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1):3–23, 1997.