Computational Geometry for the Gourmet Old Fare and New Dishes

(Invited Paper)

Bernard Chazelle
Department of Computer Science
Princeton University
Princeton, NJ 08544

This is a personal account of some of the most exciting challenges computational geometry faces today. Needless to say, my choice of open problems reflects my own taste and bias. I did not wish to cover the whole spectrum of the field and give a bland, comprehensive assessment of what's ahead for computational geometers. Rather, I chose to discuss a few problems that sprang to my mind as being both fundamental in the issues they raise, and plain fun to work on.

Some of these problems are as old as the field and can be stated in one or two sentences. In this category, one of my favorites is the problem of sorting $X + X$, the set obtained by adding elements pairwise from a set X of n numbers: Can it be done in $o\big(n^2 \log n\big)$? After Fredman's 1976 paper [31] that showed that, in essence, $O(n^2)$ comparisons suffice, precious little progress has been made. What makes Fredman's result impractical is that to find the right sequence of comparisons can only be done at this point with an exponential-size look-up table. The importance of this problem in geometry is that without a solution to it, hopes to sort the slopes formed by n points in the plane optimally or equivalently, sort the vertices of a line arrangement along a given direction, all vanish. This is old, hard and still open, but hardly an isolated case by a long shot. I will discuss more of these old unsolved cases and then gradually move on to newer trends, e.g., probabilistic algorithms and their derandomization, or the decomposition of real-algebraic manifolds. For each open problem mentioned I will also try to indicate what seems to be missing in our current understanding and what might be reasonable lines of attack.

CAPSULE 1: *Polygonal Curves*

The time has come to elucidate the mystery of polygonal curves once and for all. After many years of effort [19,25,35,42,43,69,71], the problem of triangulating a simple polygon was finally resolved last year [13]. It now appears to me that simplicity is *not* the issue but connectivity is. I conjecture that the planar subdivision created by a polygonal curve can be triangulated in linear time, whether the curve is simple or not. If the curve has self-intersections, then of course the term "linear" is to be understood with respect to the output size. More generally, if we are in the presence of p connected polygonal curves (simple or not) I believe that a triangulation should be computable in time $O(k + p \log p)$, where k is the size of the triangulation. We assume here that we are not allowed to add new vertices, except for the intersections. Such a bound would be tight and would unify in a beautiful manner a number of seemingly separate optimal algorithms: the one just mentioned for triangulating a simple polygon, another for intersecting line segments [14], and finally methods for merging convex planar subdivisions [39,44,53]. Such a lofty goal might be difficult to attain, and a good starting place might be the easier problem of intersecting two simple polygons.

CAPSULE 2: *Dynamic Point Location*

The problem of locating a point in a planar subdivision has been studied thoroughly and, by and large, it is completely solved. The dynamic version of the problem is interesting in its own right and also because it is tied to efficient solutions to static point location in a three-dimensional polyhedral cell complex [60]. The dynamic operations we have in mind consist of inserting or deleting an edge. The best known solutions are due to Cheng and Janardan [20] and Goodrich and Tamassia [37], which improve on an earlier solution by Preparata and Tamassia [59]. The storage is linear, the update time is $O(\log n)$, but the query time is $O(\log^2 n)$. The reason for the relatively high query time is that the data structure forces the algorithm into repeated binary searches. Fractional cascading [17] is a general technique for speeding up such operations, and it is indeed used in most of the optimal static solutions. Given a graph G with sorted lists of numbers attached to its nodes, fractional cascading is a way of interconnecting those lists so that binary searching for a key in the lists along a path of G can be done in time proportional to the length of the path, with logarithmic additive overhead.

What seems to be needed for efficient dynamic point location is an implementation of dynamic fractional cascading. Not dynamic in the sense of modifying the lists at the nodes (this is well understood [17,49]) but in the sense of adding and deleting edges from the graph G. These changes represent balanced search tree rotations in the context of dynamic point location. A naive implementation which consists of updating the list interconnections in the fractional cascading structure is doomed because it might entail too much work. A more "lazy" approach whereby only portions of the interconnections are updated in some incremental, persistent manner might lead somewhere, but the answer is far from clear at this point.

CAPSULE 3: *Convex Hulls*

The convex hull of n points in the plane can be computed in $O(n \log n)$ time by a variety of methods [29,58]. But what if we want to maintain the convex hull dynamically under insertions and deletions? The data structure should let us access the current convex hull at any time and perhaps allow us to compute such things as the vertex closest to a query line. Is it possible to achieve $O(\log n)$ time per update? The best bound known at this time is $O\big(\log^2 n\big)$ (Overmars and van Leeuwen [56]). The case of insertions-only [58] or deletions-only [8,41] is well understood. Since a single insertion (or deletion) might cause a massive change in the hull, one must represent the changes in a lazy fashion by using a hierarchical representation of a convex hull. But even then, alternating deletions and insertions can create havoc. There is nothing new in this phenomenon. The bane of dynamic structures is that one element whose insertion or deletion deeply modifies the combinatorial structure of the object represented, and which an ill-intentioned adversary will take pleasure in inserting and removing in alternation. Again, a deeper understanding of laziness and persistence seems in order.

In arbitrary (but fixed) dimension, one of the most classical geometric problems of all is still partly open: computing the convex hull of n points in d-space. Seidel has given an algorithm that is optimal in even dimensions [66] and another algorithm [67] that reports every face of the hull in logarithmic time, with a quadratic additive overhead. Clarkson and Shor [24] have given an optimal randomized algorithm, later simplified by Seidel [68]. But how long shall we have to wait for an optimal $O\big(n^{\lfloor d/2 \rfloor}\big)$-time algorithm, let alone one that is output-sensitive? Perhaps a better understanding of derandomization (see Capsule 8) might allow us to make the randomized algorithms deterministic without loss of efficiency.

CAPSULE 4: *Polyhedra in 3-Space*

Let us move up to three dimensions and discuss the current status of convex polyhedra. In 1983 Dobkin and Kirkpatrick [28] proposed an ingenious representation of a convex polytope as a nested sequence of coarser and coarser approximating polytopes. They used it successfully to give polylogarithmic algorithms for detecting intersection between two polytopes. Later, by building on these ideas, I showed how to compute the intersection explicitly in linear time [9]. But what happens if the polyhedra are not convex? Mehlhorn and Simon [50] gave a solution for the case where one of them is convex, but the general problem remains wide open. Interesting partial results were given in [4].

In the same vein, here is another basic intersection problem whose answer is not yet known. Is it possible to preprocess a polytope of n vertices using linear or, say, $n \times \text{polylog}(n)$ space, so that its intersection with a query plane can be computed in time $O(k + \log n)$, where k is the size of the intersection? To achieve $O((k+1) \log n)$ is easy if we first look for a point of contact and then walk around the intersection by repeated binary searches around the faces of the polytope. Again, the bottleneck appears to lend itself to fractional cascading, but the appearance is deceptive. A closer look reveals that the searches, although implemented on standard balanced search trees, are really two-dimensional. So, perhaps, we need to generalize fractional cascading in the manner described below.

Consider p planar maps, each of size $O(n)$, which we wish to preprocess for the purpose of performing point location in each of them. Using standard techniques, a point can be located in each map in a total of $O(p \log n)$ time, using $O(pn)$ storage. Note that this does not require spreading information around among the different maps. If we do so, as in fractional cascading, can we hope to achieve $O(p+\log n)$ time, while retaining optimal $O(pn)$ space or slightly worse? Small steps toward that goal can be made by trying to merge together the maps in small batches, but even then one has to be very cautious. Indeed, the fundamental difference with the world of fractional cascading is that unlike linear lists, two maps of size n might merge to produce a map of size $\Omega\big(n^2\big)$. Even if the maps are parallel strips (with different slopes) how to proceed is unclear. Understanding this problem would be very useful, not only for the intersection problem mentioned earlier, but also for several searching problems whose current solutions involve data structures storing Voronoi diagrams at the nodes of a master tree structure which is then searched by repeated planar point locations.

CAPSULE 5: *Hidden Surface Removal*

What is in our mind the most tantalizing open problem in 3d computational geometry is to resolve the complexity of hidden surface removal. To take a simple example, consider n non-intersecting opaque triangles, and think of computing the visible scene from a fixed viewpoint. Since Sutherland et al.'s classic survey [70], some progress has been made but not a great deal. In practice, it seems that the only tangible improvements have come from better hardware via image-space methods. If the description size of the visible scene is k, the goal is to find an output-sensitive algorithm whose complexity might be something like $O(n+k)$ times a polylog(n) factor. No one knows how to do this, which does not surprise me much, because I conjecture that it cannot be done.

A more realistic goal might be to shoot for a running time of the form $O\big(n^{1+\varepsilon} + kf(n)\big)$, where ε is a small positive constant and $f(n)$ is a slow-growing function. The particular cases of polyhedral terrains [63] or rectilinear scenes [5,36,61] are relatively well understood (even though some unanswered questions still remain). But the general case is completely open. Under the restrictive assumption of a partial order among the triangles, partial results have been given in [54,55,3]. See also [34,57].

At the most basic level, it seems that the main bottleneck in hidden surface removal is what is commonly referred to as *Hopcroft's problem*: Given n points and n lines in the plane, check whether none of the points lie on any of the lines. An elegant feature of this problem is that it is invariant under duality. Its relevance to hidden surface removal can be seen by the following example. Think of n uncooked spaghetti lying flat in a pan, with a napkin on top of them hiding everything. Now assume that somebody punches n tiny holes in the napkin. The visible scene will be able to tell whether one of the spaghetti shows up through any one of the holes, which is just another way of stating Hopcroft's problem. The fastest (deterministic) algorithm for that problem is due to Agarwal [2] and runs in time $O\big(n^{4/3}\log^2 n\big)$, or even $O\big(n^{4/3}\log n\big)$, by using Matoušek's latest results on derandomization (see Capsule 8). The factor $n^{4/3}$ happens to be the order of magnitude of the maximum size of n faces chosen in an arrangement of n lines [23], and this is no accident. Indeed, if the algorithm for Hopcroft's problem is to perform only tests of the form: "Is this input

point below or above that input line?", then by placing the n points inside the largest faces of the worst-case arrangement we immediately obtain an $\Theta(n^{4/3})$ lower bound on the computation time. Of course, no one says that the algorithm needs to be so restrictive. But nevertheless, we conjecture that on, say, a RAM or a pointer machine, no algorithm can run in time $n \times \text{polylog}(n)$. An answer to this question would shed a tremendous amount of light on the entire issue of hidden surface removal.

A related problem is to detect whether n points are in general position. This can be done in $O(n^2)$ time [18,30], but there is little reason to believe that this is optimal. On the other hand, the existence of, say, an $O(n \log n)$ algorithm appears unlikely. Again, the sparsity of lower bounds is a serious impediment to our current understanding of these problems.

CAPSULE 6: *Lower Bounds for Multidimensional Searching*

This generalized form of searching deals with geometric objects that cannot be structured by a total order. Hit detection in computer graphics, range searching in databases, point location in geographical maps are typical instances of multidimensional searching. In simplex range searching, for example, one is asked to preprocess n weighted points in d-space, so as to support queries of the form: "What is the cumulative weight of the points inside a given query simplex?" While efficient solutions have been obtained or at least seem in sight, the issue of lower bounds has proven rather elusive. Strong lower bounds have been established for the semigroup case, where no subtraction of weights are allowed [10,11,12,32,33]. This case is actually not nearly as restrictive as it sounds, because adding weights in a semigroup might mean taking their maximum, or combining auxiliary data structures associated with them, which are operations which do not admit of inverses. At any rate, to generalize these lower bounds to the group model, where an inverse operation is allowed seems very challenging. A partial result is given in [73] but it requires a restrictive assumption on what is an allowable data structure. The difficulty of the problem is to assess the true power of inclusion-exclusion relations. An apt analogy is how difficult it is to generalize lower bounds for monotone circuits to circuits allowing negation.

CAPSULE 7: *Higher-Dimensional Nonlinear Shapes*

Triangulating an arrangement of n hyperplanes in E^d is to partition its faces into a regular cell complex, whose cells are simplices of all dimensions between 0 and d. There are standard ways of doing that [21], but sometimes it is useful to have what is called a *vertical decomposition*, that is, a regular cell complex whose cells have a cylindrical structure oriented along the reference axes. This is especially true if we deal with nonlinear surfaces, such as for example, real-algebraic varieties (more specifically, real zero-sets of multivariate polynomials with rational coefficients.)

The need to analyze and understand the geometry of large collections of real-algebraic varieties arises in motion-planning, where the obstacles are modeled as hypersurfaces and the robot is a point in real higher-dimensional space. A milestone in that area is the paper by Schwartz and Sharir [65]; see also [7]. An outstanding open question in that area, which might be called *computational real-algebraic geometry*, is that of triangulating an arrangement of real-algebraic varieties [26,62,72]. Collins' decomposition [26] offers a solution to that problem, but one exceedingly expensive, since

it uses storage doubly-exponential in the number of variables (which is the dimension of the ambient space). Jointly with Edelsbrunner, Guibas, and Sharir [15], we have given a sign-invariant stratification scheme that is singly-exponential (note that no solution can be sub-exponential). A sign-invariant stratification is simply a partition of $\Re^d$ into simple cells over which each polynomial remains sign-invariant. A simple cell is defined here as a smooth manifold (i) topologically equivalent to a k-ball ($k \leq d$) and (ii) specified by constant-size quantifier-free formula in the theory of reals. Our method produces $O\big(n^{2d-2}\big)$ cells, where n is the number of polynomials defining the varieties. (Actually, it is a bit better, but the true bound is too technical to bother with here.) Although the algorithm represents an attractive alternative to Collins' decomposition, it raises three major open questions: One is to reduce the bound to $O\big(n^d\big)$, which is easily shown to be tight. Another is to reduce the complexity of the algebraic numbers used to describe the cells. As things stand, these numbers are represented by a recursive scheme which creates polynomials of degree doubly exponential in the number of variables. Although these degrees are still bounded by a constant, they could be huge in practice. Although it is now known that eliminating quantifiers is inherently doubly-exponential [27], many restricted problems related to the theory of reals can be solved in singly-exponential time [6,7,38,64]. Perhaps the degrees of the polynomials can also be reduced to singly-exponential. Finally, the third open problem is turning a stratification into a regular cell complex, that is, ensuring that neighboring cells are glued together nicely.

Capsule 8: *Derandomization*

There has been a flurry of activity in the area of probabilistic geometric algorithms lately, pionneered by the works of Clarkson [21,22], Haussler and Welzl [40], Reif and Sen [63]. The idea of random sampling, in particular, has proven very fruitful. A typical use of that tool is to prepare the grounds for divide-and-conquer despite the apparent lack of order in the underlying universe. For example, given a collection H of n hyperplanes in E^d we can pick r of them at random and triangulate their arrangement. It then happens that with high probability no simplex is cut by more than $O\big((n \log r)/r\big)$ lines. Matoušek defines an *ε-cutting* for H to be any collection of (possibly unbounded) d-dimensional simplices which together cover E^d and such that the interior of each simplex intersects at most εn hyperplanes [47]. It is possible to show that with a bit of additional work a random sample of r hyperplanes in H can be made to form a $(1/r)$-cutting of optimal $O\big(r^d\big)$ size [16]. Methods for derandomizing these algorithms were given in [1,16,45,46,47], and the whole question is close to being solved.

But how hard is to derandomize the numerous geometric algorithms which do no rely on random sampling, but on the incremental processing of the input according to a random permutation. Building on the works of Clarkson and Shor [24], Seidel has given eloquent evidence of the power of randomizing over the input order by exhibiting particularly simple algorithms for linear programming in fixed dimension and convex hull computation [68]. See also [51,52,69]. A random permutation requires $\Theta(n \log n)$ truly random bits, but it is likely that much less randomness is required by those incremental probabilistic algorithms. But how much exactly? Is it possible to derandomize some of those algorithms without paying a huge overhead?

Conclusions:

It is time to bring this quick overview to a close. There are many interesting problems that I have not touched upon here but which deserve intense scrutiny nevertheless. Some concern important combinatorial issues in discrete geometry, such as zone theorems, and the maximum size of k-sets. Others address the practical aspects of implementing and debugging geometric algorithms, and in particular, robustness in the face of round-off errors.

REFERENCES

1. Agarwal, P.K. *Partitioning arrangements of lines I: An efficient deterministic algorithm,* Disc. Comput. Geom. 5 (1990), 449–483.
2. Agarwal, P.K., *Partitioning arrangements of lines: II. Applications,* Disc. Comput. Geom. 5 (1990), 533–573.
3. Agarwal, P.K., Sharir, M. *Applications of a new partitioning scheme,* manuscript, 1990.
4. Aronov, B., Sharir, M. *Triangles in space, or building and analyzing castles in the air*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 381-391.
5. Bern, M. *Hidden surface removal for rectangles,* J. Comput. Sys. Sci. 40 (1990) 49–69.
6. Caniglia, L., Galligo, A. and Heintz, J. *Some new effectivity bounds in computational geometry*, Proc. 6th Internat. Conf. on Applied Algebra, Algorithmic and Error Correcting Codes, Rome, July 1988.
7. Canny, J.F. *A new algebraic method for motion planning and real geometry*, Proc. 28th Annu. IEEE Symp. on Foundat. of Computer Science (1987), 39–48.
8. Chazelle, B. *On the convex layers of a planar set,* IEEE Trans. Informat. Theory IT–31 (1985), 509–517.
9. Chazelle, B. *An optimal algorithm for intersecting three-dimensional convex polyhedra*, Proc. 30th Ann. IEEE Symp. Found. Comp. Sci. (1989).
10. Chazelle, B. *Lower bounds on the complexity of polytope range searching,* J. American Math. Soc. 2 (1989), 637–666.
11. Chazelle, B. *Lower bounds for orthogonal range searching: I. The reporting case,* J. ACM 37 (1990), 200–212.
12. Chazelle, B. *Lower bounds for orthogonal range searching: II. The arithmetic model,* J. ACM 37 (1990), 439–463.
13. Chazelle, B. *Triangulating a simple polygon in linear time,* Proc. 31st Annu. IEEE Symp. Foundat. Comput. Sci., (1990), 220–230. To appear in Discrete Comput. Geom. (1991).

14. Chazelle, B., Edelsbrunner, H. *An optimal algorithm for intersecting line segments in the plane*, Proc. 29th Ann. IEEE Symp. Found. Comp. Sci. (1988), 590–600.
15. Chazelle, B., Edelsbrunner, H., Guibas, L.J., Sharir, M. *A singly-exponential stratification scheme for real semi-algebraic varieties and its applications*, ICALP (1989) 179–193.
16. Chazelle, B., Friedman, J. *A deterministic view of random sampling and its use in geometry,* Combinatorica 10 (1990), 229–249.
17. Chazelle, B., Guibas, L.J. *Fractional cascading: I. A data structuring technique*, Algorithmica, 1 (1986), 133–162.
18. Chazelle, B., Guibas, L.J, Lee, D.T. *The power of geometric duality*, BIT 25 (1985), 76–90.
19. Chazelle, B., Incerpi, J. *Triangulation and shape-complexity*, ACM Trans. on Graphics 3 (1984), 135–152.
20. Cheng, S., Janardan, R. *New results on dynamic point location* Proc. 31st Ann. IEEE Symp. Foundat. Comput. Sci. (1990), 96–105.
21. Clarkson, K.L. *A randomized algorithm for closest-point queries*, SIAM J. Comput. 17 (1988), 830–847.
22. Clarkson, K.L. *New applications of random sampling in computational geometry*, Disc. Comp. Geom. 2 (1987), 195–222.
23. Clarkson, K.L., Edelsbrunner, H., Guibas, L.J., Sharir, M., Welzl, E. *Combinatorial complexity bounds for arrangements of curves and surfaces,* Disc. Comput. Geom. 5 (1990), 99–160.
24. Clarkson, K.L., Shor, P.W. *Applications of random sampling in computational geometry, II*, Disc. Comp. Geom. 4 (1989), 387–421.
25. Clarkson, K.L., Tarjan, R.E., Van Wyk, C.J. *A fast Las Vegas algorithm for triangulating a simple polygon*, Disc. and Comput. Geom. 4 (1989), 432–432.
26. Collins, G.E. *Quantifier elimination for real closed fields by cylindric algebraic decomposition*, Proc. 2nd GI Conf. on Automata Theory and Formal Languages, Springer-Verlag, LNCS 35, Berlin (1975), 134–183.
27. Davenport, J., Heintz, J. *Real quantifier elimination is doubly exponential*, J. Symbolic Comput. 5 (1988), 29-35.
28. Dobkin, D.P., Kirkpatrick, D.G. *Fast detection of polyhedral intersection*, Theoret. Comput. Sci. 27 (1983), 241–253.
29. Edelsbrunner, H. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, Germany, 1987.
30. Edelsbrunner, H., O'Rourke, J., Seidel, R. *Constructing arrangements of lines and hyperplanes with applications*, SIAM J. Comput. 15 (1986), 341–363.
31. Fredman, M.L. *How good is the information theory bound in sorting?*, Theoret. Comput. Sci, 1, pp. 355–361, 1976.
32. Fredman, M.L. *A lower bound on the complexity of orthogonal range queries*, J. ACM, 28 (1981), 696–705.

33. Fredman, M.L. *Lower bounds on the complexity of some optimal data structures*, SIAM J. Comput. 10 (1981), 1–10.
34. Fuchs, H., Kedem, Z., Naylor, B. *On visible surface generation by a priori tree structures*, Computer Graphics (SIGGRAPH'80), 124-133.
35. Garey, M.R., Johnson, D.S., Preparata, F.P., Tarjan, R.E. *Triangulating a simple polygon*, Inform. Process. Lett. 7 (1978), 175–180.
36. Goodrich, M.F., Atallah, M., Overmars, M. *An input-size/output-size trade-off in the time complexity of rectilinear hidden surface removal,* Proc. 16th ICALP.
37. Goodrich, M.F., Tamassia, R. *Dynamic trees and dynamic point location*, Johns Hopkins Univ. Tech. Rep., 1990.
38. Grigor'ev, D. and Vorobjov, N. *Solving systems of polynomial inequalities in subexponential time*, J. Symbolic Comput. 5 (1988), 37-64.
39. Guibas, L.J., Seidel, R. *Computing convolutions using reciprocal search*, Proc. 2nd Ann. ACM Symp. Comput. Geom. (1986), 90–99.
40. Haussler, D., Welzl, E. *Epsilon-nets and simplex range queries*, Disc. Comp. Geom. 2, (1987), 127–151.
41. Hershberger, J., Suri, S. *Finding tailored partitions,* Proc. 5th Ann. ACM Symp. Comput. Geom. (1989), 255–265.
42. Hertel, S., Mehlhorn, K. *Fast triangulation of a simple polygon*, Proc. Conf. Found. Comput. Theory, New York, Lecture Notes on Computer Science 158 (1983), 207–218.
43. Kirkpatrick, D.G., Klawe, M.M., Tarjan, R.E. $O(n \log\log n)$ *polygon triangulation with simple data structures*, Proc. 6th Ann. ACM Symp. Comput. Geom. (1990), 34–43.
44. Mairson, H.G., Stolfi, J. *Reporting and counting intersections between two sets of line segments*, Proc. NATO Advanced Study Inst. Theoret. Found. Comput. Graphics and CAD, Il Ciocco, Castelvecchio Pascoli, Italy, Springer-Verlag, 1987.
45. Matoušek, J. *Construction of ε-nets,* Disc. Comput. Geom. 5 (1990), 427–448.
46. Matoušek, J. *Approximations and optimal geometric divide-and-conquer,* KAM Series (tech. report) 90-174, Charles University, 1990. Also to appear in Proc. 23rd ACM Symp. Theory of Comput., 1991.
47. Matoušek, J. *Cutting hyperplane arrangements,* to appear in Disc. Comput. Geom., 1991. Also, in Proc. 6th ACM Symp. Comput. Geom. (1990), 1–9.
48. Mehlhorn, K. *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*, Springer-Verlag, Heidelberg, Germany, 1984.
49. Mehlhorn, K., Näher, S. *Dynamic fractional cascading,* Algorithmica 5 (1990), 215–242.
50. Mehlhorn, K., Simon, K. *Intersecting two polyhedra one of which is convex*, Univ. Saarland, Tech. Report, Saarbrücken, West Germany, 1986.
51. Mulmuley, K. *A fast planar partition algorithm*, Proc. 29th Ann. IEEE Symp. Found. Comp. Sci. (1988).

52. Mulmuley, K. *A fast planar partition algorithm, II*, Proc. 5th Ann. ACM Symp. Comp. Geo. (1989), 33–43.
53. Nievergelt, J., Preparata, F.P. *Plane-sweep algorithms for intersecting geometric figures*, Comm. ACM, 25 (1982), 739–747.
54. Overmars, M., Sharir, M. *Output-sensitive hidden surface removal algorithms,* Proc. 30th Ann. IEEE Symp. Foundat. Comput. Sci. (1989), 598–603.
55. Overmars, M., Sharir, M. *Merging visibility maps,* Proc. 6th Ann. ACM Symp. Comput. Geom. (1990), 168–176.
56. Overmars, M.H., van Leeuwen, J. *Maintenance of configurations in the plane*, Journal of Computer and System Sciences 23 (1981), 166–204.
57. Paterson, M.S., Yao, F.F. *Binary partitions with applications to hidden-surface removal and solid modelling,* Proc. 5th Ann. ACM Symp. Comput. Geom. (1989), 23-32.
58. Preparata, F.P., Shamos, M.I. *Computational Geometry*, Springer-Verlag, New York, 1985.
59. Preparata, F.P., Tamassia, R. *Fully dynamic techniques for point location and transitive closure in planar structures*, Proc. 29th Ann. IEEE Symp. Found. Comp. Sci. (1988), 558–567.
60. Preparata, F.P., Tamassia, R. *Efficient spatial point location*, Proc. 1989 Workshop on Algorithms and Data Structures.
61. Preparata, F.P., Vitter, J., Yvinec, M. *Computation of the axial view of a set of isothetic parallelepipeds,* ACM Trans. on Graphics 5 (1990), 278–300.
62. Prill, D. *On approximations and incidence in cylindrical algebraic decompositions*, SIAM J. Comput. 15 (1986), 972–993.
63. Reif, J., Sen, S. *An efficient output-sensitive hidden surface removal algorithm and its parallelization,* Proc. 4th Ann. ACM Symp. Comput. Geom. (1988), 193–200.
64. Renegar, J. *A faster PSPACE algorithm for deciding the existential theory of the reals*, Proc. 29th Annu. IEEE Symp. on Foundat. of Computer Science (1988), 291–295.
65. Schwartz, J.T., Sharir, M. *On the "piano movers" problem. II: General techniques for computing topological properties of real algebraic manifolds*, Adv. in Appl. Math. 4 (1983), 298–351.
66. Seidel, R. *A convex hull algorithm optimal for point sets in even dimensions,* Univ. British Columbia, tech. Rep. 81–14, 1981.
67. Seidel, R. *Constructing higher-dimensional convex hulls at logarithmic cost per face,* Proc. 18th Ann. ACM Symp. Theory Comput. (1986), 404–413.
68. Seidel, R. *Linear programming and convex hulls made easy,* Proc. 6th Ann. ACM Symp. Comput. Geom. (1990), 211–215.
69. Seidel, R. *A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons*, manuscript, 1990.
70. Sutherland, I.E., Sproull, R.F., Schumaker, R.A. *A characterization of ten hidden surface algorithms*, Computing Surveys 6 (1974), 1-55.
71. Tarjan, R.E., Van Wyk, C.J. *An $O(n \log\log n)$-time algorithm for triangulating a simple polygon*,

SIAM J. Comput. 17 (1988), 143–178.

72. Whitney, H. *Elementary structure of real algebraic varieties*, Annals of Math. 66 (1957).
73. Willard, D.E. *Lower bounds for dynamic range query problems that permit subtraction*, Proc. 13th ICALP, 1986.