Intersection Algorithms:
A Comparative Study

Sabine Stifter
Eugen Ardeleanu*
Research Institute for Symbolic Computation
Johannes Kepler University Linz, A-4040 Austria
Sabine.Stifter@risc.uni-linz.ac.at
Eugen.Ardeleanu@risc.uni-linz.ac.at

Abstract

This report presents an overview on intersection algorithms in 2 and 3 dimensions. It contains a classification according to the strategy used (divide-and-conquer, prune-and-search, duality) and complexity issues. The Roider method is placed in this framework and connections to linear programming and collision detection are analyzed.

Key words: *Geometric algorithms, Roider method, Linear programming, Collision detection*

1 Introduction

Intersection tests are ubiquitous in computational geometry and in a wide range of applications to robotics, computer graphics, simulation, etc. We review here some of the most important ones, pointing out the basic strategies and time and space complexity. A hybrid $2D$ Roider method mixing some of the most efficient techniques is suggested.

*Supported by the Austrian National Science Foundation (FWF) project no. P8620-TEC.

The report is organized as follows: Section 2. reviews convexity testing and convex hull construction algorithms; Section 3. analyses some proeminent intersections tests in 2 and 3 dimensions; Section 4. gives an account on the use of linear programming in $2D$ and $3D$; Section 5. addresses the duality technique; Section 6. suggests a hybrid Roider method; and Section 7. concludes the paper.

2 Convexity

There are wellknown optimal $\mathcal{O}(n \log n)$ algorithms for finding the convex hull of n points in the plane, cf. [1, 4]:

- *Graham scan*: a local method method based on $< \pi$ *property of interior angles in a convex polygon* that determines the convex hull in linear time after the vertices were sorted;
- *Jarvis march*: a method based on the property according to which all the points lie on one side of each edge of the convex hull, $\mathcal{O}(n^2)$ worst-case complexity, but *linear* if the convex hull has a constant number of edges;
- *Quickhull*: a *divide-and-conquer* method whose *merge* step is based on calculation of *supporting lines* (lines passing through a vertex of the convex polygon such that the interior of the polygon lies entirely on one side of the line).

The above mentioned merge technique has also an inherent interest because it provides answer to the problem of *finding the convex hull of the union of two convex polygons*. Within this method we can also detect some analogy with the Roider method. For these reasons we will now study in some more detail the following algorithm due to Shamos, cf. [4], with a slight modification.

Given two convex polygons P_1 and P_2, first find a point p internal to P_1, e.g the centroid of any 3 vertices of P_1. Then determine whether p is internal to P_2 by some point-location method in $\mathcal{O}(n)$ time. If p is internal to P_2 then, in the figure above, the vertices of both P_1 and P_2 occur in sorted angular order about p and they can be merged in $\mathcal{O}(n)$ time to obtained a sorted list of vertices of both P_1 and P_2. Performing the Graham scan (in linear time) obtain the hull of $P_1 \cup P_2$. If p is not internal to P_2 then,

in the figure above, P_2 lies in a wedge whose apex angle is $\leq \pi$ defined by p and two supporting lines of P_2. Vertices u, v of P_2 and w, t of P_1, which can be found in linear time by a single pass around P_2 and P_1, partition P_2 and P_1 into two chains of vertices monotonic in polar angle about p; the toward p convex chain of P_2 and concave chain of P_1 can be discarded, since their vertices lie in the interior of the convex hull, the other ones constitute two sorted lists that can be merged in linear time to form a list of vertices of $P_1 \cup P_2$, sorted about p. A final Graham scan gives the hull.

The idea of supporting lines can be further exploited to obtain an on-line $\mathcal{O}(n \log n)$ time, real-time ($\mathcal{O}(\log n)$ update time) algorithm for computing the convex hull of n points in the plain, and furthermore, through Kallay's *beneath-beyond method*, cf. [4], to obtain an on-line $\mathcal{O}(n^{\lfloor d/2 \rfloor + 1})$ time algorithm for computing the convex hull of n points in d-dimensional space.

Efficient methods to construct supporting lines are also the key to Kirkpatrick-Seidel convex hull algorithm, where resort is made to the linear time Megiddo-Deyr algorithm for linear programming, cf. [1, 4].

Yet another algorithm for convex hull due to Seidel and akin to Kallay's is based on duality:

where the construction of supporting lines from p to P is equivalent, in the dual space, to the intesection of $Dual(P)$ with $Dual(p)$.

3 Intersection tests

We treat first convex planar figures. First of all, there are optimal $\mathcal{O}(n \log n)$ algorithms for finding the intersections of n segments in the plane based on the *sweep-line method.* As a consequence, the intersection of n half-planes can be computed in $\mathcal{O}(n \log n)$ time, and this is optimal, cf. [1, 3, 4]. (Hence the kernel of a polygon can be found in $\mathcal{O}(n \log n)$ time; there exist however optimal linear time algorithms.) Furthermore, the common intersection of n convex k-gons can be found in $\mathcal{O}(nk \log n)$ time. Similar sweep-line techniques are used in geometric searching problems, cf. [3, 4]:

Point-location in an n-vertex planar subdivision can be performed in $\mathcal{O}(\log n)$ optimal query time using $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^2)$ preprocessing time.

The much easier problem of *point location in a simple (convex, star-shaped) polygon* can be solved in $\mathcal{O}(n)$ ($\mathcal{O}(\log n)$ time with $\mathcal{O}(n)$ storage, after $\mathcal{O}(n)$ preprocessing time) by *counting the number of intersections* of a straight line through the given point with the given polygon (*division into wedges and binary search*).

The last technique also displays some analogy with the Roider method, cf. [7].

Passing now to 3-dimensional intersection tests we mention Muller-Preparata method, cf. [4] which computes the intersection of two convex polyhedrons in $\mathcal{O}(n \log n)$ time if one knows a point in the intersection (*witness to intersection*. The method is based on duality. We remark here that such a witness can be found via the Roider method, cf. [8]. Reciprocally, the original Muller-Preprata algorithm for finding a witness to intersection (or reporting disjointness), simplified by Dyer, and which is based on minimizing a certain function representing the distance between relevant parts of the boundary of the two polyhedrons, will be used in our design of a hybrid Roider method.

4 Linear Programming

The standard formulation of the linear programming problem in 2 variables is similar to the problem of intersecting n planes (giving the *feasible region*), cf. [4, 5]:

$$\min(ax + by) : a_i x + b_i y + c_i \leq 0, i = 1, ..., n$$

The *objective function* defines a family of parallel lines $ax + by + c$, where c is a real parameter. The lines of this family that support the feasible region pass through the (optimum) vertices corresponding to the extreme values of the objective function. Since support lines of a convex polygon can be constructed in $\mathcal{O}(\log n)$ time, the 2-variable linear programming can be transformed to half-planes intersection, which can be solved in $\mathcal{O}(n \log n)$ time.

The wellknown *Simplex algorithm* for linear programming visits every vertex of the feasible region for a total of $\mathcal{O}(n^2)$ (similar to Jarvis' march), so it is not optimal (its complexity is exponential in the dimension). In practice however the Simplex algorithm exhibits a very good performance due to the fact that it is a linear average-case algorithm.

Building on the observation that the construction of the feasible region is not necessary to solve the linear programming problem, the Megiddo-Dyer optimal $\mathcal{O}(n)$ algorithm (adaptable also to 3 and more dimensions) discards redundant constraints (relative to the half-plane intersection) and constraints not containing an optimum vertex by applying the linear transformation:

$$ax + by \mapsto y, x \mapsto x$$

of the objective function into one of the coordinates

$$\min y : (a_i - \frac{a}{b} b_i)x + \frac{b_i}{b} y + c_i \leq 0, i = 1, ..., n$$

It is interesting to note that Megiddo-Dyer algorithm can be adapted also for situations not properly belonging to linear programming, e.g. in the presence of quadratic constraints as in case of finding the smallest enclosing circle of a set of n points, which can thus be computed in optimal $\mathcal{O}(n)$ time.

It can easily be shown that the problem of *linear separability* of two sets of points can be stated as a linear programming problem, cf. [2, 4], being thus solvable in linear time in 2 or 3 dimensions.This observation, together with duality and Muller-Preparata algorithm for polyhedrons intersection, leads to an optimal $\mathcal{O}(n \log n)$ algorithm for finding the intersection of n half-spaces.

5 Duality

There is a close relationship between finite sets of points and finite sets of hyperplanes: for each finite set of points P in a d-dimensional euclidian space E^d there is a set of hyperplanes H in a bijective correspondence such that certain statements about P are true if and only if their corresponding dual statements about H are true, e.g. for d=2, three points in P are colinear if and only if the corresponding lines in H are concurrent. The correspondence is realized through projective transformations. Thus let B be a $(d+1) \times (d+1)$ matrix of the projective group on $d + 1$ homogeneous coordinates. Then the transformation mapping points to points and lines to lines in the plane

$$\mathbf{h} = \mathbf{x} B$$

can be interpreted in E^3 as transforming a line (represented by x) to a plane with h as normal passing by the origin. If D is the inverse duality transform then preservation of incidence implies $D = k(B^{-1})^T$ and the involutory condition $B = \pm B^T$.

We consider now the case of a non-singular 3×3 symmetric matrix B and the effect of the duality transformation in the central projection to the plane $x_3 = 1$. The equation $\mathbf{x}B\mathbf{x}^T = 0$ represents a conic in the plane in homogeneous coordinates. Fixing now a point p_0 (*the pole*) define the line $p_0 B x^T = 0$ as *the polar* of p_0 with respect to the conic B. Thus a pole on the conic lies on its own polar.

In the particular case of B defining the unit circle in the plane we obtain the following situation:

where a point $\mathbf{p} = (p_1, p_2, 1)$ is mapped to a line $l : p_1 x_1 + p_2 x_2 - x_3 = 0$ in homogeneous coordinates such that the distance from the origin of the plane to the pole is inverse proportional to the distance from the origin to the polar, cf. [4].

In case B defines the unit parabola we have the following dual transform D in the plane:

Based on the fundamental properties of the dual transform *incidence and order preservation*, cf. [1] we can derive:

- *A general method for finding supporting lines*: To find the supporting lines from a point p_1 to the unit parabola construct first the polar $D(p_1)$ of p_1 with respect to the parabola and then construct the polars of the intersection points $D(p_2)$.

- *Point-location test*: A point $p_1(p_3)$ lies "outside (inside)" the conic depending upon its corresponding polar $D(p_1)(D(p_3))$ intersecting (or not) the conic.

6 Hybrid Roider Method

A simple and elegant method for testing intersection of convex planar polygons was discovered by Roider and Stifter, cf. [7, 8, 6] and was subsequently generalized to 3-dimensional general convex sets and *collision detection*.

We will now outline a hybrid Roider method mixing the basic idea of the original and duality and linear programming following a divide-and-conquer strategy.

Given two convex planar polygons, A and B, the method starts with a point S_1 in B and constructs the supporting lines to A.

In case the wedge $T_1S_1T_2$, which contains all of A, does not intersect B, it follows that A and B are disjoint and S_1 is a witness to disjointness.

In the situation depicted below,

due to convexity, we can restrict atention to the fragments of the boundaries S_0, S_2 (not containing S_1, the "bottom" of B) and T_1, T_2 (the "top" of A). In the original method point S_2 is now chosen as S_1, supporting lines to A are constructed, etc. We suggest here other possible approaches:

The problem can be transformed into a linear separability problem for the sets of vertices lying on the bottom of B, respectively on the top of A.

A *distance function* similar to the one defined in the Muller-Preparata algorithm can be defined

$$d(S_B, T_A) = d(S_1, T_A) - d(S_1, S_B)$$

between points on the bottom of B and top of A. The problem becomes now one of finding a minimum point for this convex (hence unimodal) function, cf. [4].

This reduction of the problem is especially useful in dealing with objects whose boundaries consist of "curve segments". In this case the Roider method can serve as a preprocessing step determining the two segments of the boundary that are "closest" to each other. It then suffices to determine whether these two segments intersect or not. For this purpose, the minimum of the unimodal function can be computed. (For objects with boundaries not composed from segments (i.e. defined by a single equation) this kind of reduction might not be as useful).

From the complexity viewpoint, whether or not we can improve upon the pure Roider method, it is crucial which specific method is used for computing the minimum of the unimodal function; e.g. using an iterative method based on the construction of lines normal to the boundaries of the object renders a less efficient technique than finishing the search by the pure Roider method, cf. [6].

The situation is different for convex polygons. The iterative part (constructing tangents and intersection points) does not necessarily terminate in this case. It can be used, however, to single out the vertex of B that is closest to A, even if A itself is not a polygon. Thus, the problem of testing A and B for intersection is reduced to computing the minimal distance between a line segment and a convex curve segment. Iterating further by the Roider method one can even detect the vertex of A that is closest to B (if the minimal distance is attained at a vertex of A).

In case the minimal distance is attained on a line segment, the Roider method terminates, and an extra minimum computation is again not necessary.

These variations of the Roider method concern only the technique of determining a witness to disjointness or intersection, and not the properties of such a witness. (Note that the computed witness to disjointness will always lie between the last two intersection points found by the Roider method when applied until termination.) Thus, the witness to disjointness found by the hybrid Roider method is especially a point for which the tangents to A separate A and B. This is the prerequisite for using the results provided by the hybrid Roider method for collision (and not only intersection) test.

7 Conclusions and Further Research

We have reviewed a number of important intersections tests in 2 and 3 dimensions highlighting the basic strategies: divide-and-conquer, linear programming, duality. It became apparent that most of the optimal algorithms are refined by employing mixed strategies.

This was the main impetus for the design of a hybrid Roider method that could take advantage of different powerful methods customized to the specific of the objects involved.

Further work is necessary to finetune these options according to the class of objects tested for intersections, as well as for further generalizations in 3 and higher dimensions required for the complex applications to robot motion planning (collision-free paths), [7, 8].

Acknowledgement: This work was supported in part by a grant from the Austrian National Science Foundation (FWF) under the project no. P8620-TEC *Collision Detection by Means of the Roider Method and Linear Programming* coordinated at RISC-Linz by Prof. Sabine Stifter.

References

[1] Edelsbrunner, H., (1987). *Algorithms in Combinatorial Geometry*, Springer.

[2] Mayr, H., (1991). *Different Alternatives to Formulate the Robotics Colision Problem as an LP Model*, Proc. IFIP Conference on System Modelling and Optimzation.

[3] Moret, B. M. E., Shapiro, H. D., (1991). *Algorithms: From P to NP*, Benjamin/Cummings.

[4] Preparata, F. P., Shamos, M. I., (1988). *Computational Geometry*, Springer.

[5] Schrijver, A., (1986). *Theory of Linear and Integer Programming*, Wiley.

[6] Stifter, S., (1988). *A Medley of Solutions to the Robot Collision Problem in Two and Three Dimensions*, PhD. Thesis, Univ. Linz, RISC-Linz Tech. Rep. 88-12.0.

[7] Stifter, S., (1988). *A Generalization of the Roider Method to Solve the Robot Collision Problem in 3D*, Proc. ISSAC'88, Springer LNCS 358.

[8] Stifter, S., (1992). *The Roider Method: A Method for Static and Dynamic Collision Detection*, Advances in Computing Research, vol. 6, Issues in Robotics and Non-Linear Geometry, JAI Press.