About smallest enclosing spheres

Reynald Affeldt *

July 10, 2000

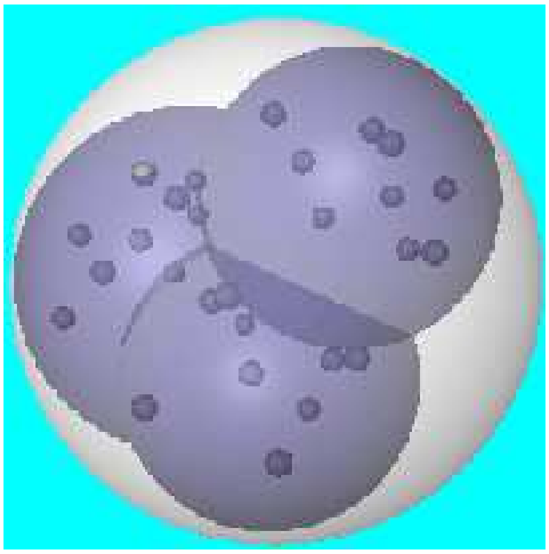

Figure 1: Enclosing Ball Software [2]

1 Introduction

In computer graphics and robotics, higher preformance is often achieved through precomputing bounding volumes like spheres that enclose more complicated shapes. Afterwards, it is easier to determine quickly the first objects hit by a ray of light in ray-tracing, or a collision-free motion for a robot.

The problem of the smallest enclosing sphere is an instance of the problem of precomputing bounding volumes. Given a set S of n points p_i in a d-dimensional space E, find the radius r and the center p_0 of the smallest sphere that contains all n points:

$$\min_{p_0 \in E} \left(\max_{p_i \in S} ||p_i - p_0|| \right)$$

In this report, we will see some solutions to the problem of the smallest enclosing sphere and a Java implementation of the lowest complexity algorithm we have found.

2 Quick Approximate solution

One can first to an approximate solution to the problem. First, we compute the diameter D of the set S: it is the distance between the points in S that are furthest apart. A randomized algorithm for computing this quantity can be found in [3] (p. 256). The computation is achieved in expected time $O(n \log n)$. Let p_1 and p_2 be any two points such that $||p_2 - p_1|| = D$. We know that every

*96393, affeldt@yl.is.s.u-tokyo.ac.jp

point of S is then contained in the intersection of the two spheres of radius D centered respectively at p_1 and p_2.

The sphere of radius D centered at p_1 is a small enclosing sphere but it is not the smallest. A slightly better approximation is to take the sphere centered at the midpoint between p_1 and p_2 and which radius is $\sqrt{3}\frac{D}{2}$. This sphere encloses the intersection of the two previous spheres, and therefore also encloses all the points from the set S. It is moreover of radius less then D.

3 Brute-force searching

3.1 Two dimensions

Let us first consider the problem in two dimensions. We can consider all the pairs of points and, for each pair of points, calculate the minimal circle covering that pair: the radius of such a circle is just half the distance between the two points and the center is just their midpoint. Then we can consider all the triples of points and calculate the minimal circle covering that triple: it is the circumscribed circle whose center is at the same distance from all the points of the triple and whose radius is given right away. We do not keep all the circles: we throw away those that do not cover all the set of points S and take the one with the minimal radius among the remainding circles.

3.2 Three dimensions

We know consider the problem in three dimensions. We first consider all the pairs of points and calculate the minimal sphere covering that pair: the radius of such a sphere is just half the distance between the two points and the center is just their midpoint. We consider all the triples and determine for each of them the corresponding plane. We find the smallest covering circle belonging to that plane and record the result of the three-dimensional revolution of this circle around one of its diameter. We eventually take all the quadruples of points and calculate the minimal sphere covering each quadruple. This is the circumscribed sphere.

Given the points $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, $p_3 = (x_3, y_3, z_3)$, and $p_4 = (x_4, y_4, z_4)$, the equation of the circumscribed sphere is given by the determinant:

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ x_1^2+y_1^2+z_1^2 & x_1 & y_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0.$$

The resulting equation has the form:

$$x^2 + y^2 + z^2 - 2ax - 2by - 2cz + d + a^2 + b^2 + c^2 - r^2 = 0$$

with a center $p_0 = (a, b, c)$ and a radius $r = \sqrt{a^2 + b^2 + c^2 - d}$. The sphere does not exist if the latter radius is a complex number. Here are the developed solutions that directly give a, b, c, and d, with first an intermediate result s (those formulaes are used in the small implementation we will present in section 6):

$$s = \begin{array}{l} x_1\,(y_2z_3 - y_2z_4 - y_3z_2 + y_3z_4 + y_4z_2 - y_4z_3) \\ -x_2\,(y_1z_3 - y_1z_4 - y_3z_1 + y_3z_4 + y_4z_1 - y_4z_3) \\ +x_3\,(y_1z_2 - y_1z_4 - y_2z_1 + y_2z_4 + y_4z_1 - y_4z_2) \\ -x_4\,(y_1z_2 - y_1z_3 - y_2z_1 + y_2z_3 + y_3z_1 - y_3z_2) \end{array}$$

$$a = \frac{\begin{array}{l} -(-(x_1^2+y_1^2+z_1^2)(y_2z_3 - y_2z_4 - y_3z_2 + y_3z_4 + y_4z_2 - y_4z_3) \\ +(x_2^2+y_2^2+z_2^2)(y_1z_3 - y_1z_4 - y_3z_1 + y_3z_4 + y_4z_1 - y_4z_3) \\ -(x_3^2+y_3^2+z_3^2)(y_1z_2 - y_1z_4 - y_2z_1 + y_2z_4 + y_4z_1 - y_4z_2) \\ +(x_4^2+y_4^2+z_4^2)(y_1z_2 - y_1z_3 - y_2z_1 + y_2z_3 + y_3z_1 - y_3z_2)) \end{array}}{2s}$$

$$b = \frac{\begin{array}{l} -(-(x_1^2+y_1^2+z_1^2)(-x_2z_3 + x_2z_4 + x_3z_2 - x_3z_4 - x_4z_2 + z_3x_4) \\ +(x_2^2+y_2^2+z_2^2)(-x_1z_3 + x_1z_4 + x_3z_1 - x_3z_4 - x_4z_1 + z_3x_4) \\ -(x_3^2+y_3^2+z_3^2)(-x_1z_2 + x_1z_4 + x_2z_1 - x_2z_4 - x_4z_1 + z_2x_4) \\ +(x_4^2+y_4^2+z_4^2)(-x_1z_2 + x_1z_3 + x_2z_1 - x_2z_3 - x_3z_1 + z_2x_3)) \end{array}}{2s}$$

$$c = \frac{\begin{array}{l} -(-(x_1^2+y_1^2+z_1^2)(x_2y_3-x_2y_4-x_3y_2+x_3y_4+x_4y_2-x_4y_3) \\ +(x_2^2+y_2^2+z_2^2)(x_1y_3-x_1y_4-x_3y_1+x_3y_4+x_4y_1-x_4y_3) \\ -(x_3^2+y_3^2+z_3^2)(x_1y_2-x_1y_4-x_2y_1+x_2y_4+x_4y_1-x_4y_2) \\ +(x_4^2+y_4^2+z_4^2)(x_1y_2-x_1y_3-x_2y_1+x_2y_3+x_3y_1-x_3y_2)) \end{array}}{2s}$$

$$d = \frac{\begin{array}{l} (-(x_1^2+y_1^2+z_1^2)(-x_2y_3z_4+x_2y_4z_3+x_3y_2z_4-x_3y_4z_2-x_4y_2z_3+x_4y_3z_2) \\ +(x_2^2+y_2^2+z_2^2)(-x_1y_3z_4+x_1y_4z_3+x_3y_1z_4-x_3y_4z_1-x_4y_1z_3+x_4y_3z_1) \\ -(x_3^2+y_3^2+z_3^2)(-x_1y_2z_4+x_1y_4z_2+x_2y_1z_4-x_2y_4z_1-x_4y_1z_2+x_4y_2z_1) \\ +(x_4^2+y_4^2+z_4^2)(-x_1y_2z_3+x_1y_3z_2+x_2y_1z_3-x_2y_3z_1-x_3y_1z_2+x_3y_2z_1)) \end{array}}{s}$$

Once we have computed all those spheres, we throw away the sphere which do not cover all the points from the set S. From the remaining spheres, we just need to take the one with the minimal radius.

3.3 d dimensions

The complexity of such a brute-force method is n^{d+1} where d is the dimension of the space we are working in.

4 From the convex hull

4.1 Two dimensions

Let us first consider the two dimensional instance. We have a set S of points p_i in the plane and we would like to find the smallest enclosing circle. Let us first take the convex hull (i.e., the smallest convex set that contains all of the n points of the set S). The problem can be solved by a randomized incremental algorithm whose complexity is given by $O(n \log n)$.

The circle we are looking for has the following property: it must pass through at least three points of the convex hull. Ab absurdo, it is passes through only two points of the convex hull, then it can be replaced by a smaller circle that however sill contains all the points of the set S.

From now, we consider all the possible triples that it is possible to build with points from the convex hull. For each triple, we build the corresponding circumscribed circle. Then we check all the circles produced so far to find those that contain all the points of the set S. Eventually we choose the circle with the smallest radius.

4.2 Three dimensions

We now consider the three dimensional instance. It is basically the same algorithm except that we now take all possible choices of four points from the convex hull. They determine a certain number of spheres we inspect to find the ones that contain all the points from the set S. Eventually, we take the sphere with the smallest radius.

Because the convex hull of a set of points contains less points than the set in question, this solution is faster than the brute-force method we have seen at the previous section.

When performing the above algorithm one may take care of special cases in which points are in such a position that there is no solution. For each construct to be built, we eliminate the cases of three points on a line for the two dimensional problem and of four points on a plane for the three dimensional problem.

5 Welzl's Randomized Algorithm

5.1 Two dimensions

We have seen in the previous section that the smallest enclosing circle of a set S is necessarily determined by three points of S. Let us call S' the set of those three points. The smallest enclosing circle for S and S' is the same:

$$\mathsf{sec}(S') = \mathsf{sec}(S).$$

We deduce from this fact that:

$$\begin{gathered} p \in S - S' \Rightarrow \mathsf{sec}(S - \{p\}) = \mathsf{sec}(S) \\ \Updownarrow \\ \mathsf{sec}(S - \{p\}) \neq \mathsf{sec}(S) \Rightarrow p \in S' \subseteq S. \end{gathered}$$

This suggests an incremental construction of the smallest enclosing circle of a set S of points. If we already know $\mathsf{sec}(\{p_1, p_2, \dots, p_i\})$, then if $p_{i+1} \in \mathsf{sec}(\{p_1, p_2, \dots, p_i\})$, then $\mathsf{sec}(\{p_1, p_2, \dots, p_i\}) = \mathsf{sec}(\{p_1, p_2, \dots, p_{i+1}\})$. If $p_{i+1} \notin \mathsf{sec}(\{p_1, p_2, \dots, p_i\})$, then a new smallest enclosing circle must be computed that contains p_{i+1}, and this circle will be $\mathsf{sec}(\{p_1, p_2, \dots, p_{i+1}\})$. The latter problem seems easier than the general one since we know for sure that the smallest enclosing circle we seek for will pass through p_{i+1}.

This is reminiscent of Seidel's linear programming algorithm (see [3], p. 268). Motwani and Raghavan give a specialized version of the algorithm that perfectly matches our problem, namely the BasisLP algorithm:

Algorithm BasisLP

Input: S, R
Output: A basis for S

0. If $S = R$, output R
1. Pick a random constraint $h \in S - R$
   $R' = \mathsf{BasisLP}(S - \{h\}, T)$
2.1 If h does not violate R', output R'
2.2 else output BasisLP(S, R′ ∪ {h})

S is here a set of constraints. We want to reduce this set to a small set of constraints from which a solution is readily computable. R is a germ for this so-called basis of S. The idea of the algorithm is to pick up randomly a constraint and to check if it is redundant with the others. If it is, we can decrease the set of constraints by one. If it is not, we must residualize it. This algorithm is known to have a linear complexity if the dimension of the problem is constant.

In the context of the smallest enclosing circle, the constraint h is the belonging of a point p to R'. The result we are expecting is a set of constraints that enables us to compute readily the smallest enclosing circle, i.e. $d+1$ points in the d-dimensional problem. Let us call SEC the smallest enclosing circle procedure. The above algorithm leads us to the following instance:

Algorithm SEC

Input: S, $R = \emptyset$
Output: The smallest enclosing circle for S

0. If $S = \underline{\emptyset}$, output $\underline{\mathsf{SEC}(\emptyset, R)}$
1. Pick a random point $p \in S - R$
   $R' = \mathsf{SEC}(S - \{p\}, R)$
2.1 If $p \in R'$, output R'
2.2 else output $\mathsf{SEC}(S \underline{- \{p\}}, R' \cup \{p\})$

The underlined parts exhibit the (slight) differences with the BasisLP algorithm. Because the size of S decreases at each recursive call, termination of the algorithm is ensured. In [1], Welzl shows that this randomized algorithm computes the smallest enclosing disk of a set of n points in expected $O(n)$ time.

5.2 d dimensions

The previous algorithm is still valid for higher dimensions. The only difference is the minimal number of points that is necessary to compute a sphere.

6 Implementation

We have realized a small implementation of Welzl's algorithm in Java in order to concretely visualize what is going on.

Here are the screenshots related to the computation of the smallest enclosing sphere of a set of seven random points. Images chronologically appear from left to right. Points are partitioned between blue points (that effectively determined the sphere) and red points. All the points lie inside the sphere at the end.

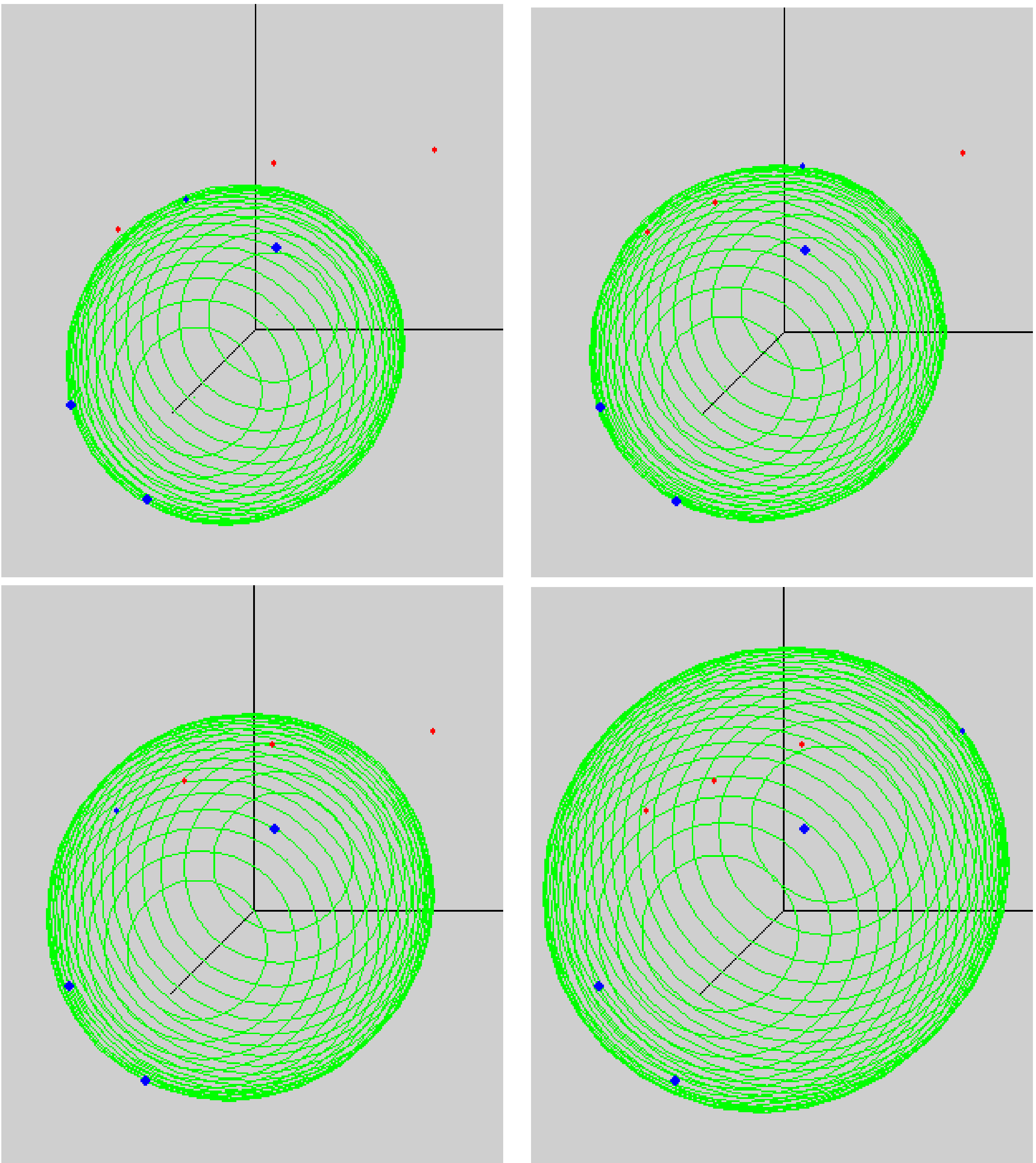

Figure 2: Welzl's Algorithm Screenshots

7 Bibliography

References

[1] Emo Welzl, *Smallest enclosing disks (balls and ellipsoids)*, New Results and New Trends in Computer Science, 1991.

[2] Dave White, *Enclosing Ball Software*, `http://www-vision.ucsd.edu/~ dwhite/ball.html`, 1996.

[3] Rajeev Motwani, Prabhakar Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[4] Eric W. Weisstein, *Eric Weisstein's World of Mathematics*, xtt-thttp://mathworld.wolfram.com/, 1996-2000.