ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters ●●● (●●●●) ●●●–●●●

Information Processing Letters

www.elsevier.com/locate/ipl

A fast deterministic smallest enclosing disk approximation algorithm

Frank Nielsen [a], Richard Nock [b,*]

[a] *Sony Computer Science Laboratories Inc., 3-14-13 Higashi Gotanda, Shinagawa-Ku, 141-0022 Tokyo, Japan*

[b] *Université des Antilles-Guyane, Campus de Schoelcher, Departement Scientifique Interfacultaire, BP 7209, Schoelcher (Martinique) 97233, France*

Received 11 May 2003; received in revised form 24 November 2004

Communicated by F. Dehne

Abstract

We describe a simple and fast $O(n \log_2 \frac{1}{\varepsilon})$-time algorithm for finding a $(1+\varepsilon)$-approximation of the smallest enclosing disk of a planar set of n points or disks. Experimental results of a readily available implementation are presented.

Keywords: Approximation algorithms; Computational geometry; Minimum enclosing ball

1. Introduction

The smallest enclosing disk (SED for short) problem dates back to 1857 when Sylvester [5] first asked for the smallest disk enclosing n points on the plane. Although $O(n \log n)$-time algorithms were designed for the planar case in the early 1970s, its complexity was only settled in 1984 with Megiddo's first linear time algorithm [2] for solving linear programs in *fixed dimension*. Unfortunately, these algorithms exhibit a large constant hidden in the big-Oh notation and do not perform so well in practice. In this note, we concentrate exclusively on the planar case approximation, and we refer readers to the papers [1,3] for experimental comparisons of recently designed algorithms that either solve the exact or approximate smallest enclosing ball problems in *unbounded dimension*. Computing smallest enclosing disks are useful for metrology, machine learning and computer graphics problems. Fast constant approximation heuristics are popular in computer graphics [4]. Let $\mathcal{P} = \{P_i = (x_i, y_i),\ i \in \{1, \ldots, n\}\}$ be a set of n planar points. We use notations $x(P_i) = x_i$ and $y(P_i) = y_i$ to mention point coordinates. Let $\mathrm{Disk}(C^*, r^*)$ be the smallest enclosing disk of $\mathcal{P}$ of center point C^* (also called circumcenter or Euclidean 1-center) and minimum ra-

* Corresponding author.
E-mail addresses: frank.nielsen@acm.org (F. Nielsen), rnock@martinique.univ-ag.fr (R. Nock).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.12.006

dius r^*. We want to compute a $(1+\varepsilon)$-approximation, that is, a disk $\mathrm{Disk}(C, r)$ such that $r \leqslant (1+\varepsilon)r^*$ and $\mathcal{P} \subseteq \mathrm{Disk}(C, r)$. Our paper aims at designing a fast deterministic (i.e., worst-case time bounded) approximation algorithm that is suitable for real-time demanding applications. Our simple implementation[1] for point/disk sets is a mere 30-line code which do not require to compute the tedious basis primitive of the smallest disk enclosing three disks. Moreover, we exhibit a robust approximation algorithm using only algebraic predicates of degree 2 on Integer arithmetic. In Section 6, we show that our floating-point implementation outperforms or fairly competes with traditional methods while guaranteeing worst-case time.

2. Piercing/covering duality

Let us consider the general case of a disk set $\mathcal{D} = \{D_i = \mathrm{Disk}(P_i, r_i),\ i \in \{1, \ldots, n\}\}$ to explain the piercing/covering duality. Our approximation algorithm proceeds by solving dual piercing *decision problems* (DPs for short; see Fig. 1): given a set of corresponding dual disks $\mathcal{B}(r) = \{B_i = \mathrm{Disk}(P_i, r - r_i), i \in \{1, \ldots, n\}\}$, determine whether $\bigcap \mathcal{B}(r) = \bigcap_{i=1}^{n} B_i = \emptyset$ or not.

Lemma 1. *Observe that for $r \geqslant r^*$, there exists a (unique) disk B of radius $r(B) = r - r^*$ centered at $C(B) = C^*$ fully contained inside $\bigcap \mathcal{B}$.*

Proof. In order to ensure that C^* is inside each $B_i(r)$, a sufficient condition is to have $r \geqslant \max_i\{r_i + d_2(P_i, C^*)\}$. Since $B_i \subseteq \mathrm{Disk}(C^*, r^*)$, $\forall i \in \{1, 2, \ldots, n\}$, we have

$$\max_i\{r_i + d_2(P_i, C^*)\} \leqslant r^*. \qquad (\star)$$

Thus, provided $r \geqslant r^*$, we have $C^* \in \bigcap \mathcal{B}(r)$. Now, notice that $\forall i \in \{1, 2, \ldots, n\}$, $\forall 0 \leqslant r' \leqslant (r - r_i) - d_2(P_i, C^*)$, $\mathrm{Disk}(C^*, r') \subseteq B_i(r)$. Thus, if we ensure that $r' \leqslant r - \max_i(r_i + d_2(P_i, C^*))$, then $\mathrm{Disk}(C^*, r') \subseteq \bigcap \mathcal{B}(r)$. From ineq. $(\star)$, we choose $r' = r - r^*$ and obtain the lemma (see Fig. 1). Uniqueness follows from the proof by contradiction of [6]. □

[1] Source code in C is available at http://www.csl.sony.co.jp/person/nielsen/WIP/MEB/.

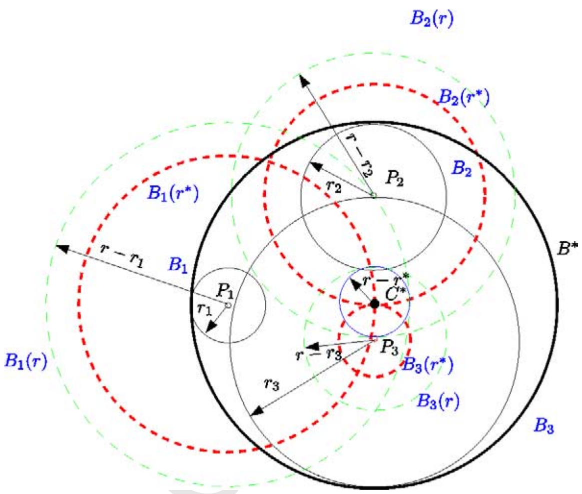

Fig. 1. Covering/piercing duality principle.

3. Algorithm outline

Our approximation algorithm proceeds by solving a sequence of dual piercing *decision problems* (see Fig. 1): given a set of disks $\mathcal{B}(r) = \{B_i = \mathrm{Disk}(P_i, r), i \in \{1, \ldots, n\}\}$, determine whether $\bigcap \mathcal{B}(r) = \bigcap_i B_i = \emptyset$ or not. We relax the 1-piercing point problem to that of a common piercing εr^*-disk (i.e., a disk of radius εr^*): report whether there exists a disk $B = \mathrm{Disk}(C, \varepsilon r^*)$ such that $B \subseteq \bigcap \mathcal{B}(r)$ or not. Algorithm 1 describes the complete approximation procedure.

3.1. Solving decision problems

We explain procedure DecisionProblem of Algorithm 1. Let $[x_m, x_M]$ be an interval on the x-axis where an εr^*-disk center might be located if it exists. (That is $x(C) \in [x_m, x_M]$ if it exists.) We initialize x_m, x_M as the x-abscissae extrema: $x_m = \max_i(x_i) - r$, $x_M = \min_i(x_i) + r$. If $x_M < x_m$ then clearly vertical line $L: x = (x_m + x_M)/2$ separates two extremum disks (those whose corresponding centers give rise to x_m and x_M) and therefore $\mathcal{B}(r)$ is not 1-pierceable (therefore not εr^*-ball pierceable). Otherwise, the algorithm proceeds by dichotomy (see Fig. 2). Let $e = (x_m + x_M)/2$ and let L denotes the vertical line $L: x = e$. Denote by $\mathcal{B}_L = \{B_i \cap L \mid i \in \{1, \ldots, n\}\}$ the set of n y-intervals obtained as the intersection of the disks of $\mathcal{B}$ with line L. We check whether $\mathcal{B}_L =$

DecisionProblem($\mathcal{P}$, xmin, xmax, r, ε):
$x_M = \text{xmin} + r$;
$x_m = \text{xmax} - r$;
while $x_M - x_m \geqslant \varepsilon$ **do**
  $l = \frac{x_M + x_m}{2}$;
  $y_m = \max_{i\in\{1,\dots,n\}} y_i - \sqrt{r^2 - (l - x_i)^2}$;
  $m = \operatorname{argmax}_{i\in\{1,\dots,n\}} y_i - \sqrt{r^2 - (l - x_i)^2}$;
  $y_M = \min_{i\in\{1,\dots,n\}} y_i + \sqrt{r^2 - (l - x_i)^2}$;
  $M = \operatorname{argmin}_{i\in\{1,\dots,n\}} y_i + \sqrt{r^2 - (l - x_i)^2}$;
  if $y_M \geqslant y_m$ **then**
    $x = l$;
    $y = (y_m + y_M)/2$;
    return true;
  else
    //m and M are arg indices of y_m and y_M;
    if $(x_m + x_M)/2 > l$ **then**
      $x_m = l$;
    else
      $x_M = l$;
return false;
SmallEnclosingDisk($\mathcal{P}$, ε):
$\text{xmin} = \min_{i\in\{1,\dots,n\}} x_i$;
$\text{xmax} = \max_{i\in\{1,\dots,n\}} x_i$;
$d_1 = \max_{i\in\{1,\dots,n\}} \|P_i - P_1\|$;
$b = d_1$;
$a = \frac{d_1}{2}$;
$\varepsilon \leftarrow \frac{1}{4}(b - a)\varepsilon$;
while $b - a > \varepsilon$ **do**
  $r = (a + b)/2$;
  pierceable = DecisionProblem($\mathcal{P}$, xmin, xmax, r, ε);
  if pierceable **then**
    $b = r$;
  else
    $a = r$;

Algorithm 1. $(1+\varepsilon)$-approximation of the minimum enclosing disk of $\mathcal{P}$.

$\{B_i \cap L = [a_i, b_i] \mid i \in \{1,\dots,n\}\}$ is 1-pierceable or not. Since $\mathcal{B}_L$ is a set of n y-intervals, we just need to check whether $\min_i b_i \geqslant \max_i a_i$ or not. If $\bigcap \mathcal{B}_L \neq \emptyset$, then we have found a point $(e, \min_i b_i)$ in the intersection of all balls of $\mathcal{B}$ and we stop recursing. (In fact we found a $(x = e, y = [y_m = \max_i a_i, y_M = \min_i b_i])$ vertical piercing segment.) Otherwise, we have $\bigcap \mathcal{B}_L = \emptyset$ and need to choose on which side of L to recurse. Without loss of generality, let B_1 and B_2 denote the two disks whose corresponding y-intervals on L are disjoint. We choose to recurse on the side where $B_1 \cap B_2$ is located (if the intersection is empty then we stop by reporting the two nonintersecting balls B_1 and B_2). Otherwise, $B_1 \cap B_2 \neq \emptyset$ and we branch on the side where $x_{B_1 B_2} = (x(C(B_1)) + x(C(B_2)))/2$ lies. At each stage of the dichotomic process, we halve the x-axis range where the solution is to be located (if it exists). We stop the recursion as soon as $x_M - x_m < \varepsilon\frac{r}{2}$. Indeed, if $x_M - x_m < \varepsilon\frac{r}{2}$ then we know that *no center of a ball* of radius εr is contained in $\bigcap \mathcal{B}$. (Indeed if such a ball exists then *both* $\bigcap \mathcal{B}_{L(x_m)} \neq \emptyset$ and $\bigcap \mathcal{B}_{L(x_M)} \neq \emptyset$.) Overall, we recurse at most $3 + \lceil \log_2 \frac{1}{\varepsilon} \rceil$ times since the initial interval width $x_M - x_m$ is less than $2r^*$ and we always consider $r \geqslant \frac{r^*}{2}$.

3.2. *Radius dichotomy search*

Finding the minimum enclosing disk radius amounts to find the smallest value $r \in \mathbb{R}^+$ such that $\bigcap \mathcal{B}(r) \neq \emptyset$. That is $r^* = \operatorname{argmin}_{r\in\mathbb{R}^+} \bigcap \mathcal{B}(r) \neq \emptyset$. We seek an $(1+\varepsilon)$-approximation of the minimum enclosing ball of points by doing a straightforward dichotomic process on relaxed decision problems as explicited by procedure SmallEnclosingDisk. We always keep a solution interval $[a, b]$ where r^* lies, such that at any stage we have $\bigcap \mathcal{B}(a - \frac{\varepsilon r^*}{2}) = \emptyset$ and $\bigcap \mathcal{B}(b) \neq \emptyset$. Without loss of generality, let P_1 denote the leftmost x-abscissae point of $\mathcal{P}$ and let $P_2 \in \mathcal{P}$ be the maximum distance point of $\mathcal{P}$ from P_1. We have $r = d_2(P_1, P_2) \geqslant r^*$ (since $\mathcal{P} \subseteq \text{Disk}(P_1, r)$). But $d_2(P_1, P_2) \leqslant 2r^*$ since both P_1 and P_2 are contained inside the unique smallest enclosing disk of radius r^*. Thus we have $r^* \in [\frac{r}{2}, r]$. We initialize the range by choosing $a = \frac{r}{2} \leqslant r^*$ and $b = r \leqslant 2r^*$. Then we solve the $\frac{\varepsilon}{4}r$-disk piercing problem with disks of radius $e = (a + b)/2$. If we found a common piercing point for $\bigcap \mathcal{B}(e)$ then we recurse on $[a, e]$. Otherwise we recurse on $[e, b]$. We stop as soon as $b - a \leqslant \varepsilon\frac{r}{4}$. (Therefore after $O(\log_2 \frac{1}{\varepsilon})$ iterations since the initial range width $b - a \leqslant r^*$.) At any stage, we assert that $\bigcap \mathcal{B}(a - \frac{\varepsilon r}{4}) = \emptyset$ (by answering that $\bigcap \mathcal{B}(a)$ does not contain any ball of radius $\frac{\varepsilon r}{4}$) and $\mathcal{B}(b) \neq \emptyset$. At the end of the recursion process, we get an interval $[a - \frac{\varepsilon r}{4}, b]$ where r^* lies in. Since $b - a \leqslant \varepsilon\frac{r}{4} \leqslant \varepsilon\frac{r^*}{2}$ and $|r^* - a| < \frac{\varepsilon r}{4} \leqslant \frac{\varepsilon r^*}{2}$ (because $\mathcal{B}(a - \frac{\varepsilon r}{4}) = \emptyset$), we get: $b \leqslant r^* + 2\varepsilon\frac{r}{4}$. Since $r \leqslant 2r^*$, we obtain a $(1+\varepsilon)$-approximation of the minimum enclosing ball of the point set. Thus, by solving $O(\log_2 \frac{1}{\varepsilon})$ decision

S0020-0190(04)00358-8/SCO AID:3208 Vol.●●●(●●●)
IPL:m3 v 1.32 Prn:21/12/2004; 13:14

ipl3208

[DTD5] P.4 (1-6)
by:violeta p. 4

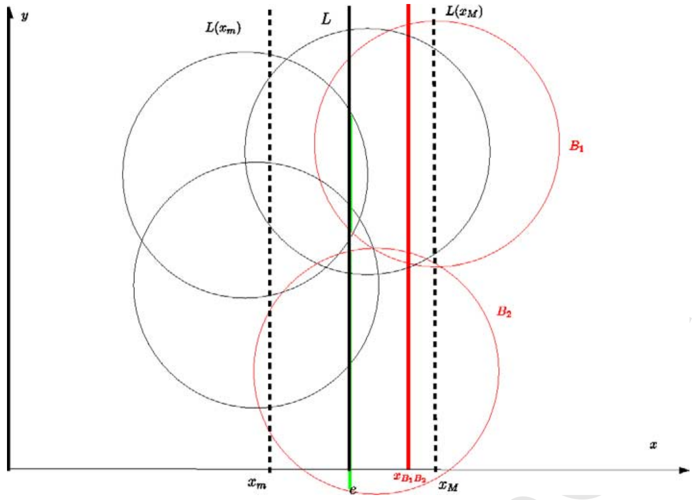

Fig. 2. A recursion step: $L: x = e$ intersects all balls. Two y-intervals do not intersect on L. We recurse on x-range $[e, x_M]$.

problems, we obtain a $O(n \log_2^2 \frac{1}{\varepsilon})$-time deterministic $(1+\varepsilon)$-approximation algorithm.

3.3. Bootstrapping

We bootstrap the previous algorithm in order to get a better $O(n \log_2 \frac{1}{\varepsilon})$-time algorithm. The key idea is to shrink potential range $[a, b]$ of r^* by selecting iteratively different approximation ratios ε_i until we ensure that, at kth stage, $\varepsilon_k \leqslant \varepsilon$. Let $\mathrm{Disk}(C, r)$ be a $(1+\varepsilon)$-approximation enclosing ball. Observe that $|x(C) - x(C^*)| \leqslant \varepsilon r^*$. We update the x-range $[x_m, x_M]$ according to the so far found piercing point abcissae $x(C)$ and current approximation factor. We start by solving the approximation of the smallest enclosing ball for $\varepsilon_1 = \frac{1}{2}$. It costs $O(n \log_2 \frac{1}{\varepsilon_1}) = O(n)$. Using the final output range $[a, b]$, we now have $b - a \leqslant \varepsilon_1 r^*$. Consider $\varepsilon_2 = \frac{\varepsilon_1}{2}$ and reiterate until $\varepsilon_l \leqslant \varepsilon$. The overall cost of the procedure is

$$\sum_{i=0}^{\lceil \log_2 \frac{1}{\varepsilon} \rceil} O(n \log_2 2) = O\left(n \log_2 \frac{1}{\varepsilon}\right).$$

We get the following theorem:

Theorem 1. *A $(1+\varepsilon)$-approximation of the minimum enclosing disk of a set of n points on the plane can be computed efficiently in $O(n \log_2 \frac{1}{\varepsilon})$ deterministic time.*

4. Predicate degree

Predicates are the basic computational atoms of algorithms that are related to their numerical stabilities. In the exact smallest enclosing disk algorithm [6], the so-called *InCircle* containment predicate of algebraic degree 4 is used on Integers. Since we only use $\sqrt{\cdot}$ function to determine the sign of algebraic numbers, all computations can be done on Rationals using algebraic degree 2. We show how to replace the predicates of algebraic degree[2] 4 by predicates of degree 2 for Integers: "Given a disk center (x_i, y_i) and a radius r_i, determine whether a point (x, y) is inside, on or outside the disk". It boils down to compute the sign of $(x - x_i)^2 + (y - y_i)^2 - r_i^2$. This can be achieved using another dichotomy search on line $L: x = l$. We need to ensure that if $y_m > y_M$, then there do exist two disjoint disks B_m and B_M. We regularly sample line L such that if $y_m > y_M$, then there exists a sampling point in $[y_M, y_m]$ that does not belong to both disks B_m and B_M. In order to guarantee that setting, we need to ensure some *fatness* of the intersection of $\bigcap \mathcal{B}(r) \cap L$ by

[2] Comparing expressions $y_1 + \sqrt{r^2 - (l - x_1)^2} > y_2 + \sqrt{r^2 - (l - x_2)^2}$ is of degree 4 for Integers. Indeed, by isolating and removing the square roots by successive squaring, the predicate sign is the same as $(2r^2 - (l - x_1)^2 - (l - x_2)^2)^2 > 4(r^2 - (l - x_1)^2)(r^2 - (l - x_2)^2)$. The last polynomial has highest monomials of degree 4.

Table 1
Timings

Method/distribution	□ Square max	⊙ Ring max	□ Square avg	⊙ Ring avg
Eberly ($\varepsilon = 10^{-5}$)	**0.7056**	**0.6374**	0.1955	0.2767
Ritter ($\varepsilon > 10^{-1}$)	**0.0070**	**0.0069**	0.0049	0.0049
ASED ($\varepsilon = 10^{-2}$)	**0.0343**	**0.0338**	0.0205	0.0286
ASED ($\varepsilon = 10^{-3}$)	**0.0515**	**0.0444**	0.0284	0.0405
ASED ($\varepsilon = 10^{-4}$)	**0.0646**	**0.0617**	0.0392	0.0449
ASED ($\varepsilon = 10^{-5}$)	**0.0719**	**0.0726**	0.0473	0.0527

Experiments done on 1000 trials for point sets of size 100000. Maximum (max) and average (avg) running times are in fractions of a second. Bold numbers indicate worst-case timings.

recursing on the x-axis until we have $x_M - x_m \leqslant \frac{\varepsilon}{\sqrt{2}}$. In that case, we know that if there was a common εr^*-ball intersection, then its center x-coordinate is inside $[x_m, x_M]$: this means that on L, the width of the intersection is at least $\frac{\varepsilon}{\sqrt{2}}$. Therefore, a regular sampling on vertical line L with step width $\frac{\varepsilon}{\sqrt{2}}$ guarantees to find a common piercing point if it exists. A straightforward implementation would yield a time complexity $\mathrm{O}(\frac{n}{\varepsilon}\log_2\frac{1}{\varepsilon})$. However it is sufficient for each of the n disks, to find the upper most and bottom most lattice point in $\mathrm{O}(\log_2\frac{1}{\varepsilon})$-time using the floor function. Using the bootstrapping method, we obtain the following theorem:

Theorem 2. *A $(1+\varepsilon)$-approximation of the minimum enclosing disk of a set of n points on the plane can be computed in $\mathrm{O}(n\log_2\frac{1}{\varepsilon})$ time using Integer arithmetic with algebraic predicates InCircle of degree* 2.

5. Extension to disks

Our algorithm extends straightforwardly for sets of disks. Consider a set of n planar disks $\mathcal{D} = \{D_1, \ldots, D_n\}$ with $C(D_i) = P_i = (x_i, y_i)$ and $r(D_i) = r_i$. Let $\mathcal{B}(r) = \{B_i \mid C(B_i) = P_i$ and $r(B_i) = r - r_i\}$. Using the dual piercing principle, we obtain that $r^* = \operatorname{argmin}_{r\in\mathbb{R}} \bigcap \mathcal{B}(r) \neq \emptyset$. (We have $C^* = \bigcap \mathcal{B}(r^*)$.) Observe also that $r^* \geqslant \max_i r_i$. Initialization is done by choosing $b = r_1 + \max_i(d_2(P_1, P_i) + r_i)$ and $a = \frac{b}{2}$. We now let

$$x_{B_1B_2} = x_{B_1} + \frac{r_2^2 - r_1^2 + (r_1+r_2)^2}{2(r_1+r_2)^2}(x_{B_2} - x_{B_1}).$$

6. Experimental results

We compare our implementation with D.H. Eberly's C++ implementation[3] using double types that guarantees precision $\varepsilon = 10^{-5}$ and has expected running time $10n$ but no known worst-case bound better than $\mathrm{O}(n!)$. We also compare our code with Ritter's fast constant approximation ($\varepsilon \simeq 10\%$) greedy heuristic used in game programming [4]. Timings are obtained on an Intel Pentium(R) 4 1.6 GHz with 1 GB of memory for points uniformly distributed inside a unit square (□) and inside a unit ring of width 0.01 (⊙). Table 1 reports our timings. The experiments show that over a thousand square/ring random point sets, our algorithm (ASED) maximum time is much smaller than that of Eberly's (in addition, this latter algorithm requires $\tilde{\mathrm{O}}(\log_2^3 n)$ calls [6] to the expensive and intricate basic primitive of computing the circle passing through three points). Source codes in C for point and disk sets are available at http://www.csl.sony.co.jp/person/nielsen/WIP/MEB/.

Acknowledgements

The authors are grateful to the anonymous referees for their helpful suggestions.

References

[1] P. Kumar, J.S.B. Mitchell, A. Yıldırım, Computing core-sets and approximate smallest enclosing hyperspheres in high di-

[3] Source code available at http://www.magic-software.com.

S0020-0190(04)00358-8/SCO AID:3208 Vol.●●●(●●●)
IPL:m3 v 1.32 Prn:21/12/2004; 13:14
ipl3208
[DTD5] P.6 (1-6)
by:violeta p. 6

mensions, in: Algorithm Engineering and Experimentation (ALENEX), SIAM, Philadelphia, PA, 2003, pp. 45–55.

[2] N. Megiddo, Linear programming in linear time when the dimension is fixed, J. ACM 3 (1) (1984) 114–127.

[3] F. Nielsen, R. Nock, Approximating smallest enclosing balls, in: Computational Geometry and Applications (ICCSA), in: Lecture Notes in Comput. Sci., vol. 3045, Springer, Berlin, 2004, pp. 147–157.

[4] J. Ritter, An efficient bounding sphere, in: A. Glassner (Ed.), Game Programming Gems, Academic Press, New York, 1990, pp. 301–303.

[5] J.J. Sylvester, A question in the geometry of situation, Quart. J. Math. 1 (79) (1857).

[6] E. Welzl, Smallest enclosing disks (balls and ellipsoids), in: H. Maurer (Ed.), New Results and New Trends in Computer Science, in: Lecture Notes in Comput. Sci., vol. 555, Springer, Berlin, 1991, pp. 359–370.