Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica,
Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

Efficient Speed-up of Smallest Enclosing Circle Algorithm*

Michal Smolik[1][0000−0001−6409−1691] and Vaclav Skala[1][0000−0001−8886−4281]

Faculty of Applied Sciences, University of West Bohemia,
Plzen, Czech Republic
{smolik,skala}@kiv.zcu.cz

Abstract. The smallest enclosing circle is a well known problem. In this paper, we propose modifications to speed-up the existing Weltzl's algorithm. We perform the preprocessing to reduce as many input points as possible. The reduction step has lower computational complexity than the Weltzl's algorithm and thus speed-ups its computation. Next, we propose some changes of Weltzl's algorithm. In the end are summarize results, that show the speed-up for 10^6 input points up to 100 times compared to the original Weltzl's algorithm. Even more, the proposed algorithm is able to process much larger data sets than the standard Weltzl's algorithm.

Keywords: Smallest enclosing circle; space subdivision; convex hull; speed-up.

1 Introduction

The smallest enclosing circle problem is defined as the following. Given a set of $2D$ points, find the circle with the smallest radius such that all the given points are contained in either inside of this circle or its circumference. The basic properties of smallest enclosing circle are:

- Center is the point which is at the minimum distance from all the points on within the circle (considered together).
- Given any three points, we can uniquely define a circle, with these points on circumference.
- Given any set of $2D$ points, the smallest enclosing circle containing all points is unique.
- Given $N \geq 2$ points in the plane, the smallest circle that encloses them contains at least two of the points on its circumference.

A naive algorithm solves the problem with time complexity $O(N^4)$ by testing the circles determined by all pairs and triples of points. There are $O(N^3)$ such

* The research was supported by the Ministry of Education, Youth, and Sport of Czech Republic - University spec. research - 1311, and by SGS 2019-016.

circles and testing all points corresponding to each circle takes $O(N)$ time, i.e. the overall algorithm runs in $O(N^4)$ time.

The two best known algorithms are the Megiddo's algorithm, which has $O(N)$ running time [6], and the Welzl's algorithm, which has expected $O(N)$ running time [12]. Megiddo's algorithm [6] is based on the technique called prune and search reducing the size of the problem by removal of $N/16$ unnecessary points. The algorithm is rather complicated and it is reflected in big multiplicative constant. The reduction needs to solve twice the similar problem where the center of the enclosing circle is located using construction of bisectors of points. The Welzl's algorithm [12] uses the power of randomized incremental construction. It also exploits the fact, that if the point is not present in the circle constructed so far, then it will be present on the boundary of the new circle. Basically, these are the known facts, and a generalized version of these facts are used in the algorithm.

Other algorithm is [11], which presents a simple iterative algorithm for computing the smallest enclosing circle. The algorithm takes $O(N \log N)$ time. This is not optimal but its simplicity makes it a better alternative for medium sized problems than both previous algorithm. The algorithm [3] computes the minimal enclosing disk in an iterative manner and needs to define some distance delta value that is used in the algorithm.

Another alternative is algorithm [4,5], which computes the smallest enclosing circle not for all input points but for at least k points. It can also compute an approximation of this circle, which can be useful in some applications as well. The algorithm of smallest enclosing circle of at least k points is described in [1,2] as well. The paper [13] summarizes four different approaches for location of minimal enclosing circle for set of fixed circles.

A convex hull was one of the first sophisticated geometry algorithms to be computed and there are many variations of it. The most common form of this algorithm involves the determination of the smallest convex set, called the "convex hull", containing a discrete set of points. There are numerous applications for convex hulls: collision avoidance, maximum distance using convex hull diameter for large data sets [7,8], hidden object determination, and shape analysis, point inside polygon [9]. In this paper, convex hull from [10] is used to speed-up the computation of the smallest enclosing circle. This convex hull algorithm uses the space subdivision to achieve the time complexity of $O(N + s \log h)$, where s is number of suspicious points to be on convex hull and h is number of points on convex hull. It is expected that $O(s \log h) \ll O(N)$, thus the expected time complexity of the the convex hull algorithm [10] is $O(N)$.

2 Proposed Approach

The smallest enclosing circle contains on its circumference only points from convex hull. To speed-up the processing time, we created two steps algorithm. The first step is the reduction of all points that cannot form the smallest enclosing

Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica, Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

circle. The second step is the computation of smallest enclosing circle. We use the well known Weltzl's algorithm [12], which is fast and easy to implement.

To gain the speed-up from the reduction of input points, the convex hull construction needs to have lower computational cost than the Weltzl's algorithm. The convex hull algorithm [10] uses space subdivision technique to speed-up the computation and results in expected $O(N)$ algorithm.

The proposed algorithm for smallest enclosing circle is summarized in Algorithm 1 and is described in more details in following sub-chapters.

Algorithm 1 Smallest enclosing circle.

```
1: input : Input $2D$ points $P$.
2: output : Smallest enclosing circle of $P$.
3: procedure SMALLEST_CIRCLE($P$)
4:     points $C := Convex\_Hull(P)$                         ▷ Convex hull from [10]
5:     random shuffle $C$
6:     $\{C_1, C_2\}$ := find two points with max distance in $C$   ▷ Max distance from [8]
7:     $C_3$ := find point with max distance to center of $C_1$ and $C_2$
8:     $C_4$ := find point with max distance to $C_3$
9:     move $[C_4, C_3, C_2, C_1]$ to the end of $C$
10:    return $Welzl(C, \{\emptyset\})$
11: end
12:
13: input : Finite sets $P$ and $R$ of $2D$ points ($\|R\| \leq 3$).
14: output : Smallest enclosing circle of $P$ with $R$ on the boundary.
15: procedure WELZL($P$, $R$)           ▷ Welzl's algorithm without randomization
16:     if $P$ is empty or $\|R\| = 3$ then
17:         return smallest circle of $R$
18:     point $p$ := last from $P$
19:     circle $D := Welzl(P - \{p\}, R)$
20:     if $p$ is in $D$ then
21:         return $D$
22:     return $Welzl(P - \{p\}, R \cup \{p\})$
23: end
```

2.1 Convex Hull Construction

The fast convex hull algorithm [10] speed-ups the construction of convex hull by reducing the input points to only a few that are suspicious to form the convex hull. Detailed algorithm is described in [10]. We summarize only the important steps without many details.

The first step is to find the axis aligned bounding box (AABB). This is done by searching only 10% of input points. From points that define this AABB is constructed a polygon. All points that are inside this polygon cannot form the convex hull and are discarded (see the gray part in Fig. 1).

Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica,
Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

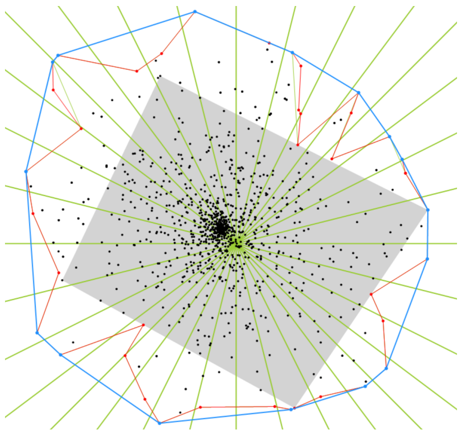


Fig. 1. Convex hull construction. Blue lines represent the convex hull.

The next step is to create a star shape division of remaining points. In each star cell, we determine the points with maximal distance from its center, that can form the convex hull (it can be more point in each cell). All other points are discarded.

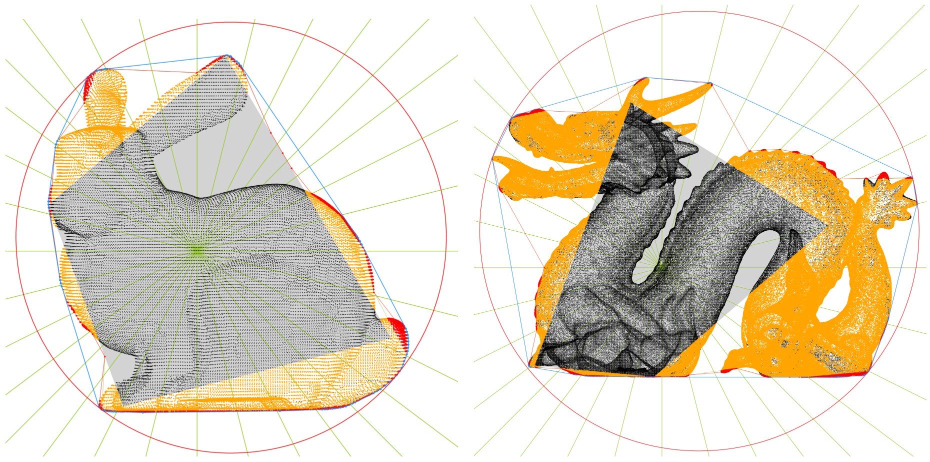
The remaining points (connected with red lines in Fig. 1) are the result of reduction. Now, we perform the actual algorithm for convex hull construction using only the suspicious points.

2.2 Smallest Enclosing Circle Computation

The Welzl's algorithm for smallest enclosing circle is recursive. Originally the algorithm is randomized. However, in this approach is used without randomization as the input points are already sorted in the required order to speed-up the computation.

One limitation of the Weltzl's algorithm is the recursion as it allows to process only smaller amounts of input points, due to the depth of recursion. We overcome this limitation by reducing the input points to only points on the convex hull. Now, the amount of points to be processed is limited only with RAM memory.

The original algorithm selects points totally in random manner. However, if the first selected points form a circle, which is big enough to contain almost all points, then the algorithm will speed-up. In the first step, we locate the two points $\{C_1, C_2\}$ with maximum distance from each other using algorithm [8]. Next, we locate the point C_3, which has the maximal distance from the center of previously located two points $\{C_1, C_2\}$. Last, we locate one more point C_4, which has the maximal distance from point C_3. All of those four points are moved to the end of input points for modified Weltzl's algorithm and the rest of points is randomly shuffled. In case, when the smallest enclosing circle is defined only by



(a) Stanford bunny: 35 947 points (b) Stanford dragon: 437 645 points

Fig. 2. The smallest enclosing circle for real data sets. The $2D$ input points are created by projection of all $3D$ points into xy plane. The models are from the Stanford $3D$ scanning repository (*http://www.graphics.stanford.edu/data/3Dscanrep/*)

two points, then this two points are exactly $\{C_1, C_2\}$ and the points $\{C_3, C_4\}$ are the same as $\{C_1, C_2\}$.

The initial input is a set of points P. The algorithm selects last one point p from P, and recursively finds the smallest circle containing $P - \{p\}$, i.e. all of the other points in P except p. If the returned circle also encloses p, it is the smallest circle for the whole of P and is returned.

Otherwise, point p must lie on the boundary of the result circle. It recurses with p as an additional point in R (points known to be on the boundary for already tested points from P).

The recursion terminates when P is empty, and a solution can be found from the points in R: for 0 or 1 points the solution is trivial, for 2 points the smallest circle has its center at the midpoint between the two points, and for 3 points the circle is the circumcircle of the triangle described by the points.

Recursion can also terminate when R has size 3 because the remaining points in P must lie within the circle described by R.

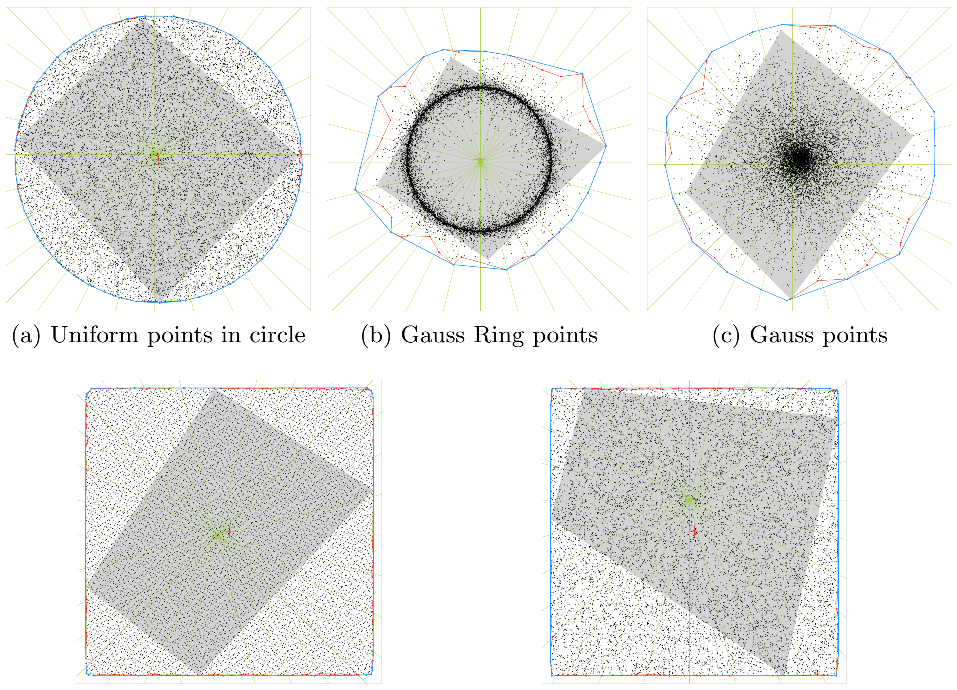
3 Experimental Results

In this chapter, we summarize the achieved results of the proposed algorithm for smallest enclosing circle for points in $2D$. The proposed approach is able to compute the smallest enclosing circle for both synthetic and real data sets as well. The synthetic data sets were used to measure the performance and the result of the proposed approach on real data sets is visualized in Fig. 2. The 3 points that define the smallest enclosing circle for bunny were actually selected between

Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica,
Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

the first 4 points that are processed in the beginning of the Weltzl's algorithm. For the dragon data set were 2 out of 3 points that define the smallest enclosing circle selected to be in the first 4 points that are processed by Weltzl's algorithm.

The proposed approach has been tested using several types of point distribution in $2D$. Some of the distributions are well known, randomly distributed uniform points inside a unit square or inside a unit circle and points with Gaussian distribution. All of these distributions are well known. The last two distributions used are Halton points and Gauss Ring points. Both of these distributions are described in [10]. The visualization of all tested distributions of points together with its convex hull is in Fig. 3.


(a) Uniform points in circle (b) Gauss Ring points (c) Gauss points

(d) Halton points (e) Uniform points in square

Fig. 3. Distributions of points used for testing. The blue line represents the convex hull of the data set.

The main purpose of the proposed approach is the speed-up of the Weltzl's algorithm. The running time of algorithm for smallest enclosing circle depends on the distribution of points and of course on the number of input points. We performed 10^3 tests for each points distribution from Fig. 3 and for each different number of input points. The times were measured for the original Weltzl's algorithm and in the first phase also for the proposed approach without the

Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica,
Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

selection of four best candidates $\{C_1, C_2, C_3, C_4\}$. As the algorithm is randomized, the running time depends on the randomization and thus we computed the minimal and maximal running time for each configuration as well as the $33^{th}\%$ fastest and $33^{th}\%$ slowest time. The visualization of this four running times is described in Fig. 4. The resulting running times are visualized in Fig. 5 - Fig. 9.

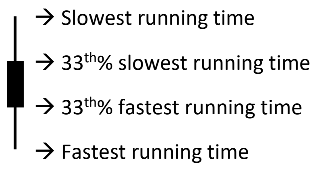
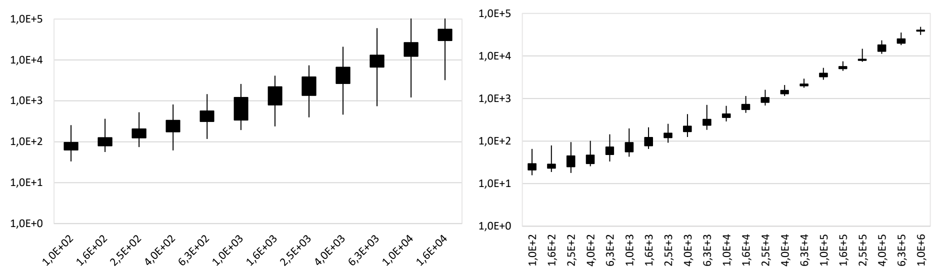


Fig. 4. Definition of symbol used in graphs with running times.



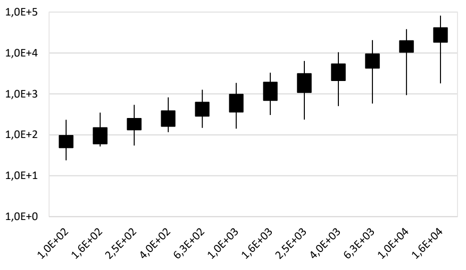
(a) Welzl's algorithm (b) Proposed approach

Fig. 5. Running times in $[\mu s]$ for uniform points in circle.

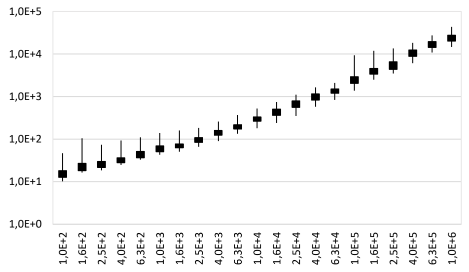
It can be seen that for the original Weltzl's algorithm is the difference between fastest and slowest running time for one distribution and one number of points almost every time at least $10\times$ different. This is a huge difference. Our proposed approach without the selection of four best candidates $\{C_1, C_2, C_3, C_4\}$ has the difference between slowest and fastest time mostly around $3.2\times$.

It can be also seen, that we measured the running times of Weltzl's algorithm for lower maximal number of input points. The reason for this is, that this was the maximal size of input data set, that could be processed due the the maximal depth of recursion.

In the second tests, we measured the running times with the same distributions and the same number of points for our proposed approach (with all steps of the algorithm, i.e. also with the selection of four best candidates $\{C_1, C_2, C_3, C_4\}$). The fastest times are the same as in Fig. 5b - Fig. 9b but the

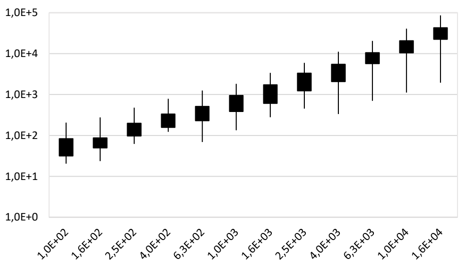


(a) Welzl's algorithm

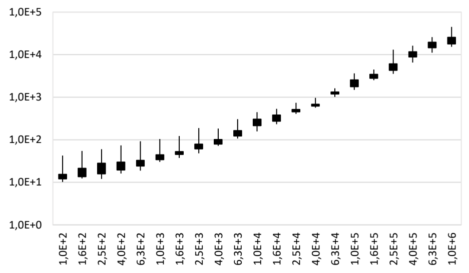


(b) Proposed approach

Fig. 6. Running times in [μs] for Gauss Ring points.

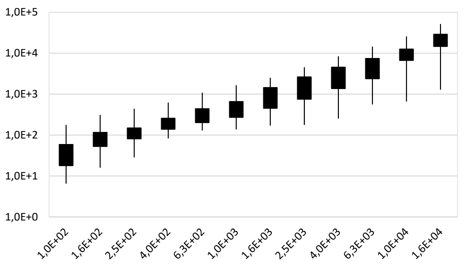


(a) Welzl's algorithm

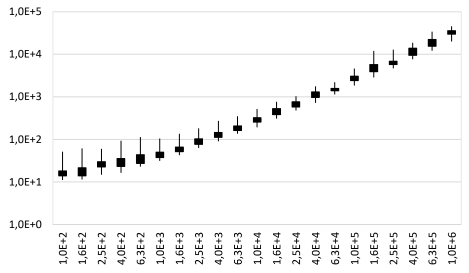


(b) Proposed approach

Fig. 7. Running times in [μs] for Gauss points.



(a) Welzl's algorithm

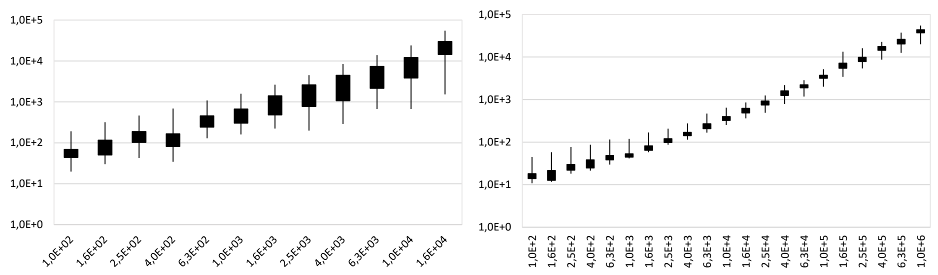


(b) Proposed approach

Fig. 8. Running times in [μs] for Halton points.

slowest times are lower. The slowest times are now about 1.9× slower than the fastest time. This is a great improvement from the previous 3.2× difference.

We also computed the speed-up of our algorithm to the Weltzl's algorithm. We used the average running times to compute the speed-up (see Fig. 10). It can be seen that with increasing number of input points, the speed-up increases



(a) Welzl's algorithm (b) Proposed approach

Fig. 9. Running times in [μs] for uniform points in square.

as well. This shows, that our proposed approach is not only faster, but it has also better time complexity than the original Weltzl's algorithm.

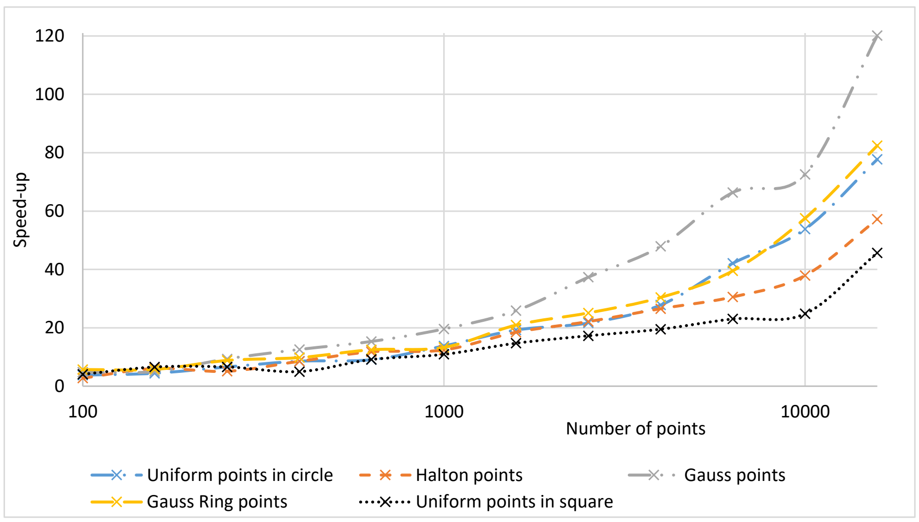


Fig. 10. Speed-up of average running times of the proposed method compared to the standard Weltzl's algorithm.

4 Conclusion

We proposed a simple and efficient speed-up of Weltzl's algorithm for computation of smallest enclosing circle of $2D$ points. The proposed approach is easy to

Smolik,M., Skala,V.: Efficient Speed-up of Smallest Enclosing Circle Algorithm, Informatica, Vol.33, No.1, pp.1-11, ISSN 0868-4952, IF=2.688, 2022
DOI: 10.15388/22-INFOR477

implement, reduces the randomness of the algorithm, and speed-ups the computation. The proposed speed-up also improves the computational complexity, which is beneficial for higher number of processed input points. Also the maximal number of points, that can be processed, was increased, as the required depth of recursion was dramatically decreased.

The proposed algorithm for computation of smallest enclosing circle of $2D$ points can be easily adapted for computation of the smallest enclosing sphere of $3D$ points.

Acknowledgments

The authors would like to thank their colleagues at the University of West Bohemia, Plzen, for their discussions and suggestions. The research was supported by the Ministry of Education, Youth, and Sport of Czech Republic - University spec. research - 1311, and by SGS 2019-016.

References

1. A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. In *Workshop on Algorithms and Data Structures*, pages 325–336. Springer, 1993.
2. A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Computational Geometry*, 4(3):119–136, 1994.
3. S. Gao and C. Wang. A new algorithm for the smallest enclosing circle. In *2018 8th International Conference on Management, Education and Information (MEICI 2018)*. Atlantis Press, 2018.
4. S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *European Symposium on Algorithms*, pages 278–288. Springer, 2003.
5. S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3):147–157, 2005.
6. N. Megiddo. Linear-time algorithms for linear programming in Rˆ3 and related problems. *SIAM journal on computing*, 12(4):759–776, 1983.
7. V. Skala. Fast oexpected (n) algorithm for finding exact maximum distance in e2 instead of o (n2) or o (n lgn). In *AIP Conference Proceedings*, volume 1558, pages 2496–2499. American Institute of Physics, 2013.
8. V. Skala and Z. Majdisova. Fast algorithm for finding maximum distance with space subdivision in e 2. In *International Conference on Image and Graphics*, pages 261–274. Springer, 2015.
9. V. Skala and M. Smolik. A point in non-convex polygon location problem using the polar space subdivision in e 2. In *International Conference on Image and Graphics*, pages 394–404. Springer, 2015.
10. V. Skala, M. Smolik, and Z. Majdisova. Reducing the number of points on the convex hull calculation using the polar space subdivision in e2. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 40–47. IEEE, 2016.

11. S. Skyum. A simple algorithm for computing the smallest enclosing circle. *Information Processing Letters*, 37(3):121–125, 1991.
12. E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New results and new trends in computer science*, pages 359–370. Springer, 1991.
13. S. Xu, R. M. Freund, and J. Sun. Solution methodologies for the smallest enclosing circle problem. *Computational Optimization and Applications*, 25(1-3):283–292, 2003.