INFORMATICA, 2022, Vol. 33, No. 3, 623–633

DOI: https://doi.org/10.15388/22-INFOR477

Efficient Speed-Up of the Smallest Enclosing Circle Algorithm

Michal SMOLIK*, Vaclav SKALA

Plzen, Faculty of Applied Sciences, University of West Bohemia, Czech Republic
e-mail: smolik@kiv.zcu.cz, skala@kiv.zcu.cz

Received: February 2021; accepted: February 2022

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 to Weltzl's algorithm. In the end are summarized 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 capable to process significantly larger data sets than the standard Weltzl's algorithm.

Key words: smallest enclosing circle, space subdivision, convex hull, speed-up, Weltzl's algorithm.

1. Introduction

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

- The maximal distance of all points to the center is minimal for the center of the smallest enclosing circle.
- Given any three points, we can uniquely define a circle, with at least two points of these points on the circumference.
- The smallest enclosing circle is unique for any set of $2D$ points.
- Given $N \geqslant 2$ points in the plane, the smallest circle that encloses them contains at least two of the points within its circumference.

The brutal force algorithm has the time complexity $O(N^4)$. The algorithm tests all possible circles, i.e. all combinations of two points $O(N^2)$ and all combinations of three points $O(N^4)$. To test if one circle is the smallest enclosing circle takes $O(N)$. This whole brutal force algorithm runs in $O(N^4)$ time.

*Corresponding author.

There are two best known algorithms, i.e. the Welzl's algorithm, which has expected $O_{expected}(N)$ running time (Welzl, 1991), and the Megiddo's algorithm, which has expected $O_{expected}(N)$ running time (Megiddo, 1983). The technique prune and search is used in Megiddo's algorithm (Megiddo, 1983) to reduce the problem size by removing approximately $N/16$ unnecessary points. The algorithm is rather complicated which resulted to a very high multiplicative constant. The reduction needs to solve twice the similar problem where the centre of the enclosing circle is located using the construction of bisectors of points. The randomized incremental construction is used in the Welzl's algorithm (Welzl, 1991). The algorithm iterates over all points. In case, the point is not inside the actual smallest enclosing circle, the point must be part of the new smallest enclosing circle in the next step. Using this basic knowledge the final minimal enclosing circle is recursively constructed.

The Skyum algorithm (Skyum, 1991) presents a simple iterative $O(N \log N)$ algorithm for computing the smallest enclosing circle. This is not optimal but its simplicity makes it a better alternative for medium-sized problems than both previous algorithms. The Gau algorithm (Gao and Wang, 2018) computes the minimal enclosing disk in an iterative manner and needs to define some distance parameter delta value that is used in the algorithm.

Other alternative algorithms (Har-Peled and Mazumdar, 2003, 2005) compute 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 the smallest enclosing circle of at least k points is described in Efrat *et al.* (1993, 1994), too. The paper (Xu *et al.*, 2003) summarizes four different approaches for the location of a minimal enclosing circle for a set of fixed circles.

One of the first sophisticated geometry algorithms developed with many variations of it is the convex hull algorithm. The convex hull is the minimal set of points, that form a boundary for all other points, i.e. all other points are located inside. There are numerous applications for convex hulls: collision avoidance, maximum distance using convex hull diameter for large data sets (Skala, 2013; Skala and Majdisova, 2015), hidden object determination, shape analysis, point inside polygon (Skala and Smolik, 2015).

In this paper, an adapted convex hull algorithm from Skala *et al.* (2016b), based on Skala (2013), is used to speed-up the computation of the smallest enclosing circle. This convex hull algorithm uses the space subdivision to achieve 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 convex hull algorithm is $O_{expected}(N)$ (Skala *et al.*, 2016b).

2. Proposed Approach

The smallest enclosing circle contains within its circumference only points from the convex hull. To speed-up the processing time, two step algorithm was developed. The first step is a removal of all points that cannot form the smallest enclosing circle. While the

second one is a computation of the smallest enclosing circle using the Weltzl's algorithm (Welzl, 1991) as it is fast and easy to implement.

In order to obtain a significant speed-up of the input points reduction, the convex hull construction needs to have a lower computational cost than the Weltzl's algorithm. The convex hull algorithm (Skala *et al.*, 2016b) uses space subdivision technique to speed-up the computation with $O_{expected}(N)$ time complexity.

The proposed algorithm for the smallest enclosing circle is summarized in Algorithm 1 and will be described in more detail in the following sub-sections.

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 (Skala et al., 2016b)
5:      random shuffle C
6:      {C1, C2} := find two points with max distance in C
7:                                  ▷ Max distance from Skala and Majdisova (2015)
8:      C3 := find point with max distance to center of C1 and C2
9:      C4 := find point with max distance to C3
10:     move [C4, C3, C2, C1] to the end of C
11:     return Welzl(C, {∅})
12: end
13:
14: input: Finite sets P and R of 2D points (‖R‖ ⩽ 3).
15: output: Smallest enclosing circle of P with R on the boundary.
16: procedure WELZL(P, R)          ▷ Welzl's algorithm without randomization
17:                                                  ▷ based on Welzl (1991)
18:     if P is empty or ‖R‖ = 3 then
19:         return smallest circle of R
20:     point p := last from P
21:     circle D := Welzl(P − {p}, R)
22:     if p is in D then
23:         return D
24:     return Welzl(P − {p}, R ∪ {p})
25: end
```

2.1. *Convex Hull Construction*

The convex hull algorithm (Skala *et al.*, 2016b) significantly speeds-up the construction of the convex hull by reducing the input points to only a few ones that are suspicious of

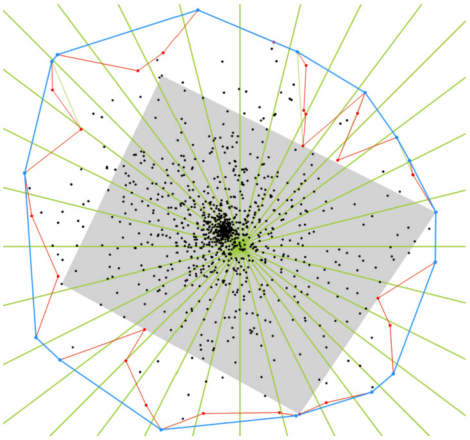


Fig. 1. Convex hull construction. Blue lines represent the convex hull.

forming the convex hull. The detailed algorithm is described in Skala *et al.* (2016b). We summarize only the important steps without details.

The first step is to find an estimation of the axis-aligned bounding box (AABB) if not known. This is done by searching only 10% of input points. A convex polygon is estimated from those points defining the estimated AABB. All points that are inside of this polygon cannot form the convex hull and thus are discarded (the gray part in Fig. 1).

The next step is to create a star shape division of the remaining points. In each cell, we determine points with the maximal distance from the centre, that can form the convex hull (there can be more points in each cell). All other points are discarded. The remaining points (connected with red lines in Fig. 1) are the result of the reduction. Now, the actual algorithm for convex hull construction using only the suspicious points is used.

2.2. *Smallest Enclosing Circle Computation*

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

One limitation of the Weltzl's algorithm is the recursion as it allows to process only limited number 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 by available RAM memory.

The original algorithm selects points totally in a random manner. However, if the first selected points form a circle, which is big enough to contain almost all the 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 the algorithm (Skala and Majdisova, 2015). 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 the points are randomly shuffled. In the case, when the smallest enclosing circle is defined only by two points, then these two points are exactly $\{C_1, C_2\}$ and the points $\{C_3, C_4\}$ are the same as $\{C_1, C_2\}$.

The input for the Weltzl's algorithm is a set of points P. In each step, the algorithm selects the 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 of point p. If the point p is also included in the returned circle, then it is the smallest enclosing circle for the whole set of points P.

Otherwise, if the point p lies outside the circle, it must lie on the boundary of the resulting circle. In the next step, 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. In case of 0 or 1 points, the circle is only none or one point. The smallest enclosing circle for 2 points is defined, that has its centre in the middle of the two points and radius half of the distance between the two points. The last case are 3 points, where the smallest enclosing circle is the circumcircle of the triangle described by the points.

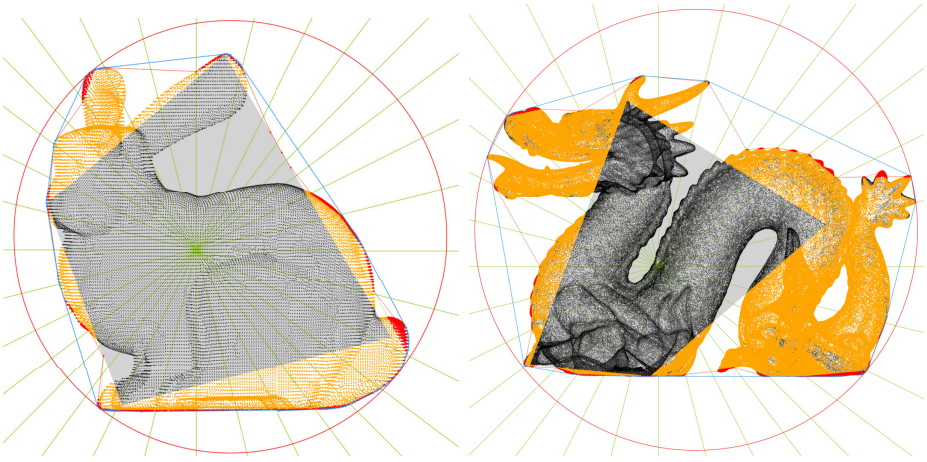
When R contains 3 points, the recursion terminates, as all points in P are already inside of the circle formed by R.

3. Experimental Results

In this chapter, we summarize the obtained results of the proposed algorithm for the smallest enclosing circle of points in $2D$. The proposed approach is capable 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 of the proposed algorithm. 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 the first 4 points that are processed at 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.

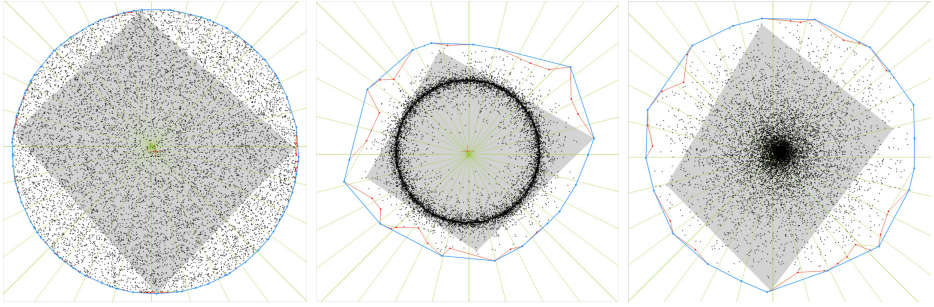
To test the proposed approach, we used several data sets with different types of point distribution in $2D$. Some of the distributions are well known, randomly distributed uniform points inside a unit square or inside a unit circle and points with Gaussian distribution. Then we also used Halton points and Gauss Ring points. The last two distributions are described in Skala *et al.* (2016b). The visualization of all tested distributions of points together with their convex hulls is presented in Fig. 3.

The main purpose of the proposed approach is the speed-up of the Weltzl's algorithm. The running time of the algorithm for the 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 running time of the algorithm for one input data set was measured as running time of $10^2 - 10^3$ repetitions divided by the number of repetitions. The random generator

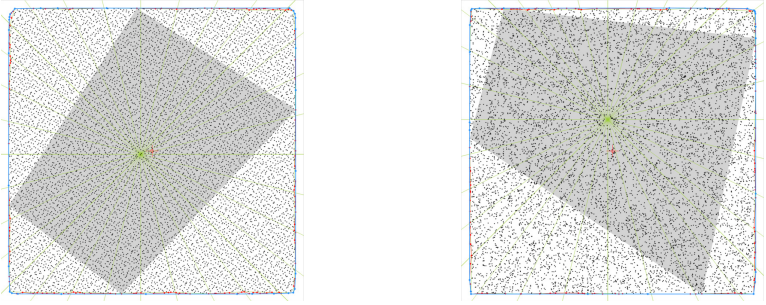


(a) Stanford bunny: 35 947 points (b) Stanford dragon: 437 645 points

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/).



(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, i.e. the initial points for Weltzl's algorithm.

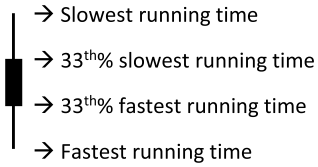
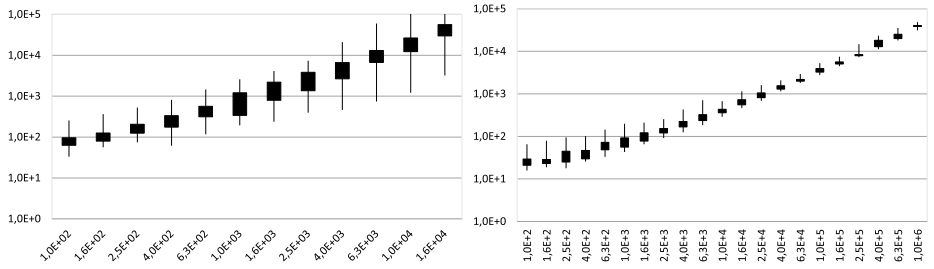
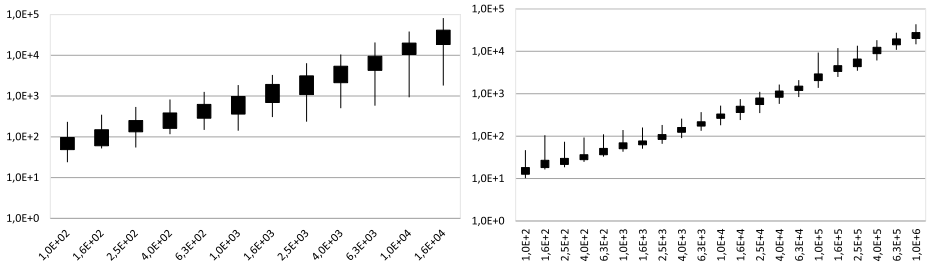


Fig. 4. Definition of symbol used in graphs with running times.



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

Fig. 5. Running times in [μs] (vertical axis) for different number of uniform points in circle (horizontal axis).

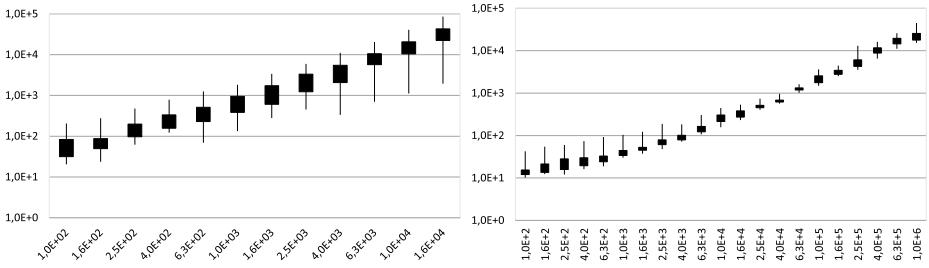


(a) Welzl's algorithm (b) Proposed approach

Fig. 6. Running times in [μs] (vertical axis) for different number of Gauss Ring points (horizontal axis).

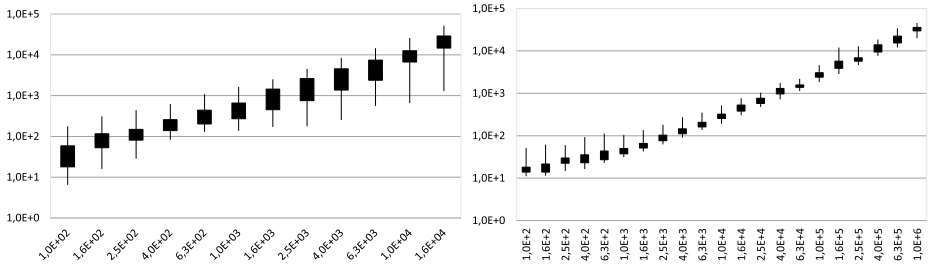
was initialized with the same seed number for each of those repetitions. The times were measured for the original Weltzl's algorithm and in the first phase also for the proposed approach without the selection of four best candidates $\{C_1, C_2, C_3, C_4\}$. As the algorithm is randomized, the running time depends on the randomization and thus we computed the minimal and maximal running time for each configuration as well as the 33th% fastest and 33th% slowest time. The visualization of these four running times is described in Fig. 4. The resulting running times are visualized in Figs. 5–9. It should be noted, that there is a logarithmic scaling on both axes. The horizontal axis on graphs with proposed approach presents higher maximal number of input points compared to graphs with standard Weltzl's algorithm.

It can be seen that for the original Weltzl's algorithm there is a difference between the fastest and the slowest running time for one distribution and one number of points almost



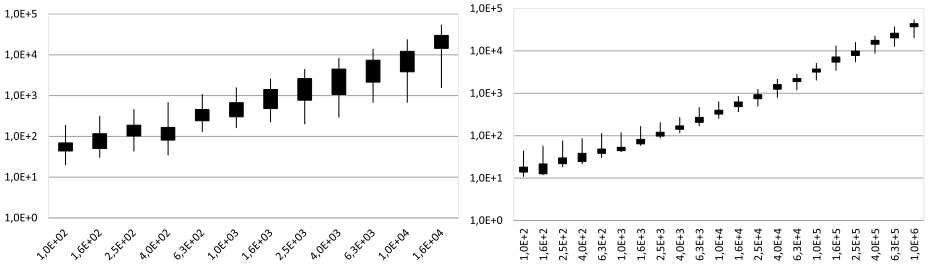
(a) Welzl's algorithm (b) Proposed approach

Fig. 7. Running times in [μs] (vertical axis) for different number of Gauss points (horizontal axis).



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

Fig. 8. Running times in [μs] (vertical axis) for different number of Halton points (horizontal axis).



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

Fig. 9. Running times in [μs] (vertical axis) for different number of uniform points in square (horizontal axis).

every time at least 10 times different. This is a huge difference in the algorithm behaviour. The 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, i.e. much better ratio resulting to stability of algorithm behaviour.

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

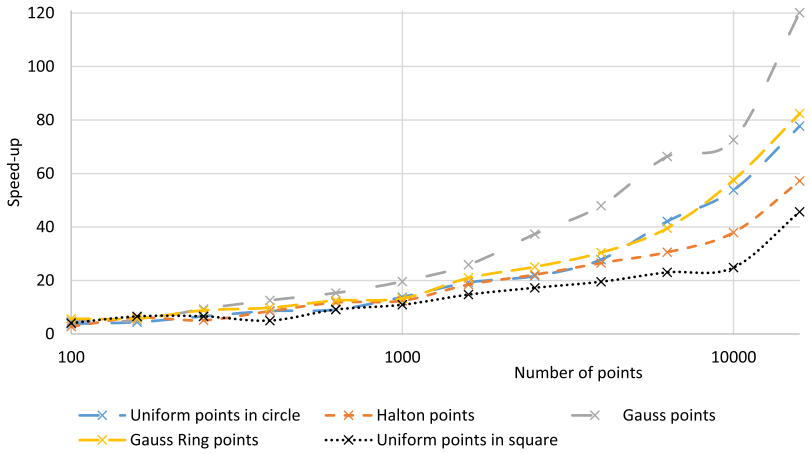


Fig. 10. Speed-up of average running times of the proposed method compared to the standard Weltzl's algorithm.

In the second set of 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 Figs. 5b–9b but the slowest times are lower. The slowest times are now only about 1.9 times slower than the fastest times. This is a great improvement from the previous 3.2 times 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 an increasing number of input points, the speed-up increases as well. This proves, that our proposed approach is not only faster, but it has also better time complexity than the original Weltzl's algorithm.

4. Conclusion

We proposed a simple and efficient speed-up of the Weltzl's algorithm for computation of the smallest enclosing circle of $2D$ points. The proposed approach is easy to implement. It reduces the randomness of the algorithm and speeds-up the computation, too. The proposed algorithm also improves the computational complexity, which is beneficial for a higher number of processed input points. Also, the maximal number of points, that can be processed, is significantly increased, as the required depth of recursion is dramatically decreased.

The proposed algorithm for computation of the smallest enclosing circle of $2D$ points can be adapted for the computation of the smallest enclosing sphere of $3D$ points using convex hull computation in $3D$ (Skala *et al.*, 2016a). Another possible improvement could be usage of the smallest enclosing circle/sphere of the points forming axis-aligned bounding box as the initial step because all points inside cannot form the smallest enclosing circle/sphere.

Acknowledgements

The authors would like to thank their colleagues at the University of West Bohemia, Plzen, for their discussions and suggestions.

Funding

The research was supported by the University of West Bohemia – Institutional research support No. 1311, and by SGS 2019-016.

References

Efrat, A., Sharir, M., Ziv, A. (1993). Computing the smallest k-enclosing circle and related problems. In: *Workshop on Algorithms and Data Structures*. Springer, pp. 325–336.

Efrat, A., Sharir, M., Ziv, A. (1994). Computing the smallest k-enclosing circle and related problems. *Computational Geometry*, 4(3), 119–136.

Gao, S., Wang, C. (2018). A new algorithm for the smallest enclosing circle. In: *2018 8th International Conference on Management, Education and Information (MEICI 2018)*, Atlantis Press.

Har-Peled, S., Mazumdar, S. (2003). Fast algorithms for computing the smallest k-enclosing disc. In: *European Symposium on Algorithms*. Springer, pp. 278–288.

Har-Peled, S., Mazumdar, S. (2005). Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3), 147–157.

Megiddo, N. (1983). Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4), 759–776.

Skala, V. (2013). Fast $O_{expected}(N)$ algorithm for finding exact maximum distance in E2 instead of $O(N^2)$ or $O(N \log N)$. In: *AIP Conference Proceedings*, Vol. 1558. American Institute of Physics, pp. 2496–2499.

Skala, V., Majdisova, Z. (2015). Fast algorithm for finding maximum distance with space subdivision in E2. In: *International Conference on Image and Graphics*. Springer, pp. 261–274.

Skala, V., Smolik, M. (2015). A point in non-convex polygon location problem using the polar space subdivision in E2. In: *International Conference on Image and Graphics*. Springer, pp. 394–404.

Skala, V., Majdisova, Z., Smolik, M. (2016a). Space subdivision to speed-up convex hull construction in E3. *Advances in Engineering Software*, 91, 12–22.

Skala, V., Smolik, M., Majdisova, Z. (2016b). 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)*. IEEE, pp. 40–47.

Skyum, S. (1991). A simple algorithm for computing the smallest enclosing circle. *Information Processing Letters*, 37(3), 121–125.

Welzl, E. (1991). Smallest enclosing disks (balls and ellipsoids). In: Maurer, H. (Ed.), *New Results and New Trends in Computer Science*, Lecture Notes in Computer Science, Vol. 555. Springer, Berlin, Heidelberg.

Xu, S., Freund, R.M., Sun, J. (2003). Solution methodologies for the smallest enclosing circle problem. *Computational Optimization and Applications*, 25(1–3), 283–292.

M. Smolik received the PhD degree in software engineering from the University of West Bohemia, Czech Republic, in 2020. He currently holds the position of researcher at the Department of Computer Graphics at the University of West Bohemia. His research areas are meshless methods for approximation and interpolation, specifically Radial basis functions. He works in the area of vector fields and develops basic algorithms connected with computer graphics.

V. Skala is a full professor of computer science at the University of West Bohemia, Plzen, Czech Republic. He received his ING. (equivalent of MSc) degree in 1975 from the Institute of Technology in Plzen and CSc. (equivalent of PhD) degree from the Czech Technical University in Prague, in 1981. In 1996, he became a full professor in computer science. His current research interests are computer graphics and visualization, applied mathematics, especially geometrical algebra, algorithms, and data structures.