Diameter and Convex Hull of Points Using Space Subdivision in E^2 and E^3

Vaclav Skala(✉)

Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, 301 00 Pilsen, CZ, Czech Republic
skala@kiv.zcu.cz
http://www.VaclavSkala.eu

Abstract. Convex hull of points and its diameter computation is a frequent task in many engineering problems, However, in engineering solutions, the asymptotic computational complexity is less important than the computational complexity for the expected data size to be processed. This contribution describes "an engineering solution" of the convex hulls and their diameter computation using space-subdivision and data-reduction approaches. This approach proved a significant speed-up of computation with simplicity of implementation. Surprisingly, the experiments proved, that in the case of the space subdivision the reduction of points is so efficient, that the "brute force" algorithms for the convex hull and its diameter computation of the remaining points have nearly no influence to the time of computation.

1 Introduction

The Convex Hull (CH) and the Convex Hull Diameter (CH-D) algorithms are applicable in many areas. Those algorithms are deeply analyzed in the *computational geometry* for asymptotic behavior, i.e. $N \mapsto \infty$. However, it is not quite what today's applications need. In engineering problems, there are two main aspects, which have to be respected:

- the number of input elements
- the dimension of the data

Geometrically oriented algorithms usually process two or three dimensional data and the number of elements is usually not higher than 10^8 in the E^2 case ($10^4 \times 10^4$), resp. 10^{12} in the E^3 case ($10^4 \times 10^4 \times 10^4$) in the real applications. It means, that the engineering solutions have to respect several factors:

- the limited size of data sets to be processed, i.e. asymptotically better algorithm is not necessarily the best one for the intended application scope,
- numerical stability and robustness, i.e. the implementation has to respect a limited precision of computation resulting from the IEEE 754-2019 floating-point representation standard, including the fact that the *Quadruple* and *Octuple* precisions are not supported on today's processors,

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 286–295, 2020.
https://doi.org/10.1007/978-3-030-58799-4_21

- memory management and data transfer via data-bus from memory to CPU/GPU and vice-versa; also the influence of caching cannot be ignored,
- simplicity and efficiency of algorithms, i.e. too complicated algorithm will not be probably used, especially if its behavior is not "stable" and predicable,

This contribution describes the principle of two basic efficient algorithms, recently designed, implemented and verified, which are based on efficient reduction of points using the space subdivision and significant reduction of points remaining for the final processing, i.e.:

- Convex Hull Diameter (CH-D) of points in E^2 - the algorithms are usually based on the Convex Hull (CH) algorithms, which are more or less based on sophisticated algorithms. However, they are not easy to implement especially in the limited precision of computation for a higher number of points.
- Convex Hull (CH) of points in E^2 - algorithm using a deterministic heuristic approach with space subdivision.

In the following, simple algorithms for finding CH-D and CH of the given points in the E^2 case are described, which are easy to implement and very efficient. It should be noted, that those algorithms can be easily extended to the E^3 case.

2 Finding a Diameter of a Convex Hull

Finding the maximum distance of points in the E^n case is usually done by a simple algorithm that has $O(N^2)$, where N is the number of the given points.

Algorithm 1: Maximum distance of two points in E^2 or E^3

```
Result: Maximum_Distance
# N - Number of points x_i, i = 1, ..., N ;
d := 0;
i := 0;
while i <= N − 1 do
    j:= i+1;
    while j <= N do
        d0 := ‖x_i − x_j‖ ; # Euclidean distance
        if d0 < d then
            d := d0 ;
        end
    end
end
Maximum_Distance := d;
```

However, such an algorithm is quite inefficient due to its $O(N^2)$ computational complexity and also of the $\|.\|$ computation, where $\sqrt{(.)}$ is used. Algorithm 2 presents simple modification using $\|.\|^2$ for the distance comparison.

Algorithm 2: Modified Maximum distance of two points in E^2 or E^3

```
Result: Maximum_Distance
# N - Number of points x_i, i = 1, ..., N ;
d := 0;
i := 0;
while i <= N − 1 do
    j:= i+1;
    while j <= N do
        d0 := ‖x_i − x_j‖^2 ; # Euclidean distance
        if d0 < d then
            d := d0 ;
        end
    end
end
Maximum_Distance := sqrt(d);
```

Such a very simple modification of the algorithm Algorithm1 has a significant influence on computational efficiency. However, for larger values of N, the time of computing is still very high. Finding the maximum distance of two points, generally in the E^n case, is equivalent to the convex hull diameter problem, which has deeply analyzed in computational geometry for a long time. There are

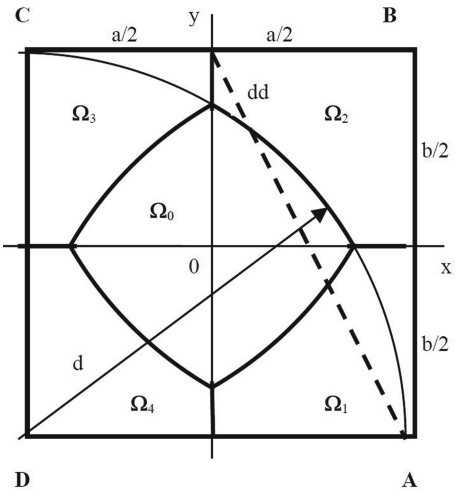

Fig. 1. Splitting data into areas by subdivision

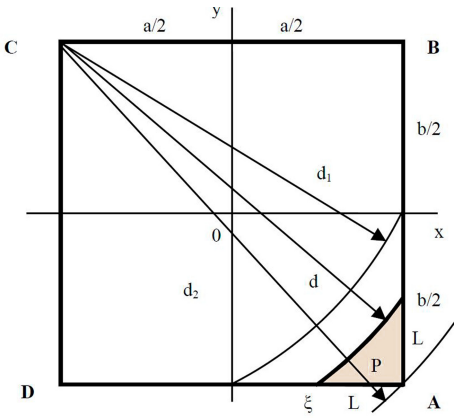

Fig. 2. Timing of the Convex Hull Diameter algorithms with space subdivision

several Convex Hull Diameter (CH-D) algorithms based on Convex Hull (CH) algorithms developed recently, e.g. Toussaint [26]. Some algorithms were deterministic, some of those based on a stochastic approach, e.g. Xue [27]. Efficient CH algorithms get more and more complex as far as implementation aspects are concerned. Recently, a simple algorithm based on the space subdivision was developed Skala [19], Fig. 1. It is primarily based on finding extreme points of the Axis Aligned Bounding Box (AABB) and the closest points to the AABB corners, which forms the first estimation of the convex hull, Fig. 1, where d is its diameter. Then the points are split to five areas $\Omega_i, i = 0, ..., 4$. The points from the central area Ω_0 cannot have any influence to the final convex hull [19,23] and can be removed automatically.

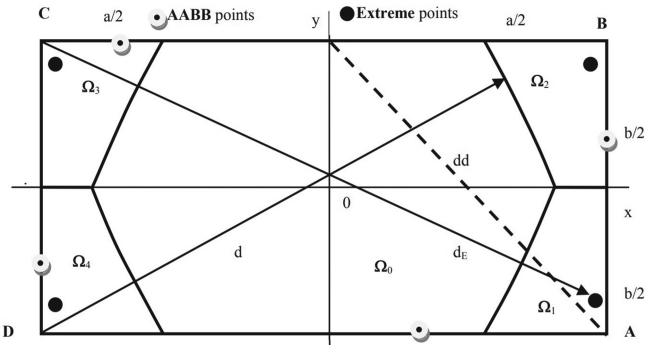

Fig. 3. Splitting data into areas by subdivision

The CH-D algorithm is based on the idea of points reduction first, followed by actual CH algorithm use for finding the diameter of the reduced data-set,

i.e. computation of the CH-D value. The influence of the simple reduction step, which is of $O(N)$ complexity, was overwhelming the expectations see Fig. 4 and Fig. 5.

$$\xi = \frac{\Omega}{\Omega_1 + \Omega_2 + \Omega_3 + \Omega_4} \tag{1}$$

where Ω is the area of the AABB box.

The expected reduction ratio Eq. 1 is given by the area bounded by d_1 and d_2, where d_1 is the worst case of the estimated diameter form the AABB box and when the data are in the squared domain, Fig. 2. If the data are in the rectangular area, see Fig. 3, then the Ω_i areas are getting significantly smaller.

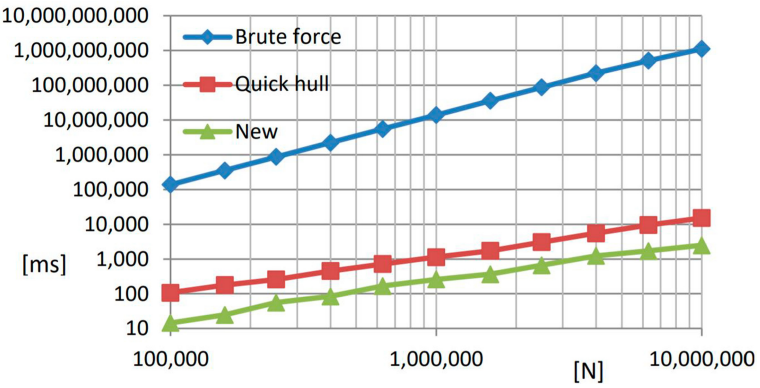

Fig. 4. Timing of the Convex Hull Diameter algorithms with space subdivision

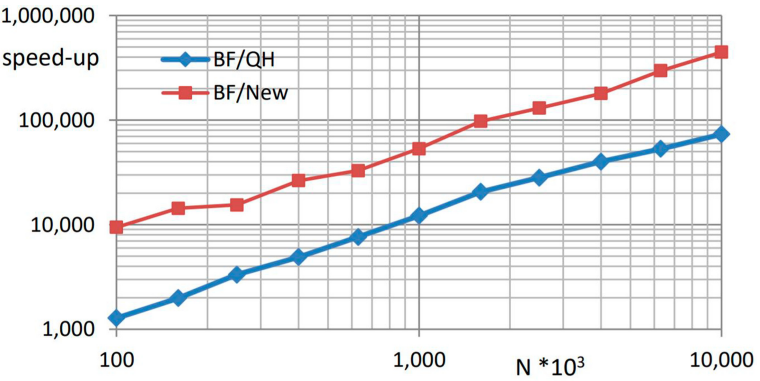

Fig. 5. Speed-up of the Convex Hull Diameter algorithm with space subdivision BF/QH - BruteForce/ QuickHull, BF/New - BruteForce/proposed

As the final data sets after this reduction step were extremely small, simple CH algorithms were used, practically without any influence to time complexity. The Halton's [11] point generation was used in the experimental evaluation and detailed description and experimental results were described in Skala [19,20,22]. The timing and speed-up are presented in Fig. 4 and Fig. 5.

It can be seen that the space subdivision significantly speed-up the CH-D algorithm. Surprisingly, the presented CH-D algorithm handle close circle generated points efficiently as well. If the AABB is not a square, the presented CH-D algorithm gets even more efficient Skala [19,22]. The extension to the E^3 case is simple, straight forward and easy to implement Skala [21].

3 Convex Hull with Space Subdivision

Several deterministic Convex Hull (CH) algorithms have been described. Table 1 presents some well-known algorithms and their asymptotic computational complexity. The estimation of the lower bound complexity was given by Yao [28]. Some of those were incremental, e.g. Kallay [13], Also a stochastic approach has been described by recently, e.g. Xue [27].

Table 1. Table of Convex Hull algorithms and their complexities

Algorithm	Complexity	Reference
Gift Wrapping	nh	Chand and Kapur, 1970 [5]
Graham Scan	$n\ log(n)$	Graham, 1972 [9]
Jarvis March	nh	Jarvis, 1973 [12]
QuickHull	nh	Barber, 1996 [2], Eddy, 1977 [8] Bykat, 1978 [3], Devai, 1979 [7]
Divide & Conquer	$n\ log(n)$	Preparata and Hong, 1977 [18]
Monotone Chain	$n\ log(n)$	Andrew, 1979 [1]
Incremental	$n\ log(n)$	Kallay, 1984 [13]
Marriage before Conquest	$n\ log(h)$	Kirkpatrick & Seidel, 1986 [14]
Chan's algorithm	$n\ log(h)$	Liu and Chen, 2007 [15]

Recently, an interesting QuickhullDisk algorithm, which is based on disks was published by Nguen [17]. Parallel algorithms output-sensitive were described by Chan [4], Gupta [10]. Manual comparison of some algorithms via multimedia exposition can be found at Loffler [16]. In engineering applications, usually a two-dimensional or three-dimensional CH algorithms are required and the number of points to be processed is up to 10^8. Therefore, a simple Smart Convex Hull (S-CH) algorithm with space subdivision was developed for the E^2 case [19]. The modification to the E^3 case was described in Skala [21]. Parallel algorithms

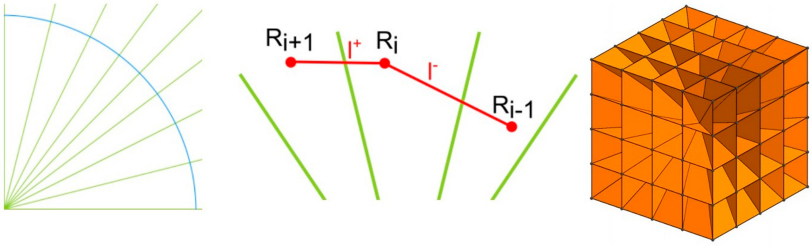

Fig. 6. The S-CH algorithm space subdivision in E^2 and E^3 taken from [21,23]

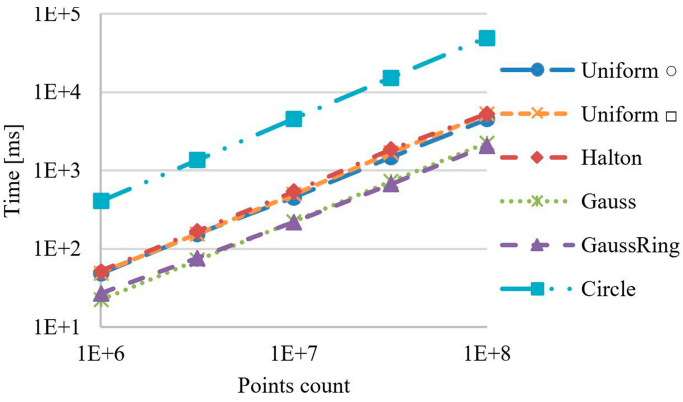

Fig. 7. Timing of the S-CH algorithm for a different distribution of points [20]

were described as well, e.g. by Sugihara[25], Gupta [10], a modification for GPU use by Stein [24].

The S-CH algorithm in the E^2 case is based on the polar space subdivision, Fig. 6, which replaces the orthogonal space subdivision in the CH-D algorithm described above. Detailed algorithm description and experimental results can be found in Skala [20]. The AABB border is split uniformly, so the angles in polar space splitting are different. Figure 7 presents the behavior of the S-CH algorithm for different data distribution including uniform on a circle.

Figure 8 presents the speed-up of the S-CH algorithm using the Graham Scan algorithm after the reduction of points, i.e. in the final step, with the selected convex hull algorithms Fig. 9.

The S-CH algorithm stores maximum distance from the origin in the angular segment and related two segments $R_{i+1}R_i$ and R_iR_{i-1} are updated so they form the approximate convex hull. Each angular segment may contain several points that form the final convex hull. The number of angular segments partially influences the time of computation, see Skala [23].

.

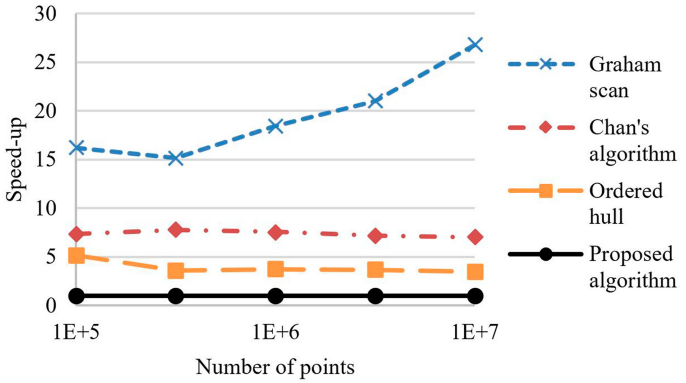

Fig. 8. Speed-up of the S-CH algorithm with the uniform distribution of points [20]

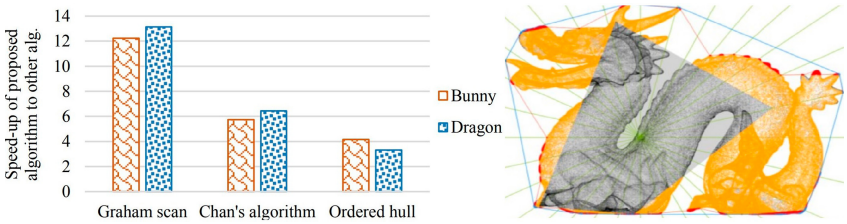

Fig. 9. Speed-up of the S-CH algorithm with the uniform distribution of points [20]

4 Conclusions

This paper presents an "engineering approach" to the Convex Hull and Convex Hull Diameter computations based on the space subdivision techniques designed for processing up to 10^8 points in the E^2 case. The algorithms are easily extensible to the E^3 case. Detailed results of experiments are given in detail in related papers, namely [20,21].

The presented approach is also used in teaching to show the influence of space subdivision to computational complexity. The presented algorithms are expected to be modified for GPU use in the future.

Acknowledgments. The author would like to thank colleagues and students at the University of West Bohemia, Pilsen, for their discussions and suggestions, especially to Zuzana Majdisova and Michal Smolik for recent implementations and experiments made, and to anonymous reviewers for their valuable comments and hints provided.

References

1. Andrew, A.: Another efficient algorithm for convex hulls in two dimensions. Inf. Process. Lett. **9**(5), 216–219 (1979)

2. Bradford, B., Dobkin, D., Dobkin, H., Huhdanpaa, D.P.: The quickhull algorithm for convex hulls. ACM Trans. Mathe. Softw. **22**(4), 469–483 (1996)
3. Bykat, A.: Convex hull of a finite set of points in two dimensions. Inf. Process. Lett. **7**(6), 296–298 (1978)
4. Chan, T.: Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete Comput. Geom. **16**(4), 361–368 (1996)
5. Chand, D., Kapur, S.: An algorithm for convex polytopes. J. ACM **17**(1), 78–86 (1970)
6. Dobkin, D., Snyder, L.: On a general method for maximizing and minimizing among certain geometric problems. In: FOCS-IEEE Symposium on Foundations of Computer Science Proceedings, pp. 9–17 (1979)
7. Dévai, F., Szendrényi, T.: Comments on convex hull of a finite set of points in two dimensions. Inf. Process. Lett. **9**(3), 141–142 (1979)
8. Eddy, W.: Algorithm 523: convex, a new convex hull algorithm for planar sets. ACM Trans. Math. Softw. (TOMS) **3**(4), 398–403 (1997)
9. Graham, R.: An efficient algorithm for determining the convex hull of a finite planar set. Inf. Process. Lett. **1**(4), 132–133 (1972)
10. Gupta, N., Sen, S.: Faster output-sensitive parallel algorithms for 3D convex hulls and vector maxima. J. Para. Distrib. Comput. **63**(4), 488–500 (2003)
11. Halton, J.: Algorithm 247: radical-inverse quasi-random point sequence. CACM **7**(12), 701–702 (1964)
12. Jarvis, R.: On the identification of the convex hull of a finite set of points in the plane. Inf. Process. Lett. **2**(1), 18–21 (1973)
13. Kallay, M.: The complexity of incremental convex hull algorithms in R^d. Inf. Process. Lett. **19**(4), 197 (1984)
14. Kirkpatrick, D., Seidel, R.: The ultimate planar convex hull algorithm? SIAM J. Comput. **15**(1), 287–299 (1986)
15. Liu, G., Chen, C.: A new algorithm for computing the convex hull of a planar point set. J. Zhejiang Univ. Sci. A **8**(8), 1210–1217 (2007)
16. Löffler, M.: A manual comparison of convex hull algorithms (multimedia exposition). In: 35th International Symposium on Computational Geometry (SoCG 2019), volume 129 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, pp. 65:1–65:2. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
17. Nguyen, K., Song, C., Ryu, J., Thanh, P.A., Hoang, N.-D., Kim, D.-S.: Quickhulldisk: a faster convex hull algorithm for disks. Appl. Math. Comput. **363**, 124626 (2019)
18. Preparata, F., Hong, S.: Convex hulls of finite sets of points in two and three dimensions. CACM **20**(2), 87–93 (1997)
19. Skala, V.: Fast $O_{\text{expected}}(N)$ algorithm for finding exact maximum distance in E^2 instead of $O(N^2)$ or $O(NlgN)$. In: ICNAAM 2013, AIP Proceedings, USA, vol. 1558, pp. 2496–2499. AIP Publishing (2013)
20. Skala, V., Majdisova, Z.: Fast algorithm for finding maximum distance with space subdivision in E^2. In: Zhang, Y.-J. (ed.) ICIG 2015. LNCS, vol. 9218, pp. 261–274. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21963-9_24
21. Skala, V., Majdisova, Z., Smolik, M.: Space subdivision to speed-up convex hull construction in E^3. Adv. Softw. Eng. **91**(C), 12–22 (2016)
22. Skala, V., Smolik, M.: Simple and fast $Oexp(N)$ algorithm for finding an exact maximum distance in E^2 instead of $O(N^2)$ or $O(NlgN)$. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11619, pp. 367–380. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24289-3_27

23. Skala, V., Smolik, M., Majdisova, Z.: Reducing the number of points on the convex hull calculation using the polar space subdivision in E^2. In: 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI 2016), pp. 40–47. IEEE (2016)
24. Stein, A., Geva, E., El-Sana, J.: Cudahull: fast parallel 3D convex hull on the GPU. Comput. Graph. **36**(4), 265–271 (2012)
25. Sugihara, K.: Robust gift wrapping for the three-dimensional convex hull. J. Comput. Syst. Sci. **49**(2), 391–407 (1994)
26. Toussaint, J., McAlear, G.T.: A simple o(n log n) algorithm for finding the maximum distance between two finite planar sets. Pattern Recogn. Lett. **1**(1), 21–24 (1982)
27. Xue, J., Li, Y., Janardan, R.: On the expected diameter, width, and complexity of a stochastic convex hull. Comput. Geom. **82**, 16–31 (2019)
28. Yao, A., Andrew, C.: A lower bound to finding convex hulls. J. ACM **28**(4), 780–787 (1981)