A New Highly Efficient Preprocessing Algorithm for Convex Hull, Maximum Distance and Minimal Bounding Circle in E^2: Efficiency Analysis ⋆

Vaclav Skala[0000−0001−8886−4281]

University of West Bohemia, Faculty of Applied Sciences
Dept. of Computer Science and Engineering
Pilsen, CZ 301 00, Czech Republic
skala@kiv.zcu.cz

Abstract. This contribution describes an efficient and simple preprocessing algorithm for finding a convex hull, maximum distance of points or convex hull diameter, and the smallest enclosing circle in E^2. The proposed algorithm is convenient for large data sets with unknown intervals and ranges of the data sets. It is based on efficient preprocessing, which significantly reduces points used in final processing by standard available algorithms.

Keywords: Preprocessing · maximum distance · smallest enclosing circle · smallest enclosing ball · algorithm complexity · preprocessing · convex hull · convex hull diameter.

1 Introduction

Many sophisticated algorithms are solving geometrical or computational problems, mostly evaluated according to their computational (asymptotic) complexity expecting the number of processed elements $N \mapsto \infty$.

Algorithms like maximum distance of points, i.e. convex hull diameter, convex hull, and minimal enclosing circle in E^2 are typical examples with known computational complexities with many modifications claiming better asymptotic computational complexity and faster run-time. Usually, a small attention is given to possible preprocessing strategies, which can significantly improve the total run-time and memory needed in some cases.

Let's consider an elementary problem: Find the maximum distance of two points in E^2 for the given a set Ω of points $\mathbf{x}_i$, $i = 1, \ldots, N$ and N is "reasonably" high. There are several strategies:

1. Simple algorithm based on mutual finding $d_{max} = \max_{i,j,\ \&\ i<j} \|\mathbf{x}_i - \mathbf{x}_j\|_2$

   It leads to $O(N^2)$ the computational complexity (the algorithm requires $N(N-1)/2$ steps), which is prohibitive even for a small N.

⋆ This work was supported by the MEYS CZ, Institutional Support for LCDRO.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7

2. Convex Hull (CH) computation, e.g. using the Kirkpatrick–Seidel algorithm [1] with $O(N \log h)$, followed by finding a maximum distance of the remaining $h \ll N$ convex hull points using the algorithm with computational complexity $O(h^2)$. It should be noted, that of the given set Ω of unordered points form a circle, the number of processed points is $h = N$.

Using the Convex Hull algorithm, can be also used to accelerate the minimum enclosing circle algorithm [2,3,9,11,12,13,14]. A simple and efficient algorithm for finding the minimum distance of points was introduced in [5,6,7,8], see Fig.2.

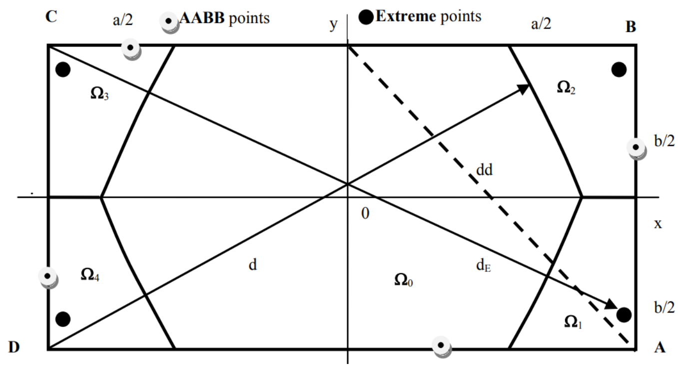


Fig. 1: Data domain subdivision; courtesy [10].

The algorithm uses a preprocessing with computational complexity $O(N)$ based on simple steps:

1. Find the Axis Aligned Bounding Box (AABB) of points in the given data set Ω.
2. Find the maximum mutual distance $d = \max\{dist(AC), dist(BD)\}$.
3. Split points into data subsets $\Omega_0, \ldots, \Omega_4$, see Fig.1, subset are defined by arcs given by the radius d and by the corners of the AABB.
   Points inside the subsets Ω_0 can be removed from the further processing directly.
4. Find the maximum mutual distance of points in pairs of subsets: (Ω_1, Ω_3), (Ω_2, Ω_4), (Ω_1, Ω_2), (Ω_2, Ω_4), (Ω_3, Ω_4), (Ω_4, Ω_1).

In the case of the uniform point's distribution, this approach leads to a maximum distance algorithm with computational complexity $O_{expected}(N)$, see Fig.2. Detailed descriptions can be found in [5,10].

Several modifications of the that based on orthogonal and polar space subdivision [6,9], extension used in 3D convex hull algorithm [8], etc. However, by a deeper analysis of the preprocessing step, additional significant improvements can be made.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7

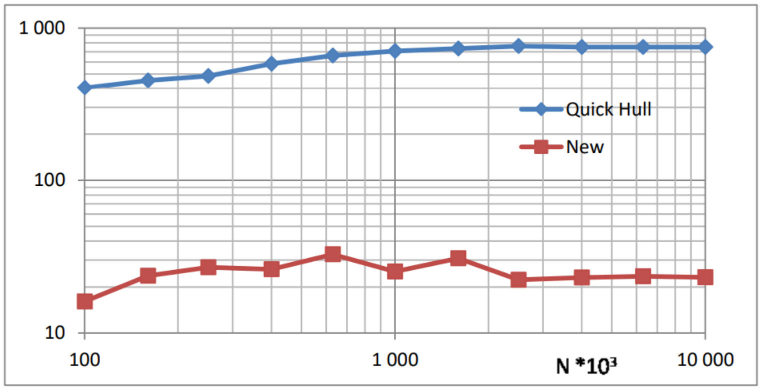
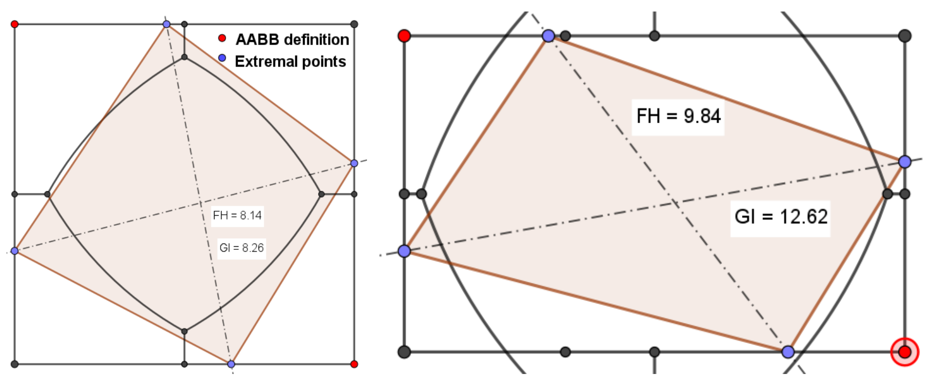


Fig. 2: Maximum distance speed-up of the Quick Hull and the Quick Hull with the original preprocessing; courtesy [10].

2 Proposed preprocessing algorithm

The original preprocessing for the reduction of points needs to find a min-max box (AABB), which requires $O(N)$ computation. Despite the liner complexity, this step is time-consuming if large data sets are to be processed, or a limited storage of incoming data is available, e.g. in the stream data processing.



(a) Squared area (b) Rectangular area

Fig. 3: Squared and rectangular areas and the selection function influence

2.1 Basic idea

Let us consider situations in Fig.3. In the case of the square data domain, Fig.3a, the tests based on a circular segment containment or a half-space test seem to

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7

be more or less equivalent. In the case of the rectangular AABB, Fig.3b, the circular segment test is more efficient. However, the areas Ω_i, $i = 1, \ldots, 4$ are too large.

It can be seen that the area of points which can be directly excluded is significantly larger. If the AABBox is known, the closest points to the AABBox corners can be found, see Fig.4, with the computational complexity $O(N)$.

If the AABBox is known, the nearest points to the AABBox vertices can be found with computational complexity $O(N)$. Then the area Ω_0 is defined by a convex polygon and all points inside to the area Ω_0 can be removed from further processing. In the following step, the remaining points will be split into the other areas Ω_i. However, this step requires additional large memory allocation.

However, the removal steps of finding AABBox and finding the closest points to the AABBox vertices lead to a more efficient algorithm.

2.2 Increasing efficiency

Let us assume, that from a small sample of points, the AABBox and the closest points to the AABBox vertices are obtained. Then the extreme points on the AABBox edges form separating half-planes defined by points A, C and B, D, see Fig.4.

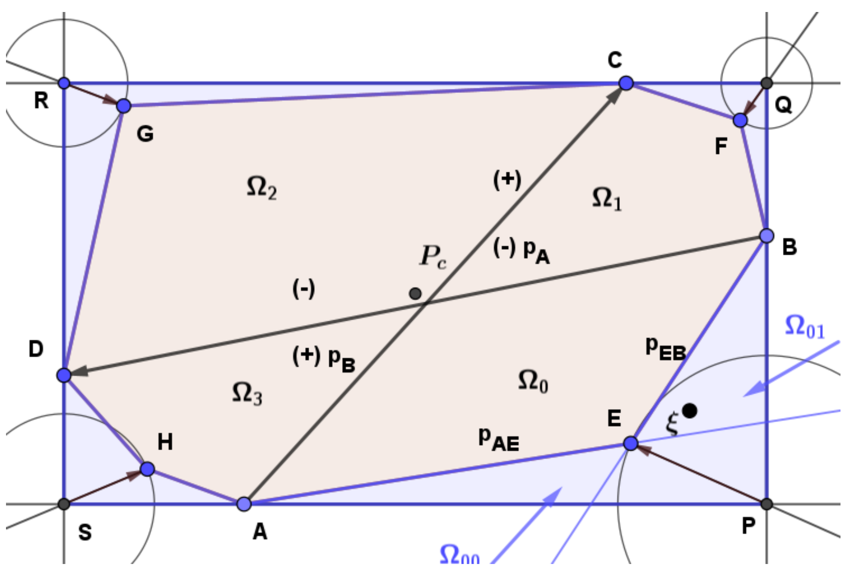


Fig. 4: Spliting the dataset to subsets Ω_i

It means, that for any point determining to which set of points it belongs to, is computationally simple. Even more, each basic area is split to areas Ω_i, Ω_{i0} and Ω_{i1} which are formed by the points E, F, G, H, i.e. the points closest the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7

AABBox vertices found so far. The point-in-area test is computationally simple as only tests for two half-planes given by points A, E and E, B are needed in the case of Ω_i, Ω_{i0} and Ω_{i1}, similarly for other areas.

Then, for all points $\boldsymbol{\xi}$ in the given data set Ω the following steps are made.

1. will fall into the convex hull of those points or one of the side areas outside; the point is excluded from further processing, or
2. change of the position of some points of the convex hull, i.e. new closest point to an AABBox vertex found, then the half-planes given by points A, E and E, B has to be recomputed, or
3. change of the position of extreme points forming AABBox, e.g. position of the point A, then the half-planes A, E and H, A have to be recomputed and check if the vertices H and E remained convex; in the concave case, the relevant vertex has to be replaced by a virtual one laying the line AB or DA, see Fig.4.

Note, that the AABBox and eight-point convex hull are changing dynamically as points are processed with the complete computational complexity $O(N)$.

However, the areas Ω_{i0} and Ω_{i1} contain also invalid points due to their incremental construction, and have to be rechecked and non-relevant points have to be removed.

After those two steps above, eight subsets Ω_{i0} and Ω_{i1}, $i = 0, \ldots, 3$ are obtained and their points are used for further processing. In the maximum distance case, the opposite areas are to be tested similarly as in [5,6].

There are two different situations in dynamically building the approximate convex hull, i.e. a new point $\boldsymbol{\xi}$:

1. is changing the AABB
2. does not change the AABB

In the case, when the new point $\boldsymbol{\xi}$ does not change the AABB, the simple half-planes tests are to be used, see Tab.1:

$p_A > 0$	$p_B > 0$	Ω_i
$+$	$+$	Ω_0
$+$	$-$	Ω_1
$-$	$+$	Ω_3
$-$	$-$	Ω_2

Table 1: Conditions for splitting the Ω dataset into datasets Ω_i

Let us consider a new point $\boldsymbol{\xi}$ in the Ω_0 area. There are the following possible cases:

– the point $\boldsymbol{\xi}$ is closer to the AABB corner P than the CH corner E, then the point $\boldsymbol{\xi}$ replaces the CH point E, Fig.5, and the line p_P has to be recomputed, i.e. the areas Ω_{00} and Ω_{01} are changed.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7

– the position of the point $\boldsymbol{\xi}$ can be in an area Ω_{00} or Ω_{01}. If the point $\boldsymbol{\xi}$ is in the area Ω_{00}, resp. Ω_{01}, the point is stored in the relevant list.
– the point $\boldsymbol{\xi}$ is not in area Ω_{00} nor Ω_{01}, the point is removed from the future processing.

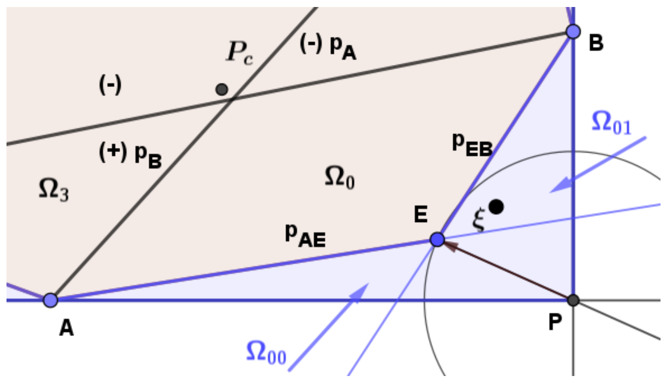


Fig. 5: Corner detail with Ω_{00} and Ω_{01} subareas specification

Now, an approximate convex hull (A-CH) has been constructed. It is defined by eight points, i.e. $A, E, B, F, C, G, D, H, A$. It means, that after this preprocessing step of all points with $O(N)$ complexity, points are:

– some points are stored in the lists Ω_{i0}, resp. Ω_{i1}, $i = 0, \ldots, 3$,
– some points have been removed directly during this preprocessing step.

Now, points in Ω_{i0} and Ω_{i1} are re-checked and points inside the A-CH are to be removed. It means, that points inside the Ω_{i0}, resp. Ω_{i1}, $i = 0, \ldots, 3$ areas form only a very small fraction of the original data set.

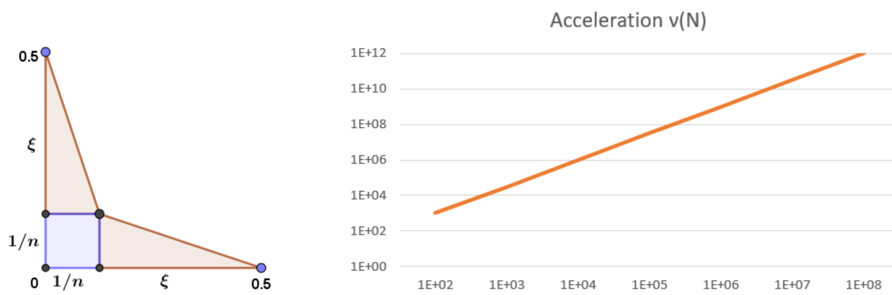
2.3 Preprocessing efficiency

The preprocessing algorithm described above is of the $O(N)$ computational complexity and computational requirements are very low, actually half-plane tests only[1]. As the preprocessing algorithm is intended for algorithms with a higher computational complexity, the efficiency of the preprocessing algorithm is an important issue.

For the sake of simplicity, let us consider data from the domain $[0, 1] \times [0, 1]$ with $N = n^2$ points, and points forming the AABBox are in the middle of the AABB edges. Fig.6a presents a detail of the AABB corner areas, formed by a square and two triangles.

[1] Half-plane test is implemented as the dot-product, and a separating plane line is determined by the outer-product (cross product in this case) [4].

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7



(a) Corner area detail

(b) Efficiency estimation $\nu = \nu(N)$

Fig. 6: Preprocessing efficiency

Then the "blue zones" in Fig.6a consists of a square of the size $1/n \times 1/n$ and two triangles of the size $1/n \times (0.5 - 1/n)$. The area contains $M = 1/(2n)$ points.

It means, that points inside the convex hull can be removed directly and only $4/(2n)$ of points of four corners need to be further processed. It leads to the preprocessing efficiency estimation of this step ν, Fig.6b:

$$\nu = \frac{N}{4M} = \frac{n^2}{4\frac{1}{2n}} = \frac{n^3}{2} \quad , \quad \nu = O(n^3) = O(N\sqrt{N}) \tag{1}$$

The efficiency of preprocessing grows $O(n^3)$, resp.$O(N\sqrt{N})$, where N is the number of points in the given data set.

3 Conclusion

The proposed preprocessing algorithm with the $O(N)$ computational complexity has been introduced. It can be used directly with an advantage for the solution of many geometrical problems with higher computational complexity the $O(N)$, e.g. for a convex hull, a diameter of a convex hull, smallest enclosing circle computations, etc. An extension for the E^3 case is expected.

Acknowledgment

The author would like to thank colleagues from Zhejiang University (Hangzhou), Shandong University in Jinan (China), colleagues and students at the University of West Bohemia in Pilsen (CZ) for the suggestions and fruitful discussions, and special thanks to Alexander Drdak for experimental counter implementation. Thanks also go to the anonymous reviewers for their critical comments, advice provided, and unknown relevant references.

References

1. Kirkpatrick, D.G., Seidel, R.: Ultimate planar convex hull algorithm? SIAM Journal on Computing **15**(1), 287 – 299 (1986). https://doi.org/10.1137/0215021
2. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica (New York) **16**(4-5), 498–516 (1996). https://doi.org/10.1007/bf01940877, the code available at https://news.ycombinator.com/item?id=14475832
3. Shen, K.W., Wang, X.K., Wang, J.Q.: Multi-criteria decision-making method based on smallest enclosing circle in incompletely reliable information environment. Computers and Industrial Engineering **130**, 1–13 (2019). https://doi.org/10.1016/j.cie.2019.02.011
4. Skala, V.: Barycentric coordinates computation in homogeneous coordinates. Computers and Graphics (Pergamon) **32**(1), 120–127 (2008). https://doi.org/10.1016/j.cag.2007.09.007
5. Skala, V.: Fast $o_{expected}(n)$ algorithm for finding exact maximum distance in E2 instead of $o(n^2)$ or $o(n\ lgn)$. AIP Conference Proceedings **1558**, 2496–2499 (2013). https://doi.org/10.1063/1.4826047
6. Skala, V.: Diameter and convex hull of points using space subdivision in E2 and E3. LNCS **12249**, 286–295 (2020). https://doi.org/10.1007/978-3-030-58799-4_21
7. Skala, V., Majdisova, Z.: Fast algorithm for finding maximum distance with space subdivision in E2. LNCS **9218**, 261–274 (2015). https://doi.org/10.1007/978-3-319-21963-9_24
8. Skala, V., Majdisova, Z., Smolik, M.: Space subdivision to speed-up convex hull construction in E3. Advances in Engineering Software **91**, 12–22 (2016). https://doi.org/10.1016/j.advengsoft.2015.09.002
9. Skala, V., Smolik, M., Majdisova, Z.: Reducing the number of points on the convex hull calculation using the polar space subdivision in E2. SIBGRAPI 2016 pp. 40–47 (2017). https://doi.org/10.1109/SIBGRAPI.2016.015
10. Skala, V.: Fast $o_{expected}(n)$ algorithm for finding exact maximum distance in e^2 instead of $o(n^2)$ or $o(n \lg n)$. In: AIP Conference Proceedings. AIP (2013). https://doi.org/10.1063/1.4826047, `http://dx.doi.org/10.1063/1.4826047`
11. Skala, V., Cerny, M., Saleh, J.Y.: Simple and efficient acceleration of the smallest enclosing ball for large data sets in e2: Analysis and comparative results. Lecture Notes in Computer Science **13350 LNCS**, 720 – 733 (2022). https://doi.org/10.1007/978-3-031-08751-6_52
12. Smolik, M., Skala, V.: Efficient speed-up of the smallest enclosing circle algorithm. Informatica (Netherlands) **33**(3), 623 – 633 (2022). https://doi.org/10.15388/22-INFOR477
13. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). LNCS **555**, 359–370 (1991). https://doi.org/10.1007/BFb0038202
14. Welzl, E.: The smallest enclosing circle - a contribution to democracy from switzerland? Algorithms Unplugged pp. 357–360 (2011). https://doi.org/10.1007/978-3-642-15328-0_36

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_7