A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E^2

Vaclav Skala[1], Michal Smolik[1]

[1] Faculty of Applied Sciences, University of West Bohemia,
Univerzitni 8, CZ 30614 Plzen, Czech Republic

Abstract. The point inside/outside a polygon test is used by many applications in computer graphics, computer games and geographical information systems. When this test is repeated several times with the same polygon a data structure is necessary in order to reduce the linear time needed to obtain an inclusion result. In the literature, different approaches, like grids or quadtrees, have been proposed for reducing the complexity of these algorithms. We propose a new method using a polar space subdivision to reduce the time necessary for this inclusion test. The proposed algorithm is robust and has a performance of $O(k)$, where $k \ll N$, k is the number of tested intersections with polygon edges, and the time complexity of the preprocessing is $O(N)$, where N is the number of polygon edges.

Keywords: Point in polygon location; Space subdivision; Non-convex polygon.

1 Introduction

In computational geometry, the point location problem asks whether a given point in the plane lies inside or outside of a convex or non-convex polygon [4], [5], [6]. This problem finds applications in areas that deal with processing geometrical data, such as computer graphics, computer vision, geographical information systems (GIS), motion planning, Computer-aided design (CAD), and has been the subject of many research papers in computer science and related application disciplines.

For an overview, we refer to Snoeyink's survey paper [9]. Probably the most common algorithm for point in polygon testing without preprocessing is the "crossing number algorithm" and the first description of the algorithm is by Shimrat [7]. It is well known that handling degenerate cases in a crossing number algorithm is not obvious. Forrest [1] nicely illustrates the problems involved. While previous cases studied on practical point in polygon testing, e.g. [2], [8], [10], [11] focus on time performance and sometimes on memory usage.

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

2 Proposed Algorithm

In this section, we will introduce a new point in non-convex polygon test algorithm in E^2. The main idea of this algorithm is to divide all input polygon edges into several subsets using polar space subdivision. Then the point in polygon test will become fast and easy.

First, in section 2.1, we will introduce the technique of space subdivision. In section 2.2, we will show how to divide polygon edges into several sectors. Finally, in section 2.3, we will propose an approach for point inside a non-convex polygon test.

2.1 Space Subdivision

The $2D$ space can be divided into several non-overlapping polar shaped sectors. This division uses a center point and angular division. Center point $\boldsymbol{C}$ is calculated as the average of all corners of the non-convex polygon.

One way to divide the space is to use a uniform division of an angle from 0 to 2π. Using this, we have to calculate the exact angle between the vector $\boldsymbol{x}' = [0,1]^T$ and the vector $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{C}$, and such a calculation uses the following formula:

$$\theta = \text{arctg2}(\boldsymbol{v}_x, \boldsymbol{v}_y)\,. \tag{1}$$

Calculation of the function $\text{arctg2}(\boldsymbol{v}_x, \boldsymbol{v}_y)$ takes a lot of time and we therefore use a simplified calculation of the approximated angle.

The simplified angle is not uniformly distributed on a circle, but it is uniformly distributed on the border of a square $\langle -1,-1\rangle \times \langle 1,1\rangle$. When calculating the angle, we have to locate the exact half of quadrant, i.e. octant, see Fig. 1, where the point is located, and then calculate the intersection with the given edge. The intersection with an edge is simple, as all edges of the box axis are aligned and intersect with the main axes at y or $x = 1$ or -1. The distribution of a simplified angle can be seen on Fig. 1.

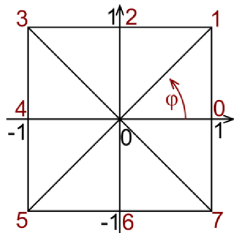
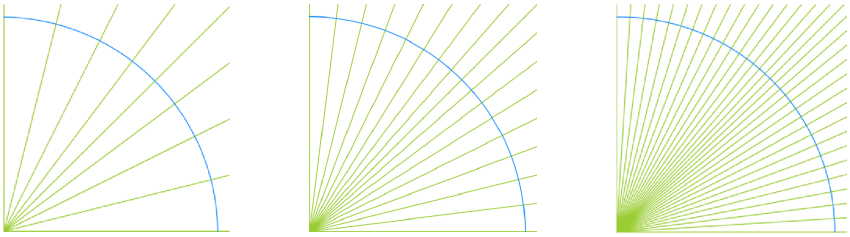


Fig. 1. Uniform distribution of a simplified angle on a unit square. Angle $\varphi \in \langle 0,8\rangle$ instead of normal values from interval $\theta \in \langle 0,2\pi\rangle$.

Calculation of a simplified angle is 1.34 times faster than the formula (1) and gives almost the same results, as can be seen in Fig. 2.



a) b) c)

Fig. 2. Division of space into 32 (a), 64 (b) and 128 (c) non-overlapping sectors using uniform distribution of a simplified angle.

Now we have a simple calculation of the simplified angle and therefore we are able to determine the index of the sector where the given point belongs.

2.2 Division of Polygon Edges into Sectors

All edges of the given polygon can be subdivided into sectors, i.e. each sector contains indices of all edges contained in the sector. The location of each edge can be determined using both endpoints of the edge. For each endpoint, we have to calculate the index of the sector (i_{start} and i_{end}) to which the vertex belongs. In the next step we have to add this edge into all sectors with indices from i_{start} to i_{end} (counterclockwise orientation), or i_{end} to i_{start} (clockwise orientation), see Fig. 3. To determine, whether the polygon edge is oriented clockwise (CW) or counterclockwise (CCW), we can use the following formula:

$$Orientation(\boldsymbol{E}_{start} - \boldsymbol{C}, \boldsymbol{E}_{end} - \boldsymbol{C}) = (\boldsymbol{v}_{start} \times \boldsymbol{v}_{end})_z = \begin{vmatrix} v_{start}^{(x)} & v_{start}^{(y)} \\ v_{end}^{(x)} & v_{end}^{(y)} \end{vmatrix}, \quad (2)$$

where $\boldsymbol{E}_{start}$ and $\boldsymbol{E}_{end}$ are starting and ending points of the polygon edge, $\boldsymbol{v}_{start}$ and $\boldsymbol{v}_{end}$ are vectors in $3D$ calculated as:

$$\begin{aligned} \boldsymbol{v}_{start} &= \begin{bmatrix} v_{start}^{(x)} & v_{start}^{(y)} & 1 \end{bmatrix}^T = [(\boldsymbol{E}_{start} - \boldsymbol{C}) \quad 1]^T \\ \boldsymbol{v}_{end} &= \begin{bmatrix} v_{end}^{(x)} & v_{end}^{(y)} & 1 \end{bmatrix}^T = [(\boldsymbol{E}_{end} - \boldsymbol{C}) \quad 1]^T, \end{aligned} \quad (3)$$

where $\boldsymbol{C}$ is the origin of polar space subdivision. The positive result confirms that the orientation of the endpoints is CCW and the negative result confirms CW orientation.

During the subdivision of all edges, we can precalculate the direction vector $\boldsymbol{e}_i$ of each edge $\boldsymbol{E}_{start}^{(i)}\boldsymbol{E}_{end}^{(i)}$ with $i \in \{1, \ldots, \text{edges\_count}\}$:

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

$$\boldsymbol{e}_i = \boldsymbol{E}_{end}^{(i)} - \boldsymbol{E}_{start}^{(i)} . \tag{4}$$

This vector will be used in the future when testing a point inside a non-convex polygon.

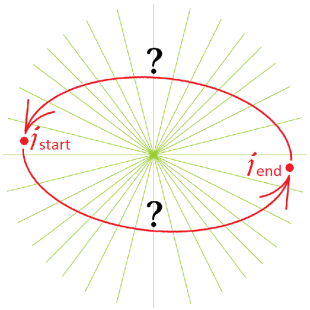


Fig. 3. Orientation of the polygon edge with endpoints in sectors with indices i_{start} and i_{end} (CCW or CW).

2.3 Point inside Non-Convex Polygon Test

To determine, whether a point is inside a non-convex polygon, we will use a ray-casting method. We create a line from the center point $\boldsymbol{C}$ to the testing point $\boldsymbol{P}$. Then we count how many times the segment line $\boldsymbol{CP}$ intersects with the polygon. The number of intersections is called the crossing number.

If the point $\boldsymbol{C}$ is inside the polygon, then if the crossing number is an odd number, the point is outside the polygon, otherwise if it is an even number, the point is inside. If the point $\boldsymbol{C}$ is outside the polygon, then if the crossing number is an odd number, the point is inside the polygon (see Fig. 4), otherwise if it is an even number, the point is outside.

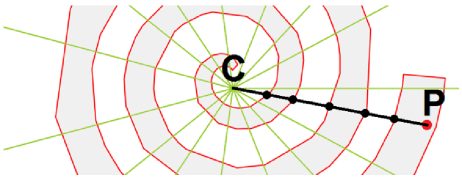


Fig. 4. The number of intersections of segment line $\boldsymbol{CP}$ with polygon edges.

The first step of the algorithm is to determine the sector to which the testing point belongs. This is done using the simplified angle calculation described in section 2.1. Each segment contains a list of polygon edges, which lie inside of it. This number of edges can be small, when using high number of divisions (i.e. sectors), compared to the number of polygon edges. The great advantage is that we do not have to test the

intersection with all polygon edges, instead of it we test the intersection only with edges contained in the particular segment.

To determine whether a line segment $\boldsymbol{CP}$ intersects a polygon edge $\boldsymbol{E}_{start}\boldsymbol{E}_{end}$ we have to use the following approach. The first criterion is that both endpoints $\boldsymbol{C}$ and $\boldsymbol{P}$ must lie on the opposite sides of the polygon edge. This is fulfilled when:

$$Orientation(\boldsymbol{e}, \boldsymbol{C} - \boldsymbol{E}_{start}) \cdot Orientation(\boldsymbol{e}, \boldsymbol{P} - \boldsymbol{E}_{start}) \leq 0\ , \tag{5}$$

where $\boldsymbol{e}$ is the precalculated directional vector, i.e. $\boldsymbol{e} = \boldsymbol{E}_{end} - \boldsymbol{E}_{start}$, and function $Orientation(\boldsymbol{a}, \boldsymbol{b})$ is defined in (2). The second criterion is that both endpoints $\boldsymbol{E}_{start}$ and $\boldsymbol{E}_{end}$ must lie on the opposite sides of the line $\boldsymbol{CP}$. This is fulfilled when:

$$Orientation(\boldsymbol{P} - \boldsymbol{C}, \boldsymbol{C} - \boldsymbol{E}_{start}) \cdot Orientation(\boldsymbol{P} - \boldsymbol{C}, \boldsymbol{C} - \boldsymbol{E}_{end}) \leq 0\ . \tag{6}$$

Now, we know how that if both conditions (5) and (6) are fulfilled then a line segment $\boldsymbol{CP}$ intersects a polygon edge $\boldsymbol{E}_{start}\boldsymbol{E}_{end}$. We count the number of intersections with polygon edges contained in a segment, i.e. the crossing number.

The only thing we need to know is if the point $\boldsymbol{C}$ is inside or outside the non-convex polygon. We create a virtual ray with starting point at $\boldsymbol{C}$ and directional vector $[1, 0]^T$. Then we compute the number of intersections with polygon edges contained in sector with index 0, as the simplified angle of vector $[1, 0]^T$ is 0. The intersection of an edge $\boldsymbol{E}_{start}\boldsymbol{E}_{end}$ with a ray is calculated as:

$$Orientation([1, 0]^T, \boldsymbol{C} - \boldsymbol{E}_{start}) \cdot Orientation([1, 0]^T, \boldsymbol{C} - \boldsymbol{E}_{end}) \leq 0\ . \tag{7}$$

If the number of intersections is odd, the point $\boldsymbol{C}$ lies inside the polygon otherwise if the number of intersections is even, the point $\boldsymbol{C}$ lies outside the polygon.

3 Experimental Results

The approach proposed has been implemented in C# using .Net Framework 4.5 and tested on data sets using PC with the configuration:

- CPU: Intel® Core™ i7 920 (4 × 2,67GHz),
- memory: 12 GB RAM,
- operating system 64bits Microsoft Windows 8

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

3.1 Examples of Testing Polygons

The proposed approach has been tested on different types of non-convex polygons. First type were randomly generated polygons (see Fig. 5) and second type were real polygons from geographic information system (GIS) (see Fig. 6).

Randomly generated polygons can have any number of edges and both have very different shapes.

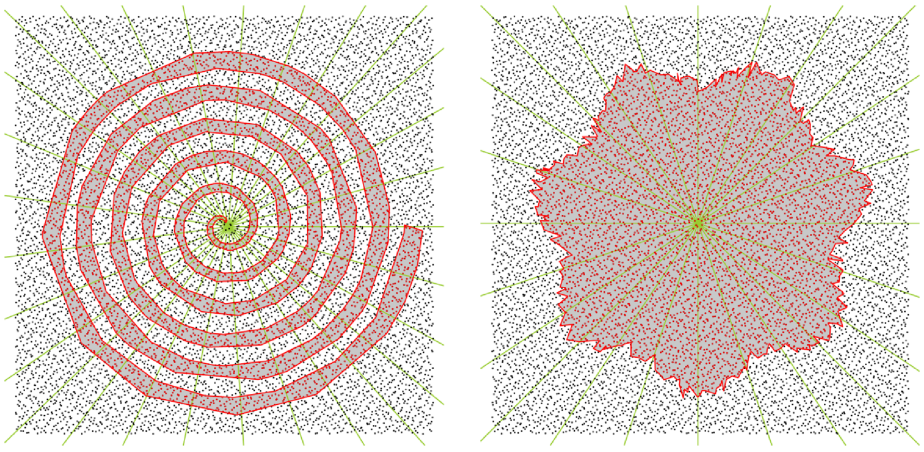


Fig. 5. Randomly generated spiral polygon (left) and star polygon (right).

Real polygons represent two regions in the Czech Republic. They were chosen as they have the most complex shapes from all regions in this country. One of them contains 1 672 edges, i.e. the Olomoucky region, and the second one contains 1 461 edges, i.e. the Kralovehradecky region.

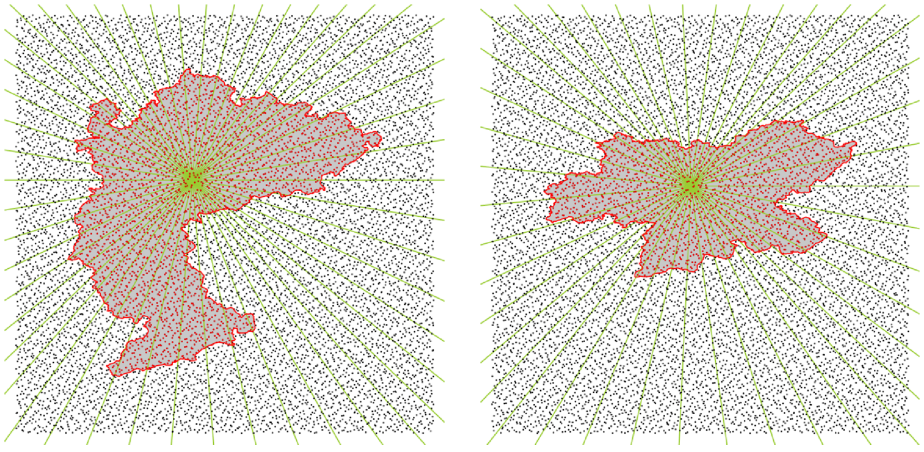


Fig. 6. Real data sets of the Olomoucky (left) and Kralovehradecky (right) region in Czech Republic. Polygon contains 1 672 edges (left) and 1 461 edges (right).

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

3.2 Distribution of Testing Points

The proposed approach has been tested using points with Halton points distribution in $2D$ [3]. Halton sequence is a deterministic sequence of numbers that produces well-spaced "draws" from the unit interval. The sequence is based on a particular prime number and is constructed based on finer and finer prime-based divisions of sub-intervals of unit interval. An example of a Halton sequence based on prime number 2 and 3 starts with the following numbers:

$$\begin{aligned} Halton(2) &= \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \dots \\ Halton(3) &= \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \dots \end{aligned} \tag{8}$$

When we pair the Halton sequences in (8) up, we get a sequence of points in $2D$ in a unit square:

$$\begin{aligned} Halton(2,3) = &\left(\frac{1}{2}, \frac{1}{3}\right), \left(\frac{1}{4}, \frac{2}{3}\right), \left(\frac{3}{4}, \frac{1}{9}\right), \left(\frac{1}{8}, \frac{4}{9}\right), \left(\frac{5}{8}, \frac{7}{9}\right), \\ &\left(\frac{3}{8}, \frac{2}{9}\right), \left(\frac{7}{8}, \frac{5}{9}\right), \left(\frac{1}{16}, \frac{8}{9}\right), \left(\frac{9}{16}, \frac{1}{27}\right), \dots \end{aligned} \tag{9}$$

It can be seen that a Halton sequence covers the space more evenly than randomly generated uniform points (see Fig. 7). This sequence of points will be used as input points for testing a point inside or outside a polygon.

Fig. 7. 10^3 $2D$ Halton points generated by $Halton(2,3)$ (left) and 10^3 $2D$ random points with uniform distribution (right).

3.3 Optimal Number of Sectors

The edges of input polygon have to be divided into several sectors such as each sector will contain only edges that are inside that sector. The important key is how many sectors should be created, i.e. what is the divide count for $2D$ polar space subdivision. We measured the proportion between the divide count and the polygon edges count.

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

The proposed algorithm consists of two phases. The first one is the preprocessing when dividing polygon edges into sectors and the second phase is the location test of a point inside or outside of polygon.

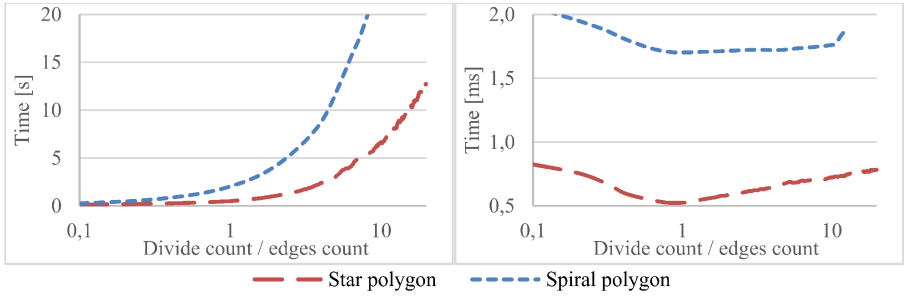


Fig. 8. Preprocessing time of the algorithm (left) and runtime of one point location test (right) The size of both polygons is $\mathbf{10^6}$ edges.

It can be seen (from Fig. 8 and Fig. 9) that with increasing number of divisions the preprocessing time increases as well. The runtime of one point location test decreases until some minimum number and then increases. The decreasing time needed for location test is caused due to decreasing number of edges in one sector. When the number of sectors is too large, the time performance becomes worse. This is due to problems with memory, as we cannot use the advantage of caching.

The optimal proportion of division count and polygon edges count is, according to the Fig. 8, equal 0.9, i.e. the division count is almost the same as the number of edges in polygon.

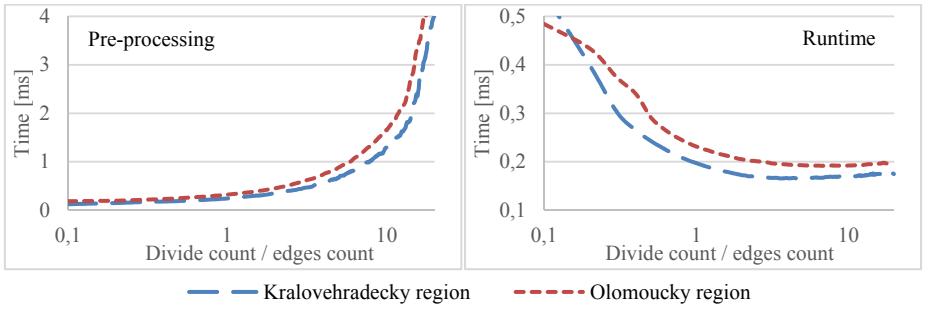


Fig. 9. Finding optimal division of space into several sectors (=division count). Preprocessing time for polygons with **1 4 61**(Kralovehradecky region) and **1 67 2**(Olomoucky region) edges (see Fig. 6) (left) and the time of location test of one point for the same polygons (right).

The optimal proportion of division count and polygon edges count is, according to the Fig. 9, for real GIS data sets equal around 4, i.e. the division count is four times higher than the number of edges in the polygon.

3.4 The Time Performance

In some applications, time performance is one of the most important criteria. We measured running times for the algorithm of point location test inside or outside a non-convex polygon. The running times were measured for different numbers of tested points generated with Halton distribution and for different polygons. The running times were measured many times and the average times are presented in Fig. 10 and Fig. 11.

As a comparison algorithm was selected the same algorithm as the proposed one, only without the space subdivision and thus without any preprocessing. This algorithm tests the point with all polygon edges and not only with edges contained in a particular sector like the algorithm proposed.

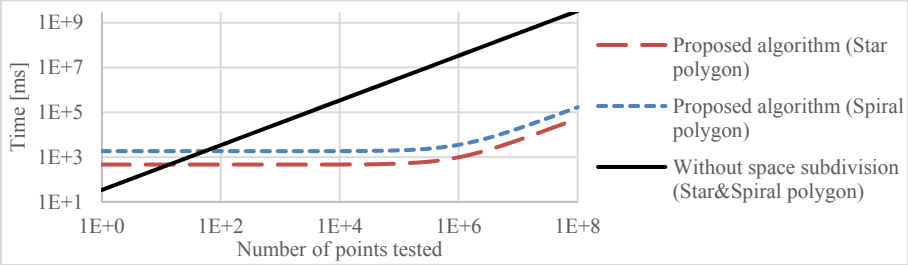


Fig. 10. The time complexity of the proposed algorithm (preprocessing time + runtime) and the algorithm without the space subdivision for different number of testing points. The same time complexity of both algorithms is for $\mathbf{14}$ (star polygon) and $\mathbf{60}$ (spiral polygon) testing points. We used the randomly generated polygons from Fig. 5 with $\mathbf{10^6}$ edges.

According to Fig. 10, we can see that the location test for a point inside or outside a star polygon is faster than for a spiral polygon, i.e. is 3.75 times faster. This is due to the fact that we have to test the segment line $\boldsymbol{CP}$ with more polygon edges. The proposed algorithm overcomes the algorithm without the space subdivision already for only 14 (star polygon) and 60 (spiral polygon) testing points.

According to Fig. 11, we can see that the location tests for a point inside or outside real GIS polygons have almost the same time performance as the number of polygon edges is almost the same. The proposed algorithm overcomes the algorithm without the space subdivision already for only 20 testing points (the Olomoucky region) and 10 testing points (the Kralovehradecky region).

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

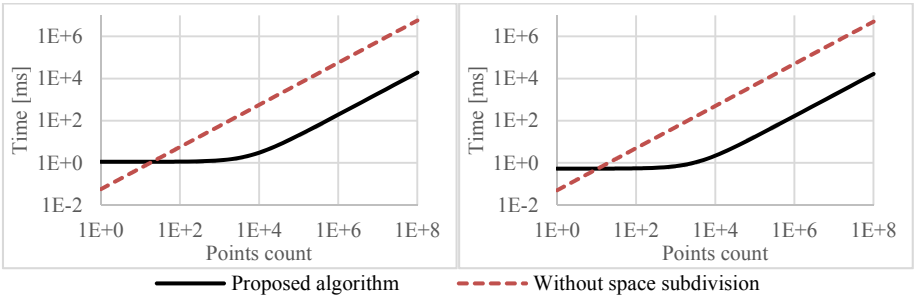


Fig. 11. The time complexity of the proposed algorithm (preprocessing time + runtime) and the algorithm without the space subdivision for different number of testing points. The same time complexity of both algorithms is for $\mathbf{2\ 0}$ (left) and $\mathbf{10}$ (right) testing points. We used the real non-convex polygons from Fig. 6: Olomoucky region (left) and Kralovehradecky region (right).

Using measured data from Fig. 10 and Fig. 11, we can compute the speed-up of the proposed algorithm to the algorithm without the space subdivision (see Fig. 12).

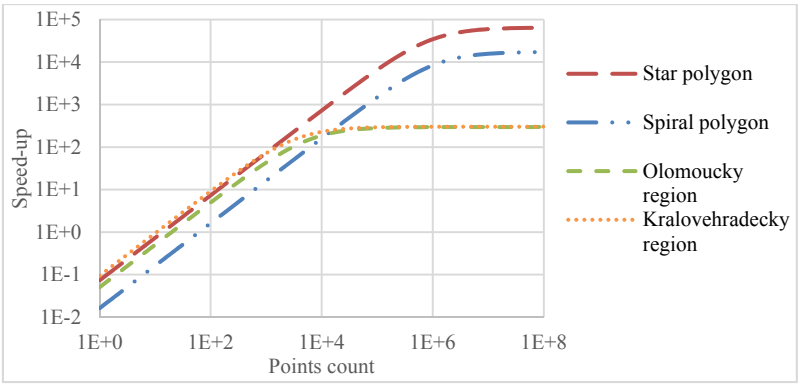


Fig. 12. The speed-up of proposed algorithm to the algorithm without the space subdivision. The star and spiral polygons contain $\mathbf{10^6}$ edges, polygon of the Olomoucky region contains $\mathbf{1\,67\ 2}$ edges and polygon of the Kralovehradecky region contains $\mathbf{1\ 4\ 61}$ edges.

It can be seen that the maximum speed-up is for the star polygon ($6 \cdot 10^4$) and the spiral polygon ($1.7 \cdot 10^4$). The speed-up is dependent on the size of the polygon, because the proposed algorithm uses for each test almost the same number of edges (independently of the size of the polygon), but the algorithm without the space subdivision uses always all edges.

A Point in Non-Convex Polygon Location Problem Using the Polar Space Subdivision in E2, ICIG 2015 proceedings Part I, LNCS 9217, China, pp.394-404, ISSN 0302-9743, ISBN 978-3-319-21977-6, Springer, 2015

4 Conclusion

A new fast and easy to implement location test algorithm of point inside or outside a polygon in E^2 has been presented. It uses the polar space division technique to speed up computation. The proposed algorithm proved robustness for different polygon types. The algorithm proposed is convenient for large polygons.

In the future, the algorithm will be modified to enable testing a point inside or outside a polyhedron in E^3.

Acknowledgments. The authors would like to thank their colleagues at the University of West Bohemia, Plzen, for their discussions and suggestions, and anonymous reviewers for their valuable comments and hints provided. The research was supported by MSMT CR project LH12181 and SGS 2013-029.

References

1 Forrest, A.R.: Computational Geometry in Practice. Fundamental Algorithms for Computer Graphics, ISBN: 978-3-540-54397-8, 1991, Vol. 17, pp. 707-724.
2 Haines, E.: Point in polygon strategies. In P. Heckbert, editor, Graphics Gems IV, Academic Press, Boston, MA, 1994, pp. 24-46.
3 Halton, J.: Algorithm 247: Radical-inverse quasi-random point sequence. ACM, 1964, Vol. 7, No. 12, pp. 701-702.
4 Jimenez, J. J., Feito, F. R., Segura, R. J.: A new hierarchical triangle-based point-in-polygon data structure. Computers & Geosciences, 2009, Vol. 35, No. 9, pp. 1843-1853.
5 Li, J., Wang, W., Wu, E.: Point-in-polygon tests by convex decomposition. Computers & Graphics, 2007, Vol. 31, No. 4, pp. 636-648.
6 Li, W., Ong, E. T., Xu, S., Hung, T.: A Point Inclusion Test Algorithm for Simple Polygons. ICCSA, 2005, pp. 769-775.
7 Shimrat, M.: Algorithm 112: position of point relative to polygon. ACM, 1962, Vol. 5, No. 8, pp. 434.
8 Skala, V.: Trading Time for Space: an O(1) Average time Algorithm for Point-in-Polygon Location Problem. Theoretical Fiction or Practical Usage? Machine Graphics and Vision, 1996, Vol. 5, No. 3, pp. 483-494.
9 Snoeyink, J.: Point location. In J. E. Goodman and J. O'Rourke, editors, Handbook of Discrete and Computational Geometry (2nd Ed.), CRC Press LLC, Boca Raton, FL, 2004, chapter 34, pp. 767-786.
10 Walker, R. J., Snoeyink, J.: Practical point-in-polygon tests using CSG representations of polygons. Algorithm Engineering and Experimentation, 1999, Vol. 1619, pp 114-128.
11 Yang, S., Yong, J. H., Sun, J. Gu, H. J., Paul, J. C.: A point-in-polygon method based on a quasi-closest point. Computers & Geosciences, 2010, Vol. 63, No. 2, pp. 205-213.