Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-1 (2018) 548–553

An Extension to Winding Number and Point-in-Polygon Algorithm

G. Naresh Kumar [*] and Mallikarjun. Bangi [**]

[*] *Scientist, DRDL, Hyderabad, Telangana, India, (e-mail: gaddam.nareshkumar@gmail.com)*
[**] *Engineer, Hyderabad, Telangana, India, (e-mail: nani.techiesite@gmail.com)*

Abstract: This work is an extension of an axis-crossing algorithm to compute winding number for solving point in polygon for an arbitary polygon. Polygons are popular drawings in computer graphics to represent different types of structures with approximations. Solutions for point-in-polygons are many, like even-odd rule, positive-negative number, and winding number. This paper mainly deals with improvements of 'A winding number and point in polygon algorithm'. Point in polygon is a fundamental problem and has various applications in ray tracing, computer graphics, image processing, gaming applications, robotics, acoustics, geo-science etc. The main focus of this paper explains about winding number for a closed polygon 'S', to test whether point 'P' lies either inside or outside with respect to positive and negative axis-crossing algorithm method.

Keywords: Axis-crossing, Computational Geometry, Polygons, Point-in-Polygon test, Winding Number.

1. INTRODUCTION

Point-in-polygon is a simple geometric problem. In many scientific and engineering applications it is required to know if a given point lies within the polygon or outside. For a closed polygon 'S' and with an arbitrary point 'P' it is to be determined whether the point lies inside or outside the polygon. Especially, in many computer graphics based systems and geometric information processing systems, it is necessary to determine whether point is contained within a polygonal boundary or not. This problem is referred as the point inclusion problem, a classical problem in computer graphics and computational geometry (Foley et al. (1982), Preparata and Shamos (1985)). Solutions to the planar point inclusion problem are well developed, and many algorithms have been developed (Shimrat (1962), Hacker (1962), Salomon (1978)). Though, most point-in-polygon algorithms have been implemented successfully, but often there are problems with singularity conditions. Some algorithms require special data representations and pre-processing routines. All of these lead to longer run times or additional storage requirements.

The geometric definitions of polygons are given in many books (Foley et al. (1982), Lee and Preparata (1984), Preparata and Shamos (1985), Forrest (1991), Giblin (2013)). An arbitary polygon is a sequence of straight line segments placed end-to-end, joining the end of the last line segment to that of the first. A polygon is the closed region bounded by straight line segments. The line segments are called 'edges' and the points where two edges meet are called 'vertices'. In general, vertices are corners at which there is a bend (slope change) between two edges.

[⋆] The Author is with Defence Research and Development Laboratory, Hyderabad, India.

Polygons are broadly classified into two major categories as shown in Fig. 1. A polygon is called simple polygon, if its line segments do not intersect. On other hand a complex polygon edges are intersects with other edges and it may contain 'holes' in the polygon itself.

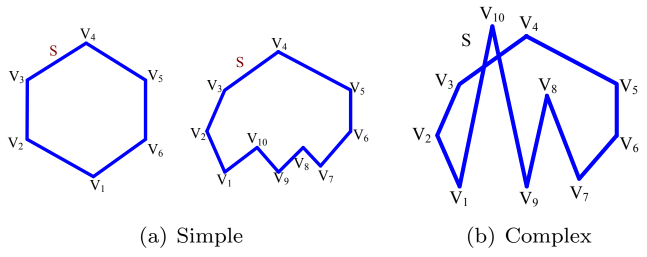


Fig. 1. Polygon classification

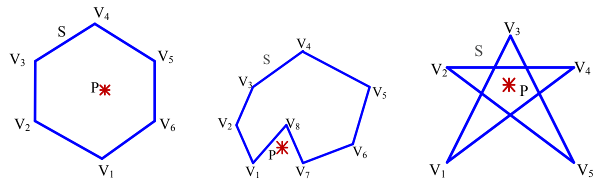
A particular subset of polygons are said to be convex, non-convex and convoluted polygons as indicated in Fig. 2.

(1) Convex polygons: The internal angles formed by connecting edges at each vertex should be less than 180^o as shown in Fig. 2(a)
(2) Non-convex polygons: At least one pair of edges connect to form an internal angle, greater than 180^o. Figure 2(b) indicates the typical non-convex polygon where vertex 'V_8' is more than 180^o internal angle.
(3) Convoluted polygons: The star has several crossing edges and the with one forming a hole in the another as shown in Fig. 2(c).

The remaining paper is organized as follows, Problem statement of point-in-polygon is presented in Section 2 and axis-crossing algorithm is discussed in Section 3. Section 4, deals with improvements on axis-crossing algorithm and

10.1016/j.ifacol.2018.05.092

Section 5 with simulation results and lastly, Section 6 concludes this work.



(a) Convex (b) Non-convex (c) Convoluted

Fig. 2. Types of point-in-polygon cases

2. POINT-IN-POLYGON PROBLEM

Point-in-polygon test is a common problem in computational geometry and computer graphics. In many applications, it is essential to know whether given point ‘P’ lies inside or outside of a given polygon ‘S’ with edges as vertices V_1, V_2, $\cdots$ V_n. The principles used by point-in-polygon algorithms considered in the literature include the line-crossing (Shimrat (1962), Anderson (1976), Bevis and Chatelain (1989), Kulowski (1992), Haines (1994), Hormann and Agathos (2001), Lewis (2002), Franklin and Randolph (2006), and Michael and Glauner (2012)) method. Using this methods it is possible to say that the given point is inside or outside by counting number of intersections of the sides with a line extending from the point to infinity. If an odd number of intersections is encountered, the point is inside the polygon; otherwise it is outside. The test line may be extended indefinitely from the point in any direction.

Most common algorithm for testing point-in-polygon without pre-processing is crossing number or line-crossing algorithm. In early 1960’s Shimrat (1962) used line crossing method for computing point-in-polygon for the first time and issues were resolved by Hacker (1962). Shimrat (1962) clearly states that his crossing number algorithm fails for a test point on the boundary. Forrest (1991) brought out the problems involved in computation of line-crossing method. Franklin and Randolph (2006) explained about point-in-polygon problem along with his Pnpoly code.

Salomon (1978) proposed a swath method, which requires pre-processing. Polygon is divided into trapezoids called swaths by using Seidel (1991) method and binary search is used to locate point w.r.to swath. Preparata and Shamos (1985) proposed wedges method, which also requires pre-processing before point-in-polygon test. Polygon is divided into wedges with a point ‘P’ inside the convex polygon ‘S’ and uses binary search for point-in-polygon test. Later by using cross-product position of a point is identified clearly. Main limitation of this approach is it works only on convex polygon.

Cell-based algorithm proposed by Žalik and Kolingerova (2001), which is especially useful to check when multiple number of points lie inside or outside. It works in two steps mainly. Grids of cells are equally generated and polygon is placed on the grid. Modified flood-fill algorithm is used to classify cells. Other step is to test all multiple points individually. If points falls on inner or outer cell results are obtained directly, and if point is on boundary of cell, local position is obtained. Bevis and Chatelain (1989) explained locating a point on a spherical surface relative to spherical polygon ‘S’ of any arbitrary shape. The test point ‘P’ lies inside, outside or on boundary of an spherical polygon. This algorithm is based on Even-Odd method.

In Hormann and Agathos (2001) Even-Odd algorithm to find point-in-polygon is explained in which a line is drawn from a point ‘P’ to ‘∞’ outside the polygon. If $\overline{P\infty}$ crosses the edges or vertices V_i of polygon $V_i = \overline{V_iV_{i+1}}$ even number of times, then the point ‘P’ is outside of the polygon, if it crosses odd number of times then point ‘P’ is inside the polygon. It not only finds crosses but also counts the number of crossings. It also shows an efficient method to find the intersection of line segment through a vertex for a given point ‘P’. However, the correctness of this even-odd counting is often corrupted by singularity conditions which occur when one or more vertices of the polygon lie exactly on the test line and also when one or more edges of the polygon are collinear with the extended test line. This failures are demonstrated with the help of red-cross polygon in section 3. These singularities can either be avoided by selecting a different test line which requires pre-process to choose an appropriate test line which is not possible is real-time application. Moreover, the end users (armed forces) may not have any background about all these algorithms.

The another popular approach is based on the winding number. Winding number (ω) is defined as how many times the polygon ‘S’ winds around that point ‘P’. If ‘P’ is inside a polygon ‘S’, then ω is nonzero. Similarly, if point ‘P’ is outside, only when the polygon doesn’t wind around the point at all that means $\omega = 0$. This paper uses a winding number as a basic technique for point-in-polygon algorithm and an extension of an algorithm given by Alciatore and Miranda (1995) . The winding number ω of a polygon ‘S’ and with an arbitary point ‘P’ not only measures how ‘S’ encloses point ‘P’, also it gives information about its orientation and count i.e., how many times polygon ‘S’ ‘winds around’ point ‘P’.

$$\omega = \left\{ \begin{array}{ll} 0; & \text{if P is outside S} \\ n > 0; & \text{if S winds P } n \text{ times counter clockwise} \\ n < 0; & \text{if S winds P } n \text{ times clockwise} \end{array} \right\}$$

The winding number is defined as a contour integral in the complex plane Churchill (1960). In generally the winding number ω of the curve in 2D xy-plane is equal to the total number of counter-clockwise turns that the point makes around the origin. Winding numbers play an important role in complex analysis. The winding number ω of a closed polygon ‘S’ in complex plane is expressed in terms of the complex coordinate form as below.

$$z = re^{i\theta} \tag{1}$$

$$dz = e^{i\theta}dr + ire^{i\theta}d\theta \tag{2}$$

and substituting Eq. 1 in Eq. 2

$$\frac{dz}{z} = \frac{dr}{r} + id\theta = d[\ln(r)] + id\theta \tag{3}$$

The total change in $\ln(r)$ is zero, and thus the $\int \frac{1}{z}dz$ is equal to i multiplied by the total change in θ. Therefore, the winding number of closed polygon 'S' about the origin is given by the expression 4.

$$\omega = \frac{1}{2\pi} \oint_S \frac{1}{z} dz \tag{4}$$

Winding number computation based on angle summation approach was presented in Gatilov (2013) to this problem. Sunday (2004) gives the basics winding number and crossing number algorithms. If a polygon is simple (i.e., it has no self intersections), then both methods give the same result for all points. But for non-simple polygons, the two methods give different results. For example, when a polygon overlaps with itself, then points in the overlap region are found to be outside using the crossing number, where as for winding number it is represented as inside. In this paper, we are discussing about improved axis-crossing algorithm for computing winding number ω of a given closed polygon 'S' with self intersections about a point 'P'.

3. AXIS-CROSSING ALGORITHM

Axis-crossing methods have been found in literature (Shimrat (1962), Anderson (1976), Kulowski (1992)) for a point in polygon algorithm. It was tested and provided successful results on different types of polygons generally encountered in scientific and engineering fields. Alciatore and Miranda (1995) proposed an efficient approach to calculate winding number (ω) for a closed planar polygon 'S' about an arbitary point 'P' using axis-crossing method rather than applying complex mathematically involved techniques to get results of winding number. Axis/ray-crossing method is an efficient algorithm extensively used in many applications. Axis-crossing method is calculated to test point-in-polygon for invariant point translations in horizontal and vertical planes. Geometry is translated to point 'P' as origin. Replace vertices V_i to $(V_i - P)$ for every point test 'i' of vertices before beginning the algorithm.

Alciatore et al. presented how to use axis-crossing method to determine the winding number, which in turn tracks direction and frequency (n) of crossing with positive x-axis. Each time polygon winds around a origin, crossing from below of positive x-axis represents counter clockwise winding, and crossing from above positive x-axis represents clockwise winding. Alciatore et al. tested their algorithm extensively and found satisfactory results, but it fails in some cases, especially in negative x-axis and when one or more edges of the polygon are collinear with the extended test line. This paper mainly gives the solutions to these problems and extension of algorithm for negative x-axis.

In-order to demonstrate the efficacy of the proposed algorithm, a red-cross polygon is considered here as indicated in Fig. 3. The main feature of this polygon is four vertices are collinear. Ten different points 'A', 'B', $\cdots$, up to 'J' are selected as depicted in Fig. 3. The point 'D' is inside the polygon and on positive x-axis, points 'B', 'C', 'E', and 'F' are on boundary of polygon and on positive x-axis, whereas points 'G', 'H', 'I', and 'J' are outside of polygon and on negative axis. Alciatore algorithm is used to test point 'D' and found inside which is true. The same experiment is repeated for points 'B', 'C', 'E' and 'F' (they are on boundary) found mixed results, such as 'B' & 'C' are outside and 'E' & 'F' are inside the polygon. Experiment is repeated for points 'G', 'H', 'I', and 'J', and found inside the polygon which is false.

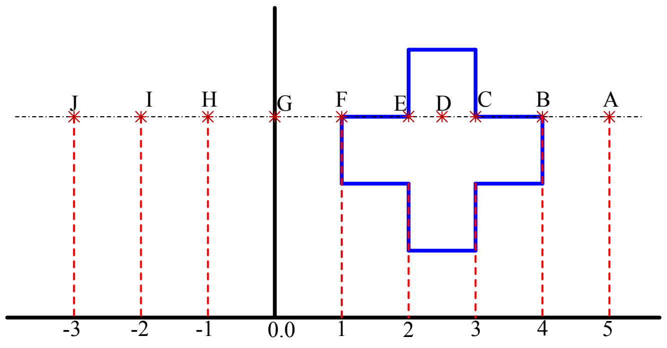


Fig. 3. Red-cross polygon

4. IMPROVED AXIS-CROSSING ALGORITHM

Winding number ω will be equal to winding a polygon about origin. i.e., for every vertex V_i of polygon, ω is updated according to the direction, crossing and intersection of every adjacent vertices $\overline{V_i V_{i+1}}$ about positive x-axis and negative x-axis. Figure 4 indicates the illustration of the terminology applicable for both positive and negative x-axis.

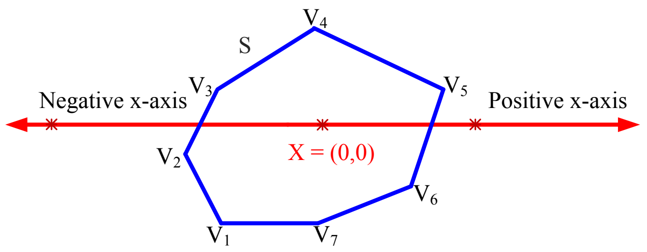


Fig. 4. New approach for axis-crossing

Improved axis-crossing algorithm addressing the negative x-axis issues are explained in detail here and pictorially shown in Fig. 5.

(1) Initialize winding number $\omega = 0.0$,
(2) Determine in which direction line segment crosses the positive x-axis or negative x-axis, for every line segments of a polygon.
(3) If crossing is from below of positive x-axis, increment $\omega = \omega + 1$, if crossing is from above of positive x-axis, decrement $\omega = \omega$ - 1,
(4) If crossing is from below of negative x-axis, decrement $\omega = \omega$ - 1, if crossing is from above of negative x-axis, increment $\omega = \omega + 1$,
(5) If crossing is from below of positive x-axis to boundary on x-axis or if crossing is from boundary of x-axis to above of positive x-axis then increment $\omega = \omega + 1/2$, if crossing is from above of positive x-axis to boundary on x-axis or if crossing is from boundary to below of positive x-axis, decrement $\omega = \omega$ - 1/2,

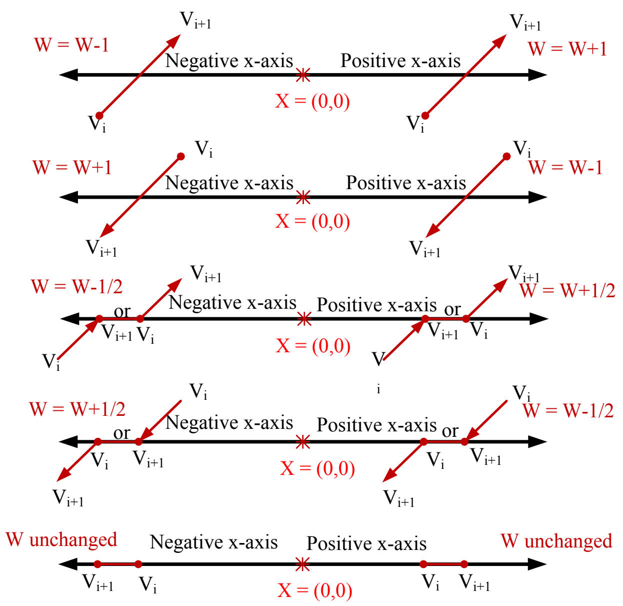


Fig. 5. Improved axis-crossing algorithm

(6) If crossing is from below of negative x-axis to boundary on x-axis or if crossing is from boundary of x-axis to above of negative x-axis then decrement $\omega = \omega$ - 1/2, if crossing is from above of negative x-axis to boundary on x-axis or if crossing is from boundary to below of negative x-axis, increment $\omega = \omega + 1/2$,
(7) If points are on the boundary of positive x-axis or on negative x-axis, then ω is unchanged.
(8) Repeat this steps for every line segment of polygon.

4.1 Algorithm

Detailed algorithm for both positive and negative x-axis is given in this section. Improvement over Alciatore algorithm is printed in blue color.

Input: Vertices $V_i(i = 1$ to n) of the polygon 'S'.
Output: Winding number 'ω' of polygon 'S' about point 'P(x, y)'.

(1) Replace each V_i by $(V_i - x)$
(2) Initialize winding number $\omega = 0$
(3) For each vertex $V_i(i = 1$ to n) of the polygon 'S'.

```
if ($y_i y_{i+1} < 0$)then [$\overline{V_i V_{i+1}}$ crosses the x-axis]
    R = $x_i + [\frac{y_i(x_{i+1}-x_i)}{y_i-y_{i+1}}]$
    [R is the x-coordinate of the intersection
    of $\overline{V_i V_{i+1}}$ and the x-axis]
    if($R > 0$) then [Positive x-axis]
        [$\overline{V_i V_{i+1}}$ crosses the positive x-axis]
        if($y_i < 0$)then
            [crossing from below of positive x-axis]
            $\omega = \omega + 1$
        else [crossing from above of positive x-axis]
            $\omega = \omega$ - 1
        elseif(($y_i = 0$) & ($x_i > 0$))then
        [$V_i$ is on positive x-axis]
            if($y_{i+1} > 0$)then
                $\omega = \omega + 0.5$
            else($y_{i+1} < 0$)then
                $\omega = \omega$ - 0.5
        elseif(($y_{i+1} = 0$) & ($x_{i+1} > 0$))then
            [$V_{i+1}$ is on positive x-axis]
            if($y_i < 0$)then
                $\omega = \omega + 0.5$
            else($y_i > 0$)then
                $\omega = \omega$ - 0.5
            end
        end
    else [Negative x-axis]
        if($y_i > 0$)then
            [crosses from below of negative x-axis]
            $\omega = \omega$-1
        else [crossing from above of negative x-axis]
        $\omega = \omega$+1
        elseif(($y_i = 0$) & ($x_i < 0$))then
        [$V_i$ is on negative x-axis]
            if($y_{i+1} < 0$)then
                $\omega = \omega + 0.5$
            else($y_{i+1} > 0$)then
                $\omega = \omega$ - 0.5
            end
        elseif(($y_{i+1} = 0$) & ($x_{i+1} < 0$))then
        [$V_{i+1}$ is on negative x-axis]
            if($y_i > 0$)then
                $\omega = \omega + 0.5$
            else ($y_i < 0$)then
                $\omega = \omega$ - 0.5
            end
        end
    end
else
    $\omega$=0
end
```

(4) Return ω = Winding number of polygon 'S' about 'P'.

5. RESULTS

The point-in-polygon tests are carried out on red-cross polygon and test points are selected in horizontal & vertical planes. The selected points are combination of inside, outside and on boundary. Both the algorithms were tested and results are indicated in Fig. 6 and 7. It is to be noted that, in all these figures marker '×' (into) represents point is inside the polygon, and marker 'O' represents the point is outside the polygon. Similarly, marker text color black is for Alciatore's algorithm and text color red for improved algorithm. Table 1 and 2 summarizes the performance of the both the algorithms for horizontal and vertical test cases. In view of the defence application, point on boundary is treated as inside the polygon and represented with marker '×' (into).

The same exercise is carried out for a set of points on x-axis (horizontal axis) and y-axis (vertical axis) which are physically on boundary, outside and inside of the offset red-cross polygon shown in Fig. 8. Similar results obtained on this polygon also. Further, both the polygons are subjected to an extensive Monte-Carlo testing with randomly generated point 'P'. Figure 9, and 10 demonstrate the the performance of the improved algorithm. Additional tests were performed for sets containing only randomly-

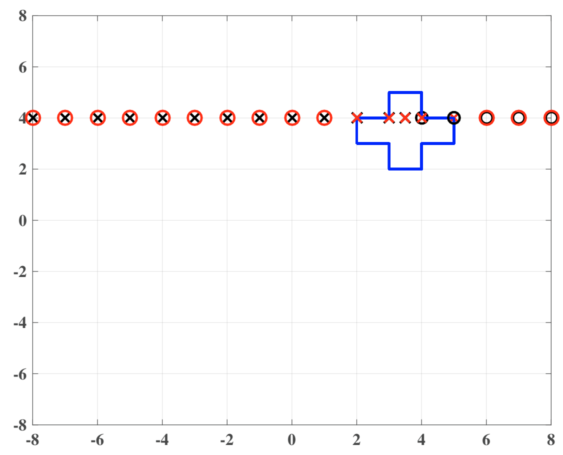


Fig. 6. Red cross polygon horizontal test case

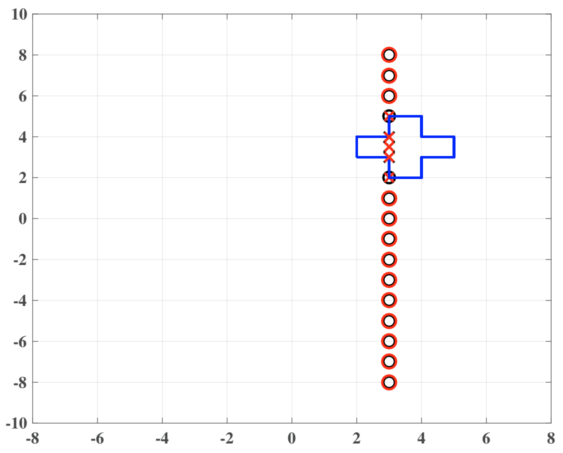


Fig. 7. Red cross polygon vertical test case

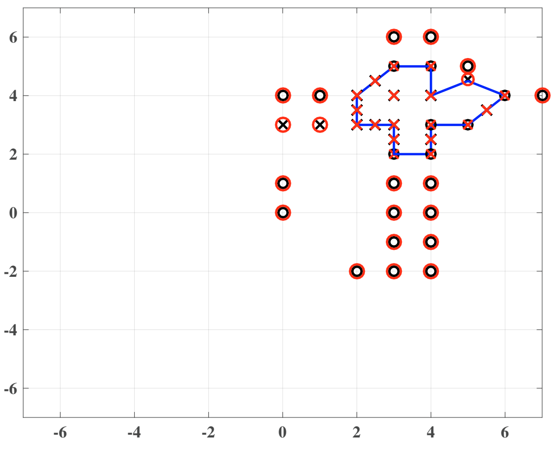


Fig. 8. Offset red-cross polygon test case

generated convex polygons, non-convex polygons, complex polygons, special cases like maps with randomly generated points on inside, outside, and on boundary to test.

6. CONCLUSION

Improvement/ extension to axis-crossing algorithm for computation of winding number is presented. Extensive studies shows that, the problems faced in previous algorithm are resolved by improvements carried out to handle the negative x-axis part. Improved algorithm can handle different types polygons without any pre-processing. It addresses all general simple closed convex polygons and complex polygons. It also calculates orientation of polygon in clockwise or anti-clockwise windings.

ACKNOWLEDGMENT

The authors thank competent authorities of Defence Research and Development Laboratory(DRDL), Hyderabad for granting permission to publish this piece of work.

REFERENCES

Alciatore, D.G. and Miranda, R. (1995). A winding number and point-in-polygon algorithm. *Glaxo Virtual*

Table 1. Comparison improved algorithm and Alciatore's for horizontal test

Case	Point	True	Algorithm Decision	
	(x,y)	Location	Alciatore's	Improved
1	8.0, 4.0	Outside	Outside	Outside
2	7.0, 4.0	Outside	Outside	Outside
3	6.0, 4.0	Outside	Outside	Outside
4	5.0, 4.0	Boundary	Outside	Inside
5	4.0, 4.0	Boundary	Outside	Inside
6	3.5, 4.0	Inside	Inside	Inside
7	3.0, 4.0	Boundary	Inside	Inside
8	2.0, 4.0	Boundary	Inside	Inside
9	1.0, 4.0	Outside	Inside	Outside
10	0.0, 4.0	Outside	Inside	Outside
11	-1.0, 4.0	Outside	Inside	Outside
12	-2.0, 4.0	Outside	Inside	Outside
13	-3.0, 4.0	Outside	Inside	Outside
14	-4.0, 4.0	Outside	Inside	Outside
15	-5.0, 4.0	Outside	Inside	Outside
16	-6.0, 4.0	Outside	Inside	Outside
17	-7.0, 4.0	Outside	Inside	Outside
18	-8.0, 4.0	Outside	Inside	Outside

Table 2. Comparison improved algorithm and Alciatore's for vertical test

Case	Point	True	Algorithm Decision	
	(x,y)	Location	Alciatore's	Improved
1	3.0, 8.0	Outside	Outside	Outside
2	3.0, 7.0	Outside	Outside	Outside
3	3.0, 6.0	Outside	Outside	Outside
4	3.0, 5.0	Boundary	Outside	Inside
5	3.0, 4.0	Boundary	Inside	Inside
6	3.0, 3.5	Inside	Inside	Inside
7	3.0, 3.0	Boundary	Inside	Inside
8	3.0, 2.0	Boundary	Inside	Inside
9	3.0, 1.0	Outside	Outside	Outside
10	3.0, 0.0	Outside	Outside	Outside
11	3.0, -1.0	Outside	Outside	Outside
12	3.0, -2.0	Outside	Outside	Outside
13	3.0, -3.0	Outside	Outside	Outside
14	3.0, -4.0	Outside	Outside	Outside
15	3.0, -5.0	Outside	Outside	Outside
16	3.0, -6.0	Outside	Outside	Outside
17	3.0, -7.0	Outside	Outside	Outside
18	3.0, -8.0	Outside	Outside	Outside

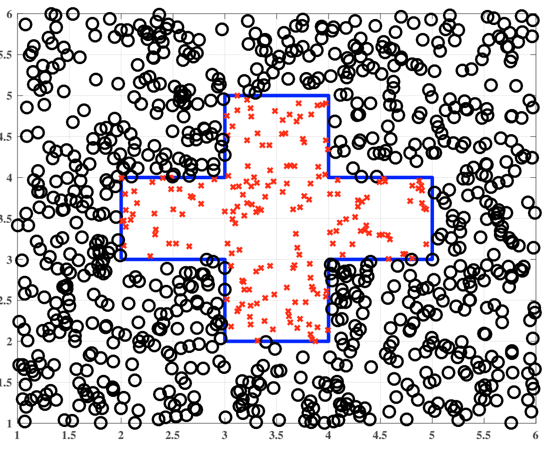


Fig. 9. Monte-Carlo on red cross polygon

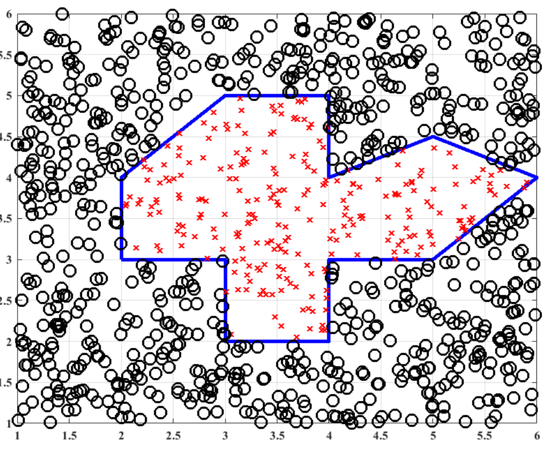


Fig. 10. Monte-Carlo on offset red cross polygon

Anatomy Project Research Report, Deparment of Mechanical Engineering, Colorado State University.

Anderson, K. (1976). Simple algorithm for positioning a point close to a boundary. *Jour. Math. Geology*, 8(1), 105–106.

Bevis, M. and Chatelain, J.L. (1989). Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. *Mathematical geology*, 21(8), 811–828.

Churchill, R.V. (1960). *Introduction to complex variables and applications.* McGraw-Hill, New York, NY, 2^{nd} edition edition. URL http://cds.cern.ch/record/498885.

Foley, J.D., Van Dam, A., et al. (1982). *Fundamentals of interactive Computer Graphics*, volume 2. Addison-Wesley Reading, MA.

Forrest, A. (1991). Computational geometry in practice. In *Fundamental Algorithms for Computer Graphics*, volume 17, 707–724. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-642-84574-1-30.

Franklin, W. and Randolph (2006). Pnpoly-point inclusion in polygon test. URL http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html.

Gatilov, S.Y. (2013). Efficient angle summation algorithm for point inclusion test and its robustness. *Reliable Computing*, 19(1), 1–25.

Giblin, P. (2013). *Graphs, surfaces and homology: An introduction o algebric topology.* Springer science & Business Media.

Hacker, R. (1962). Certification of algorithm 112: Position of point relative to polygon. *Commun. ACM*, 5(12), 606. doi:10.1145/355580.369118. URL http://doi.acm.org/10.1145/355580.369118.

Haines, E. (1994). Point in polygon strategies. *Graphics gems IV*, 994(6), 24–46. URL http://dl.acm.org/citation.cfmid=180895.180899.

Hormann, K. and Agathos, A. (2001). The point in polygon problem for arbitrary polygons. *Computational Geometry*, 20(3), 131–144.

Kulowski, A. (1992). Optimization of a point-in-polygon algorithm for computer models of sound field in rooms. *Applied Acoustics*, 35(1), 63–74. doi:http://dx.doi.org/10.1016/0003-682X(92)90036-R. URL http://www.sciencedirect.com/science/article/pii/0003682X9290036R.

Lee, D. and Preparata, F. (1984). Computational geometry - a survey. *IEEE Transactions on Computers*, C-33(12), 1072–1101. doi:10.1109/TC.1984.1676388.

Lewis, J.L. (2002). A reliable test for inclusion of a point in a polygon. *SIGCSE Bull.*, 34(4), 81–84. doi:10.1145/820127.820172. URL http://doi.acm.org/10.1145/820127.820172.

Michael, G. and Glauner, P.O. (2012). A correct even-odd algorithm for the point-in-polygon (PIP) problem for complex polygons. *CoRR*, abs/1207.3502. URL http://arxiv.org/abs/1207.3502.

Preparata, F.P. and Shamos, M.I. (1985). Introduction. In *Computational Geometry*, 1–35. Springer.

Salomon, K.B. (1978). An efficient point-in-polygon algorithm. *Computers & Geosciences*, 4(2), 173–178.

Seidel, R. (1991). A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry*, 1(1), 51 – 64. doi: http://dx.doi.org/10.1016/0925-7721(91)90012-4. URL http://www.sciencedirect.com/science/article/pii/0925772191900124.

Shimrat, M. (1962). Algorithm 112: position of point relative to polygon. *Communications of the ACM*, 5(8), 434.

Sunday, D. (2004). Fast winding number inclusion of a point in a polygon. *Dostupné z:¡ http://softsurfer.com/Archive/algorithm_0103/algorithm_0103. htm.*

Žalik, B. and Kolingerova, I. (2001). A cell-based point-in-polygon algorithm suitable for large sets of points. *Computers & Geosciences*, 27(10), 1135–1145.