DOI: 10.13009/EUCASS2019-490

Modified-Polygon Method for Point-in-Polygon Problem

Mostafa El-Salamony and Amr Guaily**,****
**College of Engineering, Peking University, Beijing, China.*
elsalamony.mostafa@pku.edu.cn
***Faculty of Engineering, Cairo University, Cairo, Egypt.*
**** Smart Engineering Systems Research Center, Nile University, Cairo, Egypt.*
amr.guaily@cu.edu.eg

Abstract

New method is presented for the point-in-polygon problem including Concave / Convex Cases. The method is based on the very basic definition of the polygon size (line/area/volume) depending on the spatial dimensions of the shape, which results in a very fast and accurate method when compared to existing techniques. The proposed method is tested against challenging problems and the results are compared with some of the existing techniques showing the superiority of the proposed method. This method is most efficient in Computational Fluid Mechanics applications.

1. Introduction

In numerical simulations of multi-phase flows using the level sets method, it is crucial to determine for each cell to determine the type of fluid exists. Hence, initially one shall solve the PIP problem for each node to create the signed distance function (SDF) as an initial condition to start the simulation. Another important application related to internal ballistics is capturing the burning surface of a solid fuel during the combustion process. Initially, the fuel grain has a known shape, then burning starts and the fuel grain shape burnt out. To trace the grain distortion a PIP shall be solved initially for proper capturing of the burning surface at later times by the level set technique [1].
In computational geometry, the point-in-polygon (PIP) problem seeks to decide if a given point in the plane lies outside, inside, or on the boundary of a polygon. It is one special case of point location problems. PIP is one of the fundamental problems involved in numerous applications starting from fluid mechanics and robotics to computer graphics and Geographical Information Systems [2].
Applications of PIP in computer vision and length measurements are numerous [3], starting from vision-based measurement to measure the deformation in bodies [4] to motion planning [5]. Also this can be applied to Unmanned Aerial Vehicles (UAVs) and its autonomous landing process. The UAV has to detect if a certain position is suitable for landing and measure how far is it, then trace its borders by detecting the intersections of the surrounding obstacles, hence the UAV can decide whether it will land or not. This can be applied on landing decision for a helicopter [6] or a fixed wing aircraft over a conventional runway [7] or even a ship [8].

2. Previous Algorithms

One of the primary methods to solve the PIP problem and the most popular method is the Ray Crossing (ray intersection) method which was derived by Nordbeck and Rystedt in 1967 [9]. Although it can deal with all kinds of polygons, it fails to detect if the investigated point lies on the polygon circumference or if the ray intersects a polygon vertex. It is executed in O(n) in time where n is the number of the polygon boundaries. Boundary cases are treated comparatively easily. Hence, this method is stable from numerical viewpoint [10].
An important upgrade to this algorithm was developed in 1994 [11] where a modified version was proposed to calculate the ray intersection method based on the vector representation and determinants.
The second method is the Area Method [9]. Its basic idea is to connect the investigated point to all vertices and make triangles of them, then calculate the area of each triangle individually and add them. If the total area is the same as the initial polygon area then the point is inside the polygons. If the area increases, then the point is outside the polygon. On one hand, it is easy to calculate the areas of a polygon but on the other hand, it is not applicable for concave polygons because fictitious areas will be generated.
In Swath method [12], the polygon is subdivided into swaths (trapezoids), which is evaluated in time O(n log n) [13], [14]. Then by using binary search, the trapezoid is located for the investigated point in time O(log n) then the

DOI: 10.13009/EUCASS2019-490

ray Intersecting method is performed on the polygon sides of the trapezoid. This method is preferred when many points are to be tested on the same polygon but the pre-processing is a drawback because of high computational effort.
Sum of Angles method ([15]-[16]) is O(n) in time and used widely, but it suffers from two main disadvantages; it is slow because calculating the angles is computationally expensive, and it is sensitive to the truncation and rounding errors. Thus, it is less suitable for polygons with large number of vertices.
Wedge method [14] can deal with convex polygons efficiently. In the first step the polygon is divided into n wedges using a random point inside the polygon, which is O(n) in time. Then by utilizing binary search, the wedge that includes the point of interest is found out in O(log n) in time. Finally, by using cross product operator, the investigated point relative location is found out in O(1) time.
Skala [17], in 1996, suggested a method using dual representation. It requires pre-processing, which has O(n) in time. Then the restriction test needs constant time O(1). One drawback of this method is that it can only be utilized for convex polygons without holes.
Huangand and Shih [18], in 1997, proposed Grid method which is appropriate for a raster-based cases. The polygon is depicted by a group of cells or pixels. The pixel colour at a certain position can be used to detect if the polygon includes the point of interest. This algorithm can be executed in O(n) time. The authors of [18] declared that this algorithm is not appropriate for vector-based representation of polygons.
Hormann and Agathos [19] investigated in detail the basic concepts of Ray-Crossing and Angle Summation methods, and they found that they are mathematically alike. Thus they derived a new algorithm and optimized it thoroughly. It is fast but wrongly detect the point location with respect to a polygon in case that the polygon lies between two peaks (e.g. star-shaped polygon) and also to detect if the point lies on a vertex it must compare if the coordinates of the point matches one of the vertices coordinates. This procedure needs O(n) in time.
R-tree search method is proposed by Zhou et al. [20] in 2013. This method is based on classifying each point in a tree-like hierarchy then search for the point in this tree. The main disadvantage is that the polygon must be represented in the R-tree indices, which creates limitations on drawing arbitrary polygons. Another drawback is that it has to be evaluated to each single point in the domain, which means huge execution time especially if the domain contains many (thousands) points.
Suwardi et al. [21], in 2015, proposed an algorithm to deal with a polygon based on the Boolean Overlay. This method is based on counting all points in the polygon and the point of interest, then classify them using colour coding. Finally, the output is obtained by visual detection of the colour of the output figure/map. Later, Khatun and Sharma [22] modified this algorithm to include the condition of point-on-boundary by adding one more If condition to check if the point lies on the body. This algorithm needs prepossessing to detect the shape and position of the point and the polygon, which can be extracted from the Geographic Information System (GIS), but this limits the application of this method.
Using Ray Intersection method, Zhang, and Wang [23], in 2017, presented an algorithm to solve the PIP problem in 3D spherical coordinates directly instead of transforming the spherical polygon into set of planar ones by projections which is time consuming process.

3. The proposed algorithm: The Modified Polygon Method

3.1 Basic Concept

The concept of this method is evolved from the way of detection, if a point divides a line internally or externally; if the point divides the line internally, the total distance between the point and the line endings is the same as the line length. In case that the point divides the line externally, the total distance of the line passing from the line endings passing through the point will be longer than the initial length.
By applying the same concept between an arbitrary point and a side of a polygon in 2D space, the area shall be calculated instead of the length. The line passing from the start point of the side to the end point passing through the point will have longer length except if the point lies over the line. Also, the side will be split into two; one from the start point to the point and from the point to the end point. This new deformation leads to a change in the polygon initial area. In case the point lies outside the polygon, the area of the modified polygon will increase and vice versa. In case that the point lies over the polygon circumference the area will be the same. Hence, by comparing the areas of the initial and the modified polygon, position of the point with respect to the polygon can be determined exactly regardless of the shape of the polygon or any other constraint. The area of the polygon can be calculated as:

$$A = \frac{1}{2}\left|\sum_{i=1}^{N} x_1 * y_{1+1} - y_1 * x_{i+1}\right| \tag{1}$$

DOI: 10.13009/EUCASS2019-490

Where x_i and y_i are the coordinates of the vertices. Points x_{n+1} and y_{n+1} are evaluated as x_1 and y_1. The absolute value is taken in order to cancel the effect of the direction around the polygon i.e. CW or CCW. The factor ½ can be omitted for faster calculations because only the difference in the area is considered for the decision, not the value itself.
To determine which line is to be connected to the point, a simple search algorithm is used. The first step is to measure the distance between the point P and all the polygon vertices. The nearest vertex A is one of the two ends of the line that will be modified to include the point. The next step is to choose which line to be modified AB or AC. This choice is arbitrary except under one condition: if the line connecting between point and the other end (line PB) will intersect the other polygon side (line AC) as shown in Figures 1 and 2. Hence, an intersection check must be done. The time required for this process is O(n). The flow chart of the proposed algorithm is illustrated in Figure 3 and the pseudo code is in Appendix.

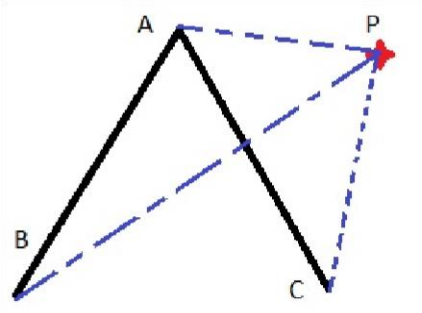

Figure 1 intersection between lines PB and AC.

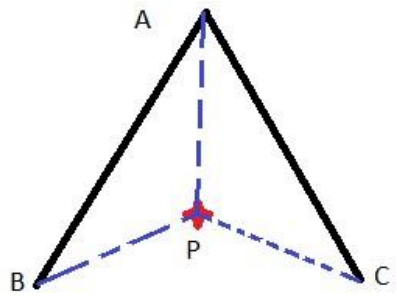

Figure 2 no intersection between lines PB and AC.

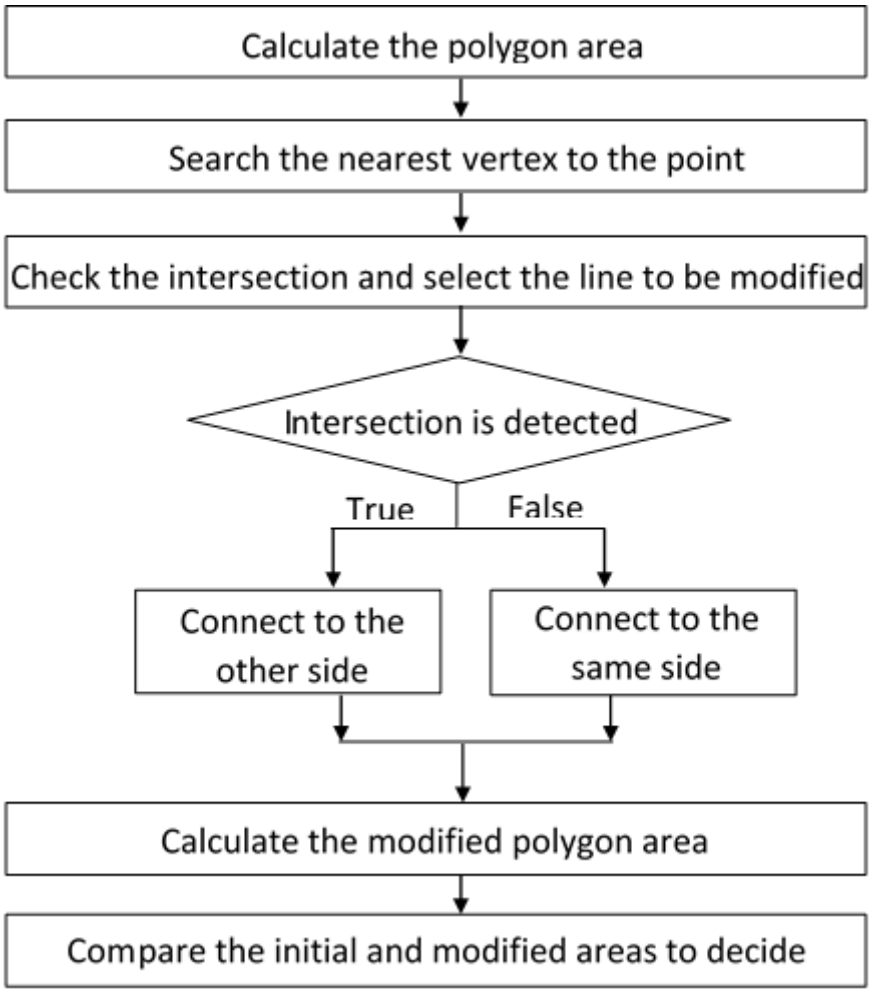

Figure 3 Flow chart of the proposed algorithm.

DOI: 10.13009/EUCASS2019-490

3.2 Comparison between the Proposed "Modified Polygon" and "Area" Methods

To stand on the differences between the proposed "Modified Polygon" method and the area method [10], one shall compare between both. The Area method is based on connecting the investigated point to all the polygon sides to create triangles. Then to sum the areas of each triangle and compare the new and initial areas. If the point lies inside the shape, the total area will be the same, while the area will increase if the point lies outside. The main drawback is that if the shape is convex (e.g. star), when connecting the triangles, they may overlap and even intersect with the polygon itself, which generates fictitious and self-intersecting areas and the method will lose its functionality.
In the Modified Polygon method, the point will be included in the polygon (i.e., the polygon will be deformed to pass through the point) and then the whole area of this modified polygon will be evaluated directly. By comparing the initial and modified areas, one of three possibilities will happen: 1) the new area is bigger which means that the point is outside the polygon, 2) the new area is smaller which means that the point is inside the polygon, and 3) the area is the same which means that the point is exactly over the polygon. The fictitious areas and the self-intersections can't happen here because of the check that is done in step 4 in the flow chart (Figure 3). One more advantage for this algorithm over the Area Method and most of the other algorithms is that it can determine if the point lies over the polygon.

4. Evaluation of the Proposed Method

4.1 Comparison with Different Methods

To evaluate the proposed method, challenging test cases are to be investigated by this method and several ones. The five-pointed star, as shown in Figure 4, is one of the most challenging problems and is introduced as a test case since it contains both convex and concave regions/angles. The comparison is made with three other methods which are a) Ray Intersection [10], b) Sum of Angles [16], and c) Algorithm number 6 of [19] as it is relatively fast. The calculation time is reported in Table 1. The average time is calculated from repeating the calculations 200 times and get the elapsed time for each trial. The reported calculations are obtained using MATLAB with an Intel I5 processor and 4 GB RAM.

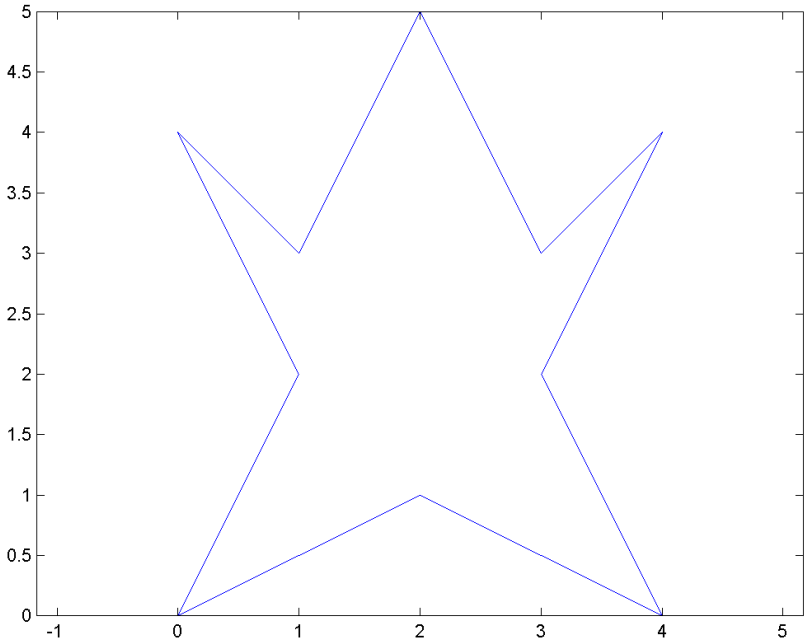

Figure 4 Generalized five-pointed star used for testing.

DOI: 10.13009/EUCASS2019-490

Table 1 results of computation time of the different methods

Method	Average time	Minimum time
Present method	3.69e-4	1.94e-4
Ray Crossing	9.67e-4	6.11e-4
Sum of Angles	34.0e-4	25.0e-4
Algorithm 6 [19]	6.39e-4	3.10e-4

As shown in Table 1, the Angle Summation method is the most expensive method, followed by the Ray Crossing and Algorithm 6, and the proposed algorithm is the fastest. By analysing the methodology of each algorithm, it can be inferred that in Ray Crossing; an intersection check is occurred between a ray passing through the point and cutting the polygon. This needs to calculate the intersection n times. For Algorithm 6, it is faster because it mainly depends on logic operators, but to check if the point is on the polygon it needs to check all the vertices coordinates which needs O(n) in time. The key behind the speed of our algorithm is that it calculates the areas using equation (1) two times which is O(n) in time also, but no more processing is needed except a single check for line intersection which is O(1). to examine if the point lies over the circumference of the polygon, the other methods uses logic operators in O(n) in time while the proposed method uses single logic operator. Besides, this algorithm is the most accurate because it depends on the very basic definition of the polygon size.

4.2 Test Case with Complex Shape

To stand over the strength points of this method, a more challenging test case shown in Figure 4 is examined, which is usually used in solid propellant fuel grains. In case that the point inside the shape, area of the shape in black will decrease. As shown in Figures 5-7, the algorithm detected correctly that the point lies inside, outside, or over the polygon respectively. Note that there is some irregularity in the shape because the coordinates were determined using digitizer software. It shall be noted that Algorithm 6 fails to detect the point position relative to the polygon in the second case of Figure 5 because the shape outer sides surround the point.

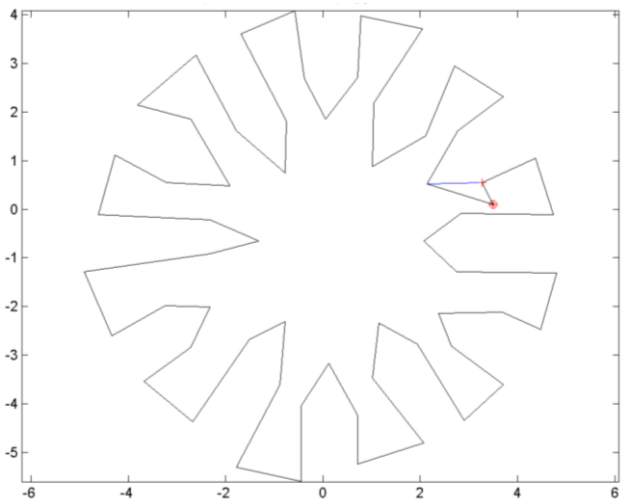

Figure 5 Fuel grain with a point lies inside.

DOI: 10.13009/EUCASS2019-490

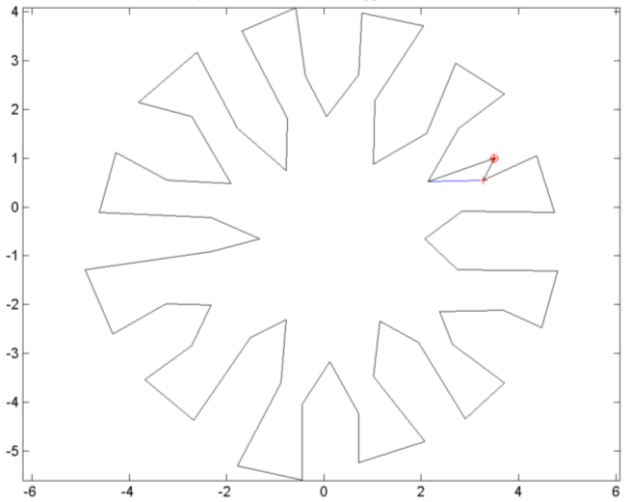

Figure 6 Fuel grain with a point lies outside.

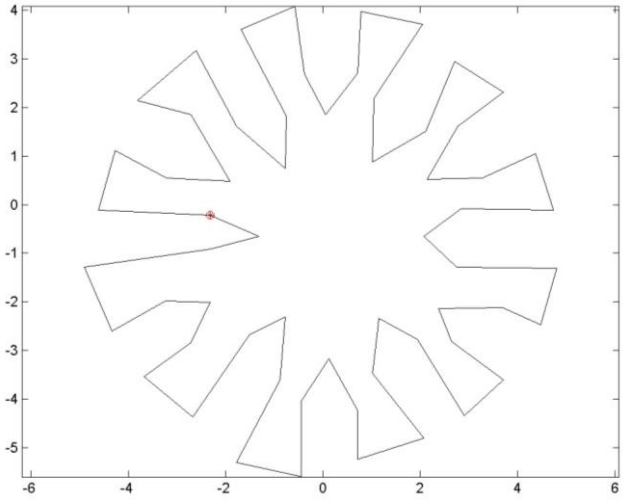

Figure 7 Fuel grain with a point lies over.

4.3 Linearity of the Method and Elapsed Time

In order to investigate the behaviour of the proposed method with the increase in the number of points, the method is tested on a square of N points, and the average and minimum time consumed are computed. The averaging is made over 2000 times and N varies from 100 to 10000. The results show this method has linear variation in time as shown in Figure 8.

DOI: 10.13009/EUCASS2019-490

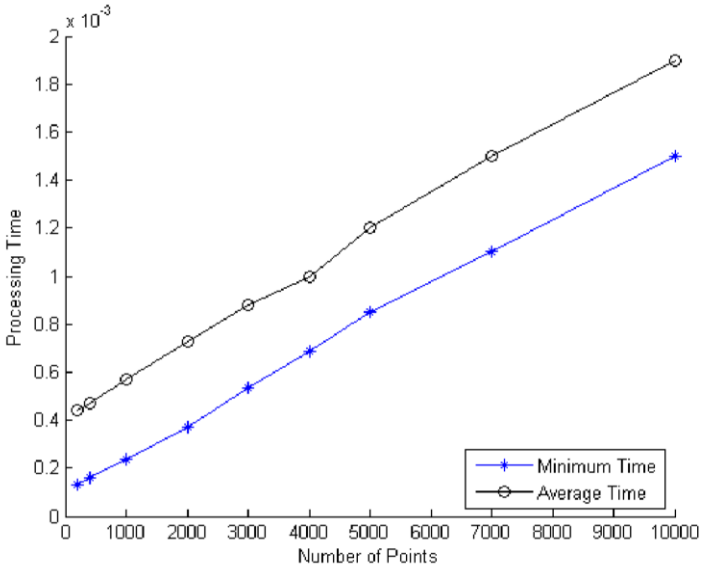

Figure 8 Elapsed time versus number of points.

4.4 Disadvantages

The current version of the algorithm may have a drawback in case of slender bodies. Consider a triangle ABC with relatively large obtuse angle B and a point P needs to be investigated lies downside angle B and the opposite side AC as shown in Figure 9. Using the proposed algorithm, the point will be connected to vertex B as it is the nearest one. Then the algorithm will insert the investigated point between AB or BC but in this particular case both choices are wrong simply because the line BP is already intersecting the original shape. Hence the correct choice is to connect P through AC. This problem can appear in any polygon with slender dimensions.

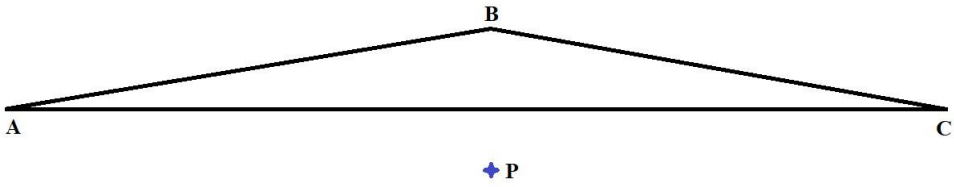

Figure 9 Line PB is self-intersecting with the initial shape.

Although this is only hypothetical and does not exist in CFD since the polygon sides are discretized over the mesh cells' boundaries. This issue can be handled by modifying the algorithm by checking the intersection of the new lines. This will be considered in a future work.

5. Conclusion

New algorithm is presented for the point-in-polygon problem. The idea behind the proposed technique is simply explained by considering the one dimensional version of the algorithm: consider a point divides a line; if the point divides the line internally, the total distance between the point and the line endings is the line length. In case it lies outside the line then the total distance of the line passing from the line endings passing through the point will be longer than the initial length. The proposed algorithm has many advantages, the most important of which are (1) it deals with convex/concave polygons. (2) It is faster when compared with pre-existing techniques. (3) It is based on the basic definition of the polygon size (line/area/volume) depending on the spatial dimensions of the shape under consideration. (4) Time consumption of this algorithm varies linearly with the number of vertices of the polygon. The algorithm is tested against different shapes and the results are compared with some of the existing techniques with a great degree of success showing the superiority of the proposed algorithm.

DOI: 10.13009/EUCASS2019-490

Appendix

Hereafter, the pseudo code of the Modified Polygon method is stated.

```
        {Calculation of the initial area using Equation (1)}
initial_area=abs(1/2*sum(x.*y([2:end,1])-y.*x([2:end,1])))
        {Looping over the polygon vertices to determine the nearest vertex}
for i=1: length (x)
   distance(i)=sqrt((check_point(1)-x(i))^2+(check_point(2)-y(i))^2)
end
[min_value pos_of_min_value]= minimum(distance)
nearest_vertex=[x(pos_of_min_value) y(pos_of_min_value)]
        {Check the new side if it intersects with the other side}
main_line=[x(pos_of_min_value) y(pos_of_min_value) x(pos_of_min_value+1) y(pos_of_min_value+1)]
added_line=[x(pos_of_min_value-1) y(pos_of_min_value-1) check_point]
out = is_intersecting(main_line,added_line)
if out= FALSE    {connect the point to this side}
    shape_x=[x(1:pos_of_min_value-1) check_point(1) x(pos_of_min_value:end)]
    shape_y=[y(1:pos_of_min_value-1) check_point(2) y(pos_of_min_value:end)]
else {connect the point to the other side}
    shape_x=[x(1:pos_of_min_value) check_point(1) x(pos_of_min_value+1:end)]
    shape_y=[y(1:pos_of_min_value) check_point(2) y(pos_of_min_value+1:end)]
end
        {Calculation of the modified area using Equation (1)}
modified_area=abs(1/2*sum(shape_x.*shape_y([2:end,1])-shape_y.*shape_x([2:end,1])))
        {Check the change in the area}
if modified_area= initial _area
    result=the point is located on the polygon
elseif modified_area> initial _area
    result= the point is located outside the polygon
else
    result= the point is located inside the polygon
end
```

References

[1] Osher, S., & Fedkiw, R. (2002). Level Set Methods and Dynamic Implicit Surfaces.

[2] Foley, J. D. (1995). Computer Graphics: Principles and Practice. Pearson Education

[3] Pobil, A. P. del., & Serna, M. A. (1995). Spatial Representation and Motion Planning. Vol. 1014, Springer Science & Business Media.

[4] Shirmohammadi, S., & Ferrero, A. (2014). Camera as the instrument: the rising trend of vision based measurement. IEEE Instrumentation & Measurement Magazine, 17(3), 41–47.

[5] Luo, P. F., Chao, Y. J., Sutton, M. A., & Peters, W. H. (1993). Accurate measurement of three-dimensional deformations in deformable and rigid bodies using computer vision. Experimental Mechanics, 33(2), 123–132.

[6] Johnson, A. E., Montgomery, J. F., & Matthies, L. H. (2005). Vision Guided Landing of an Autonomous Helicopter in Hazardous Terrain. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (pp. 3966–3971).

[7] Shang, J., & Shi, Z. (2007). Vision-based Runway Recognition for UAV Autonomous Landing. International Journal of Computer Science and Network Security 7 (3), 112-117.

[8] Xu, G., Zhang, Y., Ji, S., Cheng, Y., & Tian, Y. (2009). Research on computer vision-based for UAV autonomous landing on a ship. Pattern Recognition Letters, 30(6), 600–605.

[9] Nordbeck, S., & Rystedt, B. (1967). Computer cartography point in polygon programs. Bit Numerical Mathematics, 7(1), 39–64.

[10] Manber, U. (1989). Introduction to Algorithms: A Creative Approach. Vol. 4, Addison-Wesley Reading, MA.

[11] Taylor, G. (1994). Point in polygon test. Survey Review, 32(254), 479–484.

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

DOI: 10.13009/EUCASS2019-490

[13] Seidel, R. (2010). Reprint of: A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. Computational Geometry: Theory and Applications, 43(6), 556–564.
[14] Žalik, B., & Clapworthy, G. J. (1999). A universal trapezoidation algorithm for planar polygons. Computers & Graphics, 23(3), 353–363.
[15] Preparata, F. P., & Shamos, M. I. (1985). Convex Hulls: Basic Algorithms, 95–149.
[16] Žalik, B., & Kolingerova, I. (2001). A cell-based point-in-polygon algorithm suitable for large sets of points. Computers & Geosciences, 27(10), 1135–1145.
[17] Skala, V. (1996). Line clipping in E2 with O(1) processing complexity. Eurographics, 20(4), 523–530.
[18] Huang, C.-W., & Shih, T.-Y. (1997). On the complexity of point-in-polygon algorithms. Computers & Geosciences, 23(1), 109–118.
[19] Hormann, K., & Agathos, A. (2001). The point in polygon problem for arbitrary polygons. Computational Geometry: Theory and Applications, 20(3), 131–144.
[20] Zhou, T., Wei, H., Zhang, H., Wang, Y., Zhu, Y., Guan, H., & Chen, H. (2013). Point-polygon topological relationship query using hierarchical indices. In Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (pp. 572–575).
[21] Suwardi, I. S., Lestari, D. P., & Satya, D. P. (2015). Handling arbitrary polygon query based on the boolean overlay on a geographical information system. In 2015 2nd International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA) (pp. 1–4).
[22] Khatun, F., & Sharma, P. (2016). Arbitrary Polygon Query Handling Algorithm on GIS based on three Value Logic- An Approach. International Journal of Computer Applications, 144(1), 32–35.
[23] Li, J., Zhang, H., & Wang, W. (2017). Fast and Robust Point-in-Spherical-Polygon Tests Using Multilevel Spherical Grids. International Workshop on Next Generation Computer Animation Techniques, 56–66.