

Smart-geofencing for system of reporting inadequate regional infrastructure using crossing and winding number

Puspa Miladin Nuraida Safitri A. Basid, Fresy Nugroho

Department of Computer Science, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia

Article Info

Article history:

Received Nov 7, 2021

Revised Mar 28, 2022

Accepted Apr 14, 2022

Keywords:

Crossing number

Geofencing

Raycasting

Volunteered-geographic-information

Winding number

ABSTRACT

Infrastructure is a major factor in increasing regional productivity and become the center of economic growth. Problems arise when infrastructure improvements are only centered on downtown areas. One of the ways to develop regional infrastructure is with media reporting inadequate infrastructure. The use of smartphone technology and geofencing techniques are considered to be helpful in reporting media. Increasingly sophisticated built-in features in smartphones are undeniable proof of the technology's highly impressive advancement. Mobile applications utilizing location-based services (LBS) assisted by a global positioning systems (GPS) sensor are rapidly becoming popular. One of its most-favored features is geofencing. The volunteered-geographic-information (VGI)-based reporting system for regional infrastructure damage uses this feature to obtain the data input directly from the users. In this current research, the author uses two methods in geofencing, including crossing number (CN) and winding number (WN). Several previous studies suggest that the combination of these methods might result in faster computation, compared to the raycasting method. By using winding and crossing number methods, 98% of increased accuracy can be obtained, while a prior method could only obtain 96%. An improvement in the system's speed is also expected.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Puspa Miladin Nuraida Safitri A. Basid

Department of Computer Science, Universitas Islam Negeri Maulana Malik Ibrahim

Gajayana Street Kota Malang, Jawa Timur 65144, Indonesia

Email: puspa.miladin@uin-malang.ac.id

1. INTRODUCTION

Infrastructure is a facility that is developed and needed for government functions in the provision of water services, electricity, sewage treatment, transportation, and services that have social and economic objectives [1]. The activities of the transportation service sector are the main key in distribution activities, both people and goods. In addition, the activity of other infrastructure is one of the important factors in increasing the productivity of the population of an area. So, it can be said that infrastructure is the center of economic growth. Several regions have begun to improve the infrastructure of the area. However, problems arise when infrastructure improvements are only centered on downtown areas. Suburban areas often have less decent infrastructure than other areas. This leads to an infrastructure gap. Even though the improvement and development of infrastructure in suburban areas will have many positive impacts. Starting from the transportation sector with the freedom of isolated areas to the agricultural sector with the latest agricultural technology developments. One of the efforts in equitable distribution of improvement and development of regional infrastructure is to provide reporting media related to inadequate regional infrastructure. The author took the initiative to raise the topic of regional infrastructure improvement reporting systems using geofencing techniques on mobile devices.

Almost nothing is debatable about the rapid development of mobile applications. The advancement of smartphone technology also contributes to this, especially the unceasing innovations of built-in smartphone features. Additionally, people's eagerness to create an application that utilizes original, built-in smartphone features is creditable. Some of these aforementioned features are sensors, cameras, gyroscope, accelerometer, geolocation, and others [2]. One of the most-used features in mobile applications is geolocation. Many mobile applications utilize location-based services (LBS) [3]. They operate by providing some information of a location to a mobile device and later following the instructions given based on the user's location. In addition to using geolocation, some mobile applications add geofencing to their features [4].

Geofencing itself is a technology designed to monitor a mobile device by obtaining coordinates that will be transmitted to the control center. The application will manage a series of geographic coordinates to form what is called a geofence [5]–[8]. It can be designed to perform dynamically, perhaps forming a circle surrounding the mobile device in service or presenting a pre-determined set of borderlines, which will help the user to freely characterize a specific spot or building [5]. A previous study explores geofencing using the raycasting method, and the result suggests a 94% of accuracy level. This method operates by forming an imaginary straight line from the current point to the outside point and calculating the total number of crossings with the virtual fence, to detect the location of the coordinates.

This feature shall offer several benefits when installed into a mobile application. One of the applications utilizing it is the reporting system for regional infrastructure damage [9]. This application is volunteered-geographic-information (VGI)-based, meaning that the data input regarding the infrastructure damage is sent directly by members of the community. One of the obstacles in running a VGI-based (volunteered geographic information) application is the massive amount of data received. Moreover, some of the data might not be useable, let alone processable. This is because the data input might be generated from an area outside of the one under monitoring [10], [11]. To minimize the possibility of retrieving unusable data, geofencing is thus needed. This technology will form a virtual fence surrounding the location under monitoring, with the assistance of several computation methods to ensure that it can determine if a data input regarding the infrastructure damage is originally generated from the area under the monitoring [12].

A previous study explores geofencing using the raycasting method, and the result suggests a 94% of accuracy level [9]. This method operates by creating an imaginary straight line from the current point to the outside point and calculating the total number of crossings with the virtual fence, to detect the location of the coordinates. However, in this current research, the author uses two frequently-used methods to find out if a point is within the polygon, including crossing number (CN) and winding number (WN) [11]. Several other studies have pointed out that the computation rate resulted from the combination of the two methods is considerably faster than the raycasting one [13]. Therefore, the author considers the need to improve the geofence techniques for the reporting system for regional infrastructure damage. The testing parameters used include the accuracy of and time spent by the system in the process of identifying a user's location.

This current research aims to assess the performance of crossing number and winding number methods in identifying the coordinates in certain areas circumscribed by geofence, in comparison with prior methods. The testing parameters used include the accuracy of and time spent by the system in the process of identifying a user's location. The expected final result would be the establishment of an information system for reporting infrastructure damage with an improved accuracy level. It sets out to provide accurate information about the location of the reported infrastructure. This shall help the government to address the problem immediately.

2. LITERATURE REVIEW

There have been several studies exploring the same topic as this current research. A study by Basid *et al.* [13] became the kickoff in the development of the community complaint system which utilizes geotagging [7]. It explored the method of adding geographic information to the images of complaints sent by members of the community. It aimed to discover the geographic location of an image. Basid and Fadila [9] utilized geofencing to circumscribe the location of users who can report to the infrastructure damage complaint system. The algorithm used the raycasting technique. It resulted in 94% of accuracy, with 50 random points featured in the testing stage [4]. In this current research, the algorithm methods explored include crossing number and winding number, paired with the geofencing technique. It focuses on increasing the system's speed and accuracy in determining the location, using considerably better methods. The algorithm used in several previous studies is the raycasting method, which in this research is compared to crossing number and winding number. The system is tested using a new algorithm, and the analysis is expected to indicate which algorithm is better. The parameters used in this comparative analysis include the computational rate and the accuracy level of the system in determining geographic locations.

2.1. Geofencing

Geofencing is the technique of circumscribing an area using a series of coordinates as a virtual fence. Several mobile device applications utilize location-based services equipped with this technique. Additionally, geofencing is used as a medium for monitoring moving objects, assisted by global positioning system (GPS) sensors. Currently, the technique is explored to serve many different purposes, including [5]:

- Proximity with point of interest (POI): Used for areas with a circular virtual fence. It operates by detecting how close a moving object is to the coordinates of the destination, or POI. The only variables used are the coordinates of the destination, or the point of interest, and the distance of the object to the POI, or the radius.
- Route Adherence: Used for fast-moving objects, such as cars. It is useful to determine the route to a destination that a vehicle should take and inform the user if their monitored vehicle is on the specified route. This route is stored in the form of a series of coordinates cached in the device. They will form a circle to mark the geofenced area.
- Route and schedule adherence: In its application, it is somewhat similar to the route adherence function, except for the addition of a time variable. This shall operate by setting the time for each vehicle to arrive at the coordinates along the route. Later, the system will determine whether it arrives at the coordinates according to the pre-determined schedule [14], [15].
- Geofenced area: This technique is useful for monitoring a moving object in a virtually fenced area. The virtual fence will form a simple geo-matrix, like a polygon. This should serve as a 2-dimension representation of an area, characterizing cities, water areas, and administrative areas [16], [17].

2.2. Crossing number

The crossing number algorithm, also known as the even-odd algorithm, basically uses an infinite projection of the point in question and decides how many edges of the polygon are circumscribed by the projection [18]. This method determines how many times a projection (ray) from point P crossing the edge of the polygon boundary. When the number of crossings (CN) is odd, then point P is considered inside. Meanwhile, if the number of crossings is even, then the point P is considered outside the polygon [5].

Every time a projection crosses a line of the polygon, the status changes from (in) to (out), or vice versa. When a point is inside the polygon, the order of crossings would look like this: in > out > ... > in > out. Meanwhile, when it is outside the polygon, the order would be as follows: out > in > ... > in > out [11]. These orders are illustrated in Figure 1.

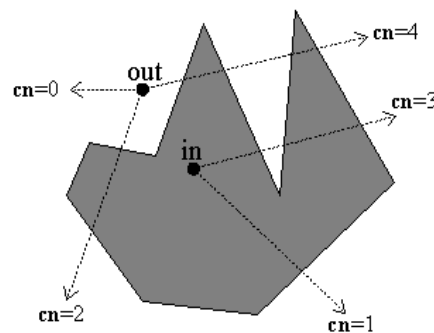


Figure 1. Projection of crossing number

The crossing number algorithm directly selects a horizontal projection extending to the right of P and is parallel to the positive x-axis. It would be easier to determine the crossings using this projection. It would be even easier to predict if such crossing is unlikely. To calculate the total number of crossings, CN, the algorithm simply loops through all the edges of the polygon, tests each crossing, and adds CN when a crossing occurs. In addition, the crossing testing must handle the special cases and points on the edges. This can be achieved by adhering to the following rules:

- a. The final endpoint at the top edge is ignored and the initial endpoint is admitted
- b. The initial endpoint at the bottom edge is ignored, and the final endpoint is admitted
- c. Horizontal edges are ignored
- d. Crossing between the edges and projections must be positioned to the right of point P [11]

2.3. Winding number

The number of windings (WN) method is employed to accurately determine whether a point is inside a closed complex polygon. It determines the number of times the polygon circles a point. The winding number (WN) is defined as the number of times polygon 'S' circles point 'P'. If 'P' is inside polygon 'S', then WN is not zero. Vice versa, if point 'P' is outside, only when the polygon does not circle the vertex at all that WN = 0. Not only does the WN of polygon 'S' with an arbitrary point 'P' measure how much 'S' surrounds point 'P', it also provides information about orientation and determines the number of times polygon 'S' winds around point 'P' [19].

$$wn = \begin{cases} 0; & \text{if } P \text{ is outside } S \\ n > 0; & \text{If } S \text{ winds around } P, \text{ clock - wise} \\ n < 0; & \text{If } S \text{ winds around } P \text{ counter clock - wise} \end{cases} \quad (1)$$

Generally, the winding number WN (P, C) can be determined based on each closed continuous curve C around point P on a 2D area. But the formula is inefficient as it still uses costly computational trigonometry. An inelaborate observation would suggest that a more efficient formula is necessary. Pick a point Q on S1. Then, when curve W (P) surrounds S1, it will cross Q a few times. If it crosses Q in a counter clock-wise direction, then it is (+1), and (-1) if it does in a clockwise direction. Thus, the accumulated number would be the exact total of W (P) surrounding S1, and equal to the winding number WN (P, C). Moreover, if the projection of infinite R is measured from P to vector Q, then the crossing where R cuts the curve C matches the point where W (P) crosses Q. If the edge crosses the positive projection upward, the crossing is considered positive (+1); however, if it crosses downward, the crossing would be considered negative (-1). The last step is cumulating all the crossings values to obtain WN (P, C). The calculation is illustrated in Figure 2 [5].

The determination of the edge crossing points with the actual projection can be avoided using the attribute is left (). However, this attribute needs to be applied differently for upward and do WN ward edges. If the top edge crosses the projection to the right of P, then P would be positioned to the left of the edge because the orientation of triangle Vi V (i + 1) P is counter clock-wise. On the other hand, if the bottom edge crosses the positive projection, then P would be positioned to the right as the orientation of triangle Vi V (i + 1) P is clock-wise [11]. The obtaining of the attribute is Left () is illustrated in Figure 3.

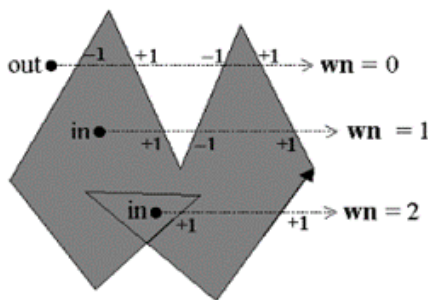


Figure 2. Calculation of winding number [20]

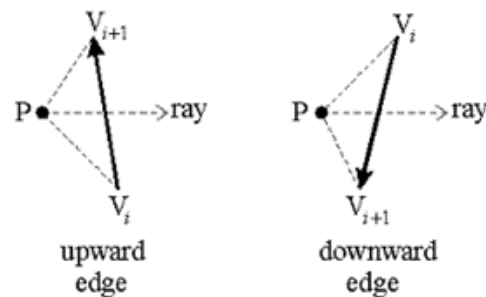


Figure 3. The calculation of the attribute is left ()

3. PROPOSED MODEL

Figure 4 presents the architectural design of the reporting system for the improvement and development of regional infrastructure. Users can submit pictures (taken directly using their built-in mobile camera or selected from the phone's gallery) through the system on their Android mobile devices [21]. After that, the process of obtaining the coordinates utilizes the user's mobile GPS [22].

The next step would be to determine the position of the coordinates using the geofencing technique, paired with the raycasting method. After identifying the location details, the next step would be to send the data to the server. The data will then be stored in the database. Next, information in the form of reports of regional infrastructure repairment and development will appear on the web admin managed by stakeholders (the government in charge) [23]. The main objective of this entire process is to provoke immediate actions following the reports.

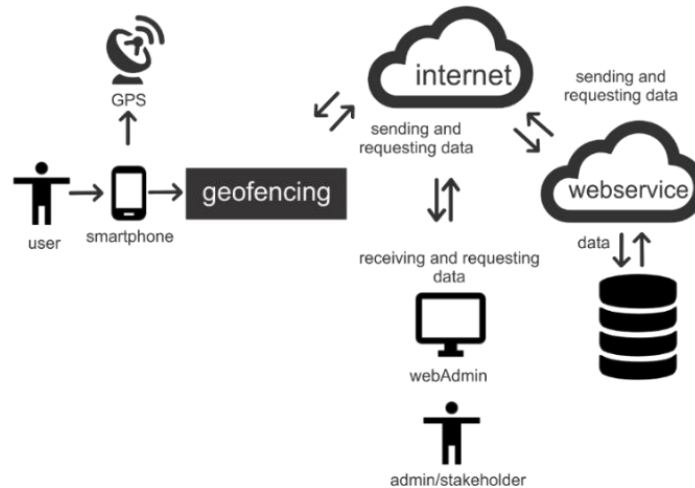


Figure 4. Architecture of the system

4. IMPLEMENTATION

4.1. Build polygon

One can assemble polygons as virtual fences on digital maps by collecting coordinates as points [18]. When a set of coordinate points are connected in a certain order, it becomes a polygon feature. Regional boundaries are not set geographically, usually form polygons [19]. In the formation of a polygon area, the first and last coordinate points must indicate the same value. As for other coordinates, they must be unique, in the sense that there cannot be two exact coordinates in a polygon [16]. The crucial initial step would be determining the coordinates to be formed into a polygon. These coordinates are obtained from the outer boundary of an area. After obtaining them, the next step would be initializing the coordinates. The coordinates are stored in a point ArrayList [24].

Validating the number of coordinates stored in the ArrayList is then required. If the number of coordinates is less than 3, it will not form a polygon; it must have at least 3 forming points. If the validation process results in more than 3 coordinates, the next step can be carried out, which is connecting the stored coordinates to the index sequence in the ArrayList. By doing this, a polygon will be formed as a virtual boundary of a geofenced area. The illustration of the block diagram is presented in Figure 5. In Figures 6 and Figure 7 are source code to build polygons.

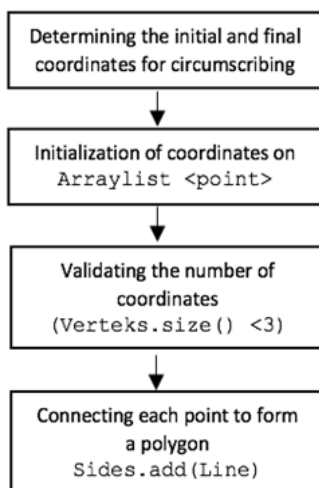


Figure 5. Flow of polygon formation

```

1  batu = Polygon.Builder()
2  .addVertex(new Point(-8.046294f, 112.641049f))
3  .addVertex(new Point(-8.041875f, 112.632466f))
4  .addVertex(new Point(-8.035454f, 112.630828f))
5  .addVertex(new Point(-8.020967f, 112.613303f))
6  .addVertex(new Point(-8.011278f, 112.618733f))
7  .addVertex(new Point(-8.009899f, 112.616644f))
8  .addVertex(new Point(-8.019097f, 112.608434f))
9  .addVertex(new Point(-8.011618f, 112.606546f))
10 .addVertex(new Point(-8.006179f, 112.596761f))
11 .addVertex(new Point(-7.992577f, 112.601335f))
12 .addVertex(new Point(-7.991599f, 112.600066f))
13 .addVertex(new Point(-7.996145f, 112.598523f))
14 .addVertex(new Point(-7.992299f, 112.589010f))
15 .addVertex(new Point(-7.986202f, 112.591158f))
16 .addVertex(new Point(-7.987073f, 112.581370f))
17 .addVertex(new Point(-7.984929f, 112.578070f))
18 .addVertex(new Point(-7.980105f, 112.581800f))
19 .addVertex(new Point(-7.984516f, 112.597951f))
20 .addVertex(new Point(-7.979662f, 112.598104f))
21 .addVertex(new Point(-7.975161f, 112.589540f))
22 .addVertex(new Point(-7.972859f, 112.590991f))
23 .addVertex(new Point(-7.971322f, 112.602829f))
24 .addVertex(new Point(-7.954791f, 112.589971f))
25  .build();
26 }
    
```

Figure 6. Source code for building polygons

```

1 private void validate()
2 {
3     if ( _vertexes.size() < 3)
4     {
5         throw new RuntimeException("Polygon must have at least 3
6 points");
7     }
8 }
10 public Polygon build()
11 {
12     validate();
13
14     // in case you forgot to close
15     if (!_isClosed)
16     {
17         // add last Line
18         _sides.add(new Line( _vertexes.get( _vertexes.size() - 1),
19 _vertexes.get(0)));
20     }
21
22     Polygon polygon = new Polygon( _sides, _boundingBox);
23     return polygon;
24 }

```

Figure 7. Source code for connect each coordinate

4.2. Determining the coordinate positions using the crossing number method

At this stage, the points (latitude and longitude) entered by the user will be checked in order to decide whether it is included in the polygon in the database. The crossing number method calculates the number of crossings (CN) of the projection from the initial point towards the edge of the polygon. The calculation of the crossing number in this study employs the following algorithm [25]. Figure 8 shows the workflow of the crossing number method, and Figure 9 shows the pseudocode for its implementation.

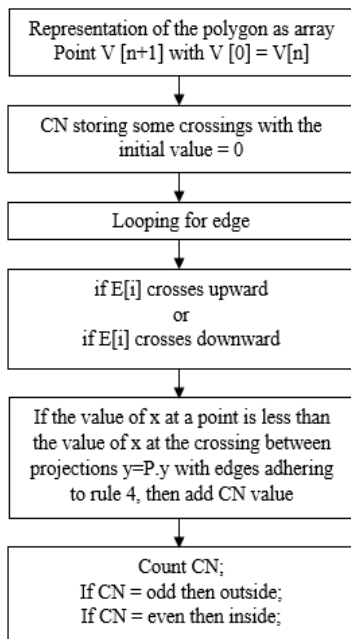


Figure 8. Flow of position calculation using the crossing number method

```

typedef struct {int x, y;} Point;
cn_PnPoly( Point P, Point V[], int n )
{
    int    cn = 0;    // the crossing number counter

    // loop through all edges of the polygon
    for (each edge E[i]:V[i]V[i+1] of the polygon) {
        if (E[i] crosses upward ala Rule #1
            || E[i] crosses downward ala Rule #2) {
            if (P.x < x_intersect of E[i] with y=P.y) // Rule #4
                ++cn; // a valid crossing to the right of P.x
        }
    }
    return (cn&1); // 0 if even (out), and 1 if odd (in)
}

```

Figure 9. Pseudocode crossing number

4.3. Determining the positions of coordinates using the winding number method

The winding number method determines whether the point entered by the user is included in the polygon by calculating the number of times the polygon's line winds the points, labeled as WN. The winding number calculation in this study follows the algorithm Figure 10 [11]. This method adapts the previous crossing number algorithm, replacing the step of cumulating the number of crossings with cumulating if the crossing is to the right, and subtracting if the crossing is to the left.

The algorithm in Figure 10 adapts the previous crossing number method, replacing the step of cumulating the number of crossings with cumulating if the crossing is to the right, and subtracting if the crossing is to the left. Figure 11 shows the pseudocode for its implementation.

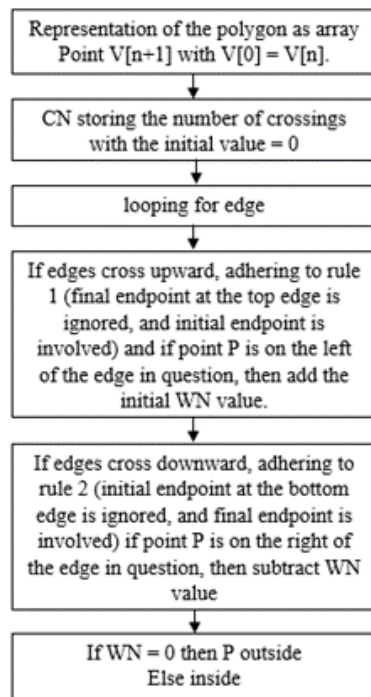


Figure 10. Flow of determining the positions using the winding number method

```

typedef struct (int x, y;) Point;
wn_PnPoly( Point P, Point V[], int n )
{
    int    wn = 0;    // the winding number counter

    // loop through all edges of the polygon
    for (each edge E[i]:V[i]V[i+1] of the polygon) {
        if (E[i] crosses upward ala Rule #1) {
            if (P is strictly left of E[i])    // Rule #4
                ++wn;    // a valid up intersect right of P.x
        }
        else
            if (E[i] crosses downward ala Rule #2) {
                if (P is strictly right of E[i])    // Rule #4
                    --wn;    // a valid down intersect right of P.x
            }
    }
    return wn;    // =0 <=> P is outside the polygon
}
  
```

Figure 11. Pseudocode winding number

5. EXPERIMENT RESULT

These methods are practically easy to carry out. However, considering the demand for efficient computation, the comparison of their outcomes becomes crucial. In practice, these two methods are easy to apply. This stage should result in the data which will later be processed to obtain the best accuracy level. The data required include the accuracy level and the time spent identifying the user's location using the geofencing technique, paired with either raycasting or winding number methods. It can be obtained by providing loads of inputs to the system, as a means of testing it. The flow of the testing is as follows:

- Determining the value of n , or the amount of testing data, the pre-established polygons, and the value between the circumscribing of testing distance in the pre-established polygons
- Determining random coordinates (x, y) equal to n in a pre-determined circumscribed area
- Conducting looping for all of the coordinate points
- Storing the testing result
- Determining the coordinates of positions inside and outside the polygons using the winding number method
- Storing the testing result

The testing incorporated 100 random coordinates, shown in Table 1 its longitude and latitude data. These coordinates were approximately located within a 100 m distance from the center point of the pre-established polygon. It was conducted with the assistance of a mobile-based system, considering the excellent qualification of the mobile devices selected. Figure 12 shows the interface of the testing process on a smartphone. Referring to the main objective, the testing instruments included the accuracy level and the system's speed [26]. The actual value was determined by comparing the testing result to the result of direct observation with the assistance of google maps visualization. Then this value was compared to the results of previous studies that explored the raycasting method. The following is the formula to obtain the accuracy level and the system's speed. Table 2 shows the results of comparison with other methods.

$$accuracy = \frac{total\ correct\ results}{total\ attempts} \times 100\% \tag{2}$$

$$speed = \frac{total\ time\ spent}{total\ data} \tag{3}$$

Test results with raycasting [9].

$$accuracy = \frac{96}{100} \times 100\% = 96 \tag{4}$$

$$speed = \frac{100}{55,1\ ms} \tag{5}$$

Test results with winning dan crossing number.

$$accuracy = \frac{100}{100} \times 100\% = 98\% \tag{6}$$

$$speed = \frac{100}{30\ ms} \tag{7}$$

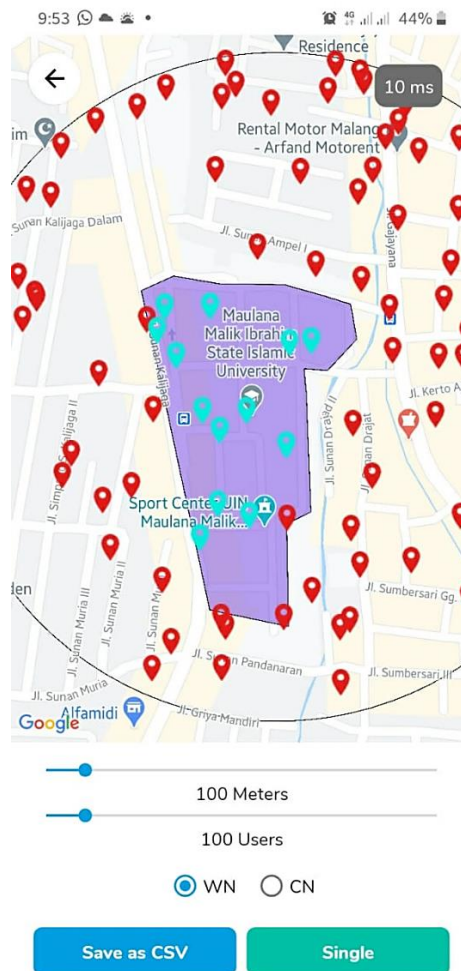


Figure 12. Interface of testing media

Table 1. Sample of test coordinate

No	Lat	Long	Inside	Validity
1	-7.952.441.865.001.140	11.260.782.520.757.100	TRUE	Valid
2	-7.951.913.090.868.190	11.260.684.648.072.500	TRUE	Valid
3	-79.540.287.038.762	11.260.942.267.979.900	FALSE	Valid
4	-7.953.068.364.315.740	11.260.532.688.716.200	FALSE	Valid
5	-79.533.425.987.745.000	11.260.763.642.323.000	TRUE	Valid
6	-7.952.922.340.608.740	11.260.552.401.807.300	FALSE	Valid
7	-7.952.153.450.694.780	11.260.927.794.382.700	TRUE	Valid
8	-7.954.608.166.636.670	11.260.802.412.478.600	FALSE	Valid
9	-79.512.084.225.200.800	11.260.997.326.479.400	FALSE	Valid
10	-79.506.493.173.990.800	11.260.945.453.610.200	FALSE	Valid
11	-79.530.164.624.474.800	11.260.956.130.405.400	FALSE	Valid
12	-7.950.211.686.420.850	11.260.638.953.209.000	FALSE	Valid
13	-7.951.527.926.481.880	11.260.681.961.876.400	TRUE	Valid
14	-7.954.578.062.513.910	11.260.841.639.930.600	FALSE	Valid
15	-7.952.324.775.311.380	1.126.067.857.256.270	TRUE	Valid

Table 2. Method comparison

No	Method	Accuration	Time speed
1.	Raycasting	96%	100/55.1 ms
2.	Winning dan crossing number	98%	100/30 ms

6. CONCLUSION

The aforementioned test result, with an equal amount of data, indicated the described accuracy level and the system's speed. The accuracy level increased due to the utilization of winding and crossing number methods, which was 98% from the 96% suggested by the previous method. Then the comparison of the system's speed indicated an increase from the previous method. The author does not claim that the current result was perfect. It depends on future researchers if they would like to use the techniques for a testing-case study. In conclusion, the current result with 100 random coordinates within a radius of 100 meters from the center point indicated that the winning number method excelled in improving accuracy level and the system's speed.





REFERENCES

- [1] D. Stone and A. Stone, "The administration of chairs," *Public Administration Review*, vol. 34, no. 1, p. 71, 1974, doi: 10.2307/974403.
- [2] M. King and O'Donnell, "Voice of the next-generation mobile developer," 2013. <http://de.slideshare.net/Ikusmer/voice-of-the-nextgeneration-mobile-developer> (accessed Jun. 15, 2021).
- [3] P. Deshmukh, A. Bhajibhakre, S. Gambhire, A. Channe, and N. Deshpande, "Survey of geofencing algorithms," *International Journal of Computer science engineering Techniques*, vol. 3, no. 2, pp. 1–5, 2018.
- [4] A. M. Talib and M. N. Jasim, "Geolocation based air pollution mobile monitoring system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 1, p. 162, Jul. 2021, doi: 10.11591/ijeecs.v23.i1.pp162-170.
- [5] F. Reclus and K. Drouard, "Geofencing for fleet & freight management," *2009 9th International Conference on Intelligent Transport Systems Telecommunications, ITST 2009*, pp. 353–356, 2009, doi: 10.1109/ITST.2009.5399328.
- [6] A. Suyama and U. Inoue, "Using geofencing for a disaster information system," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, Jun. 2016, pp. 1–5, doi: 10.1109/ICIS.2016.7550849.
- [7] P. Szczytowski, "Geo-fencing based disaster management service," in *Communications in Computer and Information Science*, vol. 498, 2015, pp. 11–21, doi: 10.1007/978-3-662-46241-6_2.
- [8] S. Schlosser, D. Toninelli, and M. Cameletti, "Comparing methods to collect and geolocate tweets in great britain," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 7, no. 1, p. 44, Jan. 2021, doi: 10.3390/joitmc7010044.
- [9] P. M. N. S. A. Basid and J. N. Fadila, "System for reporting inadequate regional infrastructure using raycasting-based geofencing technique on mobile devices," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, Oct. 2019, pp. 196–199, doi: 10.1109/ICECOS47637.2019.8984576.
- [10] G. P. O. Reddy, "Geographic information system: principles and applications," in *Geospatial Technologies in Land Resources Mapping, Monitoring and Management. Geotechnologies and the Environment*, Springer, 2018, pp. 45–62, doi: 10.1007/978-3-319-78711-4_3.
- [11] D. Sunday, "P Practical Geometry Algorithms: With C++ Code," 2021.
- [12] M. N. Stevens, H. Rastgoftar, and E. M. Atkins, "Specification and evaluation of geofence boundary violation detection algorithms," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2017, pp. 1588–1596, doi: 10.1109/ICUAS.2017.7991472.
- [13] P. M. N. S. A. Basid, H. Tolle, and F. Ramdani, "Designing module e-complaint system based on geotagging and geofencing," *International Journal of Interactive Mobile Technologies (IJIM)*, vol. 11, no. 3, p. 113, Apr. 2017, doi: 10.3991/ijim.v11i3.6557.
- [14] Z. Ozdemir and B. Tugrul, "Geofencing on the Real-Time GPS Tracking System and Improving GPS Accuracy with Moving Average, Kalman Filter and Logistic Regression Analysis," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2019, pp. 1–6, doi: 10.1109/ISMSIT.2019.8932766.





- [15] N. S. Sehwan, S. R. Sangle, and Y. N. Vadhavkar, "Real time automobile tracking system with an automated security algorithm," in *2017 International Conference on Communication and Signal Processing (ICCSPP)*, Apr. 2017, vol. 2018-Janua, pp. 2125–2129, doi: 10.1109/ICCSPP.2017.8286781.
- [16] GISGeography, "Vector vs raster: What's the difference between GIS spatial data types?," *Gisgeography.Com*, 2021. <https://gisgeography.com/spatial-data-types-vector-raster/> (accessed Jun. 15, 2021).
- [17] S. Anand, A. Johnson, P. Mathikshara, and K. R., "Low power real time gps tracking enabled with RTOS and serverless architecture," in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, Feb. 2019, pp. 618–623, doi: 10.1109/CCOMS.2019.8821738.
- [18] V. Carchiolo, M. P. Loria, M. Malgeri, P. W. Modica, and M. Toja, "An adaptive algorithm for geofencing," *Information Technology for Management: Emerging Research and Applications. AITM ISM 2018 2018*. Lecture Notes in Business Information Processing, Springer, vol 346, pp. 115–135, 2019, doi: 10.1007/978-3-030-15154-6_7.
- [19] A. Ghaffari, "Analytical design and experimental verification of geofencing control for aerial applications," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 1106–1117, Apr. 2021, doi: 10.1109/TMECH.2020.3017712.
- [20] P. S. Heckbert, *Graphics gems IV*. Academic Press, 1994.
- [21] A. G. G. Filho, P. Borba, V. H. S. Silva, A. Cerdeira, and A. P. D. Poz, "Quality control relevance on acquisition of large scale geospatial data to urban territorial management," in *2020 IEEE Latin American GRSS & ISPRS Remote Sensing Conference (LAGIRS)*, Mar. 2020, vol. XLII-3/W12, pp. 138–142, doi: 10.1109/LAGIRS48042.2020.9165682.
- [22] D. Atunggal, N. H. Ausi, and C. A. Rohkmana, "Developing android application for precise geotagging using RTK GPS module," in *2018 4th International Conference on Science and Technology (ICST)*, Aug. 2018, pp. 1–5, doi: 10.1109/ICSTC.2018.8528570.
- [23] M. R. R. Sharba, H. Adil Kadhim, S. A. W. Al-Abassi, and N. Salih Ali, "Online geocode in postal address using GPS with synchronous database accessing," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 3, p. 1487, Mar. 2020, doi: 10.11591/ijeecs.v17.i3.pp1487-1492.
- [24] "Create and monitor geofences." <https://developer.android.com/training/location/geofencing> (accessed Jun. 11, 2021).
- [25] J. O'Rourke, *Computational Geometry in C*. Cambridge University Press, 1998.
- [26] M. Alsaqer, B. Hilton, T. Horan, and O. Aboulola, "Performance assessment of geo-triggering in small geo-fences: Accuracy, reliability, and battery drain in different tracking profiles and trigger directions," *Procedia Engineering*, vol. 107, pp. 337–348, 2015, doi: 10.1016/j.proeng.2015.06.090.

BIOGRAPHIES OF AUTHORS



Puspa Miladin Nuraida Safitri A Basid     is a member of Geographic Information System Laboratory Department of Informatics Engineering, Faculty of Science and Technology UIN Maulana Malik Ibrahim, Malang. Also became a representative of Faculty of Computer Science in Team Quality Assurance Center (UPM) UIN Maulana Malik Ibrahim. She received Bachelor degree in Department of Informatics Engineering of UIN Maulana Malik Ibrahim, Malang, Indonesia, in 2014. And received M.Kom degree in the Department of Computer Science, Universitas Brawijaya, Malang, Indonesia 2017. She can be contacted at email: puspa.miladin@uin-malang.ac.id.



Fresy Nugroho     is a member of Multimedia Laboratory Department of Informatics Engineering, Faculty of Science and Technology UIN Maulana Malik Ibrahim, Malang. He received Bachelor degree in Department of Electrical Engineering of Universitas Brawijaya, Malang, Indonesia, in 1997. And received M.Tech. degree in the Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia 2010. He is currently working towards the Ph.D degree with the Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. His research interests include game technology, education, computer vision, optimization. He can be contacted at email: fresy@ti.uin-malang.ac.id.