Generation of simple polygons from ordered points using an iterative insertion algorithm

Hongyun Zhang*, Quanhua Zhao☯*, Yu Li☯

Institute for Remote Sensing Science and Application, School of Geomatics, Liaoning Technical University, Fuxin, Liaoning, China

☯ These authors contributed equally to this work.
* zqhlby@163.com

Abstract

To construct a simple polygon from a set of plane points, we propose an iterative inserting ordered points (IIOP) algorithm. Using a given a set of ordered non-collinear points, a simple polygon can be formed and its shape is dependent on the sorting method used. To form such simple polygons with a given set of plane points, the points must first be ordered in one direction (typically, the x-axis is used). The first three points in the set are used to form an initial polygon. Based on the formed polygon, new polygons can be iteratively formed by inserting the first point of from among the remaining set of points, depending on line visibility from that point. This process is carried out until all the points are inserted into the polygon. In this study, we generated 20, 50, and 80 plane points and used the proposed method to construct polygons. Experimental results show that these three polygons are all simple polygons. Through theoretical and experimental verification, we can concluded that when given a set of non-collinear points, a simple polygon can be formed.

Introduction

Constructing a simple polygon from a set of plane points is an important process in computational geometry [1] as simple polygons are widely used in computer graphics, image processing, entity construction, and other fields [2–4]. However, the fundamental problem of constructing simple polygons from a set of discrete points is still under investigation.

Currently, several methods are being used to construct a simple polygons from a set of discrete points, such as the polar coordinate sorting [5], dichotomy sorting [6], convex-hull construction [7], and visual region [8] methods. In the dichotomy sorting method [5], two outer points are determined from the maximum and minimum values of x-axis projection. The line connecting the two points can divide the set of points into two subsets. The points in these two subsets are determined from the maximum and minimum values of x-axis (or y-axis) projection. The line connecting the two points can divide the set of points into two subsets. The points in these two subsets are sorted according to the value of x-axis (or y-axis) projection and sequentially connected to form a polygon. In the polar coordinate sorting method [6], a point is randomly selected from a given set as the origin of the polar coordinate system. Afterward, the angles of all the vectors formed by the origin and other points are calculated, and all
the points are ordered according to the included angles. The sorted points are then sequentially connected to form a polygon. In the convex-hull construction method [7], a convex hull is first constructed from a given set of points. Here, iteration is started with the convex hull as the initial polygon. In each iteration, a point is randomly selected from the remaining set of points and inserted into the original polygon to form a new polygon. Finally, in the triangle-based construction method [8], an initial triangle is constructed from three randomly selected points and is considered as the initial polygon. The iterative process entails the gradual insertion of the remaining points in the polygon. Although the methods listed above have yielded good experimental results in the process of polygon construction some problems remain. For instance, during polygon construction, invalid points may be encountered. Additionally, it is impossible to prove that a simple polygon can be formed using a given set of discrete points. To solve these problems, in this study, we developed an iterative inserting ordered points (IIOP) algorithm and verified its feasibility of this algorithm theoretically and experimentally. We demonstrate that a simple polygon can be formed using a set of discrete points in which all the points are considered as polygon nodes.

The rest of this article is organized as follows. After defining some functions used in the polygon construction algorithm in the next section, we describe the process of polygon construction and theoretically demonstrate its feasibility. Later, we shall discuss several experimental results and finally present our main conclusions.

Preliminary definitions

Simple polygon

Consider a set of \( n \) points. These points can act as the nodes of a polygon \( G \), if \( G \) satisfies the following conditions.

1. A node connects only two lines.
2. Arbitrary non-adjacent lines do not intersect with each other.
3. There is no loop in the closed geometry.

Under such conditions, \( G \) can be defined as a simple polygon [9]. Fig 1 show a polygon \( G \) with a set of 7 nodes: this node set can be denoted as \( v = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \). In Fig 1(A)–1(D), each polygon has a different geometry and the polygons in Fig 1(A)–1(C) are not simple. In Fig 1(A), point \( v_4 \) connects four lines (\( v_1v_4, v_3v_4, v_4v_6, \) and \( v_4v_7 \)) and hence does not satisfy Condition 1. In Fig 1(B), lines \( v_1v_6 \) and \( v_4v_3 \) intersect, thus nullifying Condition 2. Meanwhile, Fig 1(C) does not satisfy Condition 3. However, because the polygon in Fig 1(D) satisfies all the conditions listed above, it can be classified as a simple polygon.

Visibility of a polygon line from a point

Given a polygon and a point outside of the polygon, if a node of the polygon is visible from the point, then a line formed between the node and point does not cross any other polygon line. If and only if a polygon line is visible for the point, the two endpoints of this line are visible for the point [10–12].

Fig 2 shows a polygon \( G (\Delta v_1v_2v_3) \) and a point \( v_i \) outside it. When the point \( v_i \) and node \( v_2 \) are connected we can see that the new line \( v_i,v_2 \) does not intersect with any other line of the polygon \( G \) and similar is the case with line \( v_2v_3 \). Thus, we can infer that line \( v_2v_3 \) is visible from point \( v_i \). When point \( v_i \) is connected with node \( v_3 \), the new line \( v_i,v_3 \) intersects with line \( v_2v_3 \). Therefore, lines \( v_1v_2 \) and \( v_1v_3 \) are invisible from point \( v_i \).
Line-to-line occlusion at a given point

Assume that there are two lines \((v_1v_2 \text{ and } v_3v_4)\) and one point \((v_i)\). The two lines do not intersect (or only intersect at the endpoints). With respect to point \(v_i\), line \(v_1v_2\) occludes line \(v_3v_4\) if and only if the following conditions are satisfied.
1. There is a ray \( t \) starting at point \( v_i \) and it intersects with both lines \( v_1v_2 \) and \( v_3v_4 \), at different points.

2. If the intersections of ray \( t \) with lines \( v_1v_2 \) and \( v_3v_4 \) are \( s_i \) and \( s_j \) (\( i \neq j \)), respectively, then \( d(v_i, s_i) < d(v_i, s_j) \) [13–14].

Fig 3(A) shows two disjoint lines \( v_1v_2 \) and \( v_3v_4 \). A ray \( t \) starts at point \( v_i \) and intersects with lines \( v_1v_2 \) and \( v_3v_4 \) at points \( s_i \) and \( s_j \), respectively, and thus, it satisfies Condition 1. Additionally, we can see that \( d(v_i, s_i) < d(v_i, s_j) \), which satisfies Condition 2; hence, it can be inferred that line \( v_1v_2 \) occludes \( v_3v_4 \) at point \( v_i \). Fig 3(B) shows two lines with intersecting endpoints: here, points \( v_2 \) and \( v_4 \) represent the same point. Moreover, ray \( t \) intersects lines \( v_1v_2 \) and \( v_3v_4 \) at points \( s_i \) and \( s_j \) (\( i \neq j \)), respectively, and \( d(v_i, s_i) < d(v_i, s_j) \). Therefore, line \( v_1v_2 \) occludes \( v_3v_4 \) at point \( v_i \). Fig 3(C) shows two disjoint lines \( v_1v_2 \) and \( v_3v_4 \). There is no ray starting at point \( v_i \) that can intersect lines \( v_1v_2 \) and \( v_3v_4 \) simultaneously. Hence, it does not satisfy Condition 1 and
there is no occlusion. Fig 3(D) shows two lines with intersecting endpoints. Although there is a ray starting at point \( v_i \) that can simultaneously intersect lines \( v_1v_2 \) and \( v_3v_4 \), intersection occurs at the same point. Therefore, Condition 1 is not satisfied and there is no occlusion.
Generation of a simple polygon

Consider a set of ordered points \( v = \{v_i(x_i, y_i); i = 1, \ldots, n\} \), in which no two points are coincident and no three points are collinear. In this point set, \( i \) is the index of points, \( v_i \) is used to indicate the point \( i \), \( n \) is the number of points in the set (usually \( n > 3 \)), and \((x_i, y_i)\) \( \in \mathbb{R}^2 \) represent the coordinates of point \( i \). For \( v_i \) and \( v_{i+1} \) (\( i = 1, \ldots, n-1 \)), if the \( x \) coordinates are the same \( x_{i+1} = x_i \), then \( x_{i+1} = x_i \); otherwise \( x_{i+1} > x_i \).

Usually, we regard \( x \)-axis as the horizontal axis, \( y \)-axis as the vertical axis perpendicular to the \( x \)-axis, and the intersection of the two axes as origin \( o \) (Fig 4). There are five ordered points in this figure. And their set can be represented as \( v = \{v_1, v_2, v_3, v_4, v_5\} \). After sorting, \( x_1 \geq x_2 \geq x_3 \geq x_4 \geq x_5 \). Moreover, when \( x_1 = x_2, y_2 > y_1 \). Because the \( x \)-coordinates of all the points are different, their \( y \)-coordinates were not considered when ordering.

Polygon-generation process

(1) Initialization. From a given set of ordered points \( v \), we select the first \( nc^{(0)} \) points (usually 3) to construct a polygon \( v_c^{(0)} \) (note that \( v_c \) represents both the polygon and the set of points that compose it). The set of points corresponding to this polygon can be denoted as \( v_c^{(0)} = \{v_{c1}^{(0)}, \ldots, v_{cn_c}^{(0)}\} \), where \( nc^{(0)} \) is the point number of the constructed polygon. The residual points in set \( v \) are included in the remaining set \( v_s^{(0)} = \{v/v_c^{(0)}\} = \{v_{s1}^{(0)}, \ldots, v_{sn}^{(0)}\} \), \( l = 1, \ldots, n-n^{(0)} \); set \( v_s \) is still a set of ordered points and the sorting method is consistent with that of set \( v \).

Fig 5 shows an example of polygon initialization. There are 5 ordered points in the set, i.e., \( v = \{v_1, v_2, v_3, v_4, v_5\} = \{v_l, l = 1, \ldots, n\} \), where \( n = 5 \). Select the first \( nc^{(0)} \) points to construct a polygon \( v_c^{(0)} \). Set \( v_c^{(0)} = \{v_{c1}^{(0)}, \ldots, v_{cn_c}^{(0)}\} = \{v_{ck}^{(0)}, k = 1, \ldots, nc^{(0)}\} \), where \( nc^{(0)} = 3 \) and the symbol \( ⊂ \) denotes the same point. In the remaining set, \( v_s^{(0)} = \{v/v_c^{(0)}\} = \{v_{s1}^{(0)}, v_{s2}^{(0)}, v_{s3}^{(0)}\} \).

(2) Let \( r \) be the iteration indicator. Considering point \( v_{s1}^{(r)} \) in the remaining set \( v_s^{(r)} \) as an insertion point, we can conclude that the point before point \( v_{s1}^{(r)} \) is \( v_{nc}^{(r)} \) in the set \( v \). We find two lines which connect point \( v_{nc}^{(r)} \) from the polygon \( v_c^{(r)} \) and judging the visibility of these two lines from point \( v_{s1}^{(r)} \). Then insert point into the line which is visible from it. If both lines are visible from point \( v_{s1}^{(r)} \), one line is selected randomly. The set of points of the new polygon can be rewritten as \( v_c^{(r+1)} = \{v_{ck}^{(r+1)}, k = 1, \ldots, nc^{(r+1)}\} \), where \( nc^{(r+1)} = nc^{(r)} + 1 \), and the iteration indicator is \( r = r + 1 \).

Polygon \( v_c^{(0)} \) is considered as the initial polygon, and the iteration process is started. As shown in Fig 5(B), by judging the visibility of the two lines at \( v_{nc}^{(r)} \) \( v_{s1}^{(r)} \), \( v_{s2}^{(r)} \) \( v_{s1}^{(r)} \), \( v_{s3}^{(r)} \) \( v_{s1}^{(r)} \), both \( v_{s1} \) and \( v_{s2} \) are visible from point \( v_{s1}^{(r)} \). Therefore, one line is selected randomly and the point \( v_{s1}^{(r)} \) is inserted into it; if we select line \( v_{s1} \), the resulting polygon is as shown in Fig 6(A). Consider the set of points \( v_c^{(1)} = \{v_{1}/v_{c1}^{(1)}, v_{2}/v_{c2}^{(1)}, v_{3}/v_{c3}^{(1)}, v_{4}/v_{c4}^{(1)}\} = \{v_{ck}^{(1)}\} \), where \( nc^{(1)} = 4 \) and the remaining set \( v_s^{(1)} = \{v/v_c^{(1)}\} = \{v_{s2}^{(1)}\} \). If line \( v_{s2} \) is selected, the resulting polygon is as shown in Fig 6(B). Here, \( v_c^{(1)} = \{v_{1}/v_{c1}^{(1)}, v_{2}/v_{c2}^{(1)}, v_{3}/v_{c3}^{(1)}, v_{4}/v_{c4}^{(1)}\} \) and \( v_s^{(1)} = \{v/v_c^{(1)}\} = \{v_{s2}^{(1)}\} \). Evidently, the insertion method affects the order of the set \( v_c \) and polygon geometry. However, it has no impact on the order of the sets \( v_s \) and \( v_c \).

Lemma 1 The point before point \( v_{s1} \) is \( v_{nc} \). In the case of two lines \( r_1 \) and \( r_2 \) on which \( v_{nc} \) is located, at least one line is visible from \( v_{s1} \).

**Proof:** Assume that the two lines \( r_1 \) and \( r_2 \) are not visible from \( v_{s1} \). If line \( r_1 \) is not visible from \( v_{s1} \), line \( r_2 \) occludes \( r_1 \). A ray \( t \) starting from point \( v_{s1} \) intersects lines \( r_1 \) and \( r_2 \) at points \( s_m \) and \( s_n \) respectively. According to the definition of occlusion, \( d(s_m, v_{s1}) > d(s_m, v_{s2}) \). If line \( r_2 \) is not visible from \( v_{s1} \), then \( r_1 \) occludes \( r_2 \). When ray \( t \) starts at point \( v_{s1} \), it intersects lines \( r_1 \) and \( r_2 \) at points \( s_m \) and \( s_n \) respectively. According to the definition of occlusion, \( d(s_m, v_{s1}) > d(s_m, v_{s2}) \).
"Contrary to the assumptions, there is no situation in which both lines \( r_1 \) and \( r_2 \) are not visible from the point \( v_{sl} \). Therefore, it can be demonstrated that at least one line, i.e., either \( r_1 \) or \( r_2 \) or both, is visible from point \( v_{sl} \).

Fig 7 illustrates Lemma 1. As shown in Fig 7(A), the point before point \( v_{sl} \) is \( v_{nc} \). A line passing through \( v_{nc} \) that is perpendicular to the x-axis divides the plane into L and R regions."
According to the order of the points, point \( v_{s1} \) is in the R region. The ray \( t \) starting at point \( v_{s1} \) intersects lines \( r_1 \) and \( r_2 \) at points \( s_m \) and \( s_n \), respectively. If \( d(s_m, v_{s1}) > d(s_n, v_{s1}) \), \( r_1 \) is not visible from \( v_{s1} \); similarly, if \( d(s_n, v_{s1}) > d(s_m, v_{s1}) \), \( r_2 \) is not visible from \( v_{s1} \). If there is no ray \( t \) that can simultaneously intersect lines \( r_1 \) and \( r_2 \) (except line \( v_{s1}v_{nc} \), as shown in Fig 3(D)), both lines are visible from \( v_{s1} \).

Lemma 1 indicates that either \( r_1 \) or \( r_2 \) or both lines are visible from point \( v_{s1} \). As shown in Fig 7(B), region R can be divided into three parts by extending lines \( r_1 \) and \( r_2 \), i.e., \( R = \{ R_1, R_2, R_3 \} \). When point \( v_{s1} \) is in \( R_1 \), line \( r_1 \) is visible and line \( r_2 \) is not visible. When point \( v_{s1} \) is in \( R_3 \), line \( r_2 \) is visible and line \( r_1 \) is not visible from it. When point \( v_{s1} \) is in \( R_2 \), both \( r_1 \) and \( r_2 \) are visible from it.

(3) The final polygon is generated at an iteration indicator \( \tau = n-3 \). At this time, the number of points in the remaining set is 0. The set of points corresponding to the polygon is \( v_c^{(n-3)} = \{ v_{ck}^{(n-3)}, k = 1, \ldots, n \} = v \).

The polygon-generation process is illustrated in Fig 8(A)–8(F). There are seven ordered points that can be denoted as \( v = \{ v_1, v_2, v_3, v_4, v_5, v_6, v_7 \} \). The first three points are connected to form an initial polygon \( v_c^{(0)} = \{ v_{c1}^{(0)}, v_{c2}^{(0)}, v_{c3}^{(0)} \} \) and the remaining set of points can be denoted by \( v_s^{(0)} = \{ v_{s1}^{(0)}, v_{s2}^{(0)}, v_{s3}^{(0)}, v_{s4}^{(0)} \} \). Consider the first point \( v_{s1}^{(0)} \) in \( v_s^{(0)} \); the point preceding \( v_{s1}^{(0)} \) is point \( v_{c3}^{(0)} \). Therefore, we need to judge the visibility of lines \( v_{c3}^{(0)}v_{c1}^{(0)} \) and \( v_{c3}^{(0)}v_{c1}^{(0)} \) from \( v_{s1}^{(0)} \). The line \( v_{c3}^{(0)}v_{c1}^{(0)} \) is visible from point \( v_{s1}^{(0)} \) and hence, we insert point \( v_{s1}^{(0)} \) into the line to form the new polygon \( v_c^{(1)} = \{ v_{c1}^{(1)}, v_{c2}^{(1)}, v_{c3}^{(1)}, v_{c4}^{(1)} \} \) and the remaining set of points can be written as \( v_s^{(1)} = \{ v_{s1}^{(1)}, v_{s2}^{(1)}, v_{s3}^{(1)} \} \), as shown in Fig 8(C). After the iterative insertion process is completed, the final polygon shown in Fig 8(F) is obtained.

Fig 5. Example of polygon initialization. (a) Random point distribution and (b) polygon initialization.

https://doi.org/10.1371/journal.pone.0230342.g005
Fig 6. New polygon after one iteration. (a) Polygon 1 and (b) polygon 2.
https://doi.org/10.1371/journal.pone.0230342.g006

Fig 7. Example of line-to-point visibility. (a) Visibility example and (b) visibility from different regions.
https://doi.org/10.1371/journal.pone.0230342.g007
Polygon-generation algorithm

Given a set of ordered points \( v \), the steps required to generate a polygon can be represented using the following pseudo-codes.

Initialization: Initial polygon \( v_c \) and \( v_s \) is the set of remaining points.

For \( \tau = 0 \) to \( n-3 \)

1. The number of nodes in polygon \( v_c^{(\tau)} \) is denoted as \( n_c^{(\tau)} \).
2. Find two sides of point \( v_{nc}^{(\tau)} \) from polygon \( v_c^{(\tau)} \) and denote them as \( r_1 \) and \( r_2 \).
3. Select the first node \( v_{s1}^{(\tau)} \) from \( v_s^{(\tau)} \).
4. Judge the visibility of \( r_1 \) and \( r_2 \) from \( v_{s1}^{(\tau)} \).
5. Insert \( v_{s1}^{(\tau)} \) into \( r_1 \) or \( r_2 \), whichever line is visible from it.
6. Generate polygon \( G = v_c^{(\tau+1)} \).

End

Generate polygon \( G = v_c \).
Experiments and results

This study proposes a simple polygon-construction method, which proves that in a given set of points, all points can be regarded as polygon nodes and can form a simple polygon. In the current experimental section, the number of simple polygons generated is counted and the criterion of whether all given points can be used as polygon nodes to form simple polygons is used to evaluate the proposed algorithm.

Relationship between the number of polygons and the number of points

From Lemma 1 it can be seen that a simple polygon can be formed from a set of points. Additionally, the number of polygons generated is not unique according to the constructing method. From a set of ordered points \( v = \{v_i(x_i, y_i); i = 1, \ldots, n\} \). Evidently, the first three points \( v_1, v_2, \) and \( v_3 \) can be used to construct one polygon. Based on our proposed method, point \( v_4 \) is inserted into the polygon generated by the first three points, i.e., point \( v_4 \) is inserted on either line \( v_1v_3 \) or line \( v_2v_3 \). Therefore, when both lines are visible from point \( v_4 \), two polygons can be formed which implies that a maximum of two polygons can be formed. In the two polygons formed from the four points described above, when point \( v_5 \) is inserted, each polygon yields two new polygons. Therefore, a maximum of four polygons can be formed from five points. By analogy, according to the given method, a maximum of \( 2^{n-3} \) polygons can be constructed from a set of \( n \) non-collinear points.

As shown in Fig 9(A), from a set of five ordered points \( v = \{v_1, v_2, v_3, v_4, v_5\} \), the first three points \( v_1, v_2, \) and \( v_3 \) are selected to construct a triangle \( \triangle v_1v_2v_3 \). Subsequently, point \( v_4 \) is inserted into the triangle formed by the first three points. Because the points follow a particular order, point \( v_4 \) can exist only in the \( R \) region, which is divided into three parts denoted as \( R = \{R_1, R_2, R_3\} \). When point \( v_4 \) is present in \( R_1 \) or \( R_3 \), only one of the two lines containing point \( v_3 \)
is visible from it. Therefore, only one polygon can be constructed. When point \( v_4 \) is in \( R_2 \), both lines containing \( v_3 \) are visible from it, and hence, it can form two polygons. According to the proposed polygon-construction method, a maximum of two polygons can be constructed using four points (Fig 9(B)). Similarly, when point \( v_5 \) is inserted four polygons can be constructed (Fig 9(C)).

Relationship between polygon shape and point order

In this section, we shall discuss the relationship between the number of points and the number of polygons that can be constructed with the same coordinate order. The influence of different
coordinate axis directions on polygon shape when the relative position of a group of points is unchanged is also considered.

As shown in Fig 10, given a set of points \( v = \{v_i, i = 1, \ldots, 8\} \), the positional relationship between the points remains unchanged. Fig 10(A)–10(C) depict the polygons formed when the \( x \)-axis direction is horizontal, rotated by 45\(^\circ\) to the left, and rotated by 90\(^\circ\) to the left.
respectively. Note that when the $x$-axis is in different directions, the order between the points as well as the shape of the polygon are different. Therefore, it can be concluded that the shape of the polygon depends on the arrangement of points, which in turn is dependent on the direction of the coordinate axis. However, according to the theoretical proof, given a set of non-collinear points, a simple polygon can be formed regardless of the direction of the coordinates.

**Polygon generation experiment**

To verify the feasibility of the proposed method, 20, 50, and 80 points were randomly generated and a simple polygon was then generated from each of these sets using the proposed algorithm. Fig 11 shows the plane distribution of 20, 50, and 80 random points. The experimental results are shown in Fig 12. As described in the section on the relationship between number of polygons and number of points, the number of simple polygons generated by the proposed algorithm is not unique. Fig 12 shows two different geometric polygons formed by each group of points. Fig 12(A1)–12(A2), 12(B1)–12(B2) and 12(C1)–12(C2) represent polygons consisting of 20, 50, and 80 points, respectively, and it can be noticed that all the constructed polygons are simple polygons.

Some regions of the polygon with 80 random points cannot be seen clearly. We considered the polygon generated in Fig 12(C1) as an example and enlarged a part of its area (Fig 13). Figs
13(B)–13(D) represent enlarged images of areas in the blue, green, and black boxes in Fig 13 (A), respectively. We can see that the polygon with 80 random points is still a simple polygon. Moreover, the points in the graph are randomly generated, which verifies the feasibility of the proposed method for constructing simple polygons.
Comparative experiments

Because both the polar coordinate sorting [5] and dichotomy sorting [6] methods can construct simple polygons using all the given random points as polygon nodes, we compared our proposed method with these two methods for generating simple polygons from three groups of different number of random points (see Fig 11). Fig 14 shows the result obtained using the

Fig 14. Polygon generation using the polar coordinate sorting method. Polygon consisting of (a) 20 (b) 50, and (c) 80 points.

https://doi.org/10.1371/journal.pone.0230342.g014
polar coordinate sorting method, where the black dot represents pole position. According to this algorithm, for a set of random points at a given position, this method can only generate a simple polygon with a fixed geometry. Fig 15 shows the result obtained using the dichotomy sorting method. It can be seen that the number of polygons generated is unique when the direction of a given dichotomy is fixed. However, the proposed method can still construct
polygons with different geometric shapes when the coordinate axis direction is given, which implies that the approach proposed in this study is more flexible.

Table 1 shows the time required by the three methods to generate polygons for different random points; it can be inferred that the polar coordinate sorting method is the fastest, while the proposed method is the slowest. However, this slow rate can be attributed to its ability to generate polygons of different geometric shapes for which more time is required.

The number of simple polygons constructed by the polar coordinate sorting and dichotomy sorting methods is unique, while the number of simple polygons constructed by the convex-hull construction method [7] is non-unique. However, the latter cannot generate a simple polygon using all the given random points as polygon nodes.

| Time (s)       | Number of random points |
|----------------|-------------------------|
| Proposed method| 0.25 0.30 0.37          |
| Polar coordinate sorting method| 0.07 0.09 0.11          |
| Dichotomy sorting method    | 0.09 0.12 0.13          |

https://doi.org/10.1371/journal.pone.0230342.t001

Fig 16. Polygon generation using the proposed method. (a) Random point distribution, (b) polygon initialization, (c) result of one iteration, (d) result of two iterations, (e) result of three iterations, and (f) the final polygon.

https://doi.org/10.1371/journal.pone.0230342.g016
Consider a set of seven ordered points $v = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, as shown in Fig 16(A). The polygon-construction process using the proposed method is shown in Fig 16(B–E) and Fig 16(F) shows the final polygon, which is a simple polygon.

Fig 17. Polygon generation using the convex-hull construction method. (a) Polygon initialization, (b) result of one iteration, (c) result of two iterations, and (d) result of three iterations.

https://doi.org/10.1371/journal.pone.0230342.g017
Fig 17 shows the results of polygon generation using the convex-hull construction method. First, the convex hull of a given node set is generated (Fig 17(A)); later, a point is randomly selected from the remaining points and inserted into the line visible from it to form a new polygon. The final polygon is shown in Fig 17(D). However, none of the polygon edges are visible from point $v_4$. Therefore, in this situation, point $v_4$ is called an invalid point. Finally, we can infer that polygon construction based on the convex-hull method includes a strong element of randomness owing to which it is impossible to construct a simple polygon with all the points in a given set. Unlike the method proposed in this study, using which a simple polygon can be constructed from a given set of non-collinear points, in the convex-hull construction method, invalid points may occur.

**Conclusion**

In this study, we designed an IIOP algorithm to generate a simple polygon from a set of ordered points. Through theoretical and experimental verification of this algorithm, the following three conclusions could be drawn. (1) Given a set of non-collinear points, a simple polygon can be formed. (2) The shape of a simple polygon depends on the sorting method used. (3) The number of polygons is related to the number of random points. When the sorting method is fixed, at least 1 or a maximum of $2^{n-3}$ simple polygons can be formed, where $n$ is the number of non-collinear points in a set.

**Acknowledgments**

We would like to thank Editage (www.editage.cn) for English language editing.

**Author Contributions**

**Methodology:** Hongyun Zhang, Quanhua Zhao, Yu Li.

**References**

1. Vaclav Skala. Point-in-Convex Polygon and Point-in-Convex Polyhedron Algorithms with O(1) Complexity using Space Subdivision. ICNAAM 2015. AIP Publishing LLC, 2015; 22–28. https://doi.org/10.1063/1.4952270
2. Martin N, Roman P, Ignaz R. Partitioning graph drawings and triangulated simple polygons into greedily routable regions. Algorithms and Computation (ISAAC’15). Springer Berlin Heidelberg, 2015. https://doi.org/10.1007/978-3-662-48971-054
3. Bose P, Kostitsyna I, Langerman S. Self-approaching paths in simple polygons. Computational Geometry, 2019. https://doi.org/10.1016/j.comgeo.2019.101595
4. Oh E, De Carufel JL, Ahn HK. The geodesic 2-center problem in a simple polygon. Computational Geometry, 2018; 74: 21–37. https://doi.org/10.1016/j.comgeo.2018.02.008
5. Laszlo MJ. Computational geometry and computer graphics in C++. 1st ed. Upper Saddle River, NJ: Prentice Hall, 1996; 109–112. https://doi.org/10.1109/5.163408
6. Weixin W, Yongqing H. The algorithm of constructing polygons by sorting the scattered points. Journal of Shenyang Ligong University. 1997; 16(4): 11–14. (in Chinese)
7. Muravitsky V, Tereshchenko V. Generating a simple polygonalizations. In 2011 15th International Conference on Information Visualisation, IEEE. 2011; 668–671. https://doi.org/10.1109/IV.2011.88
8. Dailey D, Whitfield D. Constructing random polygons. In Proceedings of the 9th ACM SIGITE Conference on Information Technology Education. 2008; 119–124. https://doi.org/10.1145/1414558.1414592
9. Li F, Klette R. Euclidean shortest paths in a simple polygon. Algorithms, Architectures and Information Systems Security. 2015. https://doi.org/10.1142/9789812836243_0001
10. Cardinal J, Hoffmann U. Recognition and complexity of point visibility graphs. Discrete and Computational Geometry. 2017; 57: 164–178. https://doi.org/10.1007/s00454-016-9831-1
11. Himmel AS, Hoffmann C, Kunz P, Froese V, Sorge M. Computational complexity aspects of point visibility graphs. Discrete Applied Mathematics. 2019; 254: 283–290. https://doi.org/10.1016/j.dam.2018.06.016

12. Bahoo Y, Banyassady B, Bose P K, Durocher S, Mulzer W. A time–space trade-off for computing the k-visibility region of a point in a polygon. Theoretical Computer Science, 2019; 789: 13–21. https://doi.org/10.1016/j.tcs.2018.06.017

13. Chen DZ, Wang H. Visibility and ray shooting queries in polygonal domains. Computational Geometry—Theory and Applications. 2015; 48(2): 31–41. https://doi.org/10.1016/j.comgeo.2014.08.003

14. Chen DZ, Wang H. Weak visibility queries of line segments in simple polygons. Computational Geometry. 2015; 48(6): 443–452. https://doi.org/10.1016/j.comgeo.2015.02.001