Fast and Robust Point-in-Spherical-Polygon Tests Using Multilevel Spherical Grids

Jing Li[1,2(✉)], Han Zhang[1], and Wencheng Wang[2,3]

[1] State Key Laboratory of Integrated Information System Technology, Institute of Software, Chinese Academy of Sciences, Beijing, China
lijing2015@iscas.ac.cn
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] The University of Chinese Academy of Sciences, Beijing, China

Abstract. Point-in-spherical-polygon determination is widely required in applications over the earth. However, existing methods for point-in-polygon tests cannot be applied directly, due to non-Euclidean computation over the spherical surface. Thus, it is general to transform a spherical polygon into a planar polygon via projection, and then perform point-in-polygon tests. Unfortunately, this is expensive and may cause determination errors. In this paper, we propose to subdivide the spherical grid cells iteratively, and apply the ray-crossing method locally in grid cells, which is by one of our previous works. As a result, this not only avoids expensive transformation computation, but also guarantees robust determination for point-in-spherical-polygon tests. Experimental results attest our effectiveness, and show an acceleration of five orders of magnitude over a popularly used method.

Keywords: Point-in-spherical-polygon test · Multilevel grid · Ray-crossing

1 Introduction

Applications over the earth have been increasing rapidly, e.g. space exploration, mobile communication, and internet monitoring. Here, a fundamental operation is determining whether a point is inside a spherical polygon, called as a *point-in-spherical-polygon test*. As the concerned polygon over the earth generally has many edges, and the query points are normally in a very large number, it is a challenge to run fast point-in-spherical-polygon tests.

Though there have been a lot of methods for point-in-polygon tests on a plane, they cannot be directly employed for point-in-spherical-polygon tests, due to the non-Euclidean computation over the spherical surface. With regard to this, it is commonly adopted to transform a spherical polygon to a planar polygon by projecting it onto a plane through a certain projection, e.g. cylindrical projection, and then perform point-in-polygon tests on the plane. Besides a high cost, such a treatment unfortunately suffers from the distortion of transforming a curved

J. Chang et al. (Eds.): AniNex 2017, LNCS 10582, pp. 56–66, 2017.
https://doi.org/10.1007/978-3-319-69487-0_5

spherical edge to a straight planar edge, which would cause determination errors due to the precision problem.

Realizing that performing the analysis directly in the non-Euclidean spherical surface [1] can avoid the precision problem caused by projection, Bevis and Chatelain [2] extended the traditional ray-crossing method to treat spherical polygons. It is a popular robust method for point-in-spherical-polygon tests till now [3,4]. It works by first specifying a point X inside the polygon, then connecting the query point Q with X and counting the polygon edges that XQ intersects with. If the counted number is even, this means Q is inside the polygon, otherwise it is not. Similar to the planar ray-crossing method, this method needs to check every edge of the spherical polygon. Moreover, it requires a considerable amount of trigonometric calculations to determine whether two spherical line segments intersect with each other, which is more costly than additions or multiplications. Therefore, though the method can avoid errors due to projection, its testing speed is not fast. This also prevents it from applications, especially those that need to process many points in real time.

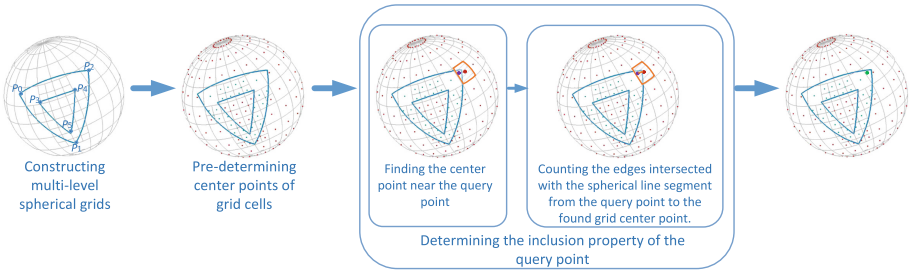

Fig. 1. The workflow of our method for fast and robust point-in-spherical-polygon tests.

In this paper, we address this challenge by presenting a fast point-in-spherical-polygon test method with the distortion problem well solved. In our method, we first build grid cells along longitudes and altitudes, and iteratively subdivide the grid cells that contain polygonal edges until the edges in a grid cell are in a smaller number than a set threshold value. Afterwards, with such a constructed multilevel spherical grid, we adopt one of our previous works to perform point-in-polygon tests [5] for query points. It is by predetermining the inclusion property of the center points of the cells as a prior, and then applying the ray crossing method locally in the cell that contains the query point by producing the line segment from the query point to the center point of the cell. Since each cell contains not many edges, the method is very fast. For the details, please refer to reference [5].

During the grid construction and point determination, one basic operation is to judge whether two spherical line segments intersect with each other, which is more difficult than the intersection test between two line segments on the plane. Instead of solving the problem by projection or by calculating angles in

the non-Euclidean space as done in existing works, we perform the test directly in the three dimensional space. As we know, the smallest distance between two points on the sphere surface is the smaller arc of the *great circle* passing through these two points, the located plane of the circle contains the sphere center. The edges of a polygon and the spherical line segments we generate to connect center points and query points on the sphere surface are all on their corresponding great circles. Thus, to determine whether a spherical line segment intersects with an edge of the spherical polygon, we can calculate the relative positions of the two vertices of the spherical line segment against the plane that contains the edge. In this way, not only much computation can be saved, but also errors caused by projection are avoided. Figure 1 shows the workflow of our method.

2 Our Algorithm

2.1 Conventions and Definitions

Before introducing our algorithm, we first clarify some definitions and conventions. Following the definitions in [2], we first define the great circle as discussed in Sect. 1. With any pair of points on a great circle, the circle is partitioned into two arcs, where the shorter arc is called a *minor arc* and the longer arc a *major arc*. Generally, an edge of a polygon on the sphere surface is a minor arc. For simplicity, in the following description, an arc generally refers to a minor arc without special explanation.

Suppose a spherical polygon S is composed of n points, $P_0, P_1, ..., P_{n-1}$ on the sphere surface, which are connected in a sequence to form n arcs $A_0 = P_0P_1, A_1 = P_1P_2, ..., A_{n-1} = P_{n-1}P_0$, to be the edges of the spherical polygon. For simplicity, we refer a polygon to a spherical polygon in the following discussion.

Suppose S is viewed from outside the sphere and it does not contain holes. We define the region on the left-hand side is inside the spherical polygon while the region on the right-hand side is outside, when traveling along the polygon edges in the anti-clockwise sequence. When S contains nested holes, we define the holes at the odd levels of the hierarchy for the nested holes are oriented in a clockwise direction, whereas the holes at the even levels are oriented in the opposite direction (e.g. anti-clockwise direction). Under such conventions, the definition of the inside/outside region of a polygon remains unchanged. For the polygon in Fig. 1, $P_0P_1P_2$ is the outer boundary of the polygon and $P_3P_4P_5$ is the boundary of a hole at the first level.

The location of a point P on the sphere surface is specified by its latitude (λ) and longitude (ϕ), denoted by $P(\lambda, \phi)$, $\lambda \in [-90°, 90°]$, $\phi \in [-180°, 180°]$. The point-in-spherical-polygon test is defined as given a query point Q and a polygon S on the sphere surface, determine if Q is inside S.

2.2 Constructing Multilevel Spherical Grids

In the spherical space, the longitude-latitude grid (called as *grid* in short) is the most commonly used coordinate system. It uniformly partitions the lines of lon-

gitude and latitude into segments and then generates regions by these segments. Owing to the uniform property, indexing in a grid is straightforward and easy to implement. However, for global applications, the distribution of polygons is usually very uneven. Polygons with tiny edges (comparing to the sphere's radius) may distribute sparsely on the whole surface of the sphere. In such a case, a uniform grid may contain a large number of small cells, most of which are empty, wasting too much storage. Therefore, we construct multilevel spherical grids. That is, we first construct a global uniform longitude-latitude grid without a high resolution to cover the whole sphere surface, and dispatch polygon edges into its cells. Then, we subdivide the cells having too many polygon edges into uniform sub-grids, which form the second-level grids. Such a subdivision process is repeated iteratively until the edges in each grid cell are fewer than a set threshold or the constructed levels reach a set value. As the cells to be subdivided are only in the regions with many edges, there are many large cells to reduce the number of cells and the storage requirement.

The resolutions for grids on each level are set as follows. According to our observation, in global applications, spherical polygons usually have many short edges and occupy small regions. So, it is better to construct a coarse grid for the first-level grid and finer grids for other levels. Therefore, we give a fixed low resolution for the first-level grid and calculate the resolutions for the sub-grids at other levels by the number of edges they contain. Equation 1 lists the formula for calculating the grid resolutions.

$$\begin{cases} M_\lambda(i,j) = R_\lambda, M_\phi(i,j) = R_\phi & i = 1, j = 1 \\ M_\lambda(i,j) = l_\lambda(i,j)\sqrt{kN(i,j)/(l_\lambda(i,j)l_\phi(i,j))} & 1 < i \le MAXD, 1 \le j \le M_i \\ M_\phi(i,j) = l_\phi(i,j)\sqrt{kN(i,j)/(l_\lambda(i,j)l_\phi(i,j))} & \end{cases} \tag{1}$$

In Eq. (1), (i,j) denotes the j^{th} grid on the i^{th} level, M_λ and M_ϕ are the number of cells for the grid in the latitude direction and the longitude direction respectively. l_λ and l_ϕ are the size of the grid in the latitude direction and the longitude direction respectively, which is measured by the degree of latitude and longitude. N is the number of edges in the grid. $MAXD$ is the set maximum level of the multilevel grids. M_i is the total number of grids at the i^{th} level. k is a coefficient. R_λ and R_ϕ are two empiric values. In our implementation, $MAXD, k, R_\lambda$ and R_ϕ are set to 4, 1.0, 10 and 20 respectively.

In dispatching an edge into grid cells, we first trace the edge and find the cells crossed by the edge, which is by calculating the intersections of the edge with the boundaries of grid cells sequentially along the edge, then record the edge in all these crossed cells. Note that cares should be taken for the polygon edge crossing the line at Longitude 180°, because the value of longitude jumps at the crossing point. In this case, we split the edge into two edges at the crossing point and dispatch the two edges respectively.

2.3 Predetermining Center Points of Grid Cells

After the multilevel grids are constructed, we determine the inclusion property of all the center points in the grid cells. These predetermined center points will be used as priors to promote the determination of query points. We start the determination of center points from the first-level grid, and then process the grids level by level until the grids at the deepest level are processed.

For each grid at each level, we take the similar measure as [5] to determine the center points by using the ray-crossing method locally. That is, for two neighboring center points, O_1 and O_2, we connect them with an arc segment. If the inclusion property of O_1 is known, the inclusion property of O_2 can be derived by counting the intersections between the arc segment and the polygon edges in the cells crossed by the arc segment using the rules listed in Table 1.

Table 1. Rules of determining the inclusion property of a point by another center point with known property.

Property of O_1	Priority of the intersection count	Property of O_2
Inside	Odd	Outside
Inside	Even	Inside
Outside	Odd	Inside
Outside	Even	Outside

To determine all the center points, we connect center points one by one with arc segments to form a traversal path, as illustrated in Fig. 2. When the first center point on the path is determined, the following center points on the path can be determined one by one with the counted intersections locally as described above. Similar to [5], a singular case may appear that a center point overlaps an edge or a vertex of the polygon. To this, we take the same measure as [5] that just mark the center point as singular and continue to treat the next center point on the path. Figure 2 shows this process.

Though the basic processing steps are similar to [5], there are two main differences when performing it on the sphere surface. The first one is for determining the intersection between two line segments. Here, we need to calculate the intersection of two arc segments (one is an edge of the polygon, and the other is an arc segment connecting two center points). The second is how to determine the first center point on a traversal path. In the planar case, it can be determined by finding a point outside the grid's bounding box, which is sure to be outside the polygon. However, this is infeasible for the first-level grid on the sphere surface as the grid covers the whole sphere surface. For a grid on the other levels, it is still sophisticated to find a point with known inclusion property outside the grid because the region outside the grid may belong to the grids on other levels. For these problems, we propose two solutions, as discussed below.

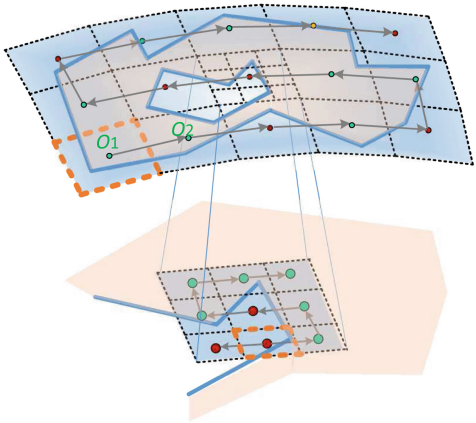

Fig. 2. The process of determining center points of multilevel grids. Inside, outside and singular center points are denoted by green, red and orange points respectively. Cells marked by orange boundaries are the starting cells of the paths for determination. (Color figure online)

To determine whether two arc segments intersect with each other, we make use of the relative positions of the vertices of an arc segment against the plane passing through the other arc segment and the sphere's center. Specifically, suppose AB and CD are two arc segments to be tested, O is the sphere's center, γ_{AB} is the plane passing through AB and O, and γ_{CD} is the plane passing through CD and O, then AB intersects with CD if and only if A and B are on different sides of γ_{CD} and C and D are on different sides of γ_{AB}, as showed in Fig. 3.

Note that in the determination, one arc segment is an edge of the polygon, and the other is an arc segment connecting two center points. According to the definition in Sect. 2.1, an arc segment refers to a part of a great circle. This means the arc is on the plane of the great circle through the sphere's center. In the implementation, longitude and latitude are transformed to rectangular coordinates and all the calculations are carried out in a rectangular coordinate system. The relative position of a vertex against a plane can be obtained by the normal of the plane. Taking the vertex C and the plane γ_{AB} as an example, suppose $\overline{OA}$, $\overline{AB}$ and $\overline{AC}$ are directed straight line segments and $w = (\overline{OA} \times \overline{AB}) \cdot \overline{AC}$, then if $w > 0$, C is on the left side of γ_{AB} (or the arc segment AB), similarly, if $w < 0$, C is on the right side of γ_{AB} (or the arc segment AB).

In [2], determining the intersection of arcs is handled directly on the sphere's surface by comparing the angles between arcs, which involves many trigonometric functions. Comparing to this, the measure proposed here can save a large amount of computation as only simple additions and multiplications are needed.

In determining the first center point on the traversal path for a grid, instead of finding a point outside the grid, we determine it by the edges in the same cell. As showed in Fig. 4, O_i is the center point of a non-empty cell $Cell_i$ (i.e. it contains edges). AB is an edge in $Cell_i$ and C is the midpoint of AB. We generate

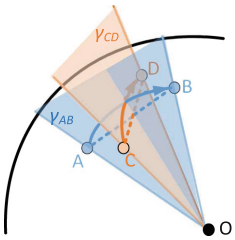

Fig. 3. Determining the intersection of two arc segments directly in the three dimensional space.

an arc segment by connecting O_i and C, and count the intersections between O_iC and the edges in the cell. Then we use the following rules to determine the inclusion property of O_i: if O_i is on the left side of AB, then O_i is inside the polygon when the counted number for intersections is even, otherwise it is outside. Similarly, if O_i is on the right side of AB, then O_i is inside the polygon when the counted number for intersections is odd, otherwise it is outside. In Fig. 4, O_iC does not intersect any edge and O_i is on the left side of AB, so O_i is inside the polygon. The relative position of O_i to AB is calculated as described above.

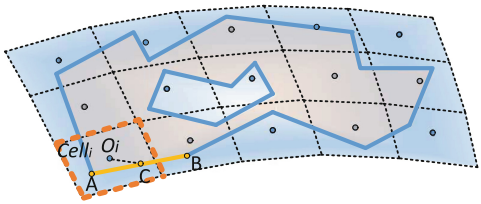

Fig. 4. Determining the inclusion property of the first center point on a traverse path

2.4 Point-in-Spherical-Polygon Test

With the constructed multilevel grids and the predetermined grid center points, we determine a query point as follows. Firstly, we search for a non-singular grid center point close to the query point in the multilevel grids. Specifically, we start the search from the first-level grid. We first find the grid cell that the query point falls in. If the cell's center point is singular, we search other cells around the cell gradually from near to far until a cell with non-singular center point is found. If the cell is a leaf node (i.e. it is not subdivided), the search is finished. If the cell is at a middle node, indicating that it is subdivided and has a sub-grid, we enter the sub-grid and search for a cell which has a non-singular center point and near the query point. Repeat such a searching process until a leaf node with a non-singular center point is found. Then, an arc segment is generated by connecting the cell's center point and the query point. By counting the intersections of the

arc segment with the edges in the cells crossed by the arc segment, the query point can be determined by the same rule for predetermining center points, as showed in Table 1.

Figure 5 shows an example. In the figure, Q falls in the cell bounded by orange lines. As the cell has a sub-grid, we enter the sub-grid and search for the cell that Q falls in. Thus, the cell bounded by purple lines is found. As the cell does not have a sub-grid and it has a non-singular center point O, then O is selected as the nearest non-singular center point to Q. So we connect Q with C by an arc segment. As O is inside the polygon and QC dose not intersect any edge, it is determined that Q is inside the polygon.

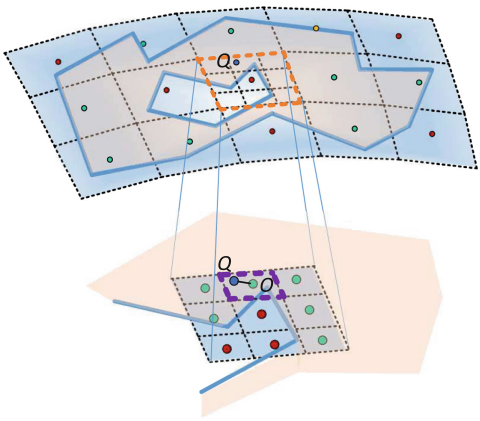

Fig. 5. Determining a query point locally with multilevel grids and predetermined center points

3 Results and Discussion

To test the performance of our method, we performed tests on a laptop installed with an Intel(R) Atom(TM) x7 1.6 GHz CPU, 4 GB RAM and Win10 operation system. We made two sets of experiments, one for testing the time and storage performance, and the other for testing the accuracy.

For the first set of experiments, we selected 7 spherical polygons with similar shape, whose number of edges are in a range from 2,927 to 92,781. Figure 6 shows one of them. For each polygon, we generated 1,000,000 query points randomly distributed within the bounding box of the polygon. We tested the time and storage cost for our new method. We also implemented the method of [2] for comparison, which is still one of the popular solutions for point-in-spherical-polygon tests. As described in Sects. 2 and 3, this method needs a large amount of trigonometric calculation to determine the intersection of arcs, which is too slow for testing 1,000,000 points. Because of this, we improved the method of [2] by using our method proposed in Sect. 2 to determine the intersection between arc

segments. As the method does not need auxiliary structure and preprocessing, we only recorded its total time for determining all query points. Table 2 lists the statistical results.

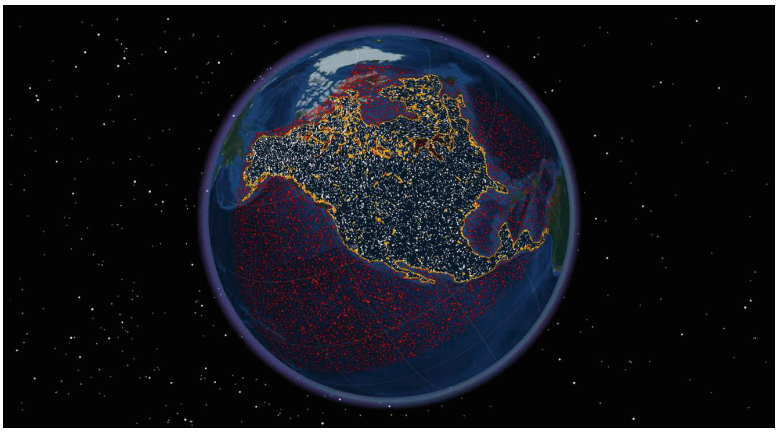

Fig. 6. The visualization of the test results against one of the tested polygons, NorthAmerica7, where inside and outside points are colored in white and red respectively. (Color figure online)

Results show that the storage of the new method increases linearly with the number of edges. For the preprocessing time, it increases a bit faster than the linear growth. As for the total time of determining the query points, it decreases with the increase of the number of edges. This is because test polygons are obtained by sampling one original data. When a polygon has more edges, the edges tend to be shorter, which is in favor of decreasing the number of intersections because shorter edges are likely to overlap fewer cells. Comparing to the improved method of [2], our method achieves an acceleration of about five orders of magnitude, as given in Table 2. Though our method needs additional space for storing grids and extra time for constructing grids, both the storage cost and preprocessing time are within a reasonable range. Note that even including the preprocessing time, our new method is still much faster, achieving an acceleration of several magnitudes. Clearly, with these advantages, our method is very suitable for fast determining many query points against large scale polygons with many edges, which is an increasingly popular requirement in modern global applications.

For the second set of experiments, we select a polygon, Antarctica, which overlaps the South Pole, to test the determination accuracy of the new method. As errors are more likely to appear near the boundary of a polygon, we generated 9,409 query points around the polygon's boundary, which has 54 edges. For each point, the distance from the point to the nearest edge is within 5% of the average length of polygon edges. To compare with our method, we implemented

Table 2. Performance comparison of our method against the improved method of [2]

Polygon	$Edge\#$	St(KB)[a]	T_p(s)[b]	T_q(s)[c]	T_{q-SRC}(s)[d]	$Acce.ratio_1$[e]	$Acce.ratio_2$[f]
NorthAmerica0	2,927	244	0.068	0.96	2,652	2,762	2,579
NorthAmerica1	4,909	425	0.12	0.88	4,381	4,977	4,380
NorthAmerica2	9,253	831	0.23	0.71	8,318	11,714	8,848
NorthAmerica3	19,637	1721	0.54	0.6	17,668	29,446	15,497
NorthAmerica4	32,138	2750	0.94	0.54	28,953	53,616	19,562
NorthAmerica5	47,982	4092	1.51	0.49	43,369	88,507	21,684
NorthAmerica6	70,926	5947	2.49	0.45	63,883	141,961	21,728
NorthAmerica7	92,781	7729	3.56	0.43	83,635	194,499	20,960

[a] St: the storage required for the auxiliary structure by our method.
[b] T_p: the preprocessing time of our method, which includes both the time for constructing the grids and the time for predetermining the center points.
[c] T_q: the total time for determining all the query points by our method.
[d] T_{q-SRC}: the total time for determining all the query points by the improved ray-crossing method.
[e] $Acce.ratio_1$: the acceleration ratios, computed as $Acce.ratio_1 = (T_{q-SRC} - T_q)/T_q$.
[f] $Acce.ratio_2$: the acceleration ratios, computed as $Acce.ratio_2 = (T_{q-SRC} - T_p - T_q)/(T_p + T_q)$.

a projection-based method which is usually used in practice. It first projects a polygon to a plane by azimuthal projection and then uses a planar ray-crossing method.

When a spherical polygon is projected on the plane, the polygon edges will become planar curves. So the planar polygon obtained by the projection is a polygon with curved edges. To carry out ray-crossing method on the plane, straight edges are generated by connecting the vertices of the planar polygon. Therefore, if a query point's projection falls in the region between a curved edge and its corresponding straight edge, errors may appear, e.g., the point's projection is determined outside the polygon by the straight edge while in fact it is inside the polygon.

Such errors caused by projection were verified by our results. Figure 7 shows the polygon Antarctica and the query points after the azimuthal projection. Figure 7(a) and (b) show the test results by the projection-based method and our method respectively, where inside and outside points are marked by purple and orange points respectively. The regions causing errors are illustrated in green. By the enlarged views, it is clear that points falling in the problem regions are mistakenly classified as outside the polygon. In contrast, all such points are correctly determined by our method as showed by the enlarged view in Fig. 7(b). This is because our method does not use projection which intrinsically avoids such errors. Figure 7(c) shows the projection of mistakenly determined points with the projection based method, which are illustrated by red points. With a statistical investigation, about 15.6% of query points are judged incorrectly by the projection-based method while all the query points are correctly determined by our method.

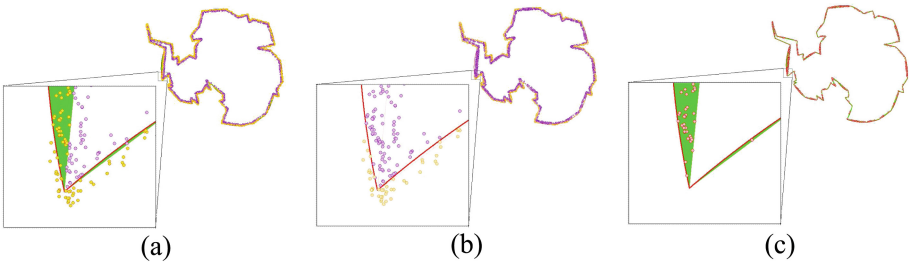

Fig. 7. Results of the accuracy test of the polygon "Antarctica".

4 Conclusion

In summary, we present a fast and robust method for point-in-spherical-polygon tests. By constructing multi-level grids and predetermining grid center points, point-in-spherical-polygon test can be carried out rapidly with local operations. Here, we develop a novel measure to fast determine whether two arc segments are intersected with each other, through computing relative positions of points against a plane. As a result, we can significantly promote point-in-spherical-polygon tests. As shown by experimental results, our robustness and efficiency are attested, and we can even obtain an acceleration of five orders of magnitude over a popularly used method.

Acknowledgment. This work is partially supported by the sub-project of the National Key Research and Development Program of China (No. 2016QY01W0101), the National Natural Science Foundation of China (Nos. 60873182, 61379087, U1435220, and 61661146002), and the EU FP7 funded project AniNex (FP7-IRSES-612627).

References

1. Raskin, R.G.: Spatial analysis on the sphere. Technical report 94-7. University of California, Santa Barbara (1994)
2. Bevis, M., Chatelain, J.-L.: Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. Math. Geol. **21**(8), 811–828 (1989)
3. Nedrich, M., Davis, W.J.: Detecting behavioral zones in local and global camera views. Mach. Vis. Appl. **24**, 579–605 (2013)
4. Bello, L., Coltice, N., Tackley, P.J., Muller, R.D., Cannon, J.: Assessing the role of slab rheology in coupled plate-mantle convention models. Earth Planet. Sci. Lett. **430**, 191–201 (2015)
5. Li, J., Wang, W.-C.: Fast and robust GPU-based point-in-polyhedron determination. Comput. Aided Des. **87**, 20–28 (2017)