COMPUTER GRAPHICS *forum*

DOI: 10.1111/j.1467-8659.2009.01481.x

COMPUTER GRAPHICS *forum*
Volume 28 (2009), number 8 pp. 2264–2274

Robust and Optimized Algorithms for the Point-in-Polygon Inclusion Test without Pre-processing

J. J. Jiménez, F. R. Feito and R. J. Segura

Computer Science Department, University of Jaén, Edificio A3, Campus las Lagunillas s/n. E-23071, Jaén, Spain

Abstract
In this work we present new point inclusion algorithms for non-convex polygons. These algorithms do not perform any pre-processing or any type of decomposition nor features classification, which makes them especially suitable for deformable or moving polygons. The algorithms are more accurate and robust than others in the sense that they consider the inclusion of the point in the vertices and edges of the polygon, and deal with the special cases correctly. In order to perform this inclusion test efficiently, they use the sign of the barycentric coordinates of the test point with regard to the triangles formed by the edges and an origin that depends on the test point. This set of triangles, which is a special simplicial covering of the polygon, is constructed after a transformation of the polygon that simplifies the calculations involved in the inclusion test. Then, an appropriate ordering of the rejection tests allows us to optimize this method. Our algorithms have been tested for robustness and compared with ray-crossing methods, showing a significant improvement.

Keywords: inclusion test, point-in-polygon, rejection test, barycentric coordinates, non-convex polygon

Categories and Subject Descriptors (according to ACM CCS): I.3.5[Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

A fundamental problem in computer graphics and computational geometry is the determination of the containment of a point in a polygon. There are multiple solutions to this problem [Hai94, HS97, HA01], usually classified according to whether they perform some type of pre-processing or not. Among algorithms without pre-processing, we can emphasize the crossings-count algorithm [Hai94, Shi62], a ray-crossing method, and a triangle-based method from Feito et al. [FTU95]. We should also mention algorithms based on the sum of angles [Hai94, Shi62] and on the sign of offset [Tay94].

Many of the algorithms developed for the inclusion problem accelerate their calculations by means of certain type of pre-processing, classifying the edges of the polygon either through some type of arrangement, or by using some type of space decomposition [ZK97]. It is also possible to pre-process certain information of the polygon such as the equation of edges or signs of the simplices, in order to obtain an improvement in the inclusion time when this information does not change.

Performing a classification of the edges by means of some type of space decomposition, or decomposing a complex polygon into convex pieces [ZJRZ03, LWW07], allows to reduce the time complexity. These techniques usually consist of classifying the point into the space decomposition, and only considering for the inclusion test the edges of the polygon classified previously in the same spatial area where the point is situated. These types of techniques imply a pre-processing of the polygon and an additional cost for the storage of the data structures associated with the space decomposition and the classification of the edges. Another pre-processing technique consists of the arrangement of the edges of the polygon [WLW05, MRF06], so that by means of a search on the ordered list of edges the time complexity of the inclusion method can be reduced.

Specific algorithms exist for special cases, such as algorithms for convex or star-shaped polygons [PS85, SE03], usually better than generic algorithms, but in some situations checking if a polygon satisfies this requirement requires great deal of pre-processing time.

Nevertheless, there are some cases in which either the polygon is in movement, as in the case of collision detection in urban environments [TC00], or the polygon can be deformed, as in applications of medical images or microbiology. For this reason certain information that we can consider constant, when the polygon is static, changes between frames, and there is no advantage on pre-calculating this information. In the case of performing space decomposition and classifying the edges of the polygon, the update between frames of the data structures associated is usually more expensive than the inclusion test itself. In other situations the information about the polygon appears suddenly, without enough time to perform any pre-processing, as in the case of image transmission [BE01]. And in others the large amount of information about the polygons and the limited memory make it unfeasible to store the additional information necessary for performing this space decomposition or features classification.

Due to the previous limitations, we present new and optimized triangle-based algorithms for the point inclusion test for non-convex polygon. These algorithms obtain a result exclusively from the vertices of the polygon, without any type of pre-processing or space decomposition, one reason why they are especially appropriate for moving or deformable polygons or for the previous situations. However, they also offer good results for static polygons, being able like other methods to use space decomposition or an arrangement of edges in order to improve the inclusion time.

2. Overview

Our algorithms are based on the use of non-disjoint triangles (simplicial covering of the polygon) formed by a common point (origin of the covering) and each one of the edges of the polygon: on the subsequent inclusion test of the point in each one of these triangles by means of the barycentric coordinates, and on obtaining the number of positive and negative triangles (counterclockwise (CCW) and clockwise (CW) oriented, respectively) in which this point is included. The main advantage of these algorithms consists of performing a transformation that reduces the calculations needed to obtain the sign of the barycentric coordinates of the point with regard to these triangles. In addition, they differ from the previous triangle-based algorithm [FTU95] in the order of the operations performed and the combination of the sign of the barycentric and Euclidean coordinates, instead of the simple study of the area of the triangles; also, a dynamic covering is performed that depends on the test point, instead of a fixed covering based on the origin of the coordinate system.

The method proposed allows us to determine whether the point lies on one of the edges or vertices of the polygon. In addition, it deals with the special cases correctly, obtaining robust algorithms.

In the next section, we propose the new method as a generalization of the previous triangle-based algorithm [FTU95]. We will explain the steps necessary for its optimization, followed by a justification of its robustness. Finally, we will carry out a study in which we verify the robustness of the algorithms developed and compare them with crossings-count [Hai] implementations for moving and deforming polygons.

3. The New Method

The triangle-based algorithm properly considers those special cases in which the test point lies on the border of the polygon, performing a robust calculation based on the signed area of the triangles of the polygon's covering with origin at the origin of coordinates.

However, the nature of the calculations and some optimizations performed by the authors based on the supposition of a rigid and static polygon would mean that these optimizations cannot be performed when the polygon moves through a scene or is deformed dynamically. In addition, the foundation used for the inclusion of points in triangles [SFM*05] is a simplification of a more general theory based on the signed barycentric coordinates [Bad90].

In the next section, we will discuss the generalization of the triangle-based algorithm. In order to do this, we will use the sign of the barycentric coordinates of the test point with regard to the triangles of a special covering of the polygon. In this situation we will obtain an algorithm based on the concept of the sign of the barycentric coordinates and an appropriate ordering of operations in an early-rejection approach. Then, we will perform a specific transformation that allows us to simplify the operations. The appropriate order of operations, the special covering used with the subsequent simplification of calculations, and the early-rejection approach make the new method faster than the previous triangle-based algorithm. Finally, we will obtain other algorithm that combines the calculation of the sign of the barycentric coordinates with the sign of the Euclidean coordinates in a similar approach to the crossings-count method. These algorithms do not need to store any previous information in pre-processing time and are suitable for moving or deformable polygons.

3.1. Generalization of the triangle-based algorithm

Now, let us see the generalization of the triangle-based algorithm using the barycentric coordinates. In order to do this, it is necessary to establish the basis of the point-in-triangle inclusion, distinguishing among several cases, such as points included strictly in the triangle or points lying on its border.

In order to classify a point with regard to a triangle, we will use three unique values that in turn represent the barycentric

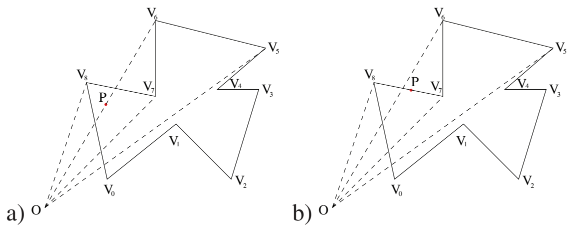

Figure 1. *(a) Inclusion of P on an edge shared by two triangles of the covering. (b) Inclusion of P on an edge.*

coordinates of a point with regard to a triangle in 2D. We will use the classical definition of signed barycentric coordinates (α, β, γ) of a point P with regard to a triangle $V_0V_1V_2$ [Eri05]:

$$\alpha = \frac{|PV_1V_2|}{|V_0V_1V_2|}, \beta = \frac{|PV_2V_0|}{|V_0V_1V_2|}, \gamma = \frac{|PV_0V_1|}{|V_0V_1V_2|}.$$

3.1.1. *Point-in-polygon inclusion*

The triangle-based algortihm [FTU95] is based on the summation of the sign of the triangles of the covering of the polygon in which the test point is situated. The algorithm computes this sign for each one of these triangles, and considers some special cases such as the inclusion on an edge shared by two triangles of the covering (Figure 1(a)), computing $+\frac{1}{2}$ multiplied by the sign of each one of the triangles. When the test point lies on one edge of the polygon (Figure 1(b)), the algorithm returns that the point is inside the polygon. The algorithm adds the values obtained for the sign of each triangle and obtains one inclusion when the sum of these values is equal to one.

The new algorithms improve the triangle-based algorithm and differ from it in the following terms:

- They use the sign of the barycentric coordinates for the point-in-triangle inclusion, which supposes a generalization of the signed area method, these coordinates being independent of the orientation of the triangle.
- They use the sign of the barycentric coordinates for testing the special situations of a point lying on an edge of the polygon or on the border of the triangle, instead of using a different method such as the supporting line equation.
- They group in sets some related situations in which a point is inside or on the border of the triangle: on the one hand, the situation of a point strictly inside the triangle, and on the other hand, the situation of a point lying on an edge or vertex of the polygon, finally, the situations of a point lying on an edge with an extreme on the origin of the covering (original edge). These groupings give fewer conditions.

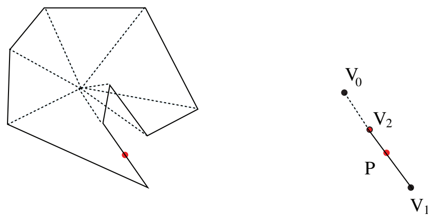

Figure 2. *Degenerate triangle ($V_0V_1V_2$) of the covering of the polygon. The algorithm checks the inclusion of the point P in the edge V_1V_2 by means of a classical algorithm.*

- They use the sets of related situations ordered according to probability. First if the point is outside the triangle, after that if the point is strictly inside the triangle, next if the point is lying on the border of the polygon, and finally if the point is on an original edge.
- They choose the appropriate origin of the triangle covering, so that the determinants obtained in the calculation of the barycentric coordinates are simplified.
- They deal properly with the case of a point situated on the origin of the covering. In this situation it is not possible to calculate the inclusion state by means of the previous triangle-based method.
- They consider the special cases of triangles with zero signed area (Figure 2).
- They optimize the calculation of the sign of barycentric coordinates by obtaining the signs of determinants.

3.2. Decreasing calculations and optimizations

The previous improvements mean an increase in the speed of the algorithms. Nevertheless, the fundamental advance consists of arranging the operations in a suitable order for a quick rejection of most of the triangles, as well as performing a transformation that allows us to reinterpret the point-in-triangle inclusion. These improvements decrease the calculations needed to obtain the determinants for the sign of the barycentric coordinates. There is no need for calculating the barycentric coordinates, only their signs matter. We will see this advance in two distinct algorithms, one in a purely triangle-based style based on the ordering of the calculations, and the other in a combination of the sign of the barycentric and Euclidean coordinates.

3.2.1. *Order of calculations*

One improvement of this method consists of grouping the different situations in different sets with a common result and ordering these sets. We can choose an approach consisting of dealing first with the early inclusion test or dealing first with the early rejection test of each triangle.

If we analyze the nature of polygons, we observe that for most of them at any given test point is included in a

Table 1: *Different cases for the determinants involved in the barycentric calculation and result of the point in triangle inclusion.*

Signs	Sign($\lvert OV_iV_j\rvert$)	Result	Interpretation
$+ + -$			
$+ - +$			
$- + -$			
$- - +$			
$0 + -$			
$0 - +$	Indifferent	$+0$	P out OV_iV_j
$- + +$			
$+ - -$			
$- + 0$			
$+ - 0$			
$+ 0 -$			
$- 0 +$			
$+ + +$	$+$	$+2$	P on OV_iV_j
$- - -$	$-$	-2	P on OV_iV_j
$+ + 0$	$+$	$+1$	P on OV_i
$+ 0 +$	$+$	$+1$	P on OV_j
$- - 0$	$-$	-1	P on OV_i
$- 0 -$	$-$	-1	P on OV_j
$0 + +$	$+$		P on V_iV_j
$0 + 0$	$+$		P on V_i
$0 0 +$	$+$	Return(1)	P on V_j
$0 - -$	$-$		P on V_iV_j
$0 - 0$	$-$		P on V_i
$0 0 -$	$-$		P on V_j
$+ 0 0$	$+$	?	P on O
$- 0 0$	$-$	?	P on O
$0 0 0$	0	If (P in V_iV_j) return(1)	Degenerate

These cases are grouped and ordered by probability. 'Signs' represents the sign of $|PV_iV_j|$, $|OPV_j|$ and $|OV_iP|$.

small fraction of the number of triangles of the covering. For example, for convex or star-shaped polygons with the origin of the covering situated at the centroid of the polygon, a test point will be included only in one triangle independently of the number of edges. For non-convex polygons with a few number of concavities (these types of polygons are typical in geographical information system (GIS) for the representation of geographical regions), some test points are inside several triangles. Only in non-convex polygons with a great number of concavities and for specific positions for the test point does the number of triangles in which the point is included exceed the number of triangles in which the point is not included (i.e. saw-shaped polygons).

Due to the previous reason, we have chosen an early rejection test order, in which we first discard triangles in which the point is not included. Certainly, this approach works best for a large number of edges.

In Table 1 we can observe the sign of the determinants involved in the calculation of the barycentric coordinates

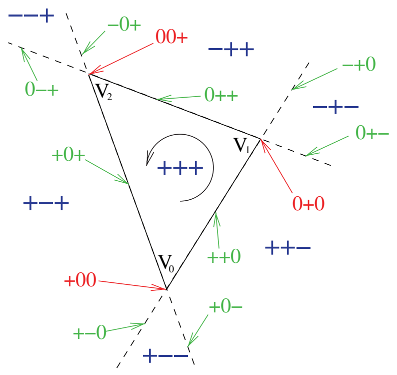

Figure 3. *Signs of the numerator of the determinants involved in the barycentric calculation for a positive triangle.*

of a point with regard to a triangle (Figure 3). These signs have been grouped by the result obtained and arranged by probability from top to bottom.

From this table we can obtain the following conclusions:

- The sign of the numerator is directly obtained using the expression $|PV_iV_j| + |OPV_j| + |OV_iP| = |OV_iV_j|$.
- When there are simultaneously two different signs (positive and negative) in the numerator of two of the barycentric coordinates the sign of the denominator is not deduced directly, but this sign is not needed to determine that the point is in effect outside the triangle.
- We only need the sign of the numerators to obtain the position of the test point with regard to the triangle when the point is inside the triangle or lies on its border.
- The combination of signs of the numerators gives us a result for the point-in-polygon test with two degenerations, one when the test point lies on the origin O of the covering (this case can be avoided by choosing a different point for O), and one when O, V_i and V_j are aligned, in which case the algorithm checks if the test point is inside V_iV_j.

If we apply these observations and Table 1 to the barycentric coordinates theory, we obtain Algorithm 1 [JFS]. In this algorithm the sequence *if-else if* is used in order to avoid, where possible, the evaluation of the other conditions.

3.2.2. *A dynamic covering*

As origin of the covering, we could use the point $O = (0, 0)$ so that the determinants obtained in the calculation of the barycentric coordinates are simplified. This configuration may be applied to Algorithm 1, obtaining a simplification

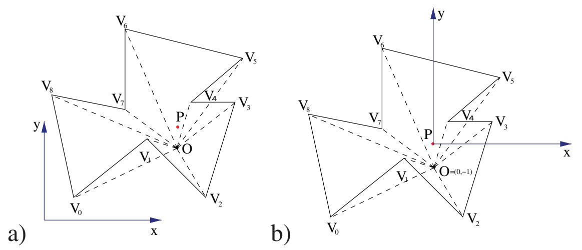

Figure 4. *(a) Covering with $O = (x_p, y_p - 1)$. (b) Covering after the translation $T(-x_p, -y_p)$.*

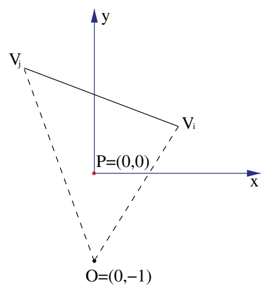

Figure 5. *Position of the test point P with regard to the triangle OV_iV_j in the actual configuration.*

of the calculations. So, the test point is a variable and additional configurations could be generated and optimized.

Nevertheless, instead of a fixed point, we can use as origin of the covering a point that depends on the test point. In this way we can assert that the test point will never be located at the origin of the covering, it not being necessary to contemplate this special situation.

Given a test point $P = (x_p, y_p)$, we use as the origin of the covering the point $O = (x_p, y_p - 1)$ (Figure 4(a)). If we then apply a transformation to the scene in which the test point will be located on the origin of coordinates, that is to say $T(-x_p, -y_p)$, we obtain a configuration in which the new test point is $P = (0, 0)$ and the new origin of the covering is the point $O = (0, -1)$ (Figure 4(b)).

Because the simplicial covering depends on the test point, it is not possible to previously store some information like the sign of the triangles of the covering. When the origin of the covering changes between frames, the triangles of the covering and possibly their signs change. This will not be a disadvantage because the calculations are simplified.

In the situation described, the determinants involved in the calculation of the barycentric coordinates of the test point P with regard to a triangle $T = OV_iV_j$ of the covering are the following (Figure 5):

$$|OV_iV_j| = \frac{1}{2}\begin{vmatrix} 0 & x_i & x_j \\ -1 & y_i & y_j \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{2}(x_i y_j - x_j y_i + x_i - x_j)$$

$$|PV_iV_j| = \frac{1}{2}\begin{vmatrix} 0 & x_i & x_j \\ 0 & y_i & y_j \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{2}(x_i y_j - x_j y_i)$$

$$|OPV_j| = \frac{1}{2}\begin{vmatrix} 0 & 0 & x_j \\ -1 & 0 & y_j \\ 1 & 1 & 1 \end{vmatrix} = -\frac{1}{2}x_j$$

$$|OV_iP| = \frac{1}{2}\begin{vmatrix} 0 & x_i & 0 \\ -1 & y_i & 0 \\ 1 & 1 & 1 \end{vmatrix} = \frac{1}{2}x_i$$

When calculating the sign of these expressions, the $\frac{1}{2}$ multiplication can be avoided.

By means of this transformation, the calculations needed for the point-in-triangle have been simplified and some calculations could be reused (i.e. $x_i y_j - x_j y_i$). In the implementation of the Algorithm 1 [JFS], we apply the transformation $T(-x_p, -y_p)$ to each one of the vertices of the polygon and we replace the sign of the determinants with the previous expressions. In this case the position of P on O is impossible and this situation is not considered.

3.2.3. *Barycentric and Euclidean coordinates*

We can obtain an algorithm in a different manner to that used for Algorithm 1 also using the dynamic covering. In order to do this, instead of verifying the point inclusion in the triangles of the covering by only using the sign of the barycentric coordinates, we also use the Euclidean coordinates. This approach gives us a mechanism for obtaining a new algorithm for the point-in-polygon test, which brings the triangle-based method and the crossings-count algorithm closer together. We can say that from a triangle-based approach we obtain a robust and efficient algorithm that is similar to the crossings-count algorithm in the operations performed but that considers all the special cases.

Let us analyze each one of the situations of P with regard to the triangle of the covering OV_iV_j, to obtain an algorithm that discards triangles that do not satisfy these conditions sequentially:

- **Situation 1:** When $\beta < 0$ or $\gamma < 0$, the point is not inside the triangle. This situation occurs when edge V_iV_j does not cross the y-axis. This is equivalent to checking if ($x_i > 0$ and $x_j > 0$) or ($x_i < 0$ and $x_j < 0$), or equally if $x_i \cdot x_j > 0$ (Figure 6(a)).

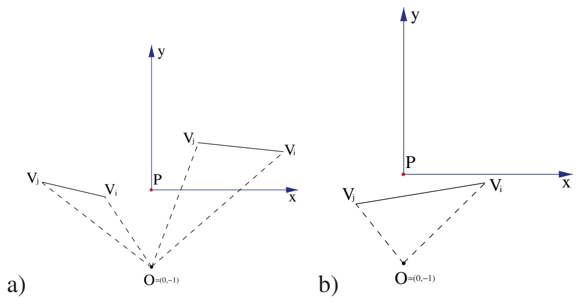

Figure 6. *(a) Triangles that do not cross y-axis. (b) Triangles that are under x-axis.*

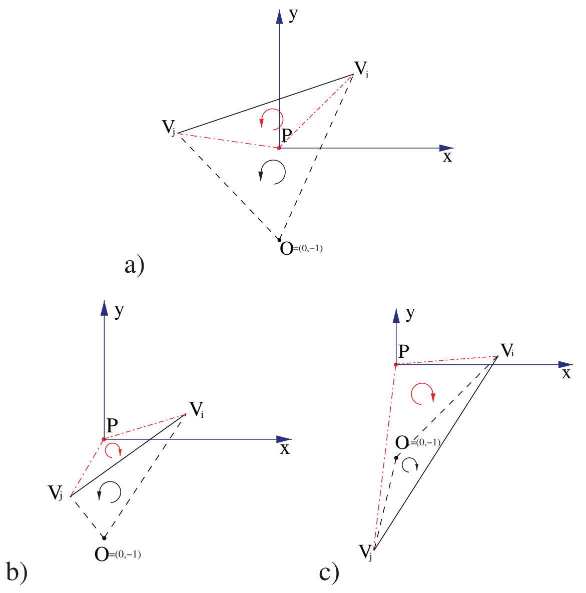

Figure 7. *$x_i > x_j$. (a) V_iV_j is above P and O. (b) V_iV_j is between P and O. (c) V_iV_j is under P and O.*

- **Situation 2:** There is no inclusion of P in the triangle for triangles with negative y coordinates, that is to say, if $y_i < 0$ and $y_j < 0$ (since $y_0 = -1$) (Figure 6(b)).

Now, we calculate the sign of α for triangles not discarded in Situations 1 and 2. In order to do this, we need the signs of $|OV_iV_j|$ and $|PV_iV_j|$, but we can avoid this calculation if we consider some additional situations:

- If $x_i > x_j$, we examine the position of edge V_iV_j respect to points P and O. The situations of V_iV_j are above P and O (Figure 7(a)), between P and O (Figure 7(b)), under P and O (Figure 7(c)). The situations of V_iV_j on P or O will be seen later. The first situation fulfils $|OV_iV_j| > 0$ and $|PV_iV_j| > 0$, the second one $|OV_iV_j| > 0$ and $|PV_iV_j| < 0$, the third one $|OV_iV_j| < 0$ and $|PV_iV_j| < 0$. We can see that P is inside OV_iV_j iff $|PV_iV_j| > 0$ and $x_i > x_j$ (Figure 7(a)). When $|PV_iV_j| < 0$ and $x_i > x_j$, P is outside the triangle independently of the sign of $|OV_iV_j|$.
- If $x_i < x_j$, we obtain similar conclusions than the previous situation, interchanging the points V_i and V_j.
- If $x_i = x_j$, we obtain a degenerate triangle, because $|OV_iV_j| = 0$. There are three situations: P on V_iV_j (Figure 8(a) and (b)), P under V_iV_j (Figure 8(c)) or P above V_iV_j (Figure 8(d)). The last case was discarded in Situation 2. We can see that P is on V_iV_j iff ($y_i \leq 0$ or $y_j \leq 0$) and $x_i = x_j$.

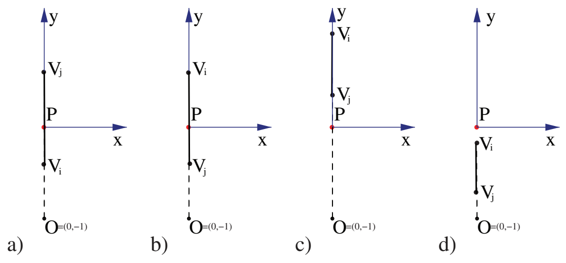

Figure 8. *Degenerate triangles ($x_i = x_j$). (a) and (b) P lies on edge V_iV_j. (c) P lies under edge V_iV_j. (d) P lies above edge V_iV_j (discarded in Situation 2).*

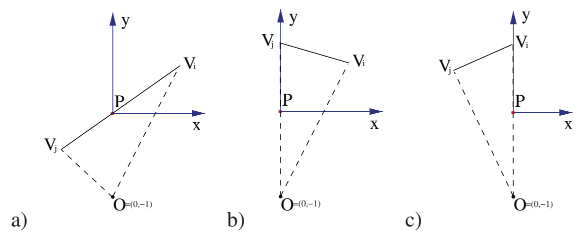

Figure 9. *Point on an edge. (a) P lies on an edge of the polygon. (b) P lies on edge OV_j. (c) P lies on edge OV_i.*

Finally, we analyze some special cases contemplated in the algorithm. Those cases in which P lies on one edge of the triangle, or O lies on the edge V_iV_j:

- P lies on V_iV_j (including their vertices) when $\alpha = 0$, that is to say, when $|PV_iV_j| = 0$ and the triangle was not discarded in Situations 1 and 2 (Figure 9(a)).
- P lies on OV_j (not including their vertices) when $x_j = 0$ and the triangle was not discarded in previous situations (Figure 9(b)).

- P lies on OV_i (not including their vertices) when $x_i = 0$ and the triangle was not discarded in previous situations (Figure 9(c)).
- When O lies on V_iV_j (including their vertices), OV_iV_j is a degenerate triangle. This case should not be considered especially because although $|OV_iV_j| = 0$ the previous conditions 'if $|PV_iV_j| < 0$ and $x_i < x_j$' and 'if $|PV_iV_j| > 0$ and $x_i < x_j$' deal with this situation and return a correct result.

The algorithm for the point-in-polygon inclusion test that is able to cope with all previous situations (Algorithm 2) and its implementation can be seen in [JFS].

3.3. Robustness

The new algorithms are more robust than traditional algorithms for the point-in-polygon problem from a numerical point of view. The numeric robustness of an algorithm can be seen from the precision of the algorithms and from the treatment of degenerate cases.

Precision errors could occur in the determinant calculations, in the comparisons and in accumulations:

- If we reduce the number of terms for the determinant calculation the precision increases. If we use the dynamic covering proposed, to obtaining two of the determinants ($|OPV_j|$ and $|OV_iP|$), no operation is needed, for the determinant $|PV_iV_j|$ the number of calculations is reduced, and the determinant $|OV_iV_j|$ is not necessary.
- The errors committed in the determinant $|PV_iV_j|$ or the errors due to the numeric representation used could cause an error in the comparisons. These errors could cause an incorrect classification of the point in the border of the triangles of the covering, but these errors do not cause an incorrect classification in the polygon. If the point lies on edge OV_i or edge OV_j and is incorrectly classified as being inside an adjacent triangle then the algorithms return a correct result. If the point lies on edge V_iV_j but is incorrectly classified, it could cause a precision error, which may be minimized by using tolerances [GSS89].

Degenerate cases are obtained when the determinant of a triangle is zero (three points aligned) or near zero (slim triangles or triangles where all three points are near the same). The situation of degenerate triangles is suitably dealt with in the algorithms. Since the determinant of the triangle $|OV_iV_j|$ is not used, only slim triangles with points situated near the y-axis are dealt with. In this situation if an error in the calculation of the inclusion of the test point on the edges OV_i or OV_j occurs, a correct result is obtained as we have seen previously. The inclusion of the point on the almost vertical edge V_iV_j is resolved using the sign of the y coordinates. If all three points are almost the same, the algorithms discard the triangle because the test point is situated one unit from the origin O, not being possible its inclusion in the triangle.

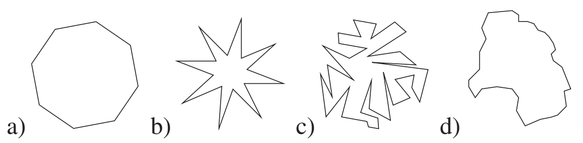

Figure 10. *Polygons used in tests. (a) Convex. (b) Star-shaped. (c) Non-convex with a great number of concavities. (d) Non-convex with a few number of concavities.*

4. Experimental Results and Discussion

We have tested the new algorithms in two ways. In the first we check their robustness. In the second we carry out a comparative study between the crossings-count algorithm in two versions: original [Hai94] and an improved version by its authors [Hai] in which division is not performed (crossings-multiply), and the new algorithms (Algorithms 1 and 2 [JFS]).

For robustness, two sets of polygons have been generated, one set in integer coordinates and other in real coordinates. For both sets of polygons, two additional sets of test points have been generated, one defined in integer and the other in real coordinates. These test points have been situated in strategic positions, covering all possible cases and degenerations, such as points situated inside or outside the polygon, on vertices or edges, on edges shared by two triangles, on degenerate triangles or slim triangles, and test points in which the origin of the covering lies on a vertex or edge.

For timing we have generated a set of polygons with different numbers of vertices. A translation followed by a small rotation and deformation has been performed on the polygon for each random test point. The deformation consists of moving some vertices randomly. For each polygon we have generated 1 000 000 random points within its bounding box and we have measured the improvement obtained in the inclusion test by the new algorithm against classical algorithms. We discard points that are outside the bounding box with a simple test. This test has not been included in the time study. In Figure 10 we can see different types of polygons used in our tests: convex and non-convex polygons with different degrees on the number of concavities.

The algorithms have been implemented in C language using an Intel Pentium IV with 1.6 GHz processor on a Linux system with gcc and compiler optimization O2, and using an Intel Core 2 Duo 2.4 GHz processor on a Windows system with Microsoft Visual Studio and the compiler optimization SSE2. For crossings-count and crossings-multiply algorithms the implementation of [Hai] has been used. In

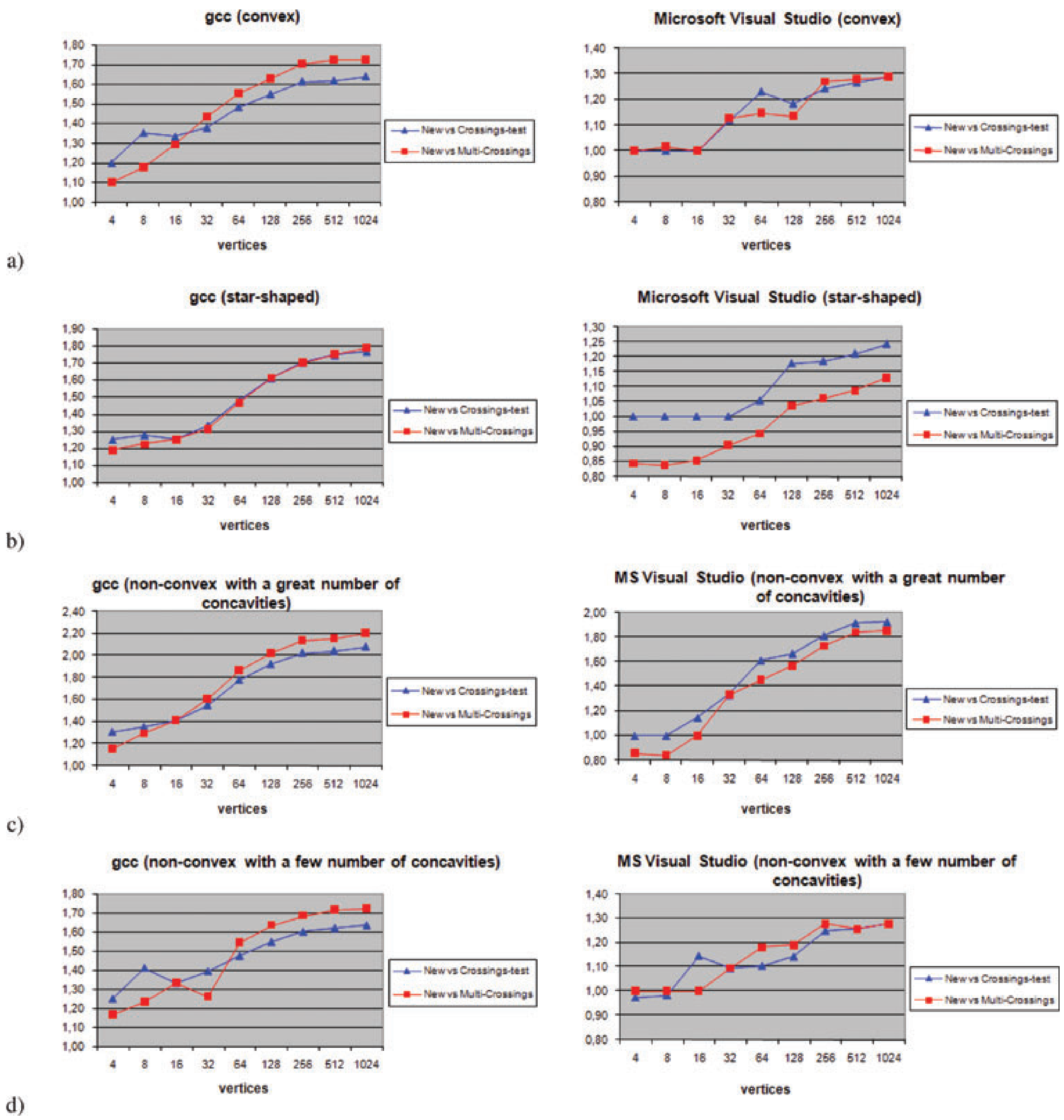

Figure 11. *Ratio improvement of the new algorithm with regard to crossings-test and multi-crossings algorithms. In x-axis we show the number of vertices of the polygon. On the left we show the improvement using gcc on a Linux system. On the right we show the improvement using Microsoft Visual Studio on a Windows system.*

Figure 11 we can see the improvement ratio obtained. As the implementation of the algorithms developed obtains similar times, we only show the times obtained for the average of both implementations (named 'new' algorithm).

In the robustness test we do not find inconsistencies or test points with anomalous results with the new algorithms in a percentage superior to 99.88%. The errors are due to the existence of some points classified incorrectly in the edges of the polygon (as mentioned in Section 3.3) because we do not use tolerances in the implementations.

Using the gcc compiler, we obtain a considerable improvement compared with ray-crossing algorithms. The new algorithms obtain a greater improvement with a higher number of vertices of the polygon as expected. This improvement with regard to the crossings-multiply algorithm ranges on average from approximately a ratio of 1.10 for polygons with 4 vertices to approximately 1.80 for polygons with 1024 vertices.

Using Microsoft Visual Studio compiler, the improvement is not as high as with the gcc compiler. However, in most cases the improvement is greater than with classical algorithms and ranges approximately from a ratio of 1.10 to 1.30 in average with regard to the crossings-multiply algorithm, except for non-convex polygons with a great number of concavities that achieve a greater improvement (up to a ratio of 1.85) for a large number of vertices, and for star-shaped polygons in which for fewer than 64 vertices the crossings-multiply obtains better times than the new algorithms.

5. Conclusions

We have developed new algorithms for the point-in-polygon inclusion test, especially suitable for situations in which a pre-processing of the polygon cannot be performed or for those in which this pre-processing does not suppose any advantage when the polygon moves or is deformed. We have carried out a robust and efficient implementation of these algorithms. These implementations are robust because special cases have been dealt with suitably. The algorithms are efficient because they obtain a significant improvement with regard to classical algorithms, being efficient for rigid and static polygons and for deformable or moving polygons. This is mainly because the algorithms do not pre-calculate any type of information, which makes them especially suitable for applications such as collision detection or deformable models.

The improved calculation of the sign of the barycentric coordinates, the appropriate order of operations in an early-rejection approach, and the dynamic covering with the subsequent simplification of operations, allowed us to develop optimized algorithms faster than the previous triangle-based algorithm. Algorithm 1 is more general and operations could be redefined by changing the dynamic origin of the covering selected or simply using a fixed origin. With the appropriate parameters it obtains nearly the same times as Algorithm 2, which is an algorithm that approaches the crossings-count algorithm in calculations, but the order of operations and the on-edge condition obtain a robust and efficient algorithm from the triangle-based algorithm.

The algorithms proposed could be extended to 3D, in order to obtain an efficient and robust point-in-polyhedron test. In this algorithm the sign of the barycentric coordinates of the test point with regard to the tetrahedra of the simplicial covering could be optimized in a similar way.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Education and Science and the European Union (via ERDF funds) through the research project TIN2007-67474-C03-03, by the Consejería de Innovación, Ciencia y Empresa of the Junta de Andalucía through the research projects P06-TIC-01403 and P07-TIC-02773, and by the University of Jaén through the research project UJA-08-16-02, sponsored by Caja Rural de Jaén.

Appendix

Algorithm 1: *Point-in-polygon inclusion algorithm by means of the interpretation of the sign of the barycentric coordinates and the appropriate ordering of the calculations.*

```
Input: polygon pol, test-point P, origin-point O
Output: inclusion state: 1 inside, 0 outside
inclusion = 0
if (P = O) choose a different O
for k=0 to number of vertices of pol do begin
  OViVj = triangle formed by O and edge k of pol
  /* reject if + and − : (?+−) or (?−+) */
  if (|OPVj|*|OViP| >= 0 ) then
    /* reject if (+−?) or (−+?) */
    if (|OPVj|*|PViVj| >= 0) then
      /* reject if (+?−) or (−?+) */
      if (|OViP|*|PViVj| >= 0 ) then
        if (|PViVj| > 0 and
        |OPVj| > 0 and |OViP| > 0 ) then
          inclusion = inclusion + 2 /* (+++) */
        else if (|PViVj| < 0 and
        |OPVj| < 0 and |OViP| < 0 ) then
          inclusion = inclusion − 2 /* (− − −) */
        else if (|PViVj| > 0 ) then
          /*(++0)or(+0+);(+00)not possib.*/
          inclusion = inclusion + 1
        else if (|PViVj| < 0 ) then
          /*(− −0)or(−0−);(−00)not possible*/
          inclusion = inclusion − 1
        else if (|OPVj| <> 0 or |OViP| <> 0 ) then
          return (1) /* (0??) */
        else if (P.in(ViVj) ) then
          return (1) /* (000) */
end
if (inclusion = 2) then return (1)
else return (0)
```

Algorithm 2: *Point-in-polygon inclusion algorithm by means of barycentric and Euclidean coordinates.*

```
Input: polygon pol, test-point P
Output: inclusion state: 1 inside, 0 outside
inc = 0
for all vertices of pol do translate(−P)
for k=0 to number of vertices of pol do begin
   (xi,yi) = coords of vertex Vi of edge k of polygon pol
   (xj,yj) = coords of vertex Vj of edge k of polygon pol
   /* reject triangles with $\beta < 0$ or $\gamma < 0$ */
   if (xi*xj <= 0) then
      /* reject triangles under x-axis */
      if (yi >= 0 or yj >= 0) then
         if (xi > xj) then begin
            a = xi*yj ; b = xj*yi
            if (a > b) then
               if (xi = 0 or xj = 0) then
                  inc = inc+1 /*OVi or OVj*/
               else inc = inc+2 /* OViVj */
            else if (a = b) then
               return (1) /* ViVj or Vi or Vj */
         end
         else if (xi < xj) then begin
            a = xi*yj ; b = xj*yi
            if (a < b) then
               if (xi == 0 or xj = 0) then
                  inc = inc−1 /*OVi or OVj*/
               else inc = inc−2 /* OViVj */
            else if (a = b) then
               return (1) /* ViVj or Vi or Vj */
         end
         else if (yi <= 0 or yj <= 0) then
            return (1) /* ViVj (xi=xj) */
end
if (inc = 2) then return (1)
else return (0)
```

References

[Bad90] Badouel D.: *An Efficient Ray-Polygon Intersection. Graphics Gems*. Academic Press, 1990.

[BE01] Bertolotto M., Egenhofer M. J.: Progressive transmission of vector map data over the world wide web. *Geoinformatica 5*, 4 (2001), 345–373.

[Eri05] Ericson C.: *Real-Time Collision Detection*. Morgan Kaufmann Publishers. Elsevier, 2005.

[FTU95] Feito F. R., Torres J. C., Ureña A.: Orientation, simplicity and inclusion test for planar polygons. *Comput. Graph. 19*, 4 (1995), 595–600.

[GSS89] Guibas L., Salesin D., Stolfi J.: Epsilon geometry: Building robust algorithms from imprecise computations. In *Proceedings of the 5th Annual ACM Symposium on Computational Geometry* (1989).

[HA01] Hormann K., Agathos A.: The point in polygon problem for arbitrary polygons. *Comput. Geom. Theory App. 20*, 3 (2001), 131–144.

[Hai] Haines E.: *CrossingsMultiplyTest*. http://www.graphicsgems.org/gemsiv/ptpoly_haines/ptinpoly.c.

[Hai94] Haines E.: *Point in Polygon Strategies. Graphics Gems IV*. Academic Press, 1994.

[HS97] Huang C., Shih T.: On the complexity of point-in-polygon algorithms. *Comput. Geosci. 23*, 1 (1997), 109–118.

[JFS] Jimenez J., Feito F., Segura R.: *Robust and Optimized Algorithms for the Point-in-Polygon Inclusion Test without Pre-processing (Appendix: Code & Algorithms).* http://wwwdi.ujaen.es/̃juanjo/inclusion/ptinpoly.c, http://wwwdi.ujaen.es/̃juanjo/inclusion/ptinpoly.pdf.

[LWW07] Li J., Wang W., Wu E.: Point-in-polygon tests by convex decomposition. *Comput. Graph. 31*, 4 (2007), 636–648.

[MRF06] Martinez F., Rueda A. J., Feito F. R.: The multi-l-rep decomposition and its applications to a point-in-polygon inclusion test. *Comput. Graph. 30*, 6 (2006), 947–958.

[PS85] Preparata F. P., Shamos M. I.: *Comput. Geom.* Springer-Verlag, 1985.

[SE03] Schneider P. J., Eberly D. H.: *Geometric Tools for Computer Graphics*. Morgan Kaufmann Publishers. Elsevier Science, 2003.

[SFM*05] Segura R. J., Feito F. R., Miras J. R., Ogayar C., Torres J. C.: An efficient point classification algorithm for triangle meshes. *J. Graph. Tools 10*, 3 (2005), 27–35.

[Shi62] Shimrat M.: Algorithm 112, position of point relative to polygon. *CACM 5*, 8 (1962), 434.

[Tay94] Taylor G.: Point in polygon test. *Sur. Rev. 32* (1994), 479–484.

[TC00] Tecchia F., Chrysanthou Y.: Real-time visualisation of densely populated urban environments: a simple and fast algorithm for collision detection. In *Proc. Eurographics Workshop on Rendering Techniques 2000* (2000), pp. 83–88.

[WLW05] WANG W., LI J., WU E.: 2D point-in-polygon test by classifying edges into layers. *Comput. Graph. 29*, 3 (2005), 427–439.

[ZJRZ03] ZALIK B., JEZERNIK A., RIZMAN ZALIK K.: Polygon trapezoidation by sets of open trapezoids. *Comput. Graph. 27*, 5 (2003), 791–800.

[ZK97] ZALIK B., KOLINGEROVA I.: A cell-based point-in-polygon algorithm suitable for large sets of points. *Comput. Geosci. 23*, 1 (1997), 109–118.