Computational Geometry 43 (2010) 474–492

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

A robust segment/triangle intersection algorithm for interference tests. Efficiency study

Juan J. Jiménez *, Rafael J. Segura, Francisco R. Feito

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

ARTICLE INFO

Article history:
Received 30 January 2008
Accepted 12 October 2009
Available online 14 October 2009
Communicated by K. Mehlhorn

Keywords:
Segment/triangle intersection
Barycentric coordinates
Rejection test
Interference test
Collision detection

ABSTRACT

In this paper, a new algorithm for the intersection between a segment and a triangle in 3D is presented. This algorithm is suitable for interference tests between moving polyhedral solids, as is shown in the times study. In this situation it is necessary to determine whether the interference between triangles takes place (boolean test), and in some applications to determine the intersection point. When solids move certain changing information, such as the triangle normal, cannot be stored so the algorithms cannot take advantage of pre-calculations. A set of tests and a study of the computational cost of the new algorithm compared with classical algorithms are provided. These algorithms and studies can be used and extended by programmers in real applications.

1. Introduction

There are many problems in Computer Graphics which are ultimately solved by determining the intersection between a segment and a triangle, because of the fact that the representation of graphic objects usually uses triangle meshes or decompositions of the objects by means of simplices [1,2]. This is the case of applications such as: ray casting, ray tracing, inclusion test, boolean operations between solids and collision detection.

Furthermore, in many problems it is not necessary to obtain the intersection point between a ray and a triangle, but rather a simple boolean test which determines whether an intersection takes place or not.

The improvement of the efficiency of these algorithms is based, principally, on obtaining a segment/triangle intersection test which quickly rejects this intersection, and calculates the intersection point in an optimum way if necessary.

In ray casting and ray tracing problems [3], this segment is replaced by a ray with a common origin point, which is placed at the observer position. Many efficient algorithms have been developed for these types of application. These algorithms are very fast if the rays have a common origin point and the scene is invariant for each ray, due to the possibility of storing pre-calculated information for all rays, such as the triangle normal for Badouel's algorithm or the signed volume of tetrahedra for Segura's.

The problem we wish to solve lies in determining the interference between two moving elements, a segment and a triangle, in a scene [4]. This problem has been traditionally solved by adapting classical ray/triangle algorithms to segment/triangle problems. For interference between polyhedra these speedy algorithms can be improved, since the adaptation of classical algorithms to new scenarios does not take advantage of specific features for these types of scenarios.

* Corresponding author.
E-mail addresses: juanjo@ujaen.es (J.J. Jiménez), rsegura@ujaen.es (R.J. Segura), ffeito@ujaen.es (F.R. Feito).
URL: http://wwwdi.ujaen.es/~gigjaen/ (J.J. Jiménez).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2009.10.001

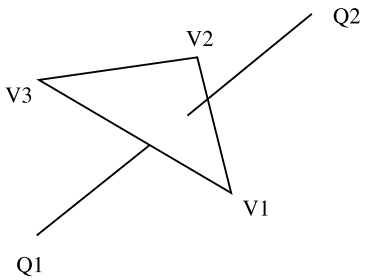

Fig. 1. Segment/triangle intersection, nomenclature.

There are several solutions to the ray/triangle or segment/triangle intersection problem. Badouel [5] designed a solution based on finding the intersection between the ray and the plane of the triangle and the subsequent projection on a plane (xy, yz or zx), and computing whether the point is inside the triangle by using 2D barycentric coordinates. This solution has a similar performance to Möller's, especially for solids formed by many triangles. On the other hand, Möller [6] developed an algorithm based on the solution of the equation system obtained with the ray equation and the equation for obtaining the intersection point between a ray and a triangle using 2D barycentric coordinates. This algorithm could be considered the fastest one up to now. Segura [7] proposed a segment/triangle intersection algorithm. This algorithm computes the sign of the volume of the tetrahedra formed by the triangle vertices and the endpoints of the segment. The boolean determination of intersection is calculated, but the intersection point has to be calculated by means of a classical ray/plane intersection algorithm. This algorithm proved in practice to be more efficient than others for static objects [8], but not so efficient for the case of moving objects because the original version uses some improvements based on precalculating information about the triangles, like the triangle normal or the signed volume of tetrahedra, which cannot be pre-calculated in the case of moving polyhedra. Other solutions include the use of Plücker coordinates [9] or optimizations for ray-tracing like the solution adopted by Kensler et al. [10].

In this work a new algorithm for the intersection between a segment and a triangle is presented, based on the sign of the barycentric coordinates of a point with regard to a tetrahedron. This algorithm is more efficient than others for several applications because it requires fewer calculations for the intersection determination. Furthermore, it is robust and stable because the type of calculations performed are based on signs of 3D barycentric coordinates, it avoids the implicit division operation for determining whether or not an intersection occurs, and it uses tolerance comparisons. The calculation of the intersection point can also be performed with a small computational cost. This algorithm, as will be seen later, is more suitable than previous ones for certain situations, especially when it is used for interference or collision detection [11] between a pair of objects.

In this paper the new algorithm is compared with classical ray/triangle and segment/triangle intersection algorithms. In order to carry out an adequate time study, the algorithms used have been transformed so that they calculate the segment/triangle intersection point and do not store any other information apart from the vertices of the triangle and the segment. Throughout this paper the number of calculations performed by the algorithms will be measured for several kinds of problems as well as the suitability of the new algorithm for certain types of applications. Finally a temporal study will be carried out for some scenarios which will show the performance of the implementation. This study will show that the algorithm developed is more efficient for interference tests and for different hit rates.

2. Previous algorithms

In this section, some of the most significant and efficient algorithms used for the ray/triangle or segment/triangle intersection are described (the term $V_1V_2V_3$ will be used for the triangle vertices, and Q_1Q_2 for the extremes of the segment, both in $\Re^3$ as can be seen in Fig. 1). We also analyze the advantages and disadvantages of these algorithms in different situations.

2.1. Badouel's algorithm

The basis of Badouel's algorithm (Algorithm 1) is shown now. First, this algorithm determines the intersection of the ray and the plane of the triangle. Next, the intersection point and the triangle are projected to xy-, yz- or zx-plane, in order to solve the point inclusion in the 2D triangle by using barycentric coordinates (Fig. 2).

Let $V_1V_2V_3$ be a triangle. The position of a point P inside a triangle can be expressed as:

$$\overrightarrow{V_1P} = \alpha \cdot \overrightarrow{V_1V_2} + \beta \cdot \overrightarrow{V_1V_3} \tag{1}$$

If $\alpha \geqslant 0$, $\beta \geqslant 0$, $\alpha + \beta \leqslant 1$, then P is inside the triangle. This equation can be broken down as follows:

INPUT: Triangle (V_1, V_2, V_3), Segment(Q_1, Q_2)

Without Back-Face culling

$E_1 = V_2 - V_1$
$E_2 = V_3 - V_1$
$N = E_1 \times E_2$
$D = Q_2 - Q_1$
$det = N \cdot D$
if (det > - ε && det < ε) return 0 rejection 1
$T = V_1 - Q_1$
$t\_param = N \cdot T / det$
if (t_param<0.0 || t_param>1.0) return 0 rejection 2
if ($N_{(x)} > N_{(y)}$)
    if ($N_{(x)} > N_{(z)}$) j=y, k=z
    else j=x, k=y
else
    if ($N_{(y)} < N_{(z)}$) j=x, k=y
    else j=x, k=z
$P_{(j)} = Q_{1(j)} + D_{(j)} * t\_param$
$P_{(k)} = Q_{1(k)} + D_{(k)} * t\_param$
$u_1 = P_{(j)} - V_{1(j)}$
$v_1 = P_{(k)} - V_{1(k)}$
$u_2 = V_{2(j)} - V_{1(j)}$
$u_3 = V_{3(j)} - V_{1(j)}$
$v_2 = V_{2(k)} - V_{1(k)}$
$v_3 = V_{3(k)} - V_{1(k)}$
if ($u_2 > -\varepsilon$ && $u_2 < \varepsilon$) {
    $\beta = u_1 / u_3$
    if ($\beta < 0$ || $\beta > 1$) return 0 rejection 3
    $\alpha = (v_1 - \beta * v_3) / v_2$
} else {
    $\beta = (v_1 * u_2 - u_1 * v_2) / (v_3 * u_2 - u_3 * v_2)$
    if ($\beta < 0$ || $\beta > 1$) return 0 rejection 3
    $\alpha = (u_1 - \beta * u_3) / u_2$
}
if ($\alpha < 0$ || $\alpha+\beta > 1.0$) return 0 rejection 4
return 1

With Back-Face culling

$E_1 = V_2 - V_1$
$E_2 = V_3 - V_1$
$N = E_1 \times E_2$
$D = Q_2 - Q_1$
$det = N \cdot D$
if (det > - ε) return 0 rejection 1
$T = V_1 - Q_1$
$t\_param = N \cdot T / det$
if (t_param<0.0 || t_param>1.0) return 0 rejection 2
if ($N_{(x)} > N_{(y)}$)
    if ($N_{(x)} > N_{(z)}$) j=y, k=z
    else j=x, k=y
else
    if ($N_{(y)} < N_{(z)}$) j=x, k=y
    else j=x, k=z
$P_{(j)} = Q_{1(j)} + D_{(j)} * t\_param$
$P_{(k)} = Q_{1(k)} + D_{(k)} * t\_param$
$u_1 = P_{(j)} - V_{1(j)}$
$v_1 = P_{(k)} - V_{1(k)}$
$u_2 = V_{2(j)} - V_{1(j)}$
$u_3 = V_{3(j)} - V_{1(j)}$
$v_2 = V_{2(k)} - V_{1(k)}$
$v_3 = V_{3(k)} - V_{1(k)}$
if ($u_2 > -\varepsilon$ && $u_2 < \varepsilon$) {
    $\beta = u_1 / u_3$
    if ($\beta < 0$ || $\beta > 1$) return 0 rejection 3
    $\alpha = (v_1 - \beta * v_3) / v_2$
} else {
    $\beta = (v_1 * u_2 - u_1 * v_2) / (v_3 * u_2 - u_3 * v_2)$
    if ($\beta < 0$ || $\beta > 1$) return 0 rejection 3
    $\alpha = (u_1 - \beta * u_3) / u_2$
}
if ($\alpha < 0$ || $\alpha+\beta > 1.0$) return 0 rejection 4
return 1

Algorithm 1. Badouel's algorithm.

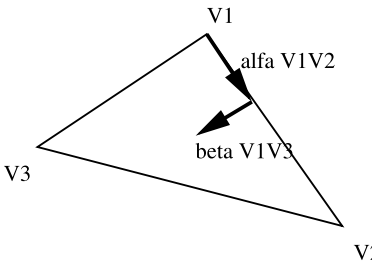

Fig. 2. Barycentric coordinates of the intersection point in Badouel's algorithm.

$$\begin{aligned} x_p - x_1 &= \alpha(x_2 - x_1) + \beta(x_3 - x_1) \\ y_p - y_1 &= \alpha(y_2 - y_1) + \beta(y_3 - y_1) \\ z_p - z_1 &= \alpha(z_2 - z_1) + \beta(z_3 - z_1) \end{aligned} \tag{2}$$

The triangle is projected to one of the xy-, xz- or yz-planes, depending on the greater value of the coefficients of the triangle's plane equation. Let i be the rejected coordinate when the triangle is projected and let j, k be the remaining coordinates. It is defined:

$$u_1 = P_{(j)} - V_{1(j)}, \qquad u_2 = V_{2(j)} - V_{1(j)}, \qquad u_3 = V_{3(j)} - V_{1(j)}$$
$$v_1 = P_{(k)} - V_{1(k)}, \qquad v_2 = V_{2(k)} - V_{1(k)}, \qquad v_3 = V_{3(k)} - V_{1(k)} \tag{3}$$

By substituting we obtain:

$$u_1 = \alpha \cdot u_2 + \beta \cdot u_3, \qquad v_1 = \alpha \cdot v_2 + \beta \cdot v_3 \tag{4}$$

This algorithm has the advantage of being faster if the normal and/or the projection plane have been precalculated. But it is less efficient than Möller's if the normal has not been previously stored, as in this paper. This algorithm has two disadvantages: one consists of the quantity of calculations carried out before rejection tests; the other consists of the greater number of calculations performed compared to Möller's algorithm, including divisions and because of this it is less robust. The algorithm obtains the barycentric coordinates of the intersection point with regard to the triangle. Because of this, the interpolation of vertices properties could be used for other kinds of algorithms, such as Gouraud's shading. This algorithm can be adapted so that the triangles not oriented in the direction of the ray can be rejected.

2.2. Möller's algorithm

Möller's algorithm (Algorithm 2) generalizes and optimizes Badouel's. It is based on the transformation of the triangle and the ray so that the triangle is translated to the origin of the coordinate system and is scaled in order to obtain a unit triangle on yz-plane with the ray aligned with the x-axis (Fig. 3).

Essentially a ray $R(t)$ is defined with an origin Q_1 and a normalized direction D, that is, $R(t) = Q_1 + t \cdot D$. In order to compare this algorithm with the proposed algorithm, a ray representation based on two points, source and destination, so that $R(t) = Q_1 + t(Q_2 - Q_1)$, Q_2 being the second extreme of the segment Q_1Q_2, will be used. A triangle is defined by means of its vertices V_1, V_2, V_3.

By using the barycentric coordinates of a point with regard to a triangle, the point $S(u, v) = (1-u-v)\cdot V_1 + u \cdot V_2 + v \cdot V_3$ is defined as belonging to the triangle if it satisfies $u \geqslant 0$, $v \geqslant 0$ and $u + v \leqslant 1$. If the ray equation is equaled to the point equation, the intersection point is calculated as follows:

$$Q_1 + t \cdot (Q_2 - Q_1) = (1 - u - v) \cdot V_1 + u \cdot V_2 + v \cdot V_3 \tag{5}$$

When this equation is solved, the following result is obtained:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_1} \cdot \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix} \tag{6}$$

being:

$$D = Q_2 - Q_1, \qquad E_1 = V_2 - V_1, \qquad E_2 = V_3 - V_1, \quad \text{and} \quad T = Q_1 - V_1 \tag{7}$$

By using this method, calculations are accelerated because it is only necessary to carry out one division, two vectorial products and four dot products in the worst case. With this method it is not necessary to store certain information such as the triangle normal. The intersection point with the triangle is calculated, and the barycentric coordinates can be used for the interpolation of properties defined at vertices, such as the color in Gouraud shading. Furthermore back-face culling can be used if necessary. Möller's optimization [12] is used for the implementation of the algorithm; this consists of performing the division early in the code plus moving a cross product out from the "if–else if"-case.

2.3. Segura's algorithm

Segura proposes a segment/triangle intersection algorithm (Algorithm 3), based on the sign study of different tetrahedra formed by the vertices of the triangle and the segment.

Let $V_1V_2V_3$ be a triangle in $\Re^3$ and Q_1Q_2 the segment extremes on different sides of the plane of $V_1V_2V_3$. These vertices are ordered so that the tetrahedron $Q_1V_1V_2V_3$ has positive orientation (that is, the signed volume of the tetrahedron is positive). Then the segment Q_1Q_2 intersects with the triangle $V_1V_2V_3$ if and only if (Fig. 4):

$$\operatorname{sign}\big(|Q_2V_1V_2Q_1|\big) \geqslant 0 \quad \text{and} \quad \operatorname{sign}\big(|Q_2V_3V_2Q_1|\big) \geqslant 0 \quad \text{and} \quad \operatorname{sign}\big(|Q_2V_1V_3Q_1|\big) \geqslant 0 \tag{8}$$

being:

$$\operatorname{sign}(x) = \left\{ \begin{array}{ll} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{array} \right\} \tag{9}$$

INPUT: Triangle (V_1, V_2, V_3), Segment(Q_1, Q_2)

Without Back-Face culling

```
D = Q2 – Q1
E1 = V2 – V1
E2 = V3 – V1
A = D x E2
det = A · E1
if (det > ε) {
        T = Q1 – V1
        u = (A · T)
        if (u < 0.0 || u > det) return 0 rejection 2
        B = T x E1
        v = (B · D)
        if (v < 0.0 || u+v > det) return 0 rejection 3
} else if (det < - ε) {
        T = Q1 – V1
        u = (A · T)
        if (u > 0.0 || u < det) return 0 rejection 2
        B = T x E1
        v = (B · D)
        if (v > 0.0 || u+v < det) return 0 rejection 3
} else return 0 rejection 1
inv_det = 1.0 / det
t_param = (B · E2) * inv_det
if (t_param<0.0 || t_param>1.0) return 0 rejection 4
α = u * inv_det
β = v * inv_det
return 1
```

With Back-Face culling

```
D = Q2 – Q1
E1 = V2 – V1
E2 = V3 – V1
A = D x E2
det = A · E1
if (det < ε) return 0 rejection 1
T = Q1 – V1
u = (A · T)
if (u < 0.0 || u > det) return 0 rejection 2
B = T x E1
v = (B · D)
if (v < 0.0 || u+v > det) return 0 rejection 3
inv_det = 1.0 / det
t_param = (B · E2) * inv_det
if (t_param<0.0 || t_param>1.0) return 0 rejection 4
α = u * inv_det
β = v * inv_det
return 1
```

Algorithm 2. Möller's algorithm.

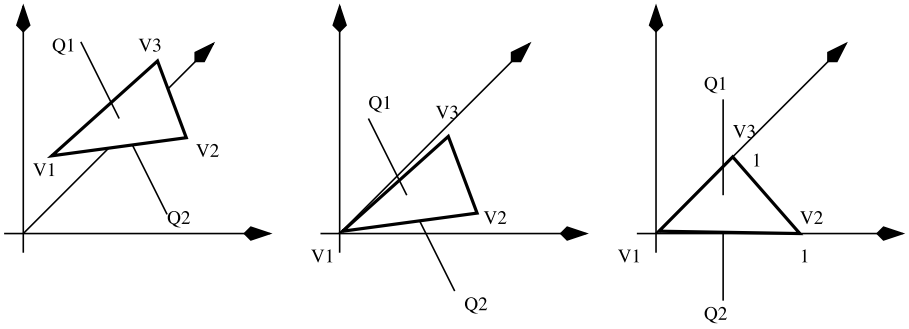

Fig. 3. Transformation and base change of ray in Möller's algorithm.

and $|ABCD|$ the signed volume of the tetrahedron $ABCD$:

$$|ABCD| = \frac{1}{6} \cdot \begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix} = \frac{1}{6} \cdot \begin{vmatrix} x_A - x_D & y_A - y_D & z_A - z_D \\ x_B - x_D & y_B - y_D & z_B - z_D \\ x_C - x_D & y_C - y_D & z_C - z_D \end{vmatrix} \tag{10}$$

This algorithm calculates whether an intersection or a non-intersection between a segment and a triangle takes place, but it does not calculate the intersection point in its original form. In order to calculate this intersection point it is necessary to apply a classical ray/plane intersection algorithm. However, the calculations used for the determination of the boolean intersection between the segment and the plane are also necessary for calculating the sign expression $\text{sign}(|Q_1V_1V_2V_3|) \geqslant 0$ equivalent to determining whether the associated tetrahedron has positive orientation. If the tetrahedron $Q_1V_1V_2V_3$ does not have positive orientation, then the segment Q_1Q_2 intersects with the triangle $V_1V_2V_3$ if and only if:

$$\text{sign}(|Q_2V_1V_2Q_1|) \leqslant 0 \quad \text{and} \quad \text{sign}(|Q_2V_3V_2Q_1|) \leqslant 0 \quad \text{and} \quad \text{sign}(|Q_2V_1V_3Q_1|) \leqslant 0 \tag{11}$$

INPUT: Triangle (V_1, V_2, V_3), Segment(Q_1, Q_2)

Without Back-Face culling

```
E1 = V2 – V1
E2 = V3 – V1
N = E1 x E2
D = Q1 – Q2
det = - N · D
if (det > - ε && det < ε) return 0 rejection 1
T = V1 – Q1
t_param = (N · T) / det
if (t_param < 0.0 || t_param > 1.0) return 0 rejection 2
A = V2 – Q2
B = V1 – Q2
W1 = A x D
s = - B · W1
if (det>0) {
        if (s < 0) return 0 rejection 3
        C = V3 – Q2
        t = C · W1
        if (t < 0) return 0 rejection 4
        W2 = C x D
        u = B · W2
        if (u < 0) return 0 rejection 5
} else {
        if (s > 0) return 0 rejection 3
        C = V3 – Q2
        t = C · W1
        if (t > 0) return 0 rejection 4
        W2 = C x D
        u = B · W2
        if (u > 0) return 0 rejection 5
}
return 1
```

With Back-Face culling

```
E1 = V2 – V1
E2 = V3 – V1
N = E1 x E2
D = Q1 – Q2
det = - N · D
if (det < ε) return 0 rejection 1
T = V1 – Q1
t_param = (N · T) / det
if (t_param < 0.0 || t_param > 1.0) return 0 rejection 2
A = V2 – Q2
B = V1 – Q2
W1 = A x D
s = - B · W1
if (s < 0) return 0 rejection 3
C = V3 – Q2
t = C · W1
if (t < 0) return 0 rejection 4
W2 = C x D
u = B · W2
if (u < 0) return 0 rejection 5
return 1
```

Algorithm 3. Segura's algorithm.

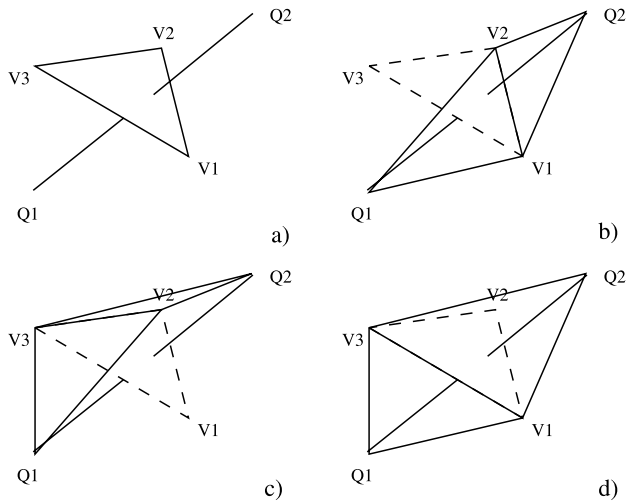

Fig. 4. Segura's algorithm interpretation. (a) Problem to be solved. (b) Tetrahedron $Q_2V_1V_2Q_1$. (c) Tetrahedron $Q_2V_3V_2Q_1$. (d) Tetrahedron $Q_2V_1V_3Q_1$.

3. New segment/triangle algorithm

A new algorithm (Algorithm 4) for the intersection between a segment and a triangle in 3D is presented. Notation presented in Fig. 1 is used.

INPUT: Triangle (V_1, V_2, V_3), Segment(Q_1, Q_2)

Without Back-Face culling

```
A = Q1 – V3
B = V1 – V3
C = V2 – V3
W1 = B x C
w = A · W1
D = Q2 – V3
s = D · W1
if ( w > ε ) {
      if ( s > ε ) return 0 rejection 2
      W2 = A x D
      t = W2 · C
      if ( t < -ε ) return 0 rejection 3
      u = - W2 · B
      if ( u < -ε ) return 0 rejection 4
      if ( w < s + t + u ) return 0 rejection 5
} else if ( w < -ε) {
      if ( s < -ε ) return 0 rejection 2
      W2 = A x D
      t = W2 · C
      if ( t > ε ) return 0 rejection 3
      u = - W2 · B
      if ( u > ε ) return 0 rejection 4
      if ( w > s + t + u ) return 0 rejection 5
} else { // w==0, swap (Q1,Q2)
      if ( s > ε ) {
            W2 = D x A
            t = W2 · C
            if ( t < -ε ) return 0 rejection 3
            u = - W2 · B
            if ( u < -ε ) return 0 rejection 4
            if ( -s < t + u ) return 0 rejection 5
      } else if ( s < -ε) {
            W2 = D x A
            t = W2 · C
            if ( t > ε ) return 0 rejection 3
            u = - W2 · B
            if ( u > ε ) return 0 rejection 4
            if ( -s > t + u ) return 0 rejection 5
      } else return 0 rejection 1
}
tt = 1 / (s – w)  //t_param=w / (s–w)
α = tt * t
β = tt * u
return 1
```

With Back-Face culling

```
A = Q1 – V3
B = V1 – V3
C = V2 – V3
W1 = B x C
w = A · W1
if ( w > ε ) {
      D = Q2 – V3
      s = D · W1
      if ( s > ε ) return 0 rejection 2
      W2 = A x D
      t = W2 · C
      if ( t < -ε ) return 0 rejection 3
      u = - W2 · B
      if ( u < -ε ) return 0 rejection 4
      if ( w < s + t + u ) return 0 rejection 5
} else if ( w < -ε) {
      return 0 rejection 1 (back)
} else { // w==0, swap (Q1,Q2)
      D = Q2 – V3
      s = D · W1
      if ( s > ε ) {
            return 0 rejection 1 (back)
      } else if ( s < -ε ) {
            W2 = D x A
            t = W2 · C
            if ( t > ε ) return 0 rejection 3
            u = - W2 · B
            if ( u > ε ) return 0 rejection 4
            if ( -s >  t + u ) return 0 rejection 5
      } else return 0 rejection 1 (coplanar)
}
tt = 1 / (s – w)      //t_param=w/(s – w)
α = tt * t
β = tt * u
return 1
```

Algorithm 4. New segment/triangle intersection algorithm.

The idea is relatively simple and consists of determining the barycentric coordinates of an extreme of the segment Q_1Q_2, for example Q_2, with regard to the tetrahedron $Q_1V_1V_2V_3$, and checking their sign. Let $Q_2 = \alpha Q_1 + \beta V_1 + \gamma V_2 + \delta V_3$, $\alpha, \beta, \gamma, \delta \in \Re$ be the equation of the position of the point Q_2 with regard to the tetrahedron $T = Q_1V_1V_2V_3$. The set $(\alpha, \beta, \gamma, \delta)$ represents the barycentric coordinates of the point Q_2 with regard to T. Note that these values fulfill the proportion $\alpha + \beta + \gamma + \delta = 1$, so that any one of them can be expressed with regard to the others: $\delta = 1 - \alpha - \beta - \gamma$.

The use of the barycentric coordinates has several advantages, such us the possibility of obtaining the exact position of a point with regard to each one of the faces of a tetrahedron, as well as the global position of this point with regard to the tetrahedron. Another advantage consists of the possibility of performing some simplifications and sharing calculations based

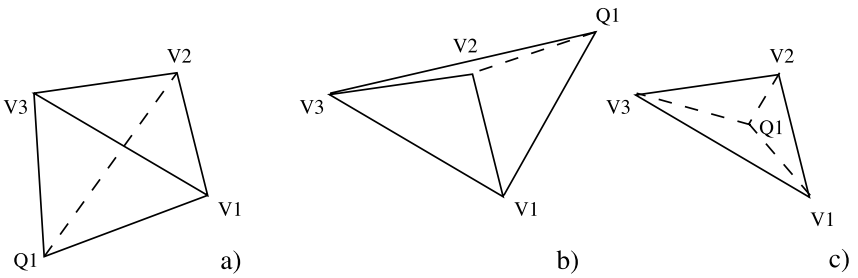

Fig. 5. Signed volume of a tetrahedron, (a) Positive, (b) negative, (c) zero.

on the geometric interpretation of these coordinates in the context of a mesh. They also allow easy detection of special cases like degenerate triangles or coplanar segments. Finally they can be used for volumetric calculations and can be used directly for the interpolation of vertex properties.

The signed volume of a tetrahedron (Eq. (10)) is used for the barycentric calculation. This signed volume can be positive or negative depending on the order of the points (Fig. 5). If the triangles of the tetrahedron have a positive orientation (counterclockwise order), the signed volume is positive, otherwise it is negative or zero if the four points are on the same plane.

Given a tetrahedron $Q_1V_1V_2V_3$, the point Q_1 is defined as the original vertex. A triangle has positive orientation if its vertices are in counterclockwise order. A tetrahedron $Q_1V_1V_2V_3$ has positive orientation if the triangle $V_1V_2V_3$ has positive orientation with regard to the original vertex and the triangles $Q_1V_2V_1$, $Q_1V_3V_2$ and $Q_1V_1V_3$ have positive orientation with regard to the remaining vertices.

By using the signed volume definition the barycentric coordinates of Q_2 with regard to the tetrahedron $Q_1V_1V_2V_3$ can be calculated as:

$$\begin{aligned}
\alpha &= \frac{|Q_2V_1V_2V_3|}{|Q_1V_1V_2V_3|} = \frac{\det(Q_2V_1V_2V_3)}{\det(Q_1V_1V_2V_3)} \\
\beta &= \frac{|Q_2Q_1V_3V_2|}{|Q_1V_1V_2V_3|} = \frac{\det(Q_2Q_1V_3V_2)}{\det(Q_1V_1V_2V_3)} \\
\gamma &= \frac{|Q_2Q_1V_1V_3|}{|Q_1V_1V_2V_3|} = \frac{\det(Q_2Q_1V_1V_3)}{\det(Q_1V_1V_2V_3)} \\
\delta &= \frac{|Q_2Q_1V_2V_1|}{|Q_1V_1V_2V_3|} = \frac{\det(Q_2Q_1V_2V_1)}{\det(Q_1V_1V_2V_3)}
\end{aligned} \tag{12}$$

The position of a point with regard to this tetrahedron can be established using the barycentric coordinates. A point Q_2 is inside the tetrahedron $Q_1V_1V_2V_3$ if $\alpha, \beta, \gamma, \delta \in [0, 1]$.

Moreover, these coordinates can be used for determining the side where the point is located with regard to the supporting planes of the triangles of the tetrahedron. Therefore a geometric interpretation of the value of each barycentric coordinate is proposed. Let $T = ABCD$ be a tetrahedron and P be a point with barycentric coordinates $(\alpha, \beta, \gamma, \delta)$ regarding T:

- A point with $\alpha = 0$ indicates that the point is on the supporting plane of *BCD*.
- A point with $\alpha > 0$ indicates that the point is on the same side as point *A* with regard to the supporting plane of *BCD*.
- A point with $\alpha < 0$ indicates that the point is on the opposite side to point *A* with regard to the supporting plane of *BCD*.

The same interpretation can be applied to β, γ and δ with regard to the supporting planes of *ADC*, *ADB* and *ACB* respectively.

The proof is simple and can be performed by using the geometric interpretation of the barycentric coordinates of a point with regard to a tetrahedron. Let $(\alpha, \beta, \gamma, \delta)$ be the barycentric coordinates of P with regard to *ABCD*. α represents the normalized signed volume of the tetrahedron *PBCD* (Fig. 6). A tetrahedron has signed volume equal to zero only if the four vertices are on the same plane. If the tetrahedron *ABCD* has a positive signed volume, the signed volume of *PBCD* will also be positive if the triangles of *PBCD* have positive orientation. This occurs only if point P is on the same side as A with regard to the supporting plane of *BCD*. If point P is on the opposite side, the triangles that form the tetrahedron *PBCD* will have negative orientation, so that the signed volume of *PBCD* will be negative. A similar analysis can be realized for a tetrahedron with negative signed volume.

By means of analogous reasoning it can be proved that β, γ, δ can be used for the determination of the position of a point with regard to the supporting planes of *ADC*, *ADB* and *ACB* respectively.

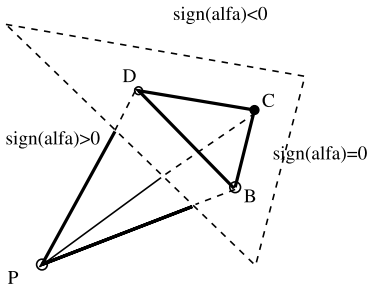

Fig. 6. Geometric interpretation of α with regard to a tetrahedron.

Returning to the problem of segment/triangle intersection, the intersection between the segment Q_1Q_2 and the triangle $V_1V_2V_3$ can be determined by means of the following theorem:

Theorem 1. *Let $V_1V_2V_3$ be a triangle and Q_1Q_2 a segment so that Q_1 is not coplanar with the plane defined by $V_1V_2V_3$. Q_1Q_2 intersects with $V_1V_2V_3$ if and only if*:

$$\operatorname{sign}(\alpha) \leqslant 0 \quad \textit{and} \quad \operatorname{sign}(\beta) \geqslant 0 \quad \textit{and} \quad \operatorname{sign}(\gamma) \geqslant 0 \quad \textit{and} \quad \operatorname{sign}(\delta) \geqslant 0 \tag{13}$$

α, β, γ and δ being the barycentric coordinates of Q_2 with regard to the tetrahedron $Q_1V_1V_2V_3$, and the sign *function as is defined in Eq.* (9)*.*

Furthermore Q_1Q_2 does not intersect with the triangle $V_1V_2V_3$ if and only if:

$$\operatorname{sign}(\alpha) > 0 \quad \text{or} \quad \operatorname{sign}(\beta) < 0 \quad \text{or} \quad \operatorname{sign}(\gamma) < 0 \quad \text{or} \quad \operatorname{sign}(\delta) < 0 \tag{14}$$

Now let us observe the efficient calculation of the sign of α, β, γ and δ (Eq. (12)). For this we must obtain:

$$a_0 = |Q_1V_1V_2V_3| = -|V_3Q_1V_1V_2| = -\big|(Q_1 - V_3)(V_1 - V_3)(V_2 - V_3)\big| \tag{15}$$

$$a_1 = |Q_2V_1V_2V_3| = -|V_3Q_2V_1V_2| = -\big|(Q_2 - V_3)(V_1 - V_3)(V_2 - V_3)\big| \tag{16}$$

$$a_2 = |Q_2Q_1V_3V_2| = -|V_3Q_2V_2Q_1| = -\big|(Q_2 - V_3)(V_2 - V_3)(Q_1 - V_3)\big| \tag{17}$$

$$a_3 = |Q_2Q_1V_1V_3| = -|V_3Q_2Q_1V_1| = -\big|(Q_2 - V_3)(Q_1 - V_3)(V_1 - V_3)\big| \tag{18}$$

$$a_4 = |Q_2Q_1V_2V_1| = -|V_1Q_2Q_1V_2| = -\big|(Q_2 - V_1)(Q_1 - V_1)(V_2 - V_1)\big| \tag{19}$$

if we define:

$$\begin{aligned} &A = Q_1 - V_3, \qquad B = V_1 - V_3, \qquad C = V_2 - V_3, \qquad D = Q_2 - V_3 \\ &E = Q_2 - V_1, \qquad F = Q_1 - V_1, \qquad G = V_2 - V_1 \end{aligned} \tag{20}$$

then we obtain:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} = \frac{1}{-|\,A \quad B \quad C\,|} \cdot \begin{bmatrix} -|\,D \quad B \quad C\,| \\ -|\,D \quad C \quad A\,| \\ -|\,D \quad A \quad B\,| \\ -|\,E \quad F \quad G\,| \end{bmatrix} = \frac{1}{A \cdot (B \times C)} \cdot \begin{bmatrix} D \cdot (B \times C) \\ -(D \times A) \cdot C \\ (D \times A) \cdot B \\ (E \times F) \cdot G \end{bmatrix} \tag{21}$$

Moreover, as Q_1 is not coplanar with $V_1V_2V_3$ it is stated that $a_0 \neq 0$ and:

$$\operatorname{sign}\left(\frac{a_i}{a_0}\right) = \frac{\operatorname{sign}(a_i)}{\operatorname{sign}(a_0)} = \operatorname{sign}(a_i) \cdot \operatorname{sign}(a_0), \quad i = 1, \ldots, 4 \tag{22}$$

we define:

$$\operatorname{sign}(i) \cdot \operatorname{sign}(j) = \left\{ \begin{array}{ll} \operatorname{sign}(j), & \text{if } \operatorname{sign}(i) > 0 \\ 0, & \text{if } \operatorname{sign}(i) = 0 \\ -\operatorname{sign}(j), & \text{if } \operatorname{sign}(i) < 0 \end{array} \right\} \tag{23}$$

and we obtain:

$$\begin{bmatrix} \operatorname{sign}(\alpha) \\ \operatorname{sign}(\beta) \\ \operatorname{sign}(\gamma) \\ \operatorname{sign}(\delta) \end{bmatrix} = \operatorname{sign}\big(A \cdot (B \times C)\big) \cdot \begin{bmatrix} \operatorname{sign}(D \cdot (B \times C)) \\ \operatorname{sign}(-(D \times A) \cdot C) \\ \operatorname{sign}((D \times A) \cdot B) \\ \operatorname{sign}((E \times F) \cdot G) \end{bmatrix} = \operatorname{sign}(w) \cdot \begin{bmatrix} \operatorname{sign}(s) \\ \operatorname{sign}(t) \\ \operatorname{sign}(u) \\ \operatorname{sign}(v) \end{bmatrix} \tag{24}$$

being:

$$s = D \cdot (B \times C), \qquad t = -(D \times A) \cdot C, \qquad u = (D \times A) \cdot B$$
$$v = (E \times F) \cdot G \quad \text{and} \quad w = A \cdot (B \times C) \tag{25}$$

Furthermore, as:

$$\alpha + \beta + \gamma + \delta = 1 \tag{26}$$

therefore:

$$\frac{s}{w} + \frac{t}{w} + \frac{u}{w} + \frac{v}{w} = 1 \tag{27}$$
$$v = w - s - t - u \tag{28}$$

for this reason it is not necessary to calculate v using the expression: $(E \times F) \cdot G$.

Based on these calculations, a new algorithm for the segment/triangle intersection (Algorithm 4) has been developed. The main advantages of this algorithm are examined below.

4. Features of the new algorithm

The proposed algorithm has some properties which make it robust and efficient for certain situations. Among these features the following could be emphasized:

4.1. Robustness

The robustness [13] of the algorithm can be considered from a numerical point of view. The numeric robustness of geometric algorithms can be seen from the precision of the algorithms (derived from the use of finite precision arithmetic) and from the treatment of degenerate cases.

Precision errors may occur in the operations carried out (additions, multiplications and divisions) and in the comparisons:

- The number of operations carried out for obtaining whether an intersection occurs or not has been counted theoretically (Table 1) and practically (Tables 2 and 3). In fact, many of the operations carried out by most of the algorithms considered are equivalent but performed in different order. In general, the new algorithm performs a minor number of operations and a greater number of rejection tests with a minor number of comparisons, avoiding the division operation which is normally less precise. It is only necessary to perform this division when the intersection point or the barycentric coordinates of the point are needed. The errors committed are minimized with regard to other segment–triangle algorithms due to the minor number of operations performed with a greater number of rejection tests.
- With regard to the error committed in the comparisons, a predetermined ϵ error [14] is used. This error could be dependent on the volume of the tetrahedron used for the calculation of barycentric coordinates, so the comparison would be less susceptible to errors. In the classical algorithms analyzed (in their original form), there are comparisons with 0.0 and with 1.0 values. If these comparisons are substituted with $\pm\epsilon$ and with $1.0 \pm \epsilon$ values, an additional summation must be added to the algorithms for each comparison, obtaining more operations than those shown in this study. In the new algorithm these comparisons using ϵ values have been performed.

In the absence of degenerate triangles, the special or degenerate cases are the ones obtained when the segment is aligned with the plane in which the triangle is located (coplanar segments). The algorithm examines the case in which Q_1 is on the plane of the triangle. In this case, it swaps the roles of Q_1 and Q_2. If Q_2 is now on the plane of the triangle, the algorithm discards the segment (Fig. 7). This is a coplanar segment and in this case the signed volume of tetrahedra $Q_1V_1V_2V_3$ and $Q_2V_1V_2V_3$ (represented by values w and s) is zero (with a predefined error [15]).

So the new algorithm discards segments which are on the $V_1V_2V_3$ plane, as do the other algorithms analyzed. For triangle meshes coplanar segments can be discarded due to the fact that there are some triangles which share an edge with the triangle in question, and the intersection with these shared triangles can be detected.

4.2. Back-face culling

It is also possible to discard some triangles which are not oriented in the direction of the ray. This verification can be performed when the volume of the tetrahedron $Q_1V_1V_2V_3$ is calculated. If this volume is positive, the triangle $V_1V_2V_3$ is not oriented in the ray's direction, and the ray is discarded. If this volume is zero Q_1 and Q_2 must be swapped, because it is possible to have a segment with Q_1 on the triangle and Q_2 on the other side of the plane. When this occurs the segment does not need to be discarded.

Table 1
Number of operations carried out for each algorithm analyzed.

Möller	No back-face					Back-face				
Section	ADD	MUL	DIV	COM	Result	ADD	MUL	DIV	COM	Result
Rejection 1	14	9	0	2	Coplanary segment	14	9	0	1	Back segment
Rejection 2	5	3	0	3.5	Segment doesn't intersect triangle	5	3	0	3	Segment doesn't intersect triangle
Rejection 3	6	9	0	2	Segment doesn't intersect triangle	6	9	0	2	Segment doesn't intersect triangle
Calculation	2	4	1	0	*t_param* calculation	2	4	1	0	*t_param* calculation
Rejection 4	0	0	0	2	Segment doesn't intersect plane	0	0	0	2	Segment doesn't intersect plane
Calculation	0	2	0	0	alfa, beta calculation	0	2	0	0	alfa, beta calculation
	27	**27**	**1**	**7.5**		**27**	**27**	**1**	**7**	
Badouel	No back-face					Back-face				
Section	ADD	MUL	DIV	COM	Result	ADD	MUL	DIV	COM	Result
Calculation	9	6	0	0	Normal calculation	9	6	0	0	Normal calculation
Rejection 1	5	3	0	2	Coplanary segment	5	3	0	1	Back segment
Calculation	5	3	1	0	*t_param* calculation	5	3	1	0	*t_param* calculation
Rejection 2	0	0	0	2	Segment doesn't intersect plane	0	0	0	2	Segment doesn't intersect plane
Rejection 3	10	6	1	6	$\beta < 0 \parallel \beta > 1$ Segment doesn't intersect triangle	10	6	1	6	$\beta < 0 \parallel \beta > 1$ Segment doesn't intersect triangle
Calculation	1	1	1	0	alfa, beta calculation	1	1	1	0	alfa, beta calculation
Rejection 4	1	0	0	2	$\alpha < 0 \parallel \alpha + \beta > 1$ Segment doesn't intersect triangle	1	0	0	2	$\alpha < 0 \parallel \alpha + \beta > 1$ Segment doesn't intersect triangle
	31	**19**	**3**	**12**		**31**	**19**	**3**	**11**	
Segura	No back-face					Back-face				
Section	ADD	MUL	DIV	COM	Result	ADD	MUL	DIV	COM	Result
Calculation	9	6	0	0	Normal calculation	9	6	0	0	Normal calculation
Rejection 1	5	3	0	2	Coplanary segment	5	3	0	1	Back segment
Calculation	5	3	1	0	*t_param* calculation	5	3	1	0	*t_param* calculation
Rejection 2	0	0	0	2	Segment doesn't intersect plane	0	0	0	2	Segment doesn't intersect plane
Rejection 3	11	9	0	2	$\text{sign}(\lvert Q_2V_1V_2Q_1\rvert) < 0$ Segment doesn't intersect triangle	11	9	0	1	$\text{sign}(\lvert Q_2V_1V_2Q_1\rvert) < 0$ Segment doesn't intersect triangle
Rejection 4	5	3	0	1	$\text{sign}(\lvert Q_2V_3V_2Q_1\rvert) \geqslant 0$ Segment doesn't intersect triangle	5	3	0	1	$\text{sign}(\lvert Q_2V_3V_2Q_1\rvert) \geqslant 0$ Segment doesn't intersect triangle
Rejection 5	5	9	0	1	$\text{sign}(\lvert Q_2V_1V_3Q_1\rvert) \geqslant 0$ Segment doesn't intersect triangle	5	9	0	1	$\text{sign}(\lvert Q_2V_1V_3Q_1\rvert) \geqslant 0$ Segment doesn't intersect triangle
	40	**33**	**1**	**8**		**40**	**33**	**1**	**6**	
New	No back-face					Back-face				
Section	ADD	MUL	DIV	COM	Result	ADD	MUL	DIV	COM	Result
Calculation	9	6	0	0	Normal calculation	9	6	0	0	Normal calculation
Rejection 1	5	3	0	4	Coplanary segment	5	3	0	2.5	Back or coplanary segment
Rejection 2	5	3	0	2.5	$\alpha > 0$ Segment doesn't intersect plane	5	3	0	2	$\alpha > 0$ Segment doesn't intersect plane
Rejection 3	5	9	0	1	$\beta < 0$ Segment doesn't intersect triangle	5	9	0	1	$\beta < 0$ Segment doesn't intersect triangle
Rejection 4	2	3	0	1	$\gamma < 0$ Segment doesn't intersect triangle	2	3	0	1	$\gamma < 0$ Segment doesn't intersect triangle
Rejection 5	2	0	0	1	$\delta < 0$ Segment doesn't intersect triangle	2	0	0	1	$\delta < 0$ Segment doesn't intersect triangle
Calculation	1	0	1	0	*t_param* calculation	1	0	1	0	*t_param* calculation
Calculation	0	2	0	0	alfa, beta calculation	0	2	0	0	alfa, beta calculation
	29	**26**	**1**	**5.5**		**29**	**26**	**1**	**5**	

Rows show the number of operations for each rejection test or for each important calculation, for example for the normal calculation, the t parameter or the barycentric coordinates.
It has been counted as half an operation when this operation is performed in 50% of the cases. The number of comparisons needed for a coplanar segment in Möller's and the new algorithm has not been accumulated because this operation has been included in the next rejection test.
It can be seen that the new algorithm uses in the worst case around the same number of operations as Möller's algorithm, carrying out one multiplication and two comparisons less than Möller's, and two more additions. The number of rejection tests is greater than with Möller's and Badouel's algorithms.

4.3. Calculation of the intersection point

If the intersection point is needed, it can be calculated with a minimal computational cost (one difference and one division). These calculations are carried out at the end of the algorithm and are based on previous calculations. This represents an advantage if the calculations are not necessary. This is the disadvantage of Badouel's algorithm, since it calculates the t parameter (*t_param*) at the beginning. The range of *t_param* is $[0, 1]$ being this parameter 0 at Q_1 and 1 at Q_2.

Table 2
Average percentages of the number of operations carried out for each type of test (1 to 3) without back-face culling.

									Random segment						Ray casting segment						Random segment with limited length					
	Möller	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Coplanar segment	14	9	0	2	14	9	0	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rej. 2	No triangle intersection	5	3	0	3.5	19	12	0	3.5	97.99	18.62	11.76	0.00	3.43	33.81	98.17	18.65	11.78	0.00	3.44	33.87	98.27	18.67	11.79	0.00	3.44	33.90
Rej. 3	No triangle intersection	6	9	0	2	25	21	0	5.5	1.95	0.49	0.41	0.00	0.11	1.01	1.79	0.45	0.38	0.00	0.10	0.92	1.69	0.42	0.35	0.00	0.09	0.87
Rej. 4	No plane intersection	2	4	1	2	27	25	1	7.5	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.01	0.01	0.00	0.00	0.02
No Rej.	Intersection	0	2	0	0	27	27	1	7.5	0.04	0.01	0.01	0.00	0.00	0.03	0.04	0.01	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
	Total									100.00	19.12	12.18	0.00	3.54	34.85	100.00	19.11	12.17	0.00	3.54	34.82	100.00	19.10	12.16	0.00	3.54	34.80
									Random segment						Ray casting segment						Random segment with limited length					
	Badouel	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Coplanar segment	14	9	0	2	14	9	0	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rej. 2	No plane intersection	5	3	1	2	19	12	1	4	67.20	12.77	8.06	0.67	2.69	24.19	26.11	4.96	3.13	0.26	1.04	9.40	97.83	18.59	11.74	0.98	3.91	35.22
Rej. 3	No triangle intersection	10	6	1	6	29	18	2	10	31.55	9.15	5.68	0.63	3.15	18.61	72.05	20.90	12.97	1.44	7.21	42.51	2.10	0.61	0.38	0.04	0.21	1.24
Rej. 4	No triangle intersection	2	1	1	2	31	19	3	12	1.21	0.38	0.23	0.04	0.15	0.79	1.80	0.56	0.34	0.05	0.22	1.17	0.07	0.02	0.01	0.00	0.01	0.04
No Rej.	Intersection	0	0	0	0	31	19	3	12	0.04	0.01	0.01	0.00	0.01	0.03	0.04	0.01	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
	Total									100.00	22.31	13.98	1.34	5.99	43.62	100.00	26.43	16.45	1.76	8.47	53.11	100.00	19.22	12.13	1.02	4.13	36.50
									Random segment						Ray casting segment						Random segment with limited length					
	Segura	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Coplanar segment	14	9	0	2	14	9	0	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rej. 2	No plane intersection	5	3	1	2	19	12	1	4	67.20	12.77	8.06	0.67	2.69	24.19	26.07	4.95	3.13	0.26	1.04	9.39	97.83	18.59	11.74	0.98	3.91	35.22
Rej. 3	No triangle intersection	11	9	0	2	30	21	1	6	16.15	4.84	3.39	0.16	0.97	9.36	37.22	11.17	7.82	0.37	2.23	21.59	1.07	0.32	0.22	0.01	0.06	0.62
Rej. 4	No triangle intersection	5	3	0	1	35	24	1	7	10.55	3.69	2.53	0.11	0.74	7.07	23.99	8.40	5.76	0.24	1.68	16.07	0.69	0.24	0.17	0.01	0.05	0.47
Rej. 5	No triangle intersection	5	9	0	1	40	33	1	8	6.07	2.43	2.00	0.06	0.49	4.98	12.68	5.07	4.18	0.13	1.01	10.39	0.40	0.16	0.13	0.00	0.03	0.33
No Rej.	Intersection	0	0	0	0	40	33	1	8	0.04	0.02	0.01	0.00	0.00	0.04	0.04	0.02	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
	Total									100.00	23.75	16.00	1.00	4.88	45.63	100.00	29.60	20.90	1.00	5.97	57.48	100.00	19.31	12.26	1.00	4.06	36.64
									Random segment						Ray casting segment						Random segment with limited length					
	New	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Coplanar segment	14	9	0	4	14	9	0	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rej. 2	No triangle intersection	5	3	0	2.5	19	12	0	2.5	67.20	12.77	8.06	0.00	1.68	22.51	15.40	2.93	1.85	0.00	0.38	5.16	97.83	18.59	11.74	0.00	2.45	32.77
Rej. 3	No triangle intersection	5	9	0	1	24	21	0	3.5	15.99	3.84	3.36	0.00	0.56	7.75	36.40	8.74	7.64	0.00	1.27	17.65	1.06	0.25	0.22	0.00	0.04	0.51
Rej. 4	No triangle intersection	2	3	0	1	26	24	0	4.5	10.60	2.76	2.54	0.00	0.48	5.78	27.03	7.03	6.49	0.00	1.22	14.73	0.72	0.19	0.17	0.00	0.03	0.39
Rej. 5	No triangle intersection	2	0	0	1	28	24	0	5.5	6.17	1.73	1.48	0.00	0.34	3.55	16.97	4.75	4.07	0.00	0.93	9.76	0.40	0.11	0.09	0.00	0.02	0.23
No Rej.	Intersection	1	2	1	0	29	26	1	5.5	0.04	0.01	0.01	0.00	0.00	0.03	0.04	0.01	0.01	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
	Total									100.00	21.10	15.46	0.00	3.06	39.62	95.83	23.45	20.06	0.00	3.81	47.32	100.00	19.14	12.23	0.00	2.54	33.90

Each row shows the number of operations for each rejection test, the number of accumulated operations until reaching this point, the percentage of cases in which the algorithm returns for each rejection test, and the weighted number of operations for each scenario. Operations have been broken down into additions, multiplications, divisions and comparisons.
There is a lower average of the number of operations for interference applications between moving objects (test 3) than with Möller's algorithm. It must be considered that the hit-rate in this case is very low, and that the time associated with all the operations is equal. In a real application, the effective cost for each type of operation must be weighted. A programmer will use this data for a specific implementation and to obtain the average of the number of operations. A possible reduction of the number of operations must be considered when the intersection point is not calculated and there is a higher number of hit rates.

Table 3
Average percentages of the number of operations carried out for each type of test (1 to 3) with back-face culling.

										Random segment						Ray casting segment						Random segment with limited length					
	Möller	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Back segment	14	9	0	1	14	9	0	1	49.98	7.00	4.50	0.00	0.50	11.99	56.81	7.95	5.11	0.00	0.57	13.63	49.93	6.99	4.49	0.00	0.50	11.98
Rej. 2	No triangle intersection	5	3	0	3	19	12	0	3	49.02	9.31	5.88	0.00	1.47	16.67	42.30	8.04	5.08	0.00	1.27	14.38	49.19	9.35	5.90	0.00	1.48	16.73
Rej. 3	No triangle intersection	6	9	0	2	25	21	0	5	0.98	0.24	0.21	0.00	0.05	0.50	0.87	0.22	0.18	0.00	0.04	0.44	0.84	0.21	0.18	0.00	0.04	0.43
Rej. 4	No plane intersection	2	4	1	2	27	25	1	7	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.01
No Rej.	Intersection	0	2	0	0	27	27	1	7	0.02	0.01	0.01	0.00	0.00	0.01	0.02	0.01	0.01	0.00	0.00	0.01	0.02	0.00	0.00	0.00	0.00	0.01
	Total									100.00	16.56	10.59	0.00	2.02	29.18	100.00	16.21	10.38	0.00	1.88	28.47	100.00	16.56	10.58	0.00	2.02	29.16
										Random segment						Ray casting segment						Random segment with limited length					
	Badouel	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Back segment	14	9	0	1	14	9	0	1	49.98	7.00	4.50	0.00	0.50	11.99	56.81	7.95	5.11	0.00	0.57	13.63	49.95	6.99	4.50	0.00	0.50	11.99
Rej. 2	No plane intersection	5	3	1	2	19	12	1	3	33.65	6.39	4.04	0.34	1.01	11.78	23.60	4.48	2.83	0.24	0.71	8.26	43.18	8.21	5.18	0.43	1.30	15.11
Rej. 3	No triangle intersection	10	6	1	6	29	18	2	9	15.77	4.57	2.84	0.32	1.42	9.14	19.24	5.58	3.46	0.38	1.73	11.16	6.46	1.87	1.16	0.13	0.58	3.75
Rej. 4	No triangle intersection	2	1	1	2	31	19	3	11	0.58	0.18	0.11	0.02	0.06	0.37	0.33	0.10	0.06	0.01	0.04	0.21	0.38	0.12	0.07	0.01	0.04	0.24
No Rej.	Intersection	0	0	0	0	31	19	3	11	0.02	0.01	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.01
	Total									100.00	18.15	11.49	0.67	2.99	33.30	100.00	18.13	11.48	0.63	3.05	33.28	100.00	17.20	10.92	0.57	2.42	31.11
										Random segment						Ray casting segment						Random segment with limited length					
	Segura	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Back segment	14	9	0	1	14	9	0	1	49.98	7.00	4.50	0.00	0.50	11.99	56.81	7.95	5.11	0.00	0.57	13.63	49.95	6.99	4.50	0.00	0.50	11.99
Rej. 2	No plane intersection	5	3	1	2	19	12	1	3	33.62	6.39	4.03	0.34	1.01	11.77	12.74	2.42	1.53	0.13	0.38	4.46	43.18	8.21	5.18	0.43	1.30	15.11
Rej. 3	No triangle intersection	11	9	0	1	30	21	1	4	8.07	2.42	1.70	0.08	0.32	4.52	15.41	4.62	3.24	0.15	0.62	8.63	3.33	1.00	0.70	0.03	0.13	1.87
Rej. 4	No triangle intersection	5	3	0	1	35	24	1	5	5.27	1.85	1.27	0.05	0.26	3.43	10.08	3.53	2.42	0.10	0.50	6.55	2.22	0.78	0.53	0.02	0.11	1.44
Rej. 5	No triangle intersection	5	9	0	1	40	33	1	6	3.03	1.21	1.00	0.03	0.18	2.43	4.94	1.98	1.63	0.05	0.30	3.95	1.29	0.52	0.43	0.01	0.08	1.03
No Rej.	Intersection	0	0	0	0	40	33	1	6	0.02	0.01	0.01	0.00	0.00	0.02	0.02	0.01	0.01	0.00	0.00	0.02	0.02	0.01	0.01	0.00	0.00	0.01
	Total									100.00	18.87	12.50	0.50	2.28	34.15	100.00	20.51	13.93	0.43	2.37	37.24	100.00	17.50	11.34	0.50	2.12	31.46
										Random segment						Ray casting segment						Random segment with limited length					
	New	Section operation				Acum. operation				Average						Average						Average					
Section	Result	+	×	/	>	+	×	/	>	%	+	×	/	>	Total	%	+	×	/	>	Total	%	+	×	/	>	Total
Rej. 1	Back segment	14	9	0	2.5	14	9	0	2.5	64.75	9.06	5.83	0.00	1.62	16.51	61.75	8.65	5.56	0.00	1.54	15.75	64.75	9.06	5.83	0.00	1.62	16.51
Rej. 2	No triangle intersection	5	3	0	2	19	12	0	2	18.85	3.58	2.26	0.00	0.38	6.22	7.80	1.48	0.94	0.00	0.16	2.57	28.39	5.39	3.41	0.00	0.57	9.37
Rej. 3	No triangle intersection	5	9	0	1	24	21	0	3	7.99	1.92	1.68	0.00	0.24	3.84	15.60	3.75	3.28	0.00	0.47	7.49	3.36	0.81	0.70	0.00	0.10	1.61
Rej. 4	No triangle intersection	2	3	0	1	26	24	0	4	5.30	1.38	1.27	0.00	0.21	2.86	9.17	2.38	2.20	0.00	0.37	4.95	2.12	0.55	0.51	0.00	0.08	1.14
Rej. 5	No triangle intersection	2	0	0	1	28	24	0	5	3.09	0.86	0.74	0.00	0.15	1.76	5.66	1.58	1.36	0.00	0.28	3.23	1.37	0.38	0.33	0.00	0.07	0.78
No Rej.	Intersection	1	2	1	0	29	26	1	5	0.02	0.01	0.01	0.00	0.00	0.01	0.02	0.01	0.01	0.00	0.00	0.01	0.02	0.01	0.00	0.00	0.00	0.01
	Total									100.00	16.81	11.79	0.00	2.60	31.20	100.00	17.85	13.33	0.00	2.82	34.00	100.00	16.20	10.78	0.00	2.44	29.43

Note: The same text as Table 2, except with back-face culling.

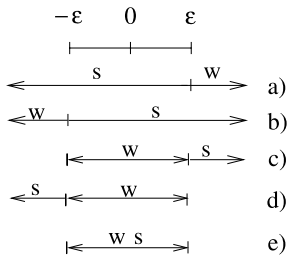

Fig. 7. Cases contemplated in Algorithm 4 for w and s (Eq. (25)). In situations (a), (b), (c) and (d) the algorithm continues. In case (e) the algorithm rejects. In situations (c) and (d) the algorithm swaps the extremes of the segment.

In the proposed algorithm the t parameter is calculated as follows, being $\vec{n}$ the normal of $V_1V_2V_3$:

$$t\_param = \frac{\vec{n}\cdot(Q_1 - V_3)}{\vec{n}\cdot(Q_2 - Q_1)} = \frac{-(B\times C)\cdot A}{-(B\times C)\cdot(D-A)} = \frac{(B\times C)\cdot A}{(B\times C)\cdot D - (B\times C)\cdot A} = \frac{w}{s-w} \tag{29}$$

4.4. Pre-processing

The triangle normal is calculated at the beginning of the algorithm, as with Badouel and Segura. If this value is calculated and stored in the triangle structure, and updated when necessary, a number of calculations can be saved. This improvement has not been used in the algorithms shown.

For ray casting or ray tracing applications, the scene does not change and rays have a common origin. In this case it is possible for Segura's algorithm to store the signed volume of the tetrahedra formed by the origin of the ray and the triangles, accelerating the calculations as in the study carried out in [8], obtaining faster times than Möller's algorithm in this scenario.

In the previous situation, the proposed algorithm can store the following information for each triangle:

$$A = Q_1 - V_3, \qquad B = V_1 - V_3, \qquad C = V_2 - V_3, \qquad W_1 = B\times C \quad \text{and} \quad w = A\cdot W_1 \tag{30}$$

Thus it is only necessary to calculate these values once for each triangle, to store and to reuse them when needed. These optimizations, not implemented in the algorithm, will be studied in the future.

4.5. Interpolation calculations

One application of the barycentric coordinates of a point with regard to a triangle is their use for the interpolation of vertex properties. The new algorithm is capable of calculating these coordinates with the calculations performed for the intersection test, as with Möller and Badouel, and these coordinates can be used for these types of rendering calculations (for example for Gouraud algorithm). The new algorithm can calculate the barycentric coordinates of the intersection point after the t parameter is calculated:

Let P be the intersection between the segment Q_1Q_2 and the triangle $V_1V_2V_3$. Its barycentric coordinates with regard to the tetrahedron $Q_1V_1V_2V_3$ are $(\alpha_P, \beta_P, \gamma_P, \delta_P)$. The barycentric coordinates of P with regard to the triangle $V_1V_2V_3$ are required. These coordinates are equal to the barycentric coordinates of P with regard to the tetrahedron $Q_1V_1V_2V_3$ when $\alpha_P = 0$.

Given that Q_1 has barycentric coordinates $(1,0,0,0)$, and Q_2 has coordinates $(\alpha,\beta,\gamma,\delta)$:

$$\begin{aligned}
P &= Q_1 + t_param\cdot(Q_2 - Q_1)\\
\alpha_P &= 1 + t_param\cdot(\alpha - 1) = 0\\
\beta_P &= 0 + t_param\cdot(\beta - 0) = t_param\cdot\beta\\
\gamma_P &= 0 + t_param\cdot(\gamma - 0) = t_param\cdot\gamma\\
\delta_P &= 0 + t_param\cdot(\delta - 0) = 1 - \beta - \gamma
\end{aligned} \tag{31}$$

These three values $(\beta_P, \gamma_P, \delta_P)$ are the barycentric coordinates of the point P with regard to the triangle $V_1V_2V_3$. In order to obtain these values it is necessary to calculate the division of:

$$\begin{aligned}
&\frac{s}{w}, \frac{t}{w}, \frac{u}{w}, \frac{v}{w} \quad \text{(Eq. (27))}, \quad \text{moreover}\\
&t_param = \frac{w}{s-w} \quad \text{(Eq. (29))}
\end{aligned} \tag{32}$$

so:

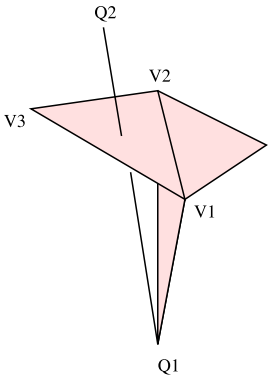

Fig. 8. Calculations for the triangle $Q_1V_1V_2$ can be shared between two triangles in a mesh.

$$\beta_P = t\_param \cdot \beta = \frac{w}{s-w} \cdot \frac{t}{w} = \frac{t}{s-w} \tag{33}$$

$$\gamma_P = t\_param \cdot \gamma = \frac{w}{s-w} \cdot \frac{u}{w} = \frac{u}{s-w} \tag{34}$$

it only being necessary to carry out one division $\text{div} = \frac{1}{s-w}$ and two multiplications.

4.6. Use of the new algorithm for triangle meshes

Many calculations can be reused for triangle meshes. For this it is necessary to know the neighbors among triangles (as in triangle strips) or to store certain information in the shared edge structures. For the shared edges the value of one barycentric coordinate of Q_2 is shared for two tetrahedra, those formed by the triangles which share this edge and Q_1 (Fig. 8).

4.7. Location of the intersection point

It is possible to determine if an intersection occurs on a vertex, on an edge or inside the triangle, without additional computational cost.

4.8. Early rejection tests

For non-intersection, the new algorithm performs a large number of tests, in order to exit early and not perform too many calculations. The calculations accumulated for every rejection test are fewer than the calculations performed by the algorithms evaluated in this study.

5. Performance

This section shows the performance of the new algorithm with regard to classical algorithms. First, the set of tests used are shown, and finally the results obtained.

5.1. Set of tests

It is difficult to develop a set of tests that would assert that a given method is faster than others, because performance depends on many different factors such as compiler, computer, application, hit rate, and type of triangle data. Löfstedt and Akenine-Möller [16] studied this situation and have developed a set of rules so that a fair comparison may be produced. A set of tests which permits developers to check whether the new algorithm suits their applications according to these rules has been established. Now the characteristics of these tests are shown:

- The number of operations carried out for each rejection test has been broken down, as well as the number of operations needed for certain calculations, such as the normal of the triangle, the t parameter and the barycentric coordinates of the intersection point. The number of operations is shown for each algorithm with and without back-face culling. Table 1 presents a comparative study of the number of operations carried out for the segment/triangle intersection. The user could check the performance of these algorithms in a real machine with a recent compiler and weight each type of operation with its calculation time in order to obtain an average time.

Fig. 9. Objects examined in the tests. Cat: 382 vertices, 760 faces. Horse: 3582 vertices, 7172 faces. Sculpture: 5689 vertices, 11 241 faces.

Performance depends on caching, branch prediction and many other effects which are extremely difficult to predict.

- Only triangle vertices are used as input data to the algorithm, since these will meet minimum storage requirements and oblige every algorithm tested to calculate everything "on-the-fly".
- The above algorithms calculate the same type of output; all calculate the t-value and the barycentric coordinates of the intersection point. The number of operations needed to perform a simple boolean intersection test, to calculate the t-value and the barycentric coordinates has been measured as well.
- The definition of "hit" in all algorithms is the same. All rays are treated as segments. Several types of tests have been developed, for ray tracing, ray casting and interference applications.
- A test with different hit rates has been provided, because some algorithms perform better for tests where the hit rate is very low and others for higher hit rates.

A set of four tests has been developed in order to study the performance of these algorithms:

- **Test 0** (hit-ratio) is based on the hit rates. Pairs of random segments with limited length and triangles have been generated and stored to obtain different hit-ratio tests. These segments have been generated (with similar dimensions to the edges of the triangle) all around the triangle in order to generate the different hit rates.
- **Test 1** (random segment) examines the behavior of the algorithms with regard to large segments, as could occur in recursive ray-tracing. In this type of application rays can rebound between objects in the scene. Therefore, large rays or segments which do not have a common origin are obtained most of the time. In this test two random points enclosed in a fixed bounding box have been generated.
- **Test 2** (ray casting segment) tries to simulate rays or segments with a common origin as occurs in ray casting or non-recursive ray tracing. The common origin for all rays is situated at the scene observer position, and the end extreme of the segment is a random point on the back plane of the scene. For this test, no information is stored in the triangle structure (for example the normal of the triangle or the signed volume of tetrahedra formed by the common origin and the triangle).
- **Test 3** (random segment with limited length) tries to simulate segments with dimensions similar to the edges of the triangles of the mesh. These segments could belong to another triangle mesh that moves around the scene. Segments have been generated by a random point inside a bounding box, and a random directional vector with constant size. This test shows us the performance of the new algorithm for interference tests, which in turn is the principal focus of this paper.

Tests 1 to 3 determine both the time and the number of operations necessary for the intersection between a ray or segment and some objects represented by triangle meshes. These studies are based on a maximum hit rate of 0.05%. Objects formed by different numbers of triangles (Fig. 9) with similar triangle lengths have been used.

It has been observed that the percentage of operations carried out within each rejection test is similar regardless of the number of triangles of each object. Because of this, the average of the operations percentage has been calculated.

Furthermore, the behavior of algorithms with triangle meshes first without and later with back-face culling has been studied.

These tests associated to ray casting and ray tracing applications have been studied in order to notice how the new algorithm deals with these types of applications in comparison with interference tests.

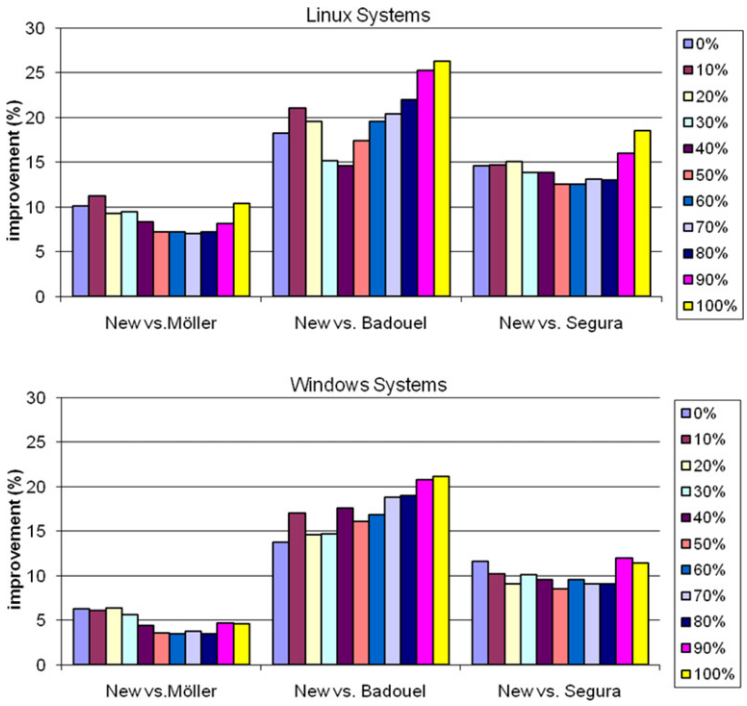

Fig. 10. Improvement percentage at different hit-ratios of the new algorithm with regard to Möller's, Badouel's and Segura's algorithm for test 0 using different operating systems.

5.2. Results

The algorithms of this study have been implemented in C, using different processors, operating systems, compilers and compiler optimizations. The set of processors used are listed below:

- Intel Centrino, 1.5 GHz, 1 GB RAM.
- Intel Core 2 Duo, 2.2 GHz, 2 GB RAM.
- Intel Core 2 Quad, 2.4 GHz, 4 GB RAM.

The set of operating systems and compilers used are the following:

- Linux OpenSUSE 10.3, with gcc and O2, O3 optimizations.
- Microsoft Windows Vista, with Microsoft Visual Studio C++ and SSE2, SSE3 optimizations.

Test 0 (different hit ratios) has been proved in these machines, operating systems and compilers. The average of the improvement percentage obtained for Linux and Windows Systems is shown in Fig. 10. These charts show the improvement of the new algorithm with regard to classical algorithms for 10 million segment/triangle pairs with different hit ratios. Each one represents the average improvement obtained using three different processors for a fixed operating system and compiler. With this data it can be concluded that the new algorithm is faster than the algorithms considered in this study. The range of improvement is between 7.50% and 12.50% approximately with regard to Möller's algorithm in Linux Systems; and between 4.00% and 6.50% approximately with regard to Möller's algorithm in Windows Systems. In general the improvement is higher for Linux Systems.

On the other hand, the number of operations performed by the algorithms is shown in Tables 1, 2 and 3. It can be observed that the new algorithm performs one operation less than Möller's algorithm (two comparisons and one multiplication less and two more additions than Möller's). If the intersection point is not needed, the new algorithm performs three operations less than Möller's algorithm (two comparisons, one multiplication and one division less and one more addition than Möller's). Tables 2 and 3 show the number of operations and the average percentage of operations for tests 1, 2 and 3. Which algorithm is faster cannot be concluded if the fact is not considered that, depending on the computer used, each type of operation will consume a different number of clock cycles. In any given computer the programmer must weight the operation time and apply this weighting to the tables shown.

The time consumed in the segment/triangle interference has been measured for 10,000 segments and some triangle meshes (Fig. 9). The results are shown in Fig. 11. In these charts we can see the improvement percentage of the new algorithm with regard to the classical algorithms with different systems.

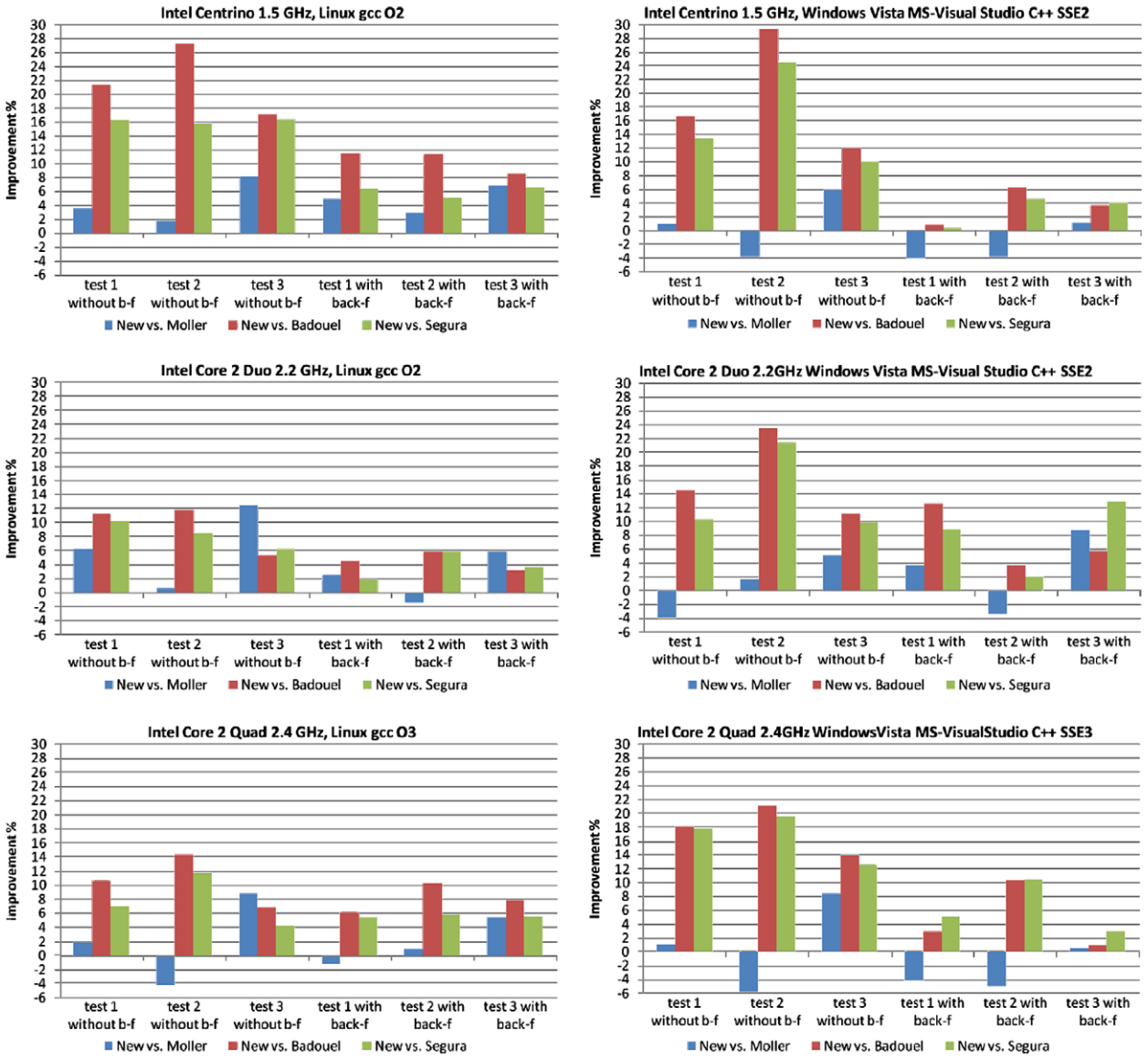

Fig. 11. Improvement percentage of the new algorithm with regard to Möller's, Badouel's and Segura's algorithm for tests 1 to 3 with and without back-face culling. Test 3 represents the improvement for interference tests.

An improvement percentage in the range from 5% to 12% can be observed for interference tests (test 3, the principal focus of the study) independently of the system used, obtaining higher improvement for Linux systems.

A negative improvement can be noticed in some situations for tests 1 and 2, when the new algorithm is compared to Möller's. This means that Möller's obtains faster times than the new algorithm using some systems. It can be considered that Möller's algorithm generally is more appropriate for ray casting situations (test 2), and that it is moderately appropriate for ray tracing situations (test 1), both algorithms obtaining similar times for this case. In general, Möller's algorithm obtains even faster times when back-face culling is used.

This data leads us to study in the future whether the new algorithm could be faster than Möller's for ray casting or ray tracing applications, taking advantage of some information stored in the triangle structure, as introduced in Section 4.

In general, a classification of the appropriateness of the algorithms studied cannot be obtained for ray casting and ray tracing problems. It is clear that this depends on the system used.

It can be observed that Segura's algorithm is faster than Möller's in some cases, but slower in general. Segura et al. [8] stated the superiority of their algorithm with regard to Möller's, but in their studies the comparisons were realized in an environment in which the mesh tested was static, and some information like the normal of the triangles or the signed volume of the tetrahedra was pre-calculated. In that context, Möller's algorithm cannot pre-calculate any information, being

slower than Segura's. In the circumstances stated in this paper, in which objects can move and pre-calculations are not possible, Segura's algorithm generally performs slower than Möller's.

For small segments (test 3) it can be concluded that the new algorithm is more efficient than the classical algorithms tested, as is usual in interference tests or collision detection, when the edges of the triangles are similarly sized most the time. If the localization of the intersection point is not needed the new algorithm will reduce the number of operations, and faster times could be achieved.

The theoretical study and the times obtained with the algorithm implementation show that the proposed algorithm is faster than classical algorithms for the interference problem.

6. Conclusions and future work

In conclusion, a segment/triangle intersection algorithm suitable for interference tests has been obtained. The number of theoretical operations carried out and the times obtained in an implementation of the algorithm improves on the number of operations and times obtained with classical algorithms. The results obtained with the new algorithm for ray casting or ray tracing suggest that the new algorithm could improve specific algorithms developed for these types of applications, if some information is stored inside the triangle structure. A study of the behavior of this algorithm for these types of applications will be carried out in the near future.

The advantage of the algorithm presented with regard to classical algorithms lies in the fact that the new algorithm is robust and more efficient than the previous ones due to the operations carried out with signs and the use of calculation, avoiding as well the division operation in order to determine whether an intersection occurs or not. The fewer number of operations, the optimized order of the calculations and the greater number of rejection tests gives us an efficient algorithm.

The new algorithm makes the most of the advantages of previous algorithms, being a simple, robust and efficient algorithm that carries out back-face culling when needed, and also calculates the barycentric coordinates of the intersection point if it is necessary. With a low computational cost the algorithm can calculate exclusively if an intersection occurs or not.

A study of the number of operations for several scenarios has been carried out. This data can be used to obtain the effective time for the calculation of intersection for different CPUs, in which the number of cycles used for each type of operation is different.

Future work to be undertaken includes a times study in which the calculations from one triangle to the adjacent one are reused effectively. The algorithm will be tested in real situations, such as ray casting, interference or collision detection applications.

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.

References

[1] T. Dey, Triangulations and csg representation of polyhedra with arbitrary genus, in: The 7th ACM Symposium on Computational Geometry, 1991, pp. 364–372.
[2] A. Paoluzzi, M. Ramella, A. Santarelli, Dimension-independent modelling with simplicial complexes, ACM TOG 12 (1993) 56–120.
[3] A. Glassner, An Introduction to Ray Tracing, Academic Press, 1989.
[4] A. Garcia-Alonso, N. Serrano, J. Flaquer, Solving the collision detection problem, IEEE Computer Graphics and Applications 3 (3) (1994) 35–43.
[5] D. Badouel, An efficient ray-polygon intersection, in: Graphics Gems, Academic Press, 1990.
[6] T. Möller, B. Trumbore, Fast, minimum storage ray/triangle intersection, Journal of Graphics Tools 2 (1) (1997) 21–28.
[7] R. Segura, F. Feito, An algorithm for determining intersection segment-polygon in 3d, Computers & Graphics 22 (5) (1998) 587–592.
[8] R. Segura, F. Feito, Algorithms to test ray-triangle intersection, comparative study, Journal of WSCG 9 (3) (2001) 76–81.
[9] R. Jones, Intersecting a ray and a triangle with Plücker coordinates, Ray Tracing News 13 (1) (2000), http://www1.acm.org/pubs/tog/resources/RTNews/html/rtnv13n1.html.
[10] A. Kensler, P. Shirley, Optimizing ray-triangle intersection via automated search, in: IEEE Symposium on Interactive Ray Tracing, 2006, pp. 33–38.
[11] M. Lin, S. Gottschalk, Collision detection between geometric models: A survey, in: IMA Conference on Mathematics of Surfaces, 1998.
[12] T. Möller, Practical analysis of optimized ray-triangle intersection, http://www.cs.lth.se/home/Tomas_Akenine_Moller/raytri/.
[13] C. Yap, Towards exact geometric computation, Computational Geometry: Theory and Applications 7 (1–2) (1997) 3–23.
[14] L. Guibas, D. Salesin, J. Stolfi, Epsilon geometry: Building robust algorithms from imprecise computations, in: Proceedings of the 5th Annual ACM Symposium on Computational Geometry, 1989.
[15] C. Hoffman, Geometric and Solid Modelling. An Introduction, Morgan Kaufmann Publishers, 1989.
[16] M. Löfstedt, T. Akenine-Möller, An evaluation framework for ray-triangle intersection algorithms, Journal of Graphics Tools 10 (2) (2005) 13–26.