A New Algorithm for Boolean Operations on General Polygons

Yu Peng, Jun-Hai Yong, Wei-Ming Dong, Hui Zhang, Jia-Guang Sun

A new algorithm for Boolean operations on general polygons

Yu Peng[a,*], Jun-Hai Yong[b], Wei-Ming Dong[a], Hui Zhang[b], Jia-Guang Sun[a,b]

[a]*Department of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China*
[b]*School of Software, Tsinghua University, Beijing 100084, PR China*

Abstract

A new algorithm for Boolean operations on general planar polygons is presented. It is available for general planar polygons (manifold or non-manifold, with or without holes). Edges of the two general polygons are subdivided at the intersection points and touching points. Thus, the boundary of the Boolean operation resultant polygon is made of some whole edges of the polygons after the subdivision process. We use the simplex theory to build the basic mathematical model of the new algorithm. The subordination problem between an edge and a polygon is reduced to a problem of determining whether a point is on some edges of some simplices or inside the simplices, and the associated simplicial chain of the resultant polygon is just an assembly of some simplices and their coefficients of the two polygons after the subdivision process. Examples show that the running time required by the new algorithm is less than one-third of that by the Rivero and Feito algorithm.

Keywords: Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems; Modeling packages; Computational geometry; Geometric modeling; Boolean operation; Polygon intersection

1. Introduction

Computing Boolean operations between general planar polygons is one of the most important problems in many areas such as computational geometry [1–4], computer graphics [5–10], geometric modeling [11,12] and computer aided design (CAD) [13–15]. In the literature, extensive research efforts have resulted in very efficient algorithms for Boolean operations on polygons [1,3–5,8–10,16–23]. However, most of these are either limited to certain types of polygons or tend to be complex and time consuming. The polygon clipping algorithms introduced by Liang and Barsky [18] and Maillot [19] assume that the clip polygon is a rectangle with edges parallel to the coordinate axes. The algorithms presented by O'Rourke [3], Preparata and Shamos [4], Foley et al. [5] and Sutherland and Hodgeman [8] are limited to convex clip polygons. That of Schutte does not permit concave polygons with holes [20]. Both the algorithm presented by Weiler [9,10] and the one by Sutherland and Hodgeman [8] are non-closed algorithms for Boolean operations on polygons, i.e. the output is possible to contain self-touching polygons, which is not allowed as input for these algorithms. Zhou [23] and Wu [22] give algorithms with a high time complexity of $O(n^2 \log n)$. Vatti's algorithm [21] and Andereev's algorithm [16] can handle the general cases in reasonable time. However, their implementation is complex [16,21]. Greiner and Hormann have presented an efficient and simple algorithm for clipping arbitrary polygons, which assumes that the twopolygons are not degenerate, i.e. each vertex of one polygon does not lie on an edge of the

*Corresponding author. Tel.: +86 10 62795455; fax: +86 10 62795460.
E-mail address: pengyu00@mails.tsinghua.edu.cn (Yu Peng).

other polygon [17]. de Berg et al. provide a note on Boolean operations for planar polygons using the map overlay algorithm [1].

Recently, Rivero and Feito [25] presented a simple and efficient algorithm for Boolean operations on planar polygons. It is suitable for general planar polygons (manifold or non-manifold, with or without holes). It is based on a mathematical model of simplicial chains [24], and is very easy to be implemented. Comparison between their work and previous work has been made in Ref. [25]. Ref. [25] shows that their algorithm is at present the most efficient algorithm among all the available ones that can be used to calculate the three Boolean operations.

This paper gives a simpler, more robust and more efficient Boolean operation algorithm with respect to the algorithm by Rivero and Feito. As with this algorithm, we also adopt the simplex theory to handle general polygons. However, we do not need to calculate the level of each simplex and the subordination between two simplices. We only need to judge whether a point is on an edge of a simplex or inside a simplex. Therefore, our algorithm is simpler than their algorithm. The new algorithm is robust. It can deal with general planar polygons (manifold or non-manifold, with or without holes, non-degenerate or degenerate). For example, it handles cases that cause the Rivero and Feito algorithm to fail. The new algorithm is efficient. Its execution time is less than one-third of one of the Rivero and Feito algorithm.

The remaining part of the paper is organized as follows. The necessary definitions and principles for our algorithm are given in Section 2. The algorithm is presented in Section 3. Some examples as well as some comparison, which are made in Section 4, will illustrate the efficiency and robustness of our algorithm. The concluding remarks are presented in Section 5.

2. Definitions and principles

A general planar polygon is a single polygon or a polygon consisting of a set of non-intersecting single polygons. The boundary of a single polygon consists of one outer contour and several non-intersecting inner contours. Each contour is represented by several directed edges and may be convex or concave. The outer contour is oriented in counterclockwise fashion and the inner contours in clockwise fashion. Thus, the interior of the polygon is on the left side of each directed edges. Some examples of general polygons are shown in Fig. 1. In our subsequent discussion, all polygons are assumed to be general planar polygons.

This paper assumes that the general polygons are defined in the first quadrant. A translation is enough to move any general polygon into the first quadrant if it is necessary to do so. We first introduce three preliminary definitions and a lemma of previous work [24,26,27] which are referred to in this paper. The definition of a simplicial chain is as follows:

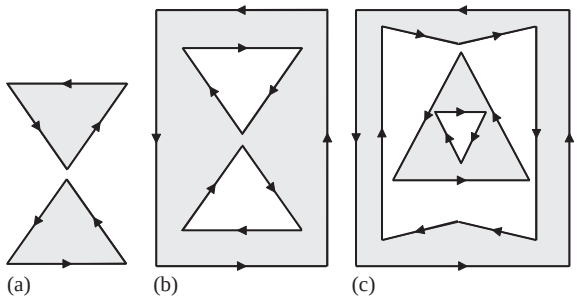

Fig. 1. Examples of general planar polygons: (a) two single polygons without inner contours; (b) a single polygon with two inner contours; (c) two single polygons with inner contours.

Definition 1. As shown in Fig. 2(a), a two-dimensional simplex is an ordered triangle. Given a sequence of simplices $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_n$, where $n \in \mathbf{Z}$, the signed areas of $\boldsymbol{S}_i$ $(i = 1, 2, \ldots, n)$ are denoted by $[\boldsymbol{S}_i]$, and the coefficients of $\boldsymbol{S}_i$ $(i = 1, 2, \ldots, n)$ are given by

$$\alpha_i = sign([\boldsymbol{S}_i]) = \begin{cases} 1 & \text{if } [\boldsymbol{S}_i] \text{ is positive,} \\ -1 & \text{if } [\boldsymbol{S}_i] \text{ is negative,} \\ 0 & \text{if } [\boldsymbol{S}_i] \text{ is zero.} \end{cases}$$

The collection of the sequence $\{\boldsymbol{S}_i\}_{i=1}^n$ and the sequence $\{\alpha_i\}_{i=1}^n$ is called a two-dimensional simplicial chain, and is denoted by $\lambda = \sum_{i=1}^n \alpha_i \cdot \boldsymbol{S}_i$.

In the remainder of the paper, when a coefficient of one simplex equals '+1' or '−1', we write '+' or '−' for short before the simplex for brevity, and we follow Ref. [24] to override the symbol "$\sum$". When it operates on two sequences (one that refers to simplices and the other to their coefficients), it signifies the collection; and when it operates on numbers, it signifies their algebraic sum. Thus, no confusion could arise over the symbol overriding.

Definition 2. A simplex is an original simplex if and only if the origin $\mathbf{O}$ is a vertex of the simplex. Let $\mathbf{OV}_1\mathbf{V}_2$ be an original simplex. The edge $\mathbf{OV}_1$ and the edge $\mathbf{OV}_2$ are called original edges, and the other edge $\mathbf{V}_1\mathbf{V}_2$ is called a non-original edge.

We suppose that the simplicial chains in the remainder of the paper are made of original simplices with their coefficients. From a two-dimensional simplicial chain, we could obtain a polygon associated with it. Fig. 2(b)

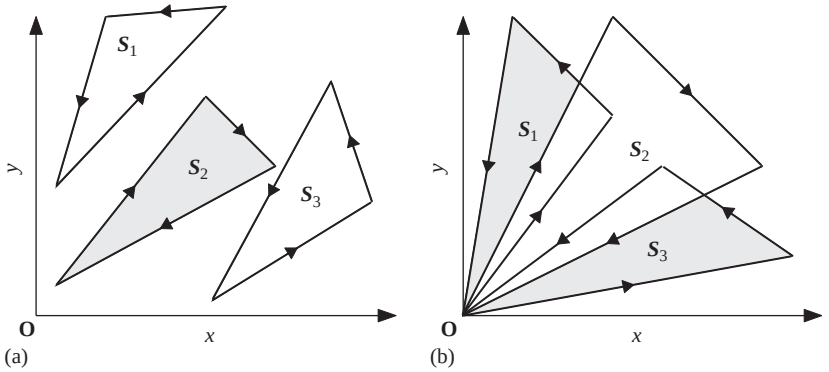

Fig. 2. Examples: (a) three simplices $\boldsymbol{S}_1$, $\boldsymbol{S}_2$ and $\boldsymbol{S}_3$ with coefficients 1, −1 (in gray) and 1, respectively; (b) a general polygon (gray area) associated with a simplicial chain $\lambda = \boldsymbol{S}_1 - \boldsymbol{S}_2 + \boldsymbol{S}_3$.

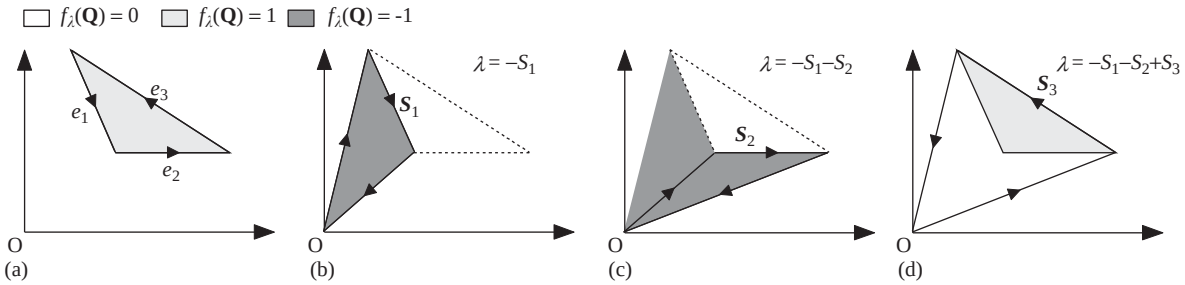

Fig. 3. An associated simplicial chain of a given triangle.

gives such an example, and the definition is as follows:

Definition 3. Given a simplicial chain

$$\lambda = \sum_{i=1}^{n} \alpha_i \cdot \boldsymbol{S}_i \text{ where } n \in \mathbf{Z} \text{ and } \alpha_i \ (i = 1, 2, \ldots, n)$$

are the coefficients of $\boldsymbol{S}_i$,

for any point $\mathbf{Q} \in \mathbf{R}^2$, a characteristic function f_λ on λ is defined by

$$f_\lambda(\mathbf{Q}) = \begin{cases} 1 & \text{if } \mathbf{Q} \text{ is on the non-original edge of some } \boldsymbol{S}_i \ (i = 1, 2, \ldots, n), \\ \sum_{i=1}^{n} \beta_i & \text{otherwise,} \end{cases}$$

where

$$\beta_i = \begin{cases} \alpha_i & \text{if } \mathbf{Q} \text{ is inside the open region of } \boldsymbol{S}_i, \\ \frac{1}{2}\alpha_i & \text{if } \mathbf{Q} \text{ is on some original edge of } \boldsymbol{S}_i, \\ 0 & \text{otherwise.} \end{cases}$$

The closed set of the point set given by

$$\mathbf{P}_\lambda = \{\mathbf{Q} | f_\lambda(\mathbf{Q}) = 1, \ \mathbf{Q} \in \mathbf{R}^2\}$$

is a polygon, called the polygon associated with λ.

Thus, from an arbitrary simplicial chain, we could obtain a polygon associated with it. Inversely, from a general polygon, we could find its associated simplicial chain as well. The method for generating an associated simplicial chain of a polygon is given by the following lemma.

Lemma 1. *Let* $\mathbf{P}$ *be a general polygon determined by* n $(n \in \mathbf{Z})$ *edges* $e_1, e_2, \ldots, e_n$. *Let* $\boldsymbol{S}_i$ *be the original simplex determined by the origin* $\mathbf{O}$ *and the edge* e_i, *and let* α_i *be the coefficient of* $\boldsymbol{S}_i$, *where* $i = 1, 2, \ldots, n$. *Then* $\mathbf{P}$ *is a general polygon associated with the simplicial chain* $\lambda = \sum \alpha_i \cdot \boldsymbol{S}_i$, *and* λ *is called the associated simplicial chain of the polygon* $\mathbf{P}$.

An example of an associated simplicial chain of a triangle is shown in Fig. 3. The associated simplicial chain of the triangle has three simplices ($\boldsymbol{S}_1, \boldsymbol{S}_2$ and $\boldsymbol{S}_3$ see Fig. 3). In this example, the coefficients of $\boldsymbol{S}_i$ $(i = 1, 2, 3)$ are −1, −1 and 1, respectively. According to Definition 3, the procedure for calculating the polygon associated with $\lambda = -\boldsymbol{S}_1 - \boldsymbol{S}_2 + \boldsymbol{S}_3$ is also given in Fig. 3. At each point $\mathbf{Q}$, the characteristic function $f_\lambda(\mathbf{Q})$ may equal 0, 1 or −1 for $\lambda = -\boldsymbol{S}_1$, $\lambda = -\boldsymbol{S}_1 - \boldsymbol{S}_2$ and $\lambda = -\boldsymbol{S}_1 - \boldsymbol{S}_2 + \boldsymbol{S}_3$, respectively. Those three different values are represented by three different gray values at the corresponding points in Fig. 3. As shown in Fig. 3, the resultant polygon (Fig. 3(d)) associated with the

simplicial chain $\lambda = -\boldsymbol{S}_1 - \boldsymbol{S}_2 + \boldsymbol{S}_3$ is exactly the given triangle (Fig. 3(a)).

In order to obtain an efficient and robust algorithm, we need some new definitions and theorems. Definition 4 gives the fundamental definition that a directed edge is contained within one general polygon.

Definition 4. A directed line segment $\vec{e}$ (for example, a directed edge of a general polygon) is contained within one general polygon $\mathbf{P}$ (denoted as $\vec{e} \subset \mathbf{P}$) if and only if exactly one of the following three conditions is satisfied:

- $\vec{e}$ is totally contained in the open region of $\mathbf{P}$ (for example $\vec{e}_1$ in Fig. 4),
- $\vec{e}$ is totally contained in the open region of $\mathbf{P}$, except at the start point or/and at the end point (for example $\vec{e}_2$, $\vec{e}_3$ in Fig. 4),
- both the start point and the end point of $\vec{e}$ lie on a directed edge of the polygon, which has the same orientation as $\vec{e}$ (for example $\vec{e}_4$, $\vec{e}_5$ in Fig. 4).

All possible cases of intersection between two edges e_1 and e_2 are as follows:

- e_1 and e_2 have no intersection point,
- e_1 and e_2 have exactly one intersection point that does not coincide with any of the vertices of e_1 or e_2,
- e_1 and e_2 have exactly one intersection point, and this point is a vertex of e_1 or e_2,
- the intersection result between e_1 and e_2 is a line segment.

In the remainder of this paper, the intersection point between two edges e_1 and e_2 is assumed to be of the second type. Thus, the number of the intersection points between two given edges is no more than 1, and that of the intersection points between the boundaries of two general polygons is always finite. For a case of the third type, a vertex of one edge, which lies on the other edge, will be called a touching point. For a case of the forth type, i.e. when one edge overlaps the other edge, the two endpoints of the shared line segment will be called touching points as well. The definition of touching points is summarized as follows.

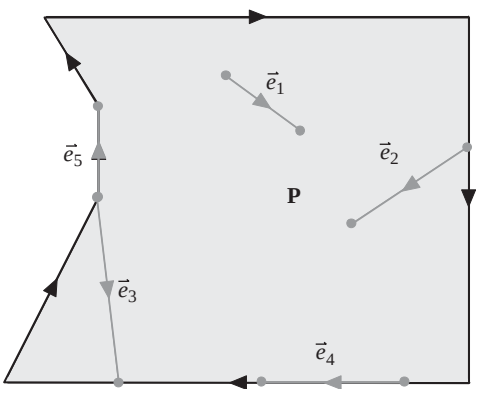

Fig. 4. Examples of directed edges contained within a general polygon.

Definition 5. Given two polygons, if an edge of one polygon overlaps an edge of the other or if a vertex of one polygon lies on an edge of the other, then the endpoints of the overlapped line segment and the vertex are called touching points.

Thus, the number of the intersection points and touching points of two general polygons is finite, and we could subdivide the two general polygons in the following way:

- calculate the intersection points and touching points of all edges from the two general polygons,
- subdivide the directed edges of the two general polygons into smaller directed edges at the intersection points and touching points,
- establish the new general polygons from the subdivided directed edges.

Fig. 5 gives an example of the subdivision process of two general polygons. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two polygons, and $\vec{e}$ be an arbitrary directed edge of $\mathbf{P}_1$. After the subdivision process, there are only three possible cases for the position relationship between $\vec{e}$ and $\mathbf{P}_2$:

- $-\vec{e}$ is a directed edge of $\mathbf{P}_2$,
- $\vec{e}$ is contained within $\mathbf{P}_2$, i.e., $\vec{e} \subset \mathbf{P}_2$,
- $\vec{e}$ has no point in the open region of $\mathbf{P}_2$.

Hence, we have the following theorem.

Theorem 1. *Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two general planar polygons after the subdivision process, and $\vec{e}$ be a directed edge of $\mathbf{P}_1$. Assume that neither $\vec{e}$ nor $-\vec{e}$ is a directed edge of $\mathbf{P}_2$. Then, $\vec{e} \subset \mathbf{P}_2$ if and only if the midpoint of $\vec{e}$ is in $\mathbf{P}_2$.*

In Theorem 1, the definition of $\vec{e} \subset \mathbf{P}_2$ is given by Definition 4. Let λ be the associated simplicial chain of $\mathbf{P}_2$, which is created according to Lemma 1. Definition 3 implies that the characteristic function $f_\lambda(\mathbf{Q})$ can decide whether a two-dimensional point $\mathbf{Q}$ belongs to $\mathbf{P}_2$. Therefore, simplex theory could be used to determine whether a directed edge is contained within a polygon or not.

Definition 6. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two general planar polygons after the subdivision process and let λ be the simplicial chain of $\mathbf{P}_2$. Then, for any directed edge $\vec{e}$ of $\mathbf{P}_1$, the characteristic function f of $\vec{e}$ on

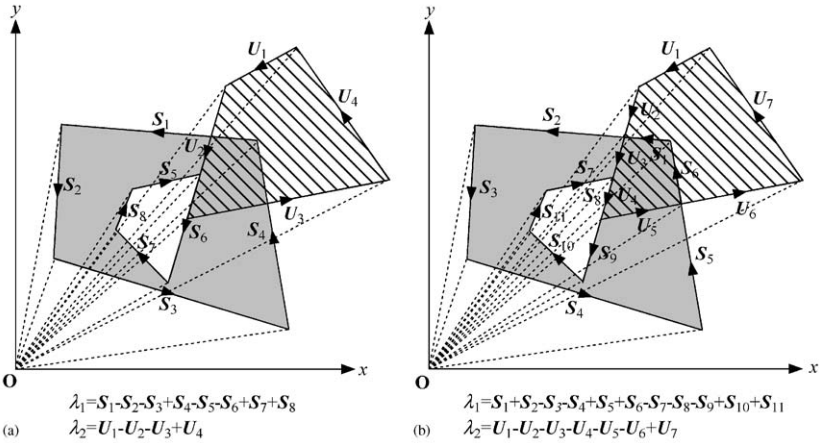

Fig. 5. An example of the subdivision process: (a) two polygons and their associated simplicial chains; (b) polygons after the subdivision process and their associated simplicial chains.

$\mathbf{P}_2$ is given by

$$f(\vec{e}) = \begin{cases} 1 & \text{if } \vec{e} \text{ is a directed edge of } \mathbf{P}_2, \\ 0 & \text{if } -\vec{e} \text{ is a directed edge of } \mathbf{P}_2, \\ f_\lambda(\mathbf{Q}) & \text{otherwise,} \end{cases}$$

where $\mathbf{Q}$ is the midpoint of $\vec{e}$, and $f_\lambda(\mathbf{Q})$ is the characteristic function f_λ on λ at the point $\mathbf{Q}$. The formula for $f_\lambda(\mathbf{Q})$ is given by Definition 3.

From Theorem 1 and Definition 6, we have the following corollary.

Corollary 1. *Let* $\mathbf{P}_1$ *and* $\mathbf{P}_2$ *be two general planar polygons after the subdivision process, and let* $\vec{e}$ *be a directed edge of* $\mathbf{P}_1$. *Then,* $\vec{e} \subset \mathbf{P}_2$ *if and only if* $f(\vec{e}) = 1$, *where* $f(\vec{e})$ *is the characteristic function f of* $\vec{e}$ *on* $\mathbf{P}_2$ *given by Definition* 6.

The following theorem uses simplex theory to obtain the Boolean operation result of two polygons.

Theorem 2. *Given two general planar polygons* $\mathbf{P}_1$ *and* $\mathbf{P}_2$ *after the subdivision process, let*

(1) $\lambda_1 = \sum_{i=1}^{m} a_i \cdot \boldsymbol{S}_i$ *and* $\lambda_2 = \sum_{j=1}^{n} b_j \cdot \boldsymbol{U}_j$ *be the associated simplicial chains of* $\mathbf{P}_1$ *and* $\mathbf{P}_2$, *respectively*,
(2) $E = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_m\}$ *and* $F = \{f_1, f_2, \ldots, f_n\}$ *be the sets of all the directed edges of the general polygons* $\mathbf{P}_1$ *and* $\mathbf{P}_2$, *respectively*,
(3) $E' = \{\vec{e} \mid \vec{e} \subset \mathbf{P}_2, \vec{e} \in E\} = \{\vec{e}_{m_1}, \vec{e}_{m_2}, \ldots, \vec{e}_{m_k}\}$, *where* $1 \leqslant m_1 \leqslant m_2 \leqslant \cdots \leqslant m_k \leqslant m$ *and*
(4) $F' = \{f \mid f \subset \mathbf{P}_1, f \in F\} = \{f_{n_1}, f_{n_2}, \ldots, f_{n_l}\}$, *where* $1 \leqslant n_1 \leqslant n_2 \leqslant \cdots \leqslant n_l \leqslant n$.

Then, the associated simplicial chain of the resultant intersection polygon $\mathbf{P}_3 = \mathbf{P}_1 \cap \mathbf{P}_2$ *is*

$$\lambda_{\mathbf{P}_3} = \sum_{i=1}^{k} a_{m_i} \cdot \boldsymbol{S}_{m_i} + \sum_{j=1}^{l} b_{n_j} \cdot \boldsymbol{U}_{n_j}, \text{ if } E' \cap F' = \phi \tag{1}$$

or

$$\lambda_{\mathbf{P}_3} = \sum_{i=1}^{k} a_{m_i} \cdot \boldsymbol{S}_{m_i} + \sum_{j=1}^{l} b_{n_j} \cdot \boldsymbol{U}_{n_j} - \sum_{h=1}^{r} b_{l_h} \cdot \boldsymbol{U}_{l_h},$$
$$\text{if } E' \cap F' = \{\vec{f}_{l_1}, \vec{f}_{l_2}, \ldots, \vec{f}_{l_r}\}, \tag{2}$$

where $n_1 \leqslant l_1 \leqslant l_2 \leqslant \cdots \leqslant l_r \leqslant n_l$.

Proof. Let $G = \{\vec{g}_1, \vec{g}_2, \ldots, \vec{g}_s\}$ be the directed edge set of $\mathbf{P}_3$, and $\hat{G} = \{\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_s\}$ be the non-directed edge set of $\mathbf{P}_3$. Let $\hat{E} = \{\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_m\}$ and $\hat{F} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n\}$ be the non-directed edge sets of $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. From $\mathbf{P}_3 = \mathbf{P}_1 \cap \mathbf{P}_2$, where $\mathbf{P}_1$and $\mathbf{P}_2$ are the polygons after the subdivision process, we have

$$\hat{G} \subseteq \hat{E} \cup \hat{F}.$$

The interior area of $\mathbf{P}_3$ is that of $\mathbf{P}_1$ or that of $\mathbf{P}_2$, and the interior area of a general polygon is on the left side of a directed edge of the polygon. Therefore, any directed edge from $\mathbf{P}_3$ has the same orientation as the corresponding directed edge from $\mathbf{P}_1$ or $\mathbf{P}_2$. Thus, we have

$$G \subseteq E \cup F.$$

For any directed edge $\vec{g}$ from $\mathbf{P}_3$, i.e. $\vec{g} \in G$, because $G \subseteq E \cup F$, we have $\vec{g} \in E$ or $\vec{g} \in F$. Without loss of generality, assume that $\vec{g} \in E$. Because $\mathbf{P}_3 = \mathbf{P}_1 \cap \mathbf{P}_2$, $\vec{g}$ is fully contained by $\mathbf{P}_2$. The interior area of $\mathbf{P}_3$ is the interior area of $\mathbf{P}_2$ as well. Therefore, if the non-directed

edge $\hat{g}$ of $\vec{g}$ is not a non-directed edge of $\mathbf{P}_2$, we have that $\vec{g} \subset \mathbf{P}_2$ through Definition 4. Otherwise, the non-directed edge $\hat{g}$ of $\vec{g}$ is equivalent to a non-directed edge $\hat{f}$ of $\mathbf{P}_2$. Because the interior area of $\mathbf{P}_3$ is on the left side of directed edges of the both $\mathbf{P}_2$ and $\mathbf{P}_3$, the corresponding directed edge f of $\hat{f}$ has the same orientation as $\vec{g}$. Thus, $\vec{g} = f$. According to Definition 4, we have $\vec{g} \subset \mathbf{P}_2$ in the case that $\hat{g}$ is a non-directed edge of $\mathbf{P}_2$. From these two cases, we can conclude that $\vec{g} \subset \mathbf{P}_2$. Hence, we have $\vec{g} \in E'$ in the case that $\vec{g} \in E$, whence $\vec{g} \in E' \cup F'$. Similarly, we could show that $\vec{g} \in E' \cup F'$ for the case where $\vec{g} \in F$ as well. Therefore, we have

$$G \subseteq E' \cup F'.$$

Similarly, we could show that $E' \cup F' \subseteq G$. From $G \subseteq E' \cup F'$ and $E' \cup F' \subseteq G$, we have $G = E' \cup F'$, i.e. we obtain all directed edges of the resultant polygon $\mathbf{P}_3$. According to Lemma 1, we build the associated simplicial chain of the general polygon $\mathbf{P}_3$ as shown in Eqs. (1) and (2). □

3. Boolean operation algorithm

The new Boolean operation algorithm is based on the simplex theory presented in Section 2. At first, the subdivision process is performed on the two general polygons $\mathbf{P}_1$ and $\mathbf{P}_2$. Then, we build the simplicial chains for both general polygons after the subdivision process according to Lemma 1, and we calculate the value of the characteristic function for each edge of one polygon on the other's original polygon (i.e. the polygon before the subdivision process) according to Corollary 1. At last, the simplicial chain of the Boolean operation resultant polygon $\mathbf{P}_3$ is obtained according to Theorem 2, and we have the polygon $\mathbf{P}_3$ from the simplicial chain according to Definition 1. Thus, the pseudo code for the Boolean operation algorithm is as shown in Fig. 6.

The subdivision process is already described in Section 2. How to build a simplicial chain is presented in Lemma 1. Fig. 7 gives the pseudo code for calculating the values of the characteristic functions of the edges of one polygon on the other polygon.

According to Definition 6, when a directed edge is shared by two polygons, both values of the characteristic functions of this edge on the polygons are 1. In Theorem 2, this edge belongs to E' as well as F'. Therefore, according to Eq. (2), the corresponding simplex of this edge will be added to the resultant simplicial chain twice and then removed once. In the above pseudo code, we force $f(u_j^*, \mathbf{P}_1) = 0$ when u_j^* is such an edge shared by both $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$. Hence, the corresponding simplex of this edge will be added to the resultant simplicial chain only once. The pseudo code for calculating the resultant simplicial chain is as shown in Fig. 8.

After the resultant simplicial chain is obtained, we could obtain the resultant polygon directly according to Definition 3. Since the two general polygons and the resultant polygon are represented by their simplicial chains (i.e. they are the same as those in the Rivero and Feito algorithm), we can use Rivero and Feito's union/subtraction approaches to calculate the union polygon and the subtraction polygon with the simplicial chain of the intersection polygon.

Let n and m be the number of edges of the two polygons $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively, and let k be the number of intersection points and touching points of $\mathbf{P}_1$ and $\mathbf{P}_2$. For the subdivision process, we extend the plane sweep algorithm introduced in Refs. [1,3,4] to calculate the intersection points and touching points. Therefore, the execution time required by the subdivision process is $O((n+m)\log(n+m)+k)$. While calculating the values

```
INTERSECTION (P1, P2, P3) // Input two general polygon P1 and P2, and output the Boolean operation resultant polygon P3=P1 ⋂ P2.
{
    SUBDIVISIONPROCESS (P1, P2, P1*, P2*);  // Perform the subdivision process on two polygons P1 and P2;
                                            // P1* and P2* are the general polygons after the subdivision process;
    BUILDSIMPLICIALCHAIN (P1*, λ1*);        // λ1* is the associated simplicial chain of polygon P1*;
                                            // λ1* is the collection of the sequence of simplices S1*, S2*,..., Sn*, and the sequence of
                                            // coefficients α(Si*) of Si*, where n is the number of edges of polygon P1*;
    BUILDSIMPLICIALCHAIN (P2*, λ2*);        // λ2* is the associated simplicial chain of polygon P2*;
                                            // λ2* is the collection of the sequence of simplices U1*, U2*,..., Um*, and the sequence of
                                            // coefficients α(Uj*) of Uj*, where m is the number of edges of polygon P2*;
    CALCHARACTEROFEDGE (P1, P2, P1*, P2*, λ1, λ2, λ1*, λ2*, f(si*, P2), f(uj*, P1));
                                            // Here, λ1 and λ2 are the associated simplicial chains of P1 and P2, respectively;
                                            // si* and uj* are the non-original edges of the simplices Si* and Uj* , respectively;
                                            // f(si*, P2) is the characteristic function of si* on P2 (see Definition 6);
                                            // f(uj*, P1) is the characteristic function of uj* on P1 if uj* is not a directed edge of P1*;
                                            // f(uj*, P1)=0 if uj* is a directed edge of P1*;
    CALRESULTCHAIN (λ1*, λ2*, f(si*, P2), f(uj*, P1), λ3);
                                            // Calculate the resultant simplicial chain λ3 Using Theorem 2;
    CALPOLYGONFROMCHAIN (λ3, P3);           // Calculate the resultant polygon P3 from λ3, according to Definition 1.
}
```

Fig. 6. Pseudo code for the intersection algorithm.

```
bool CALBETA (Q, S, α(S), β)
// Input:  a point Q, a simplex S, and its coefficient α(S);
// Output: the value of β according to Definition 3;
// Return: return true if the value of β is not 0 according to Definition 3; otherwise return false.
{
    if (Q is on an original edge of S)
      β= 0.5*α(S);
      return true;
    else if (Q is in S);
      β= α(S);
      return true;
    end if
    return false;
}

CALCHARACTEROFEDGE (P1, P2, P1*, P2*, λ1, λ2, λ1*, λ2*, f(si*, P2), f(uj*, P1))
// Input:  P1 and P2 are two polygons, and λ1 and λ2 are their simplicial chains, respectively;
//         P1* and P2* are two polygons after the subdivision process, and λ1* and λ2* are their simplicial chains, respectively;
// Output: f(si*, P2) is the characteristic function of si* on polygon P2;
//         f(uj*, P1) is the characteristic function of uj* on polygon P1 if uj* is not an edge of P1*; otherwise, f(uj*, P1)=0.
{
    Let E1 = F1 = {e | e is a directed edge of P1* as well as P2* };
    Let E2 = {e | e is a directed edge of P1*, and -e is a directed edge of P2* };
    Let F2 = {e | -e is a directed edge of P1*, and e is a directed edge of P2* };
    for each simplex Si* of λ1* do                    // Initiate the function f(si*, P2);
      f(si*, P2) = 0;                                 // Here, si* is the non-original edge of Si*;
    end for
    for each simplex Uj* of λ2* do                    // Initiate the function f(uj*, P1);
      f(uj*, P1) = 0;                                 // Here, uj* is the non-original edge of Uj*;
    end for
    for each simplex Si* of λ1* do                    // Here, we calculate the values of the characteristic functions
      if (si*∈E1)                                     // of the edges of polygon P1* on polygon P2.
        f(si*, P2) = 1;
      else
        for each simplex Uj of λ2 do
          Let Q be the midpoint of si*;
          if (si*∉E2 && CALBETA (Q, Ui, α(Ui), β))    // α(Ui) is the coefficient of the simplex Ui;
              f(si*, P2) = f(si*, P2) + β;
          end if
        end for                                       // End of the inner loop;
      end if
    end for                                           // End of the outer loop.
    for each simplex Uj* of λ2* do                    // Here, we calculate the values of the characteristic functions
      for each simplex Si of λ1 do                    // of the edges of polygon P2* on polygon P1.
        Let Q be the midpoint of uj*;
        if (uj*∉F1 && uj*∉F2 && CALBETA (Q, Sj, α(Sj), β))// α(Sj) is the coefficient of the simplex Sj;
            f(uj*, P1) = f(uj*, P1) + β;
        end if
      end for                                         // End of the inner loop;
    end for                                           // End of the outer loop.
}
```

Fig. 7. Pseudo code for calculating the values of the characteristic functions.

of the characteristic functions, two double-nested loops are used, so its execution time is $O((n(m+k)+m(n+k)) = O(2nm+k(n+m))$. The subroutine CALRESULTCHAIN uses two consecutive loops, so its execution time is $O(n+m+k)$.

Compared to the Rivero and Feito algorithm, the new algorithm is more efficient, as the execution time required by their algorithm is $O((n+k)(m+k))$. The new algorithm does not need to calculate the levels of all simplices of the two general polygons as a preprocessing step as does the Rivero and Feito algorithm, since the execution time needed for the calculation of the levels is $O((n+k)\log(n+k))$. Furthermore, the subordination problem between an edge and a polygon in the subroutine INTERSECTION of the Rivero/Feito algorithm is reduced to the problem for determining whether a

point is on some edges of some simplices or inside the simplices in the new algorithm. The determination as to whether a point belongs to a simplex used by this paper is much simpler and more efficient than the determination of the subordination between two simplices.

4. Experimental results

Rivero and Feito [25] have shown that their algorithm is robust. This paper compares the results of our algorithm with those of Rivero and Feito's. The first example shows that our algorithm is more robust than the Rivero and Feito algorithm. In this example, the Rivero and Feito algorithm produces an incorrect result, while our algorithm achieves the correct result.

The two general polygons are shown in Fig. 9(a), and the coordinates of their vertices are also shown in this figure. The first step of the Rivero and Feito algorithm calculates the intersection points of the edges of the two polygons, subdivides the edges at these intersection points, and forms the associated simplicial chains as shown in Fig. 9(b), which are

$$\lambda_{\mathbf{P}_1^*} = \boldsymbol{S}_{11} + \boldsymbol{S}_{12} + \boldsymbol{S}_2 - \boldsymbol{S}_{31} - \boldsymbol{S}_{32} + \boldsymbol{S}_{41} + \boldsymbol{S}_{42} - \boldsymbol{S}_5 + \boldsymbol{S}_{61} + \boldsymbol{S}_{62} - \boldsymbol{S}_{71} - \boldsymbol{S}_{72} + \boldsymbol{S}_{81} + \boldsymbol{S}_{82}$$

and

$$\lambda_{\mathbf{P}_2^*} = -\boldsymbol{U}_{11} - \boldsymbol{U}_{12} + \boldsymbol{U}_{21} + \boldsymbol{U}_{22} - \boldsymbol{U}_{31} - \boldsymbol{U}_{32} - \boldsymbol{U}_{41} - \boldsymbol{U}_{42} + \boldsymbol{U}_{51} + \boldsymbol{U}_{52} + \boldsymbol{U}_{61} + \boldsymbol{U}_{62} + \boldsymbol{U}_7.$$

The second step of the Rivero and Feito algorithm calculates the levels of all simplices, and classifies all simplices in layers according to their levels. In this example, the result of this step is

$\mathbf{P}_1^*$: layer 1: $\boldsymbol{S}_{11}, \boldsymbol{S}_{12}, \boldsymbol{S}_2, \boldsymbol{S}_{61}, \boldsymbol{S}_{81}, \boldsymbol{S}_{82}$;
layer 2: $\boldsymbol{S}_{31}, \boldsymbol{S}_{32}, \boldsymbol{S}_{71}, \boldsymbol{S}_{72}$;
layer 3: $\boldsymbol{S}_{41}, \boldsymbol{S}_{42}, \boldsymbol{S}_{62}$;
layer 4: $\boldsymbol{S}_5$;

and

$\mathbf{P}_2^*$: layer 1: $\boldsymbol{U}_{22}, \boldsymbol{U}_{51}, \boldsymbol{U}_{52}, \boldsymbol{U}_{61}, \boldsymbol{U}_{62}, \boldsymbol{U}_7$;
layer 2: $\boldsymbol{U}_{11}, \boldsymbol{U}_{12}, \boldsymbol{U}_{32}, \boldsymbol{U}_{41}, \boldsymbol{U}_{42}$;
layer 3: $\boldsymbol{U}_{21}$;
layer 4: $\boldsymbol{U}_{31}$.

The third step of the Rivero and Feitos algorithm, which obtains the associated simplicial chain of the resultant Boolean operation polygon $\mathbf{P}_3 = \mathbf{P}_1 \cap \mathbf{P}_2$, computes the data shown in Table 1. In Table 1, the first row and the first column list the simplices with their levels in brackets of $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$, respectively. The second row and the second column list the numbers of layers, in which the simplices contain the given simplex. In the interior of the table, the entries of two numbers in brackets represent the combination of subordination and the coefficients of the given simplices. An asterisk indicates that the corresponding data should be ignored. At each row, the sum of the first numbers of the entries at this row is

```
CALRESULTCHAIN (λ1*, λ2*, f(si*, P2), f(uj*, P1), λ3)
// Input:   the simplicial chains λ1* and λ2* of P1* and P2*, respectively;
            the characteristic functions f(si*, P2) and f(uj*, P1);
// Output:  the resultant simplicial chain λ3.
{
    for each simplex Si* of λ1* do
        if (f(si*, P2) is equal to 1)  // Here, si* is the non-original edge of Si*;
            Add the simplex Si* and its coefficient to λ3;
        end if
    end for
    for each simplex Uj* of λ2* do
        if (f(uj*, P1) is equal to 1)  // Here, uj* is the non-original edge of Uj*.
            Add the simplex Uj* and its coefficient to λ3;
        end if
    end for
}
```

Fig. 8. Pseudo code for calculating the resultant simplicial chain.

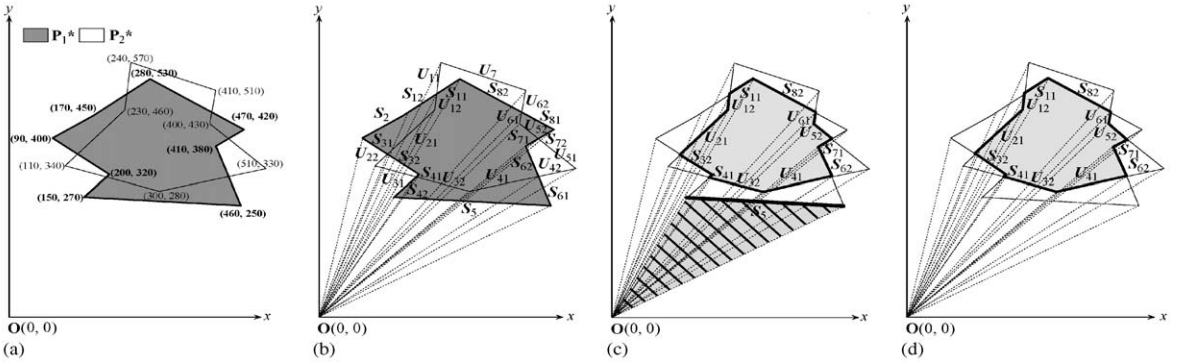

Fig. 9. Example 1: (a) two polygons; (b) associated simplicial chains of polygons after the subdivision process; (c) result calculated by the Rivero and Feito algorithm; (d) result calculated by our algorithm.

Table 1
Data calculated for Example 1 using the Rivero and Feito algorithm

Simp. $\mathbf{P}_2^*$	Simp. $\mathbf{P}_1^*$	$S_{11}(1)$	$S_{12}(1)$	$S_2(1)$	$S_{31}(2)$	$S_{32}(2)$	$S_{41}(3)$	$S_{42}(3)$	$S_5(4)$	$S_{61}(1)$	$S_{62}(3)$	$S_{71}(2)$	$S_{72}(2)$	$S_{81}(1)$	$S_{82}(1)$	Coefficient (U_j)
	LevelsDom(S_i) →	1	2,1			2,3,1	1	4,1	4,2,1	2,1	1	1			1	
	LevelsDom(U_j) ↓															
$U_{11}(2)$		(0,0)	(0,−1)	(0,0)	(0,0)	(0,1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	0
$U_{12}(2)$	1	(−1,0)	(0,0)	(0,0)	(0,0)	(0,1*)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	−1
$U_{21}(3)$	1	(1,0)	(1*,0)	(1*,0)	(0,0)	(0,−1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	1
$U_{22}(1)$	1,2	(0,0)	(0,0)	(1,0)	(−1,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	1 − 1 = 0
$U_{31}(4)$	1,2	(−1,0)	(−1*,0)	(−1*,0)	(1,0)	(1*,0)	(0,0)	(0,−1)	(0,1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(−1*,0)	−1 + 1 = 0
$U_{32}(2)$	2,3,1	(0,0)	(0,0)	(0,0)	(0,0)	(1,0)	(−1,0)	(0,0)	(0,1)	(0,0)	(0,0)	(0,0)	(1*,0)	(−1,0)	(−1*,0)	1 − 1 − 1 = −1
$U_{41}(2)$	3,2,1	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1*)	(0,0)	(−1,0)	(1,0)	(1*,0)	(−1,0)	(0,0)	1 − 1 − 1 = −1
$U_{42}(2)$		(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1*)	(0,−1)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	0
$U_{51}(1)$	2,1	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,−1)	(0,1)	(0,0)	(0,0)	(−1,0)	(1,0)	(0,0)	−1 + 1 = 0
$U_{52}(1)$	1	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,−1*)	(0,0)	(0,1*)	(0,−1)	(0,0)	(1,0)	(0,0)	1
$U_{61}(1)$	1	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,−1*)	(0,0)	(0,0)	(0,0)	(0,0)	(1,0)	(0,0)	1
$U_{62}(1)$		(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,−1*)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	0
$U_7(1)$		(0,1)	(0,1)	(0,0)	(0,0)	(0,−1)	(0,1)	(0,1)	(0,−1*)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1*)	0
Coefficient(S_i)		1	−1 + 1 = 0	0	0	1 − 1 − 1 = −1	1	−1 + 1 = 0	1 + 1 − 1 = 1	−1 + 1 = 0	1	−1	0	0	1	

recorded at the last column; and at each column, the sum of the second numbers of the entries at this column is recorded at the last row. Then, the simplices, whose corresponding numbers appear at the last column (for simplices of $\mathbf{P}_2^*$) and at the last row (for simplices of $\mathbf{P}_1^*$) are not zero, are selected to build the resultant simplicial chain. Hence, the resultant simplicial chain of the Rivero and Feito algorithm for Example 1 is

$$\lambda_{\mathbf{P}_1\cap\mathbf{P}_2} = \boldsymbol{S}_{11} - \boldsymbol{S}_{32} + \boldsymbol{S}_{41} + \boldsymbol{S}_5 + \boldsymbol{S}_{62} - \boldsymbol{S}_{71} + \boldsymbol{S}_{82} - \boldsymbol{U}_{12} + \boldsymbol{U}_{21} - \boldsymbol{U}_{32} - \boldsymbol{U}_{41} + \boldsymbol{U}_{52} + \boldsymbol{U}_{61}.$$

The resultant polygon calculated by the Rivero and Feito algorithm is in the gray area as shown in Fig. 9(c). However, this result is incorrect. It has one extra triangle with shading. The correct result is shown in Fig. 9(d).

We believe that the reason for this incorrect result is as follows. From Theorem 3 in Ref. [25], if a simplex is subordinated to two or more simplices from another polygon in the same layer, only one of these could make a contribution to the simplex. In this example, $\boldsymbol{S}_5$ is subordinated to $\boldsymbol{U}_{31}, \boldsymbol{U}_{32}, \boldsymbol{U}_{41}, \boldsymbol{U}_{42}, \boldsymbol{U}_{51}, \boldsymbol{U}_{52}, \boldsymbol{U}_{61}$ and $\boldsymbol{U}_{62}$. From this theorem, $\boldsymbol{U}_{31}$, $\boldsymbol{U}_{32}$ and $\boldsymbol{U}_{51}$ make contributions to $\boldsymbol{S}_5$, i.e. the coefficient of $\boldsymbol{S}_5$ in the resultant simplicial chain is $(-1)^*(-1) + (-1)^*(-1) + (-1)^*(1) = 1 + 1 - 1 = 1$. Therefore, the theorem leads us to the erroneous conclusion that $\boldsymbol{S}_5$ is a simplex from the resultant simplicial chain associated with the intersection polygon.

To illustrate the procedure for calculating the values of the characteristic functions, in the remainder of the paper, $\mathbf{P}_1$ and $\mathbf{P}_2$ are replaced by $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$ in the characteristic functions of the subroutine CalCharacterOfEdge, respectively. As $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$ are equivalent to $\mathbf{P}_1$ and $\mathbf{P}_2$ in shape (the latter have fewer vertices), respectively, the resultant simplicial chains associated with the intersection polygons made from them are the same.

The new algorithm first performs the subdivision process on the two general polygons not only according to the intersection points but also the touching points. In this example, after the subdivision process, the two simplicial chains are the same as those of the Rivero and Feito algorithm. The simplices and their coefficients are shown in the first row and the first column of Table 2, respectively. Table 2 also shows the procedure for calculating the values of the characteristic functions of the edges of one polygon on the other. In the interior of Table 2, each table cell contains an entry of two numbers in brackets. Suppose that each cell corresponds to a simplex $\boldsymbol{S}$ (the first row above this cell) of $\mathbf{P}_1^*$ and a simplex $\boldsymbol{U}$ (the first column on the left side of the cell) of $\mathbf{P}_2^*$, and that the non-original edges of $\boldsymbol{S}$ and $\boldsymbol{U}$ are s and u, respectively. Then, according to the subroutine CalCharacterOfEdge in our algorithm, the entry in this cell is obtained as follows. The following conditions

Table 2
Data calculated for Example 1 using the new algorithm

	$S_{11}(1)$	$S_{12}(1)$	$S_2(1)$	$S_{31}(-1)$	$S_{32}(-1)$	$S_{41}(1)$	$S_{42}(1)$	$S_5(-1)$	$S_{61}(1)$	$S_{62}(1)$	$S_{71}(-1)$	$S_{72}(-1)$	$S_{81}(1)$	$S_{82}(1)$	$f(U_j, \mathbf{P}_1^*)$
$U_{11}(-1)$	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	0
$U_{12}(-1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$U_{21}(1)$	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$U_{22}(1)$	(0, 0)	(0, 0)	(1, 0)	(−1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1 − 1 = 0
$U_{31}(-1)$	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(−1, 0)	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1 − 1 = 0
$U_{32}(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	1
$U_{41}(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, −1)	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$U_{42}(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	0
$U_{51}(1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(0, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	0
$U_{52}(1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(1, 0)	(0, 0)	1
$U_{61}(1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	1
$U_{62}(1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	0
$U_7(1)$	(0, 1)	(0, 1)	(0, 0)	(0, 0)	(0, 1)	(0, 1)	(0, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)	0
$f(S_i, \mathbf{P}_2^*)$	1	−1 + 1 = 0	0	0	−1 + 1 + 1 = 1	1	−1 + 1 = 0	−1 + 1 = 0	0	1	1	0	0	1	

are checked one by one from top to bottom. Once this cell is filled with an entry, no further more checks are needed:

- if s is a directed edge of $\mathbf{P}_2^*$, then the entry is (0,*), and the number at the last row under this cell is filled with the number 1,
- if $-s$ is a directed edge of $\mathbf{P}_2^*$, then the entry is (0,*), and the number at the last row under this cell is filled with the number 0,
- if u or $-u$ is a directed edge of $\mathbf{P}_1^*$, then the entry is (*, 0), and the number at the last column on the right side of this cell is filled with the number 0,
- if the midpoint of s lies inside $\boldsymbol{U}$, then the entry is $(0, b)$, where b is the coefficient of $\boldsymbol{U}$; otherwise, if the midpoint lies on an original edge of $\boldsymbol{U}$, then the entry is $(0, b/2)$,
- if the midpoint of u lies inside $\boldsymbol{S}$, then the entry is $(a, 0)$, where a is the coefficient of $\boldsymbol{S}$; otherwise, if the midpoint lies on an original edge of $\boldsymbol{S}$, then the entry is $(a/2, 0)$,
- if none of the above conditions are satisfied, then the entry is $(0, 0)$.

After the interior cells of the table are filled, then we begin to calculate the numbers at the last row and the last column of the table. At each row, the sum of the first numbers of each entry of this row is recorded at the last column of the table; and at each column, the sum of the second numbers of each entry of this column is recorded at the last row of the table. Then, the simplices, whose corresponding numbers appear at the last column (for simplices of $\mathbf{P}_2^*$) and at the last row (for simplices of $\mathbf{P}_1^*$) are 1, respectively, are selected to build the resultant simplicial chain. Hence, the resultant simplicial chain using the new algorithm for Example 1 is

$$\lambda_{\mathbf{P}_1 \cap \mathbf{P}_2} = \boldsymbol{S}_{11} - \boldsymbol{S}_{32} + \boldsymbol{S}_{41} + \boldsymbol{S}_{62} - \boldsymbol{S}_{71} + \boldsymbol{S}_{82} - \boldsymbol{U}_{12} + \boldsymbol{U}_{21} - \boldsymbol{U}_{32} - \boldsymbol{U}_{41} + \boldsymbol{U}_{52} + \boldsymbol{U}_{61}.$$

The resultant intersection polygon calculated by the new algorithm is indicated by the gray area shown in Fig. 9(d).

Example 2 shows that our algorithm can deal with general polygons which share some common edges. The two general polygons and their associated simplicial chains after the subdivision process are shown in Fig. 10(a). The associated simplicial chains of the general polygons are

$$\lambda_{\mathbf{P}_1^*} = \boldsymbol{S}_1 - \boldsymbol{S}_{21} - \boldsymbol{S}_{22} - \boldsymbol{S}_{23} + \boldsymbol{S}_{31} + \boldsymbol{S}_{32} + \boldsymbol{S}_{33}$$

and

$$\lambda_{\mathbf{P}_2^*} = -\boldsymbol{U}_1 - \boldsymbol{U}_2 + \boldsymbol{U}_3 - \boldsymbol{U}_{41} - \boldsymbol{U}_{42} + \boldsymbol{U}_5 + \boldsymbol{U}_6.$$

In this example, the simplex $\boldsymbol{S}_{22}$ is equivalent to the simplex $\boldsymbol{U}_2$, and the simplex $\boldsymbol{S}_{31}$ has the opposite

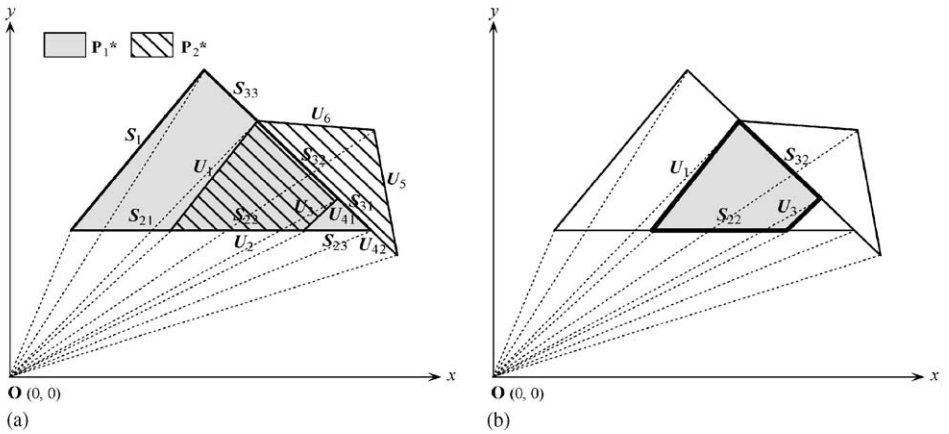

Fig. 10. Example 2: (a) two polygons with their associated simplicial chains, (b) the resultant polygon (in gray).

Table 3
Data calculated for Example 2 using the new algorithm

	$S_1(1)$	$S_{21}(-1)$	$S_{22}(-1)$	$S_{23}(-1)$	$S_{31}(1)$	$S_{32}(1)$	$S_{33}(1)$	$f(U_j, P_1^*)$
$U_1(-1)$	(0, 0)	(0, 0)	(0,*)	(0, 0)	(0,*)	(1, 0)	(0, 0)	1
$U_2(-1)$	(*, 0)	(*, 0)	(*,*)	(*, 0)	(*,*)	(*, 0)	(*, 0)	0
$U_3(1)$	(0, 0)	(0, 0)	(0,*)	(0, 0)	(1,*)	(0, 0)	(0, 0)	1
$U_{41}(-1)$	(*, 0)	(*, 0)	(*,*)	(*, −1)	(*,*)	(*, 0)	(*, 0)	0
$U_{42}(-1)$	(0, 0)	(0, 0)	(0,*)	(0, 0)	(0,*)	(0, 0)	(0, 0)	0
$U_5(1)$	(0, 0)	(0, 0)	(0,*)	(0, 1)	(0,*)	(0, 0)	(0, 0)	0
$U_6(1)$	(0, 0)	(0, 0)	(0,*)	(0, 0)	(0,*)	(0, 1)	(0, 0)	0
$f(S_i, P_2^*)$	0	0	1	−1 + 1 = 0	0	1	0	

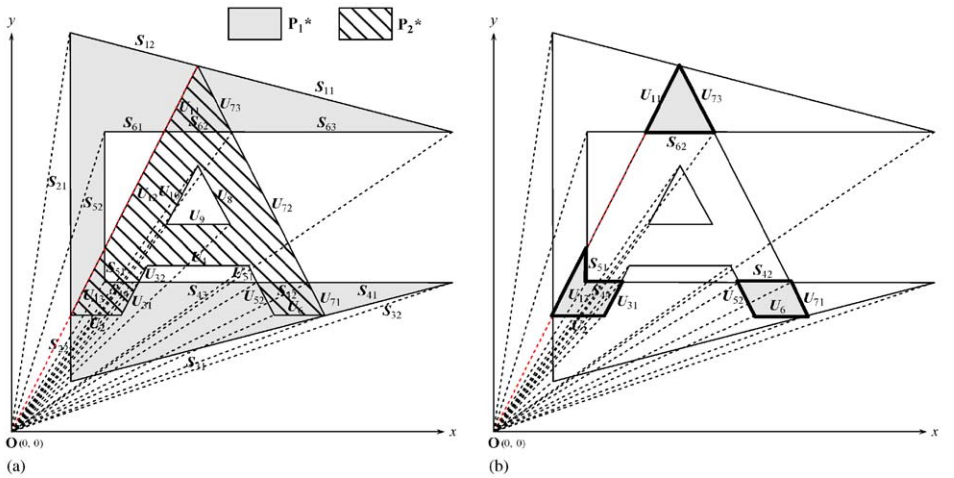

Fig. 11. Example 3: (a) two polygons with their associated simplicial chains; (b) the resultant polygon (in gray).

orientation to the simplex $\boldsymbol{U}_{41}$. Table 3 shows the procedure for calculating the values of the characteristic functions.

Therefore, we have the resultant simplicial chain

$$\lambda_{\mathbf{P}_1\cap\mathbf{P}_2} = -\boldsymbol{S}_{22} + \boldsymbol{S}_{32} - \boldsymbol{U}_1 + \boldsymbol{U}_3$$

and the resultant intersection polygon is shown in Fig. 10(b).

Example 3 is shown in Fig. 11. The two general polygons $\mathbf{P}_1$ and $\mathbf{P}_2$ are concave. $\mathbf{P}_2$ has a hole, and some vertices of $\mathbf{P}_1$ lie on some edges of $\mathbf{P}_2$. The associated simplicial chains of $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$ after the subdivision process are

$$\begin{aligned}\lambda_{\mathbf{P}_1^*} = {} & \boldsymbol{S}_{11} + \boldsymbol{S}_{12} - \boldsymbol{S}_{21} - \boldsymbol{S}_{22} - \boldsymbol{S}_{31} - \boldsymbol{S}_{32} + \boldsymbol{S}_{41} + \boldsymbol{S}_{42} \\ & + \boldsymbol{S}_{43} + \boldsymbol{S}_{44} + \boldsymbol{S}_{51} + \boldsymbol{S}_{52} - \boldsymbol{S}_{61} - \boldsymbol{S}_{62} - \boldsymbol{S}_{63}\end{aligned}$$

and

$$\begin{aligned}\lambda_{\mathbf{P}_2^*} = {} & 0\boldsymbol{U}_{11} + 0\boldsymbol{U}_{12} + 0\boldsymbol{U}_{13} - \boldsymbol{U}_2 + \boldsymbol{U}_{31} + \boldsymbol{U}_{32} - \boldsymbol{U}_4 \\ & - \boldsymbol{U}_{51} - \boldsymbol{U}_{52} - \boldsymbol{U}_6 + \boldsymbol{U}_{71} + \boldsymbol{U}_{72} + \boldsymbol{U}_{73} - \boldsymbol{U}_8 \\ & + \boldsymbol{U}_9 + \boldsymbol{U}_{10}.\end{aligned}$$

In the above formula, some coefficients are equal to 0. Those entries should not be removed. Zero coefficient values indicate that the simplices have degenerated into line segments. Table 4 shows the procedure for calculating the values of the characteristic functions.

Therefore, we have the resultant simplicial chain

$$\begin{aligned}\lambda_{\mathbf{P}_1\cap\mathbf{P}_2} = {} & \boldsymbol{S}_{42} + \boldsymbol{S}_{44} + \boldsymbol{S}_{51} - \boldsymbol{S}_{62} + 0\boldsymbol{U}_{11} + 0\boldsymbol{U}_{13} - \boldsymbol{U}_2 \\ & + \boldsymbol{U}_{31} - \boldsymbol{U}_{52} - \boldsymbol{U}_6 + \boldsymbol{U}_{71} + \boldsymbol{U}_{73}\end{aligned}$$

and the resultant intersection polygon is shown in Fig. 11(b).

Performance data of the de Berg et al. algorithm, the Rivero and Feito algorithm and the new algorithm are shown in Table 5. The data is shown in a chart in Fig. 12 which makes the comparison easier to follow. All algorithms were implemented on a personal computer with a 1.7 GHz Intel Pentium IV CPU and 256 MB RAM, and the source code of all three algorithms was compiled with the Microsoft Visual C++ 6.0 compiler using the same byte alignment (8 bytes) and optimization options. In Table 5, the numbers of edges in both general polygons are listed in the first column, i.e. both polygons have the same number of edges. The numbers below $t_{\text{de Berg}}$, t_{Rivero} and t_{New} are the execution times (in millisecond), respectively, used to calculate the Boolean operation results for the de Berg et al. algorithm, the Rivero and Feito algorithm and the new algorithm. We used a very large number of examples to test the three algorithms. The execution time is obtained by averaging. The improvement factors of the new algorithm over the other two algorithms are also listed in Table 5. Table 5 and Fig. 12 show that our algorithm is more efficient than the de Berg et al. algorithm, and the execution time required by the new algorithm is less than one-third of

Table 4
Data calculated for Example 3 using the new algorithm

	$\boldsymbol{S}_{11}(1)$	$\boldsymbol{S}_{12}(1)$	$\boldsymbol{S}_{21}(-1)$	$\boldsymbol{S}_{22}(-1)$	$\boldsymbol{S}_{31}(-1)$	$\boldsymbol{S}_{32}(-1)$	$\boldsymbol{S}_{41}(1)$	$\boldsymbol{S}_{42}(1)$	$\boldsymbol{S}_{43}(1)$	$\boldsymbol{S}_{44}(1)$		$\boldsymbol{S}_{51}(1)$	$\boldsymbol{S}_{52}(1)$	$\boldsymbol{S}_{61}(-1)$	$\boldsymbol{S}_{62}(-1)$	$\boldsymbol{S}_{63}(-1)$	$f(\boldsymbol{U}_j, \mathbf{P}_1^*)$
$\boldsymbol{U}_{11}(0)$	(0.5, 0)	(0.5, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	$0.5*2 = 1$
$\boldsymbol{U}_{12}(0)$	(0.5, 0)	(0.5, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(−0.5, 0)	(−0.5, 0)	(0, 0)	$0.5*2 - 0.5*2 = 0$
$\boldsymbol{U}_{13}(0)$	(0.5, 0)	(0.5, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0.5, 0)	(0.5, 0)	(−0.5, 0)	(−0.5, 0)	(0, 0)	$0.5*4 - 0.5*2 = 1$
$\boldsymbol{U}_2(-1)$	(1, 0)	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)		(0, 0)	(0, 0)	(0, 0)	(−1, 0)	(0, 0)	$1 + 1 - 1 = 1$
$\boldsymbol{U}_{31}(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 + 1 - 1 = 1$
$\boldsymbol{U}_{32}(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 - 1 = 0$
$\boldsymbol{U}_4(-1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, −1)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 - 1 = 0$
$\boldsymbol{U}_{51}(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	0
$\boldsymbol{U}_{52}(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$\boldsymbol{U}_6(-1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$\boldsymbol{U}_{71}(1)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(1, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	1
$\boldsymbol{U}_{72}(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 1)	(0, 0.5)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 - 1 = 0$
$\boldsymbol{U}_{73}(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0.5)		(0, 1)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	1
$\boldsymbol{U}_8(-1)$	(1, 0)	(0, 0)	(0, 0)	(0, −1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, −1)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 - 1 = 0$
$\boldsymbol{U}_9(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(−1, 0)	$1 - 1 = 0$
$\boldsymbol{U}_{10}(1)$	(1, 0)	(0, 0)	(0, 0)	(0, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 1)		(0, 0)	(0, 0)	(0, 0)	(−1, 0)	(0, 0)	$1 - 1 = 0$
$f(\boldsymbol{S}_i, \mathbf{P}_2^*)$	0	0	0	$-1+1-1+1=0$	$-1+1=0$	0	0	1	$-1+1=0$	$0.5*2-1+1=1$	1	0	0	1	0		

Table 5
Performance results using: the de Berg et al. algorithm, the Rivero and Feito algorithm and the new algorithm

Numbers of edges	$t_{\text{de Berg}}$ (ms)	t_{Rivero} (ms)	t_{New} (ms)	$t_{\text{de Berg}}/t_{\text{New}}$	$t_{\text{Rivero}}/t_{\text{New}}$
3	0.17410	0.08637	0.02679	6.50	3.22
5	0.29786	0.13235	0.03929	7.58	3.37
10	0.48708	0.27364	0.07814	6.23	3.50
20	0.97175	0.90293	0.24748	3.92	3.65
30	1.59237	1.72743	0.46227	3.44	3.74
40	2.08146	2.97129	0.77523	2.68	3.83
45	2.59044	3.95988	1.01165	2.56	3.91
50	3.14897	5.07541	1.23712	2.54	4.10

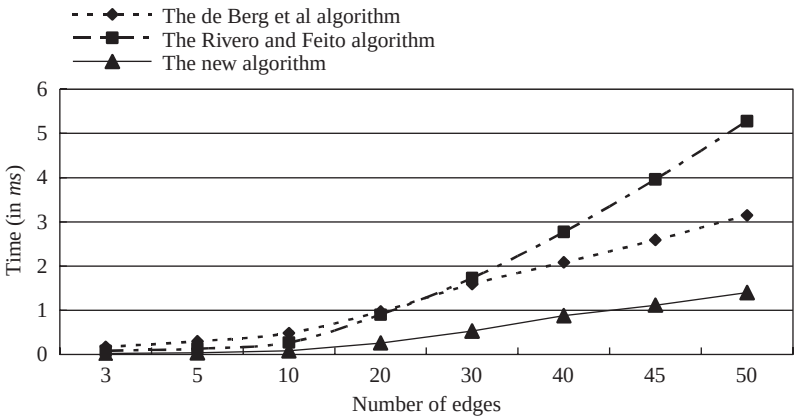

Fig. 12. Performance chart using: the de Berg et al. algorithm, the Rivero and Feito algorithm and the new algorithm.

that required by the Rivero and Feito algorithm. The de Berg algorithm needs to correct the situation of half-edges around each intersection point after it is calculated. The Rivero and Feito algorithm has to calculate the levels of all simplices of the two polygons.

5. Conclusions

A new algorithm for Boolean operations on general planar polygons is presented. The new algorithm uses simplex theory to solve the Boolean operation problem. The associated simplicial chains of general polygons have played a very important role while calculating the result of Boolean operation. At the beginning, the associated simplicial chains of the two general polygons are created; and in the end, the Boolean operation resultant polygon is obtained via its associated simplicial chain. The new algorithm is very simple. After the subdivision process, the algorithm only requires some inclusion tests to determine whether a point is on an edge of a simplex or inside a simplex, and a few addition operations of some real numbers.

Acknowledgements

The authors wish to thank the anonymous reviewers for many helpful comments and suggestions. Thanks are also due to Mr. Zi-Heng Peng for carefully proofreading this paper. The research was supported by Chinese 863 Program (2003AA4Z3110) and Chinese 973 Program (2002CB312106). The second author was supported by a project sponsored by SRF for ROCS, SEM (041501004), and a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (200342).

References

[1] de Berg M, van Krefeld M, Overmars M, Schwarzkopf O. Computational geometry: algorithms and applications. Berlin: Springer; 1997.
[2] Guibas L, Ranshaw L, Stolfi L. A kinetic framework for computational geometry. 24th Annual Symposium on Foundations of Computer Science, IEEE, 1983. p. 100–11.
[3] O'Rourke J. Computational geometry in C. Cambridge: Cambridge University Press; 1985.

[4] Preparata FP, Shamos MI. Computational geometry: an introduction. Berlin: Springer; 1985.
[5] Foley JD, Dam A, Feiner SK, Hughes JF. Computer graphics: principles and practice. Massachusetts: Addison-Wesley; 1990.
[6] Montani C, Re M. Vector and raster hidden surface removal using parallel connected stripes. IEEE Computer Graphics and Applications 1987;7(1):14–23.
[7] Sechrest S, Greenberg D. A visible polygon reconstruction algorithm. Computer Graphics 1981;15(1):17–26.
[8] Sutherland IE, Hodgeman GW. Reentrant polygon clipping. Communications of the ACM 1974;17(1):32–42.
[9] Weiler K, Atherton P. Hidden surface removal using polygon area sorting, Proceedings of the SIGGRAPH 1977; 214–22.
[10] Weiler K. Polygon comparison using a graph representation. Computer Graphics 1980;14(3):10–8.
[11] Fortune S. Polyhedral modelling with multiprecision integer arithmetic. Computer-Aided Design 1997;29(2): 123–33.
[12] Narayanaswami C, Franklin WR. Determination of mass properties of polygonal CSG objects in parallel. Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991, p. 279–88.
[13] Gardan Y, Perrin M. An algorithm reducing 3D Boolean operations to a 2D problem: concepts and results. Computer-Aided Design 1996;28(4):277–87.
[14] Hoffmann CM, Hopcroft JE, Karasick MJ. Robust set operations on polyhedral solids. IEEE Computer Graphics and Applications 1989;9(6):50–9.
[15] Persion H. NC machining of arbitrarily shaped pockets. Computer-Aided Design 1978;10(3):169–74.
[16] Andereev RD. Algorithm for clipping arbitrary polygons. Computer Graphics Forum 1989;8(2):183–91.
[17] Greiner G, Hormann K. Efficient clipping of arbitrary polygons. ACM Transactions on Graphics 1998;17(2): 71–83.
[18] Liang YD, Barsky BA. An analysis and algorithm for polygon clipping. Communications of the ACM 1983;26(11):868–77.
[19] Maillot PG. A new, fast method for 2D polygon clipping: analysis and software implementation. ACM Transactions on Graphics 1992;11(3):276–90.
[20] K. Schutte, An edge labeling approach to concave polygon clipping. Manuscript 1995;1–10.
[21] Vatti BR. A generic solution to polygon clipping. Communications of the ACM 1992;35(7):56–63.
[22] Wu YX. Polygon Boolean operation based on edges recognition algorithm. Journal of Computer-Aided Design and Computer Graphics 1994;6(4):260–5 [in Chinese].
[23] Zhou PD. Computational geometry: algorithms analysis and design. Beijing: Tsinghua University Press; 2000 [in Chinese].
[24] Feito FR, Rivero ML. Geometric modelling based on simplicial chains. Computers and Graphics 1998;22(5): 611–9.
[25] Rivero ML, Feito FR. Boolean operations on general planar polygons. Computers and Graphics 2000;24(6): 881–96.
[26] Feito FR, Torres JC, Ureña A, Orientation. simplicity and inclusion test for planar polygons. Computers and Graphics 1995;19(14):595–600.
[27] Ruiz J, Feito FR. Inclusion test for curved-edge polygons. Computers and Graphics 1997;21(6):815–24.