Efficient algorithm for general polygon clipping

Yu Peng, Junhai Yong, Hui Zhang, Jiaguang Sun

EFFICIENT ALGORITHM FOR GENERAL POLYGON CLIPPING

Yu Peng[1,2] Junhai Yong[1] Hui Zhang[1] Jiaguang Sun[1,2]

[1]School of Software, Tsinghua University, Email: pengyu00@mails.tsinghua.edu.cn
[2]Department of Computer Science and Technology, Tsinghua University

ABSTRACT: We present an efficient algorithm to determine the intersection of two planar general polygons. A new method based on rotation angle is proposed to obtain the classification of an edge with respect to a polygon. The edge candidates can be determined efficiently by a 1-dimensional range searching approach based on an AVL tree (a balanced binary search tree). The simplicial chain is used to represent the general polygons, and to determine the classification of polygon edges. Examples are given to illustrate the algorithm.

KEYWORDS: Computational geometry, Geometric modeling, Polygon intersection, Polygon Clipping, AVL tree.

1. INTRODUCTION

Polygon clipping with applications in various areas of computer graphics [7, 15] and CAD (Computer Aided Design) [4, 9] is one of the fundamental problems of the computational geometry [1, 10, 12]. In literature lots of approaches for polygon clipping are presented [5, 6, 8, 11, 13-16]. Peng et al [8] have made a good summarization of the latest research directions on this problem, and proposed a simpler, more efficient and more robust algorithm to determine the intersection of two general polygons with respect to the algorithm by Rivero and Feito [11]. The algorithm uses a process of double-nested loop to determine all edges of a polygon that are contained within the simplices of other polygon, which would spend time on some useless calculation.

A more efficient algorithm is presented in this paper. As with the algorithm by Peng et al [8], we adopt the simplex theory to handle general polygons. In Peng et al algorithm, each inclusion test between an edge midpoint of one polygon and a simplex of the other is performed. Practically, however, an edge midpoint of one polygon is contained within no or only a few simplices of the other. In the new algorithm, the process of double-nested loop is reduced to a 1D range searching using an AVL tree. The new algorithm is more efficient, as it requires half as much running time as the algorithm presented by Peng et al does.

2. PRELIMINARIES

We begin by briefly reviewing some definitions and properties about simplicial chain by Feito et al [3] and Peng el al [8]. (Refer to [2, 3, 8] for more details).

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 an 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 and the inner contours in clockwise. Thus, the interior of the polygon is on the left side of each directed edges.

A simplex $\boldsymbol{S}$ is an ordered triangle, and the coefficient of $\boldsymbol{S}$ equals *sign*(Area_sign($\boldsymbol{S}$)), where *sign* is the sign function and Area_sign is the signed area of the ordered triangle. In the remaining part of this paper, the simplex is referred to an original simplex, where a vertex of the simplex lies on the origin. The directed edge, an endpoint of which lies on the origin, is called an original edge (otherwise called a non-original edge). A simplicial chain is a collection of the sequence of the simplices $\{\boldsymbol{S}_i\}$ and the sequence of the coefficients $\{\alpha_i\}$, and it is denoted by $\lambda=\sum\alpha_i\cdot\boldsymbol{S}_i$, where i=1, 2, ..., n.

Definition 1. Given a simplicial chain $\lambda=\sum\alpha_i\cdot\boldsymbol{S}_i$, for any point $\mathbf{Q}\in \mathrm{R}^2$, a characteristic function f_λ is defined by

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

where

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

The closed set of the point set given by $\mathbf{P}_\lambda=\{\mathbf{Q} \mid f_\lambda(\mathbf{Q})=1,\ \mathbf{Q}\in\mathbf{R}^2\}$ is a polygon, and λ is called the simplicial chain associated with the polygon. Given a general polygon, we could find its associated simplicial chain.

Lemma 1. Let **P** be a polygon determined by n edges

e_1, e_2,..., e_n. Let $\boldsymbol{S}_i$ be the original simplex determined by the origin and e_i, and let α_i be the coefficient of $\boldsymbol{S}_i$, where i=1, 2, ..., n. Then $\lambda=\sum \alpha_i \cdot \boldsymbol{S}_i$ is the simplicial chain associated with polygon **P**.

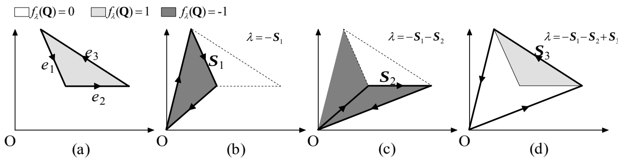

Figure 1. associated simplicial chain of given triangle

Figure 1 gives an example of an associated simplicial chain of a triangle. This paper assumes that the general polygons are in the first quadrant. A translation is enough to move any general polygons into the first quadrant if it is necessary to do so. Given two general polygons, the subdivision process is as follows. First, calculate intersection points and touching points (some edges' endpoints lying on other edges) of two general polygons. Then subdivide the directed edges of two polygons into smaller directed edges at the intersection points and the touching points. Finally, establish the new general polygons from the subdivided directed edges.

Theorem 1. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two general 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, $\vec{e}$ is inside $\mathbf{P}_2$ if and only if the midpoint of $\vec{e}$ is in $\mathbf{P}_2$.

Definition 2. Let $\mathbf{P}_1$ and $\mathbf{P}_2$ be two given general polygons after the subdivision process. Let λ be the simplicial chain of $\mathbf{P}_2$. For any directed edge $\vec{e}$ of $\mathbf{P}_1$, the characteristic function f of $\vec{e}$ on $\mathbf{P}_2$ is given by

$$f(\vec{e})=\begin{cases} 1 & , \text{ if } \vec{e} \text{ is a directed edge of } \mathbf{P}_2, \\ 0 & , \text{ if } -\vec{e} \text{ is a directed edge of } \mathbf{P}_2, \\ f_\lambda(\mathbf{Q}), & \text{otherwise.} \end{cases} \quad (3)$$

where $\mathbf{Q}$ is the midpoint of $\vec{e}$, and $f_\lambda(\mathbf{Q})$ is defined by Eqn. (1).

Corollary 1. Let $\vec{e}$ be a directed edge of $\mathbf{P}_1$. Then, $\vec{e}$ is contained within $\mathbf{P}_2$ if and only if $f(\vec{e})$=1, where $f(\vec{e})$ is defined by Eqn. (3).

The following theorem uses the simplex theory to obtain the clipped polygon of two given polygons.

Theorem 2. Given two general polygons $\mathbf{P}_1$ and $\mathbf{P}_2$ after the subdivision process. Then, the associated simplicial chain of the clipped polygon $\mathbf{P}_3$=$\mathbf{P}_1\cap\mathbf{P}_2$ is

$$\lambda_{\mathbf{P}_3}=\sum a_i \cdot \boldsymbol{S}_i+\sum a_j' \cdot \boldsymbol{S}_j'+\sum b_k \cdot \boldsymbol{U}_k \quad (4)$$

where i =1, 2, ..., n, j=1, 2, ..., n', k=1, 2, ..., l, $\{\boldsymbol{S}_i\}$ and $\{\boldsymbol{S}_j'\}$ are two subsets consisting of the simplices of $\mathbf{P}_1$, respectively. The non-original edge of each $\boldsymbol{S}_i$ is inside $\mathbf{P}_2$, and the non-original edge of each $\boldsymbol{S}_j'$ is equivalent to some directed edge of $\mathbf{P}_2$. $\{\boldsymbol{U}_k\}$ is a subset consisting of the simplices of $\mathbf{P}_2$. The non-original edge of each $\boldsymbol{U}_k$ is inside $\mathbf{P}_1$. $\{a_i\}$, $\{a_j'\}$ and $\{b_k\}$ are the coefficient sets of simplices of $\{\boldsymbol{S}_i\}$, $\{\boldsymbol{S}_j'\}$ and $\{\boldsymbol{U}_k\}$, respectively.

3. ROTATION ANGLE

In order to obtain an efficient algorithm, we need some new definitions and terminology.

Definition 3. Given a point, the rotation angle of the point is given by the counterclockwise angle that the positive x-axis makes with the segment from the origin to the point.

As shown in Figure 2, α, β and γ are the rotation angles of the points p_1 , p_2 and p_3, respectively. From the feature of the rotation angle of a point, we could have the following theorem.

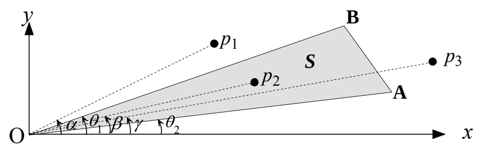

Figure 2. Rotation angle of points

Theorem 3. Given an original simplex and a point in the first quadrant. The point is inside the simplex, if and only if the following two conditions are both satisfied:

- The rotation angle of the point is in the range $[x{:}x']$.
- Both the origin and the point are on the same side of the non-original edge of the original simplex.

Where x and x' are the rotation angles of two endpoints of the non-original edge of the original simplex, respectively.

As shown in Figure 2, p_2 is inside the simplex $\boldsymbol{S}$. The rotation angle β of p_2 is in the range $[\theta_1{:}\theta_2]$, and both the origin and p_2 are on the same side of the edge $\overline{\mathbf{AB}}$, where θ_1 and θ_2 are the rotation angles of **B** and **A**, respectively. The situation of neither p_1 nor p_3 satisfies the above two conditions simultaneously, where the rotation angle of p_1 is greater than θ_1, and the origin and p_3 are not on the same side of $\overline{\mathbf{AB}}$). Hence, neither p_1 nor p_3 is inside $\boldsymbol{S}$.

Definition 4. Given a general polygon and an original simplex in the first quadrant, an edge of the polygon is called an edge candidate with respect to the simplex if and only if the rotation angle of the edge midpoint is in the range $[x{:}x']$, where x and x' are the rotation angles of two endpoints of the non-original edge of the original simplex, respectively.

4. CLIPPING ALGORITHM

The new algorithm is based on the simplex theory mentioned in Section 2. The pseudo code for the algorithm is illustrated in Figure 3.

Algorithm CLIPPING ($\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$)
Input. two general polygon $\mathbf{P}_1$ and $\mathbf{P}_2$.
Output. the clipped polygon $\mathbf{P}_3 = \mathbf{P}_1 \cap \mathbf{P}_2$.
/* $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$ are the polygons after subdivision. */
SUBDIVISIONPROCESS ($\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_1^*$,$\mathbf{P}_2^*$);
/* λ_1^* is the associated simplicial chain of $\mathbf{P}_1^*$. */
BUILDSIMPLICIALCHAIN ($\mathbf{P}_1^*$,λ_1^*);
/* λ_2^* is the associated simplicial chain of $\mathbf{P}_2^*$. */
BUILDSIMPLICIALCHAIN ($\mathbf{P}_2^*$,λ_2^*);
/* s_i^* is the non-original edges of the simplices $\boldsymbol{S}_i^*$ from λ_1^*; $f(s_i^*)$ is the characteristic function of s_i^* on $\mathbf{P}_2$ according to Definition 2; SUBJECT means that $\mathbf{P}_1^*$ is the subject polygon.*/
CALEDGECHARACTER ($\mathbf{P}_1^*$,$\mathbf{P}_2$,λ_1^*,λ_2,$f(s_i^*)$,SUBJECT);
/* u_j^* is the non-original edges of the simplices $\boldsymbol{U}_j^*$ from λ_2^*; $f(u_j^*)$ is the characteristic function of u_j^* on $\mathbf{P}_1$; CLIP means that $\mathbf{P}_2^*$ is the clip polygon.*/
CALEDGECHARACTER ($\mathbf{P}_2^*$,$\mathbf{P}_1$,λ_2^*,λ_1,$f(u_j^*)$,CLIP);
/* Calculate the resultant simplicial chain λ_3 through λ_1^* and λ_2^* using Theorem 2. */
CALRESULTCHAIN (λ_1^*,$f(s_i^*)$,λ_3);
CALRESULTCHAIN (λ_2^*,$f(u_j^*)$,λ_3);

Figure 3. Pseudo code for clipping algorithm

At first, the subdivision process is performed on the given general polygons. Then, build the simplicial chains for both polygons according to Lemma 1, and calculate the value of the characteristic function for each edge of one polygon on the other's original polygon (the polygon before the subdivision process) through Corollary 1. At last, the simplicial chain of the clipped polygon $\mathbf{P}_3$ is obtained through a subroutine CALRESULTCHAIN shown in Figure 4, and we have the polygon $\mathbf{P}_3$ from the simplicial chain.

CALRESULTCHAIN (λ,$f(s_i)$,λ_3)
Input. The simplicial chain λ and the characteristic function $f(s_i)$
Output. The resultant simplicial chain λ_3.
for each simplex $\boldsymbol{S}_i$ of λ **do**
    if $f(s_i)$=1 **then** /* s_i is the non-original edge of $\boldsymbol{S}_i$. */
        Add the simplex $\boldsymbol{S}_i$ and its coefficient to λ_3;

Figure 4. Pseudo code for calculating the resultant simplicial chain

The subdivision process is already described in Section 2. How to build a simplicial chain of a general polygon is introduced in Lemma 1. According to Definition 2 and Corollary 1, we could obtain the value of an edge characteristic function of a polygon by calculating the classification between the edge midpoint and each simplex of the other. Practically, however, an edge midpoint of one polygon is contained within no or only a few simplices of the other.

In order to accelerate the calculation of value of the edge characteristic function, we extend the 1-dimensional range searching approach [1] with the principles described in Section 3. The edge candidates can be determined by this approach efficiently. The pseudo code for calculating values of edge characteristic functions is illustrated in Figure 5.

CALEDGECHARACTER ($\mathbf{P}_1$,$\mathbf{P}_2$,λ_1,λ_2,$f(s_i)$,t)
Input. Two polygons $\mathbf{P}_1$ and $\mathbf{P}_2$ and their respective simplicial chains λ_1 and λ_2, t denotes the type of $\mathbf{P}_1$.
Output. The characteristic function $f(s_i)$ of s_i on $\mathbf{P}_2$.
/* Initiate the function $f(s_i)$; s_i is the non-original edge of $\boldsymbol{S}_i$.*/
1. **for** each simplex $\boldsymbol{S}_i$ of λ_1 **do**
2. $f(s_i) \leftarrow 0$;
/* Build the 1D range searching tree; **T** is the AVL tree */
3. BUILD1DRANGETREE (λ_1,$f(s_i)$,$\mathbf{P}_2$,$\mathbf{T}$,t);
4. **for** each simplex $\boldsymbol{U}_j$ of λ_2 **do**
/* u_j is the non-original edge of $\boldsymbol{U}_j$. */
5. $x \leftarrow$ the minor rotation angle of endpoints of u_j;
6. $x' \leftarrow$ the major rotation angle of endpoints of u_j;
/* v_s is the split node; $[x{:}x']$ is the 1D query range with $x \le x'$. */
7. $v_s \leftarrow$ FINDSPLITNODE ($\mathbf{T}$,$[x{:}x']$);
8.   **if** $v_s \ne$ **NULL then**
/* $ra(v_s)$ denotes the rotation angle stored at v_s. */
9.   **if** $ra(v_s)$ is in $[x{:}x']$ **then**
/* Calculate $f(s(v_s))$; $s(v_s)$ denotes the directed edge stored at v_s. */
10.   CALVAL ($f(s(v_s))$,$\boldsymbol{U}_j$);
11.   **if** v_s is not a leaf **then**
/* Follow the path to x and calculate the characteristic functions of the edge candidates; $lc(v_s)$ denotes the left child of v_s. */
12. $v \leftarrow lc(v_s)$;
13.   **while** $v \ne$ **NULL do**
14.   **if** $ra(v) \ge$ x **then**
15.   CALVAL ($f(s(v))$,$\boldsymbol{U}_j$);
/* Traverse the subtree rooted at $rc(v)$; $rc(v)$ denotes the right child of v. */
16.   CALSUBTREEVAL ($rc(v)$,$f(s(rc(v)))$,$\boldsymbol{U}_j$);
17. $v \leftarrow lc(v)$;
18.   **else** $v \leftarrow rc(v)$;
19.   Follow the path to x', calculate the characteristic functions of the edge candidates, which is similar to lines 12-18.

Figure 5. Pseudo code for calculating the values of edge the characteristic functions

Now we describe CALEDGECHARACTER in more details. First initiate the edge characteristic functions in line 1 and line 2. Then build the AVL tree **T** in line 3. Each node of **T** stores a directed edge of one polygon and the rotation angle of the edge midpoint. We assume that the left subtree of a node v contains all the rotation angles smaller than or equal to $ra(v)$ and that the right subtree contains all the rotation angles strictly greater than $ra(v)$. In lines 4-19, we traverse each simplex of the other polygon. For each simplex, we adopt the 1-dimensional range searching approach to determine the nodes, where the rotation angles stored at the nodes are in the range $[x{:}x']$. To find these nodes we first search for the node v_s in line 7, where the paths to x and x' split shown in Figure 6.

FINDSPLITNODE ($\mathbf{T}$,$[x{:}x']$)
Input. An AVL tree $\mathbf{T}$ and the query range $[x{:}x']$ with $x \le x'$.
Output. The node v where the paths to x and x' split.
$v \leftarrow$ the root of $\mathbf{T}$;
/* $ra(v)$ denotes the rotation angle stored at v; $lc(v)$ and $rc(v)$ denote the left

child and right child of v, respectively. */

2. **while** $v \neq$ **NULL and** $(ra(v) < x$ **or** $ra(v) \geq x')$ **do**

3. **if** $(ra(v) \geq x')$ **then**

4. $v \leftarrow lc(v)$;

5. **else** $v \leftarrow rc(v)$;

6. **return** v;

Figure 6. Pseudo code for finding the split node

Starting from v_s we then follow search path of x. At each node where the path goes left, we first calculate the characteristic function of the edge stored at it through a subroutine CALVAL, and then do the same calculation on each node in the right subtree, since this subtree is between the two search paths, i.e., the edges stored at the nodes in the subtree are the edge candidates. CALSUBTREEVAL is recursive and it is used to calculate the characteristic functions of all nodes in the subtree (The subroutine is illustrated in Figure 7). Similarly, follow the path of x' and calculate the characteristic functions of the edge candidates.

CALSUBTREEVAL $(v, f(s), \boldsymbol{U})$

Input. A node v in the AVL tree and a simplex $\boldsymbol{U}$.

Output. The edge characteristic function $f(s)$.

/* $s(v)$ denotes the directed edge stored at v; $lc(v)$ denotes the left child of v; $rc(v)$ denotes the right child of v.*/

1. **if** $v \neq$ **NULL then**

2. CALVAL $(f(s(v)), \boldsymbol{U})$;

3. CALSUBTREEVAL $(lc(v), f(s(lc(v))), \boldsymbol{U})$;

4. CALSUBTREEVAL $(rc(v), f(s(rc(v))), \boldsymbol{U})$;

Figure 7. Pseudo code for calculating the values of the edges stored at the subtree

The pseudo code for building AVL tree is shown in Figure 8. According to Definition 2, when a directed edge is shared by two polygons both values of the characteristic functions of this edge on respective polygons are 1. And according to Eqn. (4), only one corresponding simplex of this edge is added to the resultant simplicial chain. Hence, we force $f(s_i)$=1 in line 6 of BUILD1DRANGETREE, where s_i is from the subject polygon and it is shared by both polygons.

BUILD1DRANGETREE $(\lambda, f(s_i), \mathbf{P}_2, \mathbf{T}, t)$

Input. The simplicial chains λ and the type t of the polygon associated with λ.

Output. An AVL tree $\mathbf{T}$ and the characteristic function $f(s_i)$ on polygon $\mathbf{P}_2$.

1. **for** each simplex $\boldsymbol{S}_i$ of λ do

/* s_i is the non-original edge of $\boldsymbol{S}_i$; */

2. **if** s_i does not overlap any edges of $\mathbf{P}_2$ **then**

3. $p \leftarrow$ the rotation angle of the midpoint of s_i;

4. Insert node (s_i, p) into $\mathbf{T}$;

/* It is performed according to Definition 2; */

5. **else if** t = SUBJECT **and** s_i is a directed edge of $\mathbf{P}_2$ **then**

6. $f(s_i) \leftarrow 1$;

Figure 8. Pseudo code for building 1D range tree

The subroutine CALVAL is illustrated in Figure 9. It uses Theorem 3 to determine the classification of the edge midpoint in line 1, and the characteristic function of the edge s is calculated in line 3 and line 4 according to Definition 1 and Definition 2.

CALVAL $(f(s), \boldsymbol{U})$

Input. A directed edge s and a simplex $\boldsymbol{U}$.

Output. The edge characteristic function $f(s)$.

/* Obtain the classification by Theorem 3; u is the non-original edge of $\boldsymbol{U}$. */

1. **if** Both the midpoint of s and the origin are on the same side of u **then**

/* $a(s)$ denotes the rotation angle of the midpoint of s; mina(u) denotes the minor rotation angle of vertices of u; $maxa(u)$ denotes the major rotation angle of vertices of u; $c(\boldsymbol{U})$ denotes the coefficient of $\boldsymbol{U}$. */

2. **if** $a(s) = mina(u)$ **or** $a(s) = maxa(u)$ **then**

3. $f(s) \leftarrow f(s) + c(\boldsymbol{U})/2$;

4. **else** $f(s) \leftarrow f(s) + c(\boldsymbol{U})$;

Figure 9. Pseudo code for calculating the value of an edge to a simplex

5. EVALUTION

5.1 Time Complexity

Let n and m be the numbers of edges of the two polygons $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively, and k be the number of the intersection points and the touching points of $\mathbf{P}_1$ and $\mathbf{P}_2$. For the subdivision process, we adopt the plane sweep algorithm introduced in References [1, 10, 12] to calculate the intersection points and touching points. Therefore, the running time required by the subdivision process is $O((n+m)\log(n+m)+k)$. The subroutine BUILDSIMPLICIALCHAIN takes an amount of time that is linear in the number of edges of the polygon. CALEDGECHARACTER with respect to the subject polygon uses a loop, which includes the 1D range searching process. The AVL tree can be built in $O((n+k)\log(n+k))$ time. FINDSPLITNODE takes $O(\log(n+k))$ time. The time spent in a call to CALSUBTREEVAL is linear in t_i, where t_i is the number of the edge canditates of subject polygon with respect to the ith simplex of clip polygon. Hence, the total time spent in such call s is $O(t_i)$. The remaining nodes storing the edge candidates are the nodes on the search path of x or x'. Because $\mathbf{T}$ is balanced, these paths have length $O(\log(n+k))$, so the total time spent in these nodes is $O(\log(n+k))$. Hence, the time of 1D range searching process is $O((\log(n+k))+t_i)$. Since the 1D range searching process runs m times in the loop, the subroutine CALEDGECHARACTER with respect to subject polygon gives a running time of $O(m\log(n+k)+T_1))$, where $T_1=\sum t_i$. Similarly, CALEDGECHARACTER with respect to clip polygon gives a running time of $O(n\log(m+k)+T_2)$, where $T_2=\sum s_j$ *and* s_j is the number of the edge canditates of clip polygon with respect to the jth simplex of subject polygon. The subroutine CALRESULTCHAIN gives a time of $O(n+m+k)$.

5.2 Example

Next we show an example in which two polygons have some edges in common. Figure 10(a) shows the subject polygons $\mathbf{P}_1$ and the clip polygon $\mathbf{P}_2$.

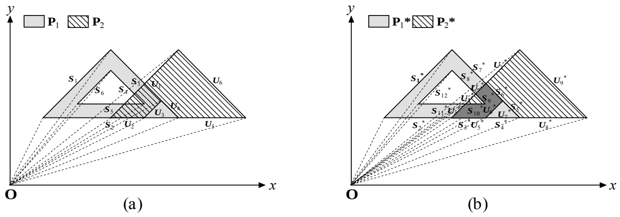

Figure 10. Example: (a) two polygons with their associated simplicial chains, (b) the clipped polygon (in deep gray)

The simplical chains of $\mathbf{P}_1$ and $\mathbf{P}_2$ are $\lambda_1=\boldsymbol{S}_1\text{-}\boldsymbol{S}_2+\boldsymbol{S}_3\text{-}\boldsymbol{S}_4+\boldsymbol{S}_5\text{-}\boldsymbol{S}_6$ and $\lambda_2=\text{-}\boldsymbol{U}_1\text{-}\boldsymbol{U}_2+\boldsymbol{U}_3\text{-}\boldsymbol{U}_4\text{-}\boldsymbol{U}_5+\boldsymbol{U}_6$, respectively. The new algorithm first performs the subdivision process on the two general polygons according to not only the intersection points but also the touching points. After the subdivision process, the simplical chains of $\mathbf{P}_1^*$ and $\mathbf{P}_2^*$ are $\lambda_1^* = \boldsymbol{S}_1^*$ - $\boldsymbol{S}_2^*$ - $\boldsymbol{S}_3^*$ - $\boldsymbol{S}_4^*+\boldsymbol{S}_5^*+\boldsymbol{S}_6^*+\boldsymbol{S}_7^*\text{-}\boldsymbol{S}_8^*\text{-}\boldsymbol{S}_9^*+\boldsymbol{S}_{10}^*+\boldsymbol{S}_{11}^*\text{-}\boldsymbol{S}_{12}^*$ and $\lambda_2^* = \text{-}\boldsymbol{U}_1^*\ \text{-}\boldsymbol{U}_2^*\text{-}\boldsymbol{U}_3^*\text{-}\boldsymbol{U}_4^*\text{-}\boldsymbol{U}_5^*+\boldsymbol{U}_6^*\text{-}\boldsymbol{U}_7^*\text{-}\boldsymbol{U}_8^*+\boldsymbol{U}_9^*$, respectively. The simplex $\boldsymbol{S}_3^*$ is equivalent to the simplex $\boldsymbol{U}_5^*$, and the non-orginal edge of $\boldsymbol{S}_5^*$ has the opposite orientation as the non-original edge of $\boldsymbol{U}_7^*$.

Table 1 shows the procedure for calculating the values of the edge characteristic functions of $\mathbf{P}_1^*$ on $\mathbf{P}_2$. The simplices of $\mathbf{P}_1^*$ and $\mathbf{P}_2$ and their coefficients are shown in the first row and the first column of Table 1, respectively. The second row lists the query range of each simplex of $\mathbf{P}_2$. The second column lists the rotation angle of the midpoint of each edge of $\mathbf{P}_1^*$. Note that the value of the charactristic functions of the edges, which overlap some edges of $\mathbf{P}_2^*$, can be determined directly through the edge directions. Hence, fill the corresponding bracket with '+' where the corresponding $\boldsymbol{S}_i^*$ and $\boldsymbol{U}_j$ have the non-original edges in common that have the same orientation, and fill the bracket with '−' where the corresponding $\boldsymbol{S}_i^*$ and $\boldsymbol{U}_j$ have the non-original edges in common that have the opposite orientations. Assume that the midpoint of the non-original edge of $\boldsymbol{S}_i^*$ and the simplex $\boldsymbol{U}_j$ are the point and the simplex in the assumed conditions of Theorem 3, respectively. In the interior of Table 1, the corresponding table cell leaves with blank where the first condition of Theorem 3 is not satisfied. The cell is filled with '×' where only the first condition of Theorem 3 is satisfied. The cell is filled with '✓' where both of the two conditions of Theorem 3 are satisfied. The last column lists the value of the characteristic function of non-original edge of respective $\boldsymbol{S}_i^*$ according to Definition 2.

Table 1. Procedure for calculating the value of edge characteristic functions of P_1^*

$\mathbf{P}_1^*\downarrow$	$\mathbf{P}_2\rightarrow$ / *range*→ / *angle*↓	$\boldsymbol{U}_1$(-1) [33.7:38.7]	$\boldsymbol{U}_2$(-1) [26.6:33.7]	$\boldsymbol{U}_3$(1) [26.6:29.1]	$\boldsymbol{U}_4$(-1) [21.8:29.1]	$\boldsymbol{U}_5$(-1) [15.9:21.8]	$\boldsymbol{U}_6$(1) [15.9:38.7]	$f(s_i^*)$
$\boldsymbol{S}_1^*$(1)	(56.3)							0
$\boldsymbol{S}_2^*$(-1)	(45.0)							0
$\boldsymbol{S}_3^*$(-1)	(+)							1
$\boldsymbol{S}_4^*$(-1)	(24.0)				✓		✓	-1+1=0
$\boldsymbol{S}_5^*$(1)	(−)							0
$\boldsymbol{S}_6^*$(1)	(32.9)		×				✓	1
$\boldsymbol{S}_7^*$(1)	(45.0)							0
$\boldsymbol{S}_8^*$(-1)	(42.1)							0
$\boldsymbol{S}_9^*$(-1)	(33.2)		×				✓	1
$\boldsymbol{S}_{10}^*$(1)	(32.6)		×				✓	1
$\boldsymbol{S}_{11}^*$(1)	(41.3)							0
$\boldsymbol{S}_{12}^*$(-1)	(48.7)							0

Similarly, Table 2 shows the procedure for calculating the values of the edge characteristic functions of $\mathbf{P}_2^*$ on $\mathbf{P}_1$. The value of last column of Table 2 is zero where the corresponding bracket of the second column is filled with '+' or '−'. The simplices, whose corresponding numbers in the last columns of Table 1 and Table 2 are 1, are selected to build the resultant simplicial chain of clipped polygon. Hence, the resultant simplicial chain is $\lambda_3 = \boldsymbol{S}_3^*+ \boldsymbol{S}_6^*\text{-}\ \boldsymbol{S}_9^*+ \boldsymbol{S}_{10}^*\text{-}\ \boldsymbol{U}_2^*\text{-}\boldsymbol{U}_4^*+\boldsymbol{U}_6^*$. The clipped polygon is indicated in deep gray shown in Figure 10(b).

Table 2. Procedure for calculating the value of edge characteristic functions of P_2^*

$\mathbf{P}_2^*\downarrow$	$\mathbf{P}_1\rightarrow$ / *range*→ / *angle*↓	$\boldsymbol{S}_1$(1) [53.1:63.4]	$\boldsymbol{S}_2$(-1) [21.8:63.4]	$\boldsymbol{S}_3$(1) [21.8:53.1]	$\boldsymbol{S}_4$(-1) [30.4:48.2]	$\boldsymbol{S}_5$(1) [30.4:49.6]	$\boldsymbol{S}_6$(-1) [48.2:49.6]	$f(u_j^*)$
$\boldsymbol{U}_1^*$(-1)	(37.9)		×	×	×	×		0
$\boldsymbol{U}_2^*$(-1)	(36.5)		×	✓	×	×		1
$\boldsymbol{U}_3^*$(-1)	(35.6)		×	✓	✓	×		1-1=0
$\boldsymbol{U}_4^*$(-1)	(34.4)		×	✓	✓	✓		1-1+1=1
$\boldsymbol{U}_5^*$(-1)	(+)							0
$\boldsymbol{U}_6^*$(1)	(27.9)		×	✓				1
$\boldsymbol{U}_7^*$(-1)	(−)							0
$\boldsymbol{U}_8^*$(-1)	(18.4)							0
$\boldsymbol{U}_9^*$(1)	(26.6)		×	×				0

5.3 Experimental results

Performance data of the Vatti algorithm, the Rivero and Feito algorithm, the Peng et al algorithm and the new algorithm are shown in Table 3 and Figure 11. All algorithms were implemented in a personal computer with 1.7GHZ Intel Pentium IV CPU and 256MB RAM, and the source code of all four algorithms are complied with the Microsoft Visual C++ 6.0 compiler using the same byte alignment (8 bytes) and optimization options. In Table 3, the numbers of edges in both general polygons are listed in the first column, i.e., both general polygons have the same number of edges. The numbers below t_V, t_R, t_P and t_N are the running times (in milliseconds) respectively used to calculate the intersection results for the Vatti algorithm, the Rivero and Feito algorithm, the Peng et al algorithm and the new algorithm. We used a very large number of examples to test the four algorithms. The running time was obtained by averaging. The improvement factors of the new algorithm over other algorithms are also listed in Table 3 (from the 6th column to the 8th column). In Figure 11 we can see graphically the evolution of the running time of polygon clipping versus the number of polygon edges used for our algorithm and the other algorithms. As we can see in the results, the new algorithm is more efficient. The running time required by the new algorithm is less than one third of that by the Rivero and Feito algorithm and half as much as that by the Peng et al algorithm.

Table 3. Performance results using: the Vatti algorithm, the Rivero and Feito algorithm, the Peng et al algorithm and the new algorithm

n	$t_V(ms)$	$t_R(ms)$	$t_P(ms)$	$t_N(ms)$	t_V/t_N	t_R/t_N	t_P/t_N
5	0.0725	0.1324	0.0393	0.0408	1.777	3.245	0.963
10	0.1129	0.2736	0.0781	0.0721	1.567	3.795	1.083
20	0.2592	0.9029	0.2475	0.1536	1.688	5.878	1.611
30	0.4336	1.7274	0.4623	0.2568	1.688	6.727	1,800
40	0.6059	2.9713	0.7752	0.3517	1.723	8.448	2.204
45	0.6847	3.9599	1.0117	0.4606	1.487	8.597	2.196
50	0.8828	5.0754	1.2371	0.5130	1.721	9.894	2.412

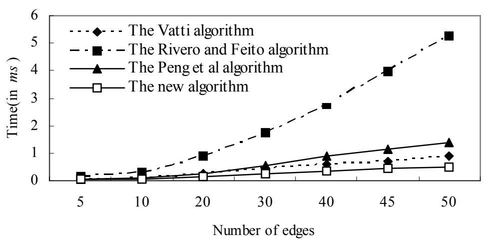

Figure 11. Performance chart using: the Vatti algorithm, the Rivero and Feito algorithm, the Peng et al algorithm and the new algorithm

6. CONCLUSIONS

In this paper, a new polygon clipping algorithm is presented. A new method based on the rotation angle of the edge midpoint is used to determine the classification of the edges of a polygon with respect to another polygon. Also the edge candidate is defined by the rotation angle and a 1-dimensional range searching approach is proposed to obtain the edge candidates for accelerating the edge classification. The algorithm is efficient, as it requires half as much running time as the algorithm by Peng et al does.

ACKNOWLEDGMENTS

The research was supported by the National Science Foundation of China (60403047) and the Chinese 973 Program (2004CB719400). The second author was supported by a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (200342) and a project sponsored by SRF for ROCS, SEM.

REFERENCE

[1] de Berg, M., van Krefeld, M., Overmars, M., Schwarzkopf, O., Computational Geometry: Algorithms and Applications, Springer-Verlag, 1997.

[2] Feito, F.R., Torres, J.C., Ureña, A., Orientation, simplicity and inclusion test for planar polygons, Computers & Graphics 1995; 19(14): 595-600.

[3] Feito, F.R., Rivero, M.L., Geometric Modelling based on simplicial chains, Computers & Graphics 1998; 22(5): 611-619.

[4] Hoffmann, C.M., Hopcroft, J.E., Karasick, M.J., Robust set operations on polyhedral solids, IEEE Computer Graphics and Applications, Vol. 9, No. 6, 1989, pp 50-59.

[5] Greiner, G., Hormann, K., Efficient clipping of arbitrary polygons, ACM Transactions on Graphics 1998; 17(2): 71-83.

[6] Liang, Y.-D., Barsky, B.A., An analysis and algorithm for polygon clipping, Communications of the ACM 1983; 26(11): 868-877.

[7] Loutrel, P.P., A solution to the hidden-line problem for computer-drawn polyhedra, IEEE Transactions on Computers 1970; 19(3): 205-213.

[8] Peng, Y., Yong, J.-H., Sun, J.-G., A new algorithm for Boolean operation on general polygons, Computers & Graphics 2005; 29(1): to appear.

[9] Persion, H., NC machining of arbitrarily shaped pockets, Computer-Aided Design 1978; 10(3): 169-174.

[10] Preparata, F.P., Shamos, M.I., Computational geometry: an introduction, Berlin: Springer-Verlag, 1985.

[11] Rivero, M.L., Feito, F.R., Boolean operations on general planar polygons, Computers & Graphics 2000; 24(6): 881-896.

[12] O'Rourke, J., Computational geometry in C, Cambridge: Cambridge University Press, 1985.

[13] Schutte, K., An edge labeling approach to concave polygon clipping, Manuscript, 1995; 1-10.

[14] Sutherland, I.E., Hodgeman, G.W., Reentrant polygon clipping, Communications of the ACM 1974; 17(1): 32-42.

[15] Weiler, K., Atherton, P., Hidden surface removal using polygon area sorting, Proceedings of the SIGGRAPH 1977; 214-222.

[16] Vatti, B.R., A generic solution to polygon clipping, Communications of the ACM 1992; 35(7): 56-63.