GENERALIZED CLIPPING OF A POLYGON AGAINST A 2D ARBITRARY WINDOW AND A 3D NON-CONVEX VOLUME

Shuwei Hua
Computer Management & Development Services, Inc.
P.O.Box 1184, Harrisonburg, VA 22801

Alade Tokuta
Department of Computer Science
James Madison University
Harrisonburg, VA 22807
(tokutaao@vax1.acs.jmu.edu)

Abstract

Several clipping algorithms are in wide use. These are separated into broad classes. They include subdivision algorithms of which the best known is the midpoint subdivision algorithm; the Cohen-Sutherland algorithm which uses outcodes generators and line window intersection calculations to determine what portion(s) of line segments may be contained in the window. Outcode determination/redetermination for the stages of the algorithm can dominate the clipping process. There are algorithms that are based on parametric expressions for lines, and which are comparatively more efficient than the simple Cohen-Sutherland algorithm. The Skala 2D line clipping algorithms use a parameter value to determine the intersection of a line segment with a convex or non-convex polygon window. The Rappoport algorithm clips any subject polygon against a convex polygonal window. The work of Weiler and Atherton algorithm allows the clipping of any subject polygon against any clip polygon. However, it is based on some assumptions. This work presents an efficient method for clipping a polygon against an arbitrary two-dimensional polygonal windows and a modification of the algorithm can also be applied to clip a polygon against a non-convex volume on 3D. The basic idea based on geometry and parametric representation of the lines, determines if an edge of a subject polygon should be totally rejected, or is totally visible. On the other hand, if an edge of a polygon has intersections with the boundary of a simple polygon window, it is easy to decide which parts of the edge are visible after sorting these intersections.

Keywords: **Clip, Parametric Representation, Simple Polygon, Non-convex Volume.**

1. Introduction

Clipping is the process of removing that portion of a scene that lies outside a region called the clip window. Clipping a scene against a window is a fundamental operation in computer graphics and the demand oftentimes require clipping many polygons against a window. In some cases the windows are simple rectangular regions.

Several clipping algorithms have been proposed and are in wide use. The Liang-Barsky[LB84] algorithm provides an efficient line clipping algorithm that uses a trivial rejection test. The generalized Cyrus-Beck[CB78] algorithm uses a value of the parameter to determine

the intersection of a line segment with an arbitrary window edge. This work builds on that idea and is similar to the work of Skala[Ska89] with respect to its line clipping; however, the approach for handling special cases is different from Skala's. Thus we develop a method and algorithm for polygon clipping. This algorithm for polygon clipping can clip an arbitrary simple polygon against an arbitrary simple polygonal window. The basic idea based on geometry and parametric representation of the lines of which a polygon is made, determines if a line should be totally rejected, or is totally visible. On the other hand, if a line has intersections with the boundary of a simple polygonal window, it is easy to decide which portions of the line is contained within the window. The organization of the presentation considers first the problem of line clipping, showing the details of its implementation in determining the visible portions of a clipped segment after the parameter values have been considered. Next, polygon clipping is derived from line clipping and the discussion is followed with clipping against a volume.

2. The Concept of the Algorithm

We assume a polygon **S** to be clipped, is made up of a set, **L**, of edges of the polygon, and a clipping polygon window, **C**, (in Figure 1), is composed of a set, **E**, of edges. Thus **E** = $\{e_i\}$ i=1,2,...,n; **L** = $\{l_j\}$ j=1,2,...,m.

The clipping process requires that we keep the portions of the edges within the interior of the polygon window **C** and remove the portions of the edges outside the window. The basic operation is to clip a polygon described by an input vertex list against the clipping polygon window and may result in one or more output vertex lists. Each vertex list correspond to an output polygon.

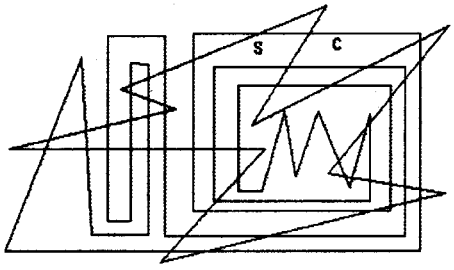

Figure 1

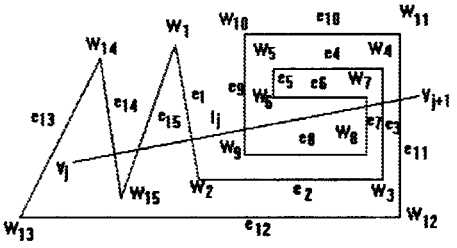

Figure 2

2.1 Parametric Representation

We consider a polygon as given by a sequence of vertices P_1, P_2,..., P_n, where the edges of the polygon are P_1P_2, P_2P_3,.., P_nP_1. The *ith* edge is considered as a vector from P_i to P_{i+1} and is represented in the parametric form as $P(t_i) = P_i + (P_{i+1} - P_i)t_i$, where $0 \le t_i \le 1$.

A polygon **S** to be clipped and a clipping polygonal window **C** can be described as a sequence of vertices $V_1(x_1, y_1)$, $V_2(x_2, y_2)$,..., $V_m(x_m, y_m)$ and $W_1(u_1, v_1)$, $W_2(u_2, v_2)$,..., $W_n(u_n, v_n)$ respectively, where the edges, l_1, l_2,..., l_m, of polygon S are V_1V_2, V_2V_3,..., V_mV_1 and the edges, e_1, e_2,..., e_n, of polygon C are W_1W_2, W_2W_3,..., W_nW_1 respectively.

We express the edges of the polygonal window as:

$$u(h_i) = u_i + (u_{i+i} - u_i)h_i \qquad (1)$$

e_i :

$$v(h_i) = v_i + (v_{i+1} - u_i)h_i \qquad (2)$$

i=1,2,...,n; when i=n, i+1=1. $h_i \in [0,1]$.

The edges l_j of the subject polygon are represented as:

$$x(t_j) = x_j + (x_{j+1} - x_j)t_j \tag{3}$$

l_j :

$$y(t_j) = y_j + (y_{j+1} - y_j)t_j \tag{4}$$

where j = 1, 2, ..., m; when j = m, j+1 = 1. $t_j \in [0,1]$.

From the above equations, a determination can be made if there is an intersection of a segment e_i and a line segment l_j.

Define

$$D = dx_j dv_i - du_i dy_j .$$

where $D \neq 0$ and $d\beta_k = \beta_{k+1} - \beta_k$

From the above equations,

$$h_{i,j} = \frac{dx_i(y_j - v_i) - dy_j(x_j - u_i)}{D} \tag{5}$$

$$t_{j,i} = \frac{du_i(y_j - v_i) - dv_i(x_j - u_i)}{D} \tag{6}$$

where h_{ij} and t_{ji} represent the parameter values for the intersection of the ith edge of polygon **C** and the jth edge of polygon **S**.

Given the edges l_j and e_i, it is obvious that there is an intersection if both $h_{i,j}$ and $t_{j,i}$ are in a range between 0 and 1. The algorithm relies on the check on these parameters values although non-admissible values of these parameters also provide necessary information for the clipping. The details will be discussed in the following sections.

2.2 Line Clipping

2.2.1 General Case

We next discuss line clipping in order to better aid the process of the polygon clipping. In the general case, consider the simpler picture in Figure 2, which contains the arbitrary edge l_j of a subject polygon and clip polygonal window.

In this algorithm, we select any arbitrary point W_1 in the vertex list of the polygonal window, as a starting point to traverse cyclically from vertex to vertex in the list, in either of the two opposite directions. We consider traversal of the polygon through e_1, e_2,..., e_{15} in Figure 2. Since there exist only one *bona fide* intersection between two line segments the *t,h* values are stored as two parameter pairs. All of the intersections of line l_j and the polygon are listed as pairs of t and h values, in an order of traversal of the edges: ($t_{j,1}$:$h_{1,j}$), ($t_{j,3}$:$h_{3,j}$), ($t_{j,7}$:$h_{7,j}$), ($t_{j,9}$:$h_{9,j}$), ($t_{j,11}$:$h_{11,j}$), ($t_{j,14}$:$h_{14,j}$), and ($t_{j,15}$:$h_{15,j}$). If we sort t_j's by their values in ascending order, we obtain the sequence of t_j's as: $t_{j,14}$, $t_{j,15}$, $t_{j,1}$, $t_{j,9}$, $t_{j,7}$, $t_{j,3}$, $t_{j,11}$. (j= 1, 2, ..m).

For a simple polygon and a sequence of the sorted t_j values, if the point corresponding to the parameter t_j in the odd positions are entering the polygonal window, those on the even positions must be leaving the window, and vice versa. Also, if the point of parameter $t_j = 0$ (starting point of the line segment l_j) is inside the polygonal window, the first point of the sequence of the sorted t_j list must be leaving the polygonal window. Thus, the position of the points of the t_j on the sorted list can be determined by the relative position of the starting point of the line segment l_j.

The starting point of a line segment being inside or outside a polygon is determined easily[Man89]. For, if the line segment l_j is extended infinitely in the value of t_j and the number of intersections of the polygon on the extended part of the line segment ($t_j < 0$) is counted, the determination as to the starting point ($t_j = 0$) of the line segment being inside or outside the polygon can be done straightforwardly. t_j for the starting point ($t_j = 0$) inside the polygon is added to the left of the sorted t_j list. Finally, if the t_j list still contains odd number of the elements, t_j of the ending point ($t_j = 1$) also should be added into the t_j list because the point must be inside the polygon. Because a line segment which may be intersected several times would result in fragmented line segments, consecutive pair of t_j values will correspond to a line fragment and thus a new line segment. A complete list of t_j's always consists of even number of t_j's. If there are k(even) elements in the list and points p_1, p_2,...,p_k are associated with the t_j's, the part of the line segment p_1p_2, p_3p_4,..., $p_{k-1}p_k$ must lie inside the polygon window. If the complete t_j's only have two elements and both are equal to either 0 or 1, it indicates that the entire segment lies outside the window except maybe for one of the end points. Thus, the list is discarded. The h value, if it exists, associated with the t value, is also eliminated from the h_i list. The h_i lists are maintained like the t_j lists.

2.2.2 Special Cases

Next we consider special cases. These, are handled at the implementation level. Consider Figure 3, where the line segment, V_jV_{j+1}, is to be clipped. The intersection of line segments V_jV_{j+1} and $W_{i-1}W_{i+1}$ is calculated and examined for the value of parameter on $W_{i-1}W_{i+1}$. For an admissible value of the parameter $\in (0, 1)$, it indicates that the line segment is entering or leaving the window, as shown in Figure 3a. If the value $\notin [0, 1]$, V_jV_{j+1} does not intersect the vertex W_i. We should not count the intersection, as shown in Figure 3b, for future polygon clipping.

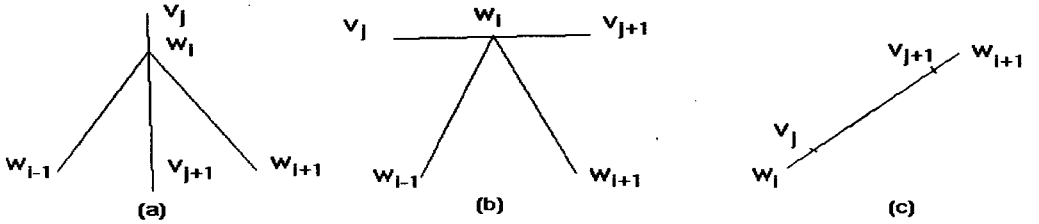

Figure 3

In Figure 3c, if a segment V_jV_{j+1} lies on a segment W_iW_{i+1} entirely, the segment V_jV_{j+1} will be disregarded. On the other hand, if a segment W_iW_{i+1} lies on a segment V_jV_{j+1} entirely, the segment W_iW_{i+1} will be not considered.

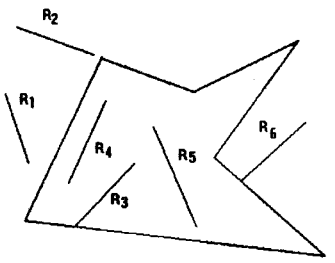

Figure 4

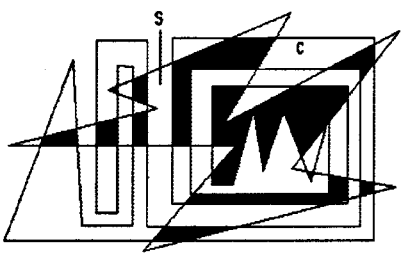

Figure 5

Other obvious special cases for which the line segment to be clipped does not intersect any edge of the window, as in Figure 4, are handled trivially as a general case described in previous section.

In Figure 4, R_i denote line segments that are clipped and do not have any intersection with the edges of the window. The algorithm does not pay a special attention to these cases. Ending points of R_3, R_4, and R_5 is included into their t_j lists.

2.3 Polygon Clipping

After we traverse through all of edges of subject polygon **S**, a pair of sorted lists t_1, t_2,...,t_m, h_1, h_2,..., h_n, as described in previous section, are obtained for each edge of subject polygon **S** and clip polygon **C**.

The process of clipping a polygon results in a region of a subject polygon **S** intersecting the region of a polygon window **C**, $\mathbf{S} \cap \mathbf{C}$ (shaded regions of Figure 5). The process results in a set of regions that are enclosed by a set of polygon edges and formed as a set of polygons. These polygons are composed of the edges or the part of edges of polygons **S** and **C**. These boundaries of polygons that are created by the clipping process mutually lie in polygon **S** and polygon **C**.

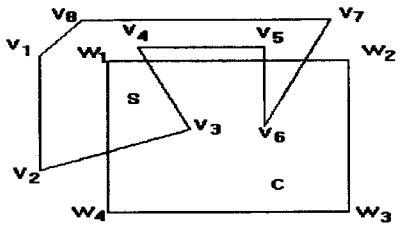

Figure 6

To describe the algorithm, we consider an example shown in Figures 6. In the example, we start to traverse the subject polygon to be clipped from vertex V_1, V_2, etc. (as shown in Figure 6):

Edge list of Polygon S:

l_1: $\langle V_1V_2\rangle$, l_2: $\langle V_2V_3\rangle$, l_3: $\langle V_3V_4\rangle$, l_4: $\langle V_4V_5\rangle$,
l_5: $\langle V_5V_6\rangle$, l_6: $\langle V_6V_7\rangle$, l_7: $\langle V_7V_8\rangle$, l_8: $\langle V_8V_1\rangle$.

Edge list of Polygon C:

e_1: $\langle W_1W_2\rangle$, e_2: $\langle W_2W_3\rangle$, e_3: $\langle W_3W_4\rangle$, e_4: $\langle W_4W_1\rangle$.

As described, a set of t and h lists can be obtained. These are shown on Table 1. In the table, the cell cross l_j and e_i indicates a pair of parameters, $(t_{j,i}:h_{i,j})$, of the intersection. "Start" and "End" indicate the parameters of the ending points of a edge.

The complete traversal is shown in Table 2, where a parameter in parentheses and a parameter without parentheses in a cell are a pair of parameters.

We skip l_1 because it is outside the window. Starting from l_2, we find that the point corresponding to $t_{2,4}$ is ***entering*** the polygon **C**. We proceed to the next point corresponding to the parameter t_2^e, which is the ending point of line segment l_2 and it is not an intersection of two edges of the polygons. We continue to traverse to the next edge, l_3. The parameter $t_{3,1}$ represents a point ***leaving*** the polygon **C**. The point of the intersection on edge e_1 of polygon **C** is associated with parameter $h_{1,3}$ and with respect to polygon **C**, that point of the parameter $h_{1,3}$ is also *leaving* the polygon **S**. This means we should traverse backwards on edge e_1, at this point.

The next parameter on the list, h_1^s, corresponds to a point inside polygon **S** and is not an intersection of two polygons. Thus next edge is e_4, on which we reach the point of parameter $h_{4,2}$ which is an intersection. This results in the switch to polygon **S**. In this case, this

intersection on l_2 is the beginning point from which we commenced traversal, thus generating a circuit in the path traversed. This path which constitutes the boundary of the polygon, divides a 2D plane into two regions: the interior, enclosed by the path and exterior that is a region other than interior on the plane. The interior enclosed by the path must be a polygon. A set of vertices that form the polygon in the example, is composed of the points corresponding to the parameters $t_{2,4}$, t_2^e, $t_{3,1}$, h_1^s. Because the search terminated on the sublist for polygon edge l_3 in the construction of the output polygon(s), we continue the search for other potential output polygons. We restart the traversal on l_5. In this case, there is no other intersection on l_1, l_2, and l_3. The next point to be considered is that of $t_{5,1}$, that is entering the polygon **C**. The next parameter, t_5^e is the ending point of the line segment l_5 and is inside polygon **C**. We go onto l_6 and the point for $t_{6,1}$ is an intersection. Next, we focus on the companion edge e_1. On e_1, the point of the parameter $h_{1,6}$ is leaving polygon **S**. We then traverse e_1 backwards to the point of $h_{1,5}$ which is the starting point of the traversal. Thus, a set of vertices for the second output polygon is obtained as the points for the parameters $t_{5,1}$, t_5^e, and $h_{1,6}$.

TABLE 1:

Edges	Start	e_1	e_2	e_3	e_4	End
Start		h_1^s				
l_1						
l_2					$t_{2,4}$: $h_{4,2}$	t_2^e
l_3	t_3^s	$t_{3,1}$: $h_{1,3}$				
l_4						
l_5		$t_{5,1}$: $h_{1,5}$				t_5^e
l_6	t_6^s	$t_{6,1}$: $h_{1,6}$				
l_7						
l_8						
End					h_4^e	

TABLE 2:

Poly	Edge	1	2		3	4
1 →	l_2	$t_{2,4}$ ($h_{4,2}$)	t_2^e (t_3^s)			
	l_3	t_3^s (t_2^e)	$t_{3,1}$ ($h_{1,3}$)			
2 →	l_5	$t_{5,1}$ ($h_{1,5}$)	t_5^e (t_6^s)			
	l_6	t_6^s (t_5^e)	$t_{6,1}$ ($h_{1,6}$)			
	e_1	h_1^s (h_4^e)	$h_{1,3}$ ($t_{3,1}$)	2 •	$h_{1,5}$ ($t_{5,1}$)	$h_{1,6}$ ($t_{6,1}$)
1 •	e_4	$h_{4,2}$ ($t_{2,4}$)	h_4^e (h_1^s)			

All edges of polygon **S** have been visited, and there are two output polygons: $\{h_1^s, t_{3,1}, t_3^s, t_{2,4}\}$ and $\{t_{5,1}, t_6^s, t_{6,1}\}$. We note that all vertices are colored white and whenever we visit a vertex, the point is marked gray to prevent it from being visited more than once.

3. Clipping a Polygon against a Non-Convex Volume in 3D

The problem of clipping a simple polygon against a concave polygonal window, as described previously, can be helpful in clipping a simple polygon against a non-convex volume in 3D.

3.1 Parametric Representation

We assume a polygon **S** to be clipped, is made up of a set, **L**, of edges of the polygon, and a clipping volume, **V**, is composed of a set, **F**, of boundary planar facets each of which is formed by a set, **E**(k), of edges. The basic operation is to clip a polygon described by an input vertex list against the non-convex volume and may result in one or more output vertex lists. Each vertex list correspond to an output polygon.

A clipping non-convex volume **V** can be defined as a sequence of polygonal facets F(1), F(2),..., F(p) and each of these facets can be defined as a sequence of vertices. The edges of polygon to be clipped and the edges of each facet of a volume can be defined as follows:

$$P(t_j) = P_j + (P_{j+1} - P_j)t_j \quad (7)$$

$$V^k(h^k) = V_i^k + (V_{i+1}^k - V_i^k)h_i^k \quad (8)$$

where t_j^k , $h_i^k \in [0 , 1]$.

3.2 Polygon Clipping

The algorithm for clipping a polygon against a volume in 3D has two major procedures that are used to determine the intersections of the plane on which a polygon to be clipped lies, with one of facets of a clipping volume and to compose polygonal windows by these intersections on the plane. The rest of processes is very similar to polygon clipping in 2D as already discussed in the previous sections.

In Figure 7, is a polygon S is to be clipped against a cubic volume (we note, the polygon and the volume do not need to be convex).

3.2.1. Intersection of a Polygon Plane and a Planar Facet

The volume is defined by a set of planar facets that form a polyhedron. The intersection of a plane containing the polygon to be clipped and a plane containing one of facets of the volume is a line that can be obtained from:

$$A_i x + B_i y + C_i z + D_i = 0 \quad (9)$$

i = 1, 2

Thus, the intersection of a plane that contains the polygon to be clipped and a bounded planar facet is a line segment through two endpoints, $V_1(x_1, y_1, z_1)$, and $V_2(x_2, y_2, z_2)$, which lie on the plane and facet. The segment can be represented parametrically as:

$$V(r) = V_1 + r(V_2 - V_1) \quad (10)$$

where $r \in [0,1]$.

If we assume that the value of parameter r in (10) is such that $r \in (-\infty , \infty)$, the equation illustrates a line through two points, V_1 and V_2, instead of a line segment. The question is how two points, V_1 and V_2, are defined before the intersection of a plane containing the polygon to be clipped and a bounded planar facet is determined. In fact, V_1 and V_2 can be arbitrarily defined. Thus y_1, z_1, y_2, and z_2 can be determined from equation (9). Therefore, we obtain the intersection

of a plane containing the polygon to be clipped and a bounded planar facet while clipping a line as defined in equation (10), $r \in (-\infty, \infty)$, against the boundary facet.

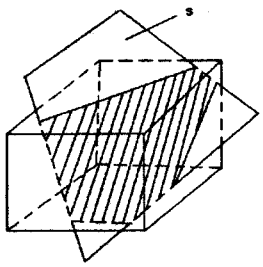

Figure 7

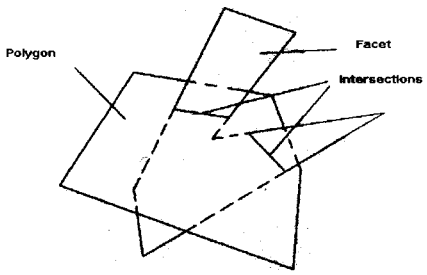

Figure 8

After the line clipping process, adjustment of V_1 and V_2 need to be made according to the ending points of the clipped line segment. Following the adjustment in V_1 and V_2, equation (10), with $r \in [0, 1]$, represents the clipped line segment which represent the intersection of a plane containing the polygon to be clipped and a bounded planar facet and that lies on the polygon plane and the facet, as shown in Figure 8.

3.2.2. Clipping Window on the Plane Containing the Polygon to Be Clipped

The clipped line segments described in previous section will form one or more polygons on the plane containing the polygon to be clipped after we have visited all the planar facets of the volume. Eventually, these polygons are considered as clipping windows. The polygon to be clipped is clipped against these windows.

A clipping window may be generated by making a chain connection of the clipped line segments until the chain forms a cycle. Thus, these segments can produce one or more clipping window on the plane if the clipping volume is non-convex. After all of clipping windows have been generated, clipping the polygon against the clipping windows is almost the same as the polygon clipping in 2D as we have discussed in previous sections, with a little modification based on 2D polygon clipping. The result of clipping the polygon against these clipping windows is what is expected for clipping the polygon against the volume.

4. Implementation

The algorithm was implemented in Borland C++ on a PC. In C++, the algorithm is implemented as a **Class**. The sorted t_j and h_i lists are built as two linked lists to maintain the t_j and h_i lists and generate as output, clipped polygons. The algorithm does not need to distinguish the subject polygon and a polygonal window. It accepts two defined polygons, one of which is a subject polygon and the other a polygon window, and the algorithm provides the same results. This releases us from the burden of knowing the difference between two input polygon parameters.

Sorting the h_i and t_j lists is an essential part of the algorithm. A straight-radix sort [Man 89], whose complexity is $O(cn)$, where c is the number of digits after the decimal points, may be applied. c can be regarded as a constant for a particular resolution. Therefore, the complexity becomes O(n). However, there exist other sort algorithms that have complexity $O(n\log n)$ and

are widely used. For n large, the straight-radix sort algorithm is better than O(n*log*n) sort algorithm. Thus, this polygon clipping algorithm can be maintained as complexity O(mn).

5. Conclusion

The algorithm extends line clipping to polygonal clipping without much difficulty. The treatment is extended to handle an arbitrary polygonal window. Consideration of some special cases are handled in the general treatment and it scales into the polygonal case. The line clipping algorithm is generally efficient. In processing output polygons, less consideration can be devoted to some special cases that might thus occur. Because of the nature of clipping a simple polygon against a nonconvex polygonal window, we need to calculate all the possible intersections of two polygon edges. The algorithm requires floating-point computation and suffers from the fact that all intersections need to be considered.

The polygon clipping algorithm jointly clips the edges of a clipping window while the edges of a polygon are clipped. Those mutually exclusive edges of a polygon and a window are eliminated. The complexity remains essentially the same. The algorithm does not restrict the orders of vertices of two input polygons to having the same directions. The order of the vertices of an input polygon can be either clockwise or anti-clockwise. Overall, the algorithm has less restrictions and easy to implement.

The basic concept of the algorithm described for the 2D case is modified for clipping against an arbitrary volume The ease provided by the polygon clipping against an arbitrary volume is due to the parametric representations and the representation of polyhedral objects in 3D. Further modifications of the algorithm could be made to satisfy Boolean operations such as union and difference on polygons.

6. Acknowledgment

The authors wish to thank the reviewers for their valuable comments.

7. References

[CB78] M. Cyrus and J. Beck, Generalized Two- and Three-Dimensional Clipping, Computers and Graphics, Vol. 3, No.1 1978, pp. 23-28.

[SS68] Robert F. Sproull and Ivan E. Sutherland, A Clipping Divider, FJCC, 1968, Thompson Books, Washington DC., pp. 765-775

[SH74] Ivan E. Sutherland and G. W. Hodgman, Reentrant Polygon Clipping, CACM, Vol. 17, No. 1, Jan. 74, pp. 32-42.

[LB84] You-Dong Liang and Brian A. Barsky, A New Concept and Method for Line Clipping, 1984, ACM Transactions on Graphics, Vol.3, No.1.

[LB83] You-Dong Liang and Brian A. Barsky, An Analysis and Algorithm for Polygon Clipping, 1983, pp. 868-877.

[NS79] William M. Newman and Robert F. Sproull, Principles of Interactive Computer Graphics, 2nd ed., McGraw-Hill Book Co., NY, 1979

[Hil90] Francis S. Hill, Jr., 1990, Computer Graphics, Macmillan Publ. Co, NY, 1990.

[PS85] Franco P. Preparata and Michael Ian Shamos, Computational Geometry -- An Introduction, Springer-Verlag New York Inc., 1985.

[Mai92] Patrick-Gilles Maillot, A New, Fast Method for 2D Polygon Clipping: Analysis and Software Implementation, ACM Transaction on Graphics, Vol. 11, No. 3, July 1992. Pages 276-290.

[FDF+90] James D. Foley, Andries Van Dam, Steven K. Feiner, and John F. Hughes, Computer Graphics and Practice, Addison-Wesley Publishing Company, 1990.

[Man89] Udi Manber, Introduction to Algorithms, Addison-Wesley Publishing Company, 1989.

[Dun83] Michael R .Dunlavey, Efficient Polygon-Filling Algorithms for Raster Displays, ACM Transactions on Graphics, Vol. 2, No. 4, October 1983, Pages 264-273.

[Pin91] David Pinedo, Window Clipping Methods in Graphics Accelerators, IEEE Computer Graphics and Applications, May 1991, Pages 75-84.

[And89] Rumen D. Andreev, Algorithm for Clipping Arbitrary Polygons, Computer Graphics Forum 8(1989)183-191.

[EE90] Herbert Edelsbrunner and Ernst Peter Mucke, ACM Transactions on Graphics, Vol. 9, No. 1, January 1990, Pages 66-104.

[NLN87] Tina M. Nicholl, D. T. Lee, and Robin A. Nicholl, An Efficient Algorithm for 2D Line Clipping: Its Development and Analysis, Computer Graphics, Vol. 2, No. 4, July 1987, Pages 253-261.

[Vat92] Bala Vatti, A Generic Solution to Polygon Clipping, Communication of the ACM, Vol. 35, No. 7, July 1992, Pages 57-63.

[Ska89] V. Skala, Algorithms for 2D Line Clipping, Eurographics 89, North Holland, 1989, Pages 355-366.

[Rap91] Ari Rappoport, An Efficient Algorithm for Line and Polygon Clipping, Visual Computer 7(1), 1991, Pages 19-28.