A Simple and Efficient Algorithm for Line and Polygon Clipping in 2-D Computer Graphics

Sushil Chandra Dimri
Professor and Head
Graphic Era University
Dehradun

ABSTRACT
The most popular lines clipping algorithms are Cohen-Sutherland and Liang-Barsky line clipping algorithm. These algorithms are complex and the steps of calculation are very high. This paper proposes a simple new line clipping algorithm for 2D space which uses the parametric equation of the line. This algorithm further easily extended to the polygon clipping by considering the edge of the polygon as a line. The proposed algorithm is numerically tested for a numbers of random line segments and the results show the simplicity and less complex behavior of the algorithm.

Keywords
Line clipping algorithm, Cohen-Sutherland line clipping algorithm, Liang-Barsky line clipping algorithm. 2D space

1. INTRODUCTION
Clipping is a basic and important problem in computer graphics. It is the process which removes that portion of an image which lies outside a specified region called the clip window. Line clipping is useful in 2D-3D designing, building architecture, different transformations, animation, VLSI circuits design, and too many more. In 2D there are 5 types of clipping – i) point clipping ii) Line clipping iii) Polygon Clipping iv) Curve clipping v) Text clipping, in this paper our focus is on line clipping only . The most popular line clipping algorithm are the Cohan-Sutherland line clipping algorithm, the Liang-Barsky line clipping, the Cyrus-Beck line clipping and the Nicholl – Lee– Nicholl line clipping algorithm [1, 2 and 3].

The Cohen–Sutherland algorithm was developed by Danny Cohen and Sutherland. This algorithm is used for line clipping. The algorithm divides a two-dimensional plane into 9 regions ,every region is assigned a 4 bit code by code assignment scheme, the code for clip window is 0000, using these 4- bit code algorithm determines whether the line is clipping candidate or not also which lines are completely out and which completely inside the clip window . For clipping candidate lines algorithm determines the point of intersection of the line with the boundaries of clip window and those portion of lines clipped which lies outside of the clip window. The algorithm needs to compute 4 bit code for each end point of a line. This way the algorithm identifies the position of the line with respect to clip window which increases the calculation. Also algorithm creates some confusion when a line passes through three regions. [1, 2, and 3]

The Liang-Barky Clipping algorithm uses the parametric representation of a line, which changes the line in to a collection of infinite number of points. Algorithm simply identifies those points of line which lies inside the clip window and clips those which are outside the clip window. The u_{max} and u_{min} value of parameter u helps to determine the point of intersection [4]. The Cyrus–Beck algorithm is a more generalized line clipping algorithm which can deal with different shape of clip window which are generally of rectangular shape. Cyrus Beck is more efficient than Sutherland–Cohen algorithm in which repetitive clipping is used. [5].

The Nicholl–Lee–Nicholl algorithm is an efficient line clipping algorithm that reduces the chances of clipping a single line segment multiple times, as may happen in the Cohen-Sutherland algorithm. This algorithm divides the region around the clipping window into a number of different sub regions, depending on the position of the initial point of the line to be clipped. However, this algorithm is only applicable in two dimensions [6]. Dörr [7] used both the parametric representation line and out code for line clipping and deviced a new algorithm. Sharma and Manohar [8] proposed an algorithm based on geometric observations. Skala [9,10 and 11] also proposed an O(lg N) algorithm for line clipping against convex window.

Day [12].proposed a new algorithm for clipping lines against rectangular windows.It is suitable for computations in both object space (floating point arithmetic) and image space (integer arithmetic). Guodong et. al., [13] clipping algorithm is to save the unnecessary intersection calculations demanded by traditional algorithms either for rejecting some totally invisible lines or for clipping some partially visible lines.

Andreev and Sofianska [14] reduced the computational load by identifying the basic cases of the locations of the line segment with respect to the window. Sobokow, et. al., [15] encoded the line instead of encoding its end points and showed improvement over the then existing algorithms. The algorithm is similar to the Cohan Sutherland algorithm but in contrast to the CS algorithm, it does not need to iterate to find out the clipped segment. The line code is an 8-bit number instead of 4-bit out-code used in the Cohan Sutherland algorithm. A large number of cases need to be considered and that's why there is a big switch statement or a long else-if ladder in the implementation algorithm.

The proposed algorithm expresses the line in parametric form, which represents the line as a collection of infinite points. Initially a lines end point coordinates are checked with the boundaries of the clip window then the line passes to next step of algorithm.

The line is then tested whether it intersect the left boundary , right boundary ,bottom boundary and upper boundary and the respective values of parameter u is recorded. If value of u is less than 0 and greater than 1 simply discard that point. Also there is a condition if x coordinate of the point (in case of bottom and upper boundary)or y coordinate (in case of left and right boundary) does not satisfies the condition then discards the point even if u lies between 0 and 1. Lastly there are only2 points which remains, protect the line between these two points and delete the rest. The proposed algorithm is

unique and simple since it does not divide the 2D plane in sub region also there is neither uses 4 bit code system, nor the use of vector algebra.

2. PROPOSED NEW ALGORITHM

AB , A(x_1.y_1) and B(x_2,y_2) ,C(x_3.y_3) and D(x_4,y_4) and E(x_5.y_5) and F(x_6,y_6) are the line segments which we want to clip against the clip window PQRS shown in figure 1.1.

The boundaries of clip window is given by x= a, x=b y=c and y=d. line segment AB intersect the clip boundaries at point T, V, L and W.

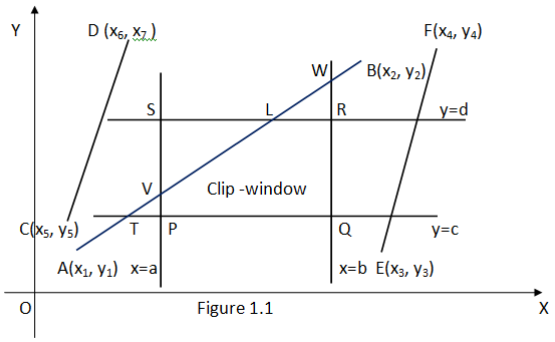

Fig 1.1 – Proposed Algorithm

3. ALGORITHM – Clip Line

(Clip Window (a, b, c, d), line(x_1, y1, x2, y2))

Initial condition check:

if (x_1 and x_2) < a), the line completely outside of clip window

If ((x_1 and x_2)>b), the line is completely outside of clip window

If (y_1 and y_2) < c, the line is completely outside of clip window

If ((y_1 and y_2)>d) the line is completely outside of clip window

If (a≤ x1, x2≤b) && (c ≤ y1, y2≤d) then line lies complete inside the clip window.

Otherwise- parametric equation of line AB

$x=x_1+u\ (x_2-x_1)$ or $x=x_1+u\ (\Delta x)$, $y= y_1+u\ (y_2-y_1)$ or $y= y_1+u\ (\Delta y)$, where 0≤u≤1

Step-1:

$a= x_1+u.\ (\Delta x)$, $u=(a- x_1\ )/\ \Delta x$, if 0 ≤ u≤1 , $y= y_1 + ((a- x_1\ ).\Delta y)/\ \Delta x$, // left boundary intersection

If (y<c && y>d) then discard the point even 0≤u≤1 otherwise record point (a, y)

Step-2:

$b= x_1+u.\ (\Delta x)$, $u=(b- x_1\ )/\ \Delta x$, if 0≤u≤1 , $y= y_1 +((b- x_1\ ).\ \Delta y)/\Delta x$ // right boundary Intersection:

If (y<c && y>d), discard the point even 0≤u≤1 otherwise record point (b, y)

Step-3:

$c= y_1+u.\ (\Delta y)$, $u=(c- y_1)/\ \Delta y$, if 0≤u≤1 $x= x_1 +(c- y_1\ ).\ \Delta x)/\ \Delta y$ // bottom boundary intersection:

If (x<a && x>b), discard the point even 0≤u≤1 otherwise record Point (x, c)

Step-4:

$d= y_1+u.\ (\Delta y)$, $u=(d- y_1\ )/\ \Delta y$, if 0≤u≤1 , $x= x_1 +((d- y_1\ ).\ \Delta x)/\Delta y$ // Upper boundary intersection:

If (x<a && x>b), discard the point even 0≤u≤1 otherwise record Point (x, d)

Join the recorded points

Finally after the clipping the line is figure 1.2

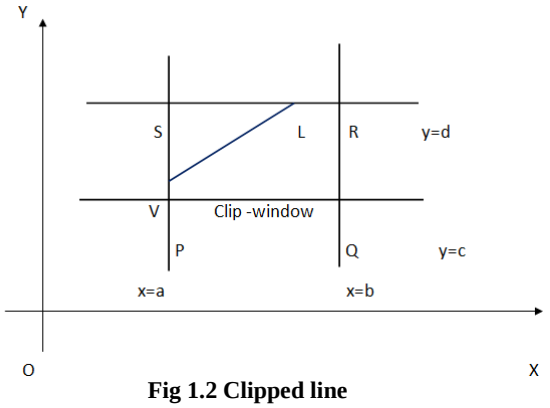

Fig 1.2 Clipped line

For polygon clipping consider each edge of the as a line and clip it using the same algorithm against the give clip window. Finally we will remain with the clipped polygon**.**

4. RESULT WITH NUMERIC VALUE

4.1: line Clipping

Clipping the line AB shown in figure 1.3 against the clip window PQRS

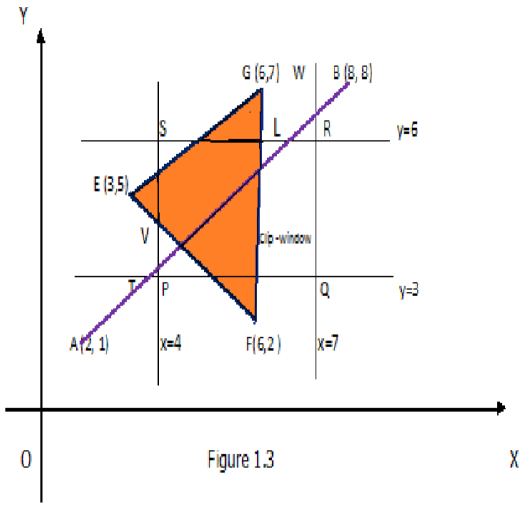

Fig 1.3 Result

Line AB is neither outside nor inside the clip window (none of the condition satisfied)

Parametric equation of line AB

$x_1 =2$, $y_1 =1$, $x_2 = 8$, $y_2 =8$, $\Delta x = x_2-x_1 =6$, $\Delta y = y2-y_1 =7$

$x=x_1+u\ (x_2-x_1)$, $x=x_1+u\ (\Delta x)$, $y= y_1+u\ (y_2-y_1)$, $y= y_1+u\ (\Delta y)$

$x=x_1+6u$, $y= y_1+7u$, Where $0\le u\le 1$,

i) For left boundary (x =4) intersection:

$a= x_1+u(\Delta x)$

$4= 2+6u$, $u=2/6 =1/3$

if $0 \le u\le 1$,then $y= y_1+7u$, $y= 1+7x1/3 =10/3 = 3.33$

If($c \le y\le d$) i.e. $3 \le y\le 6$ so record the point V (4, 3.33)

ii) For right boundary (x =7) intersection:

$b= x_1+u(\Delta x)$, $7= 2+6u$, $u=5/6$

if $0\le u\le 1$, W(b, y),

$y= y_1+7u = 1+ 7x5/6 =41/6=6.83$, point W (7, 6.83)

if ($c \le y\le d$)i.e. $3 \le y\le 6$, consider the point otherwise discard the point. Since 6.83> 6 so discard the point W.

iii) For bottom boundary (y=3) intersection:

$c= y_1+u(\Delta y)$, $3= 1+7u$,$u=2/7$, if $0\le u\le 1$

$x=x_1+6u$, $x=2+6x2/7 =26/7=3.71$, point T (3.71, c)

If ($a \le x\le b$) i.e. $4 \le x\le 7$, consider the point otherwise discard the point

$3.71 <4$, discard the point T even $0\le u\le 1$

iv) For upper boundary (y=6) intersection:

$d= y_1+u(\Delta y)$, $6 = 1 +7u$, $u =5/7$, if $0\le u\le 1$

$x=x_1+6u$, $x=2+6x5/7 =44/7=6.286$

If ($a \le x\le b$) i.e. $4 \le 6.286 \le 7$, consider the point otherwise discard the point

Even $0\le u\le 1$ so record L (6.286, 6)

Thus we are remaining with two points V and L hence after clipping the line is VL

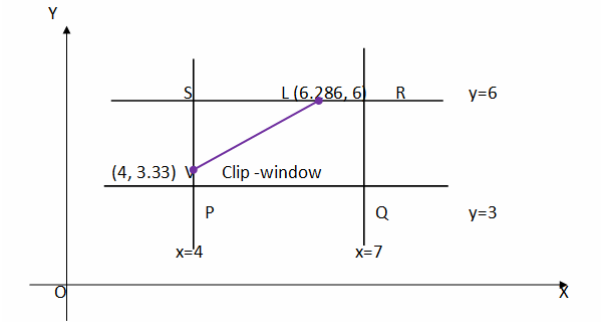

Fig 1.4 - Clipped line VL

4.2 Polygon clipping

In the Polygon EGF (E (3, 5) G (6, 7) F (6, 2)), the edges (lines) are EG, EF and GF are not in the category (out -side Clip window)

For Polygon edge EG

$x=3+u(6-3)=3+3u$, $y=5+u(7-5)=5+2u$, $0 \le u \le 1$

$x_1=3$, $x_2=6$ $x_1=3<x=4$ (left boundary)

$y_1=5$, $y_2=8$ $x_2=6<x=7$ (right boundary)

For left boundary x=4

$4=3+3u \rightarrow u=1/3$, $0 \le u \le 1$

$y=5+ (1/3)2 \rightarrow 5+2/3=5.66$

$y_1=3<y<y_2=6$ record- P_1 (4, 5.66)

For right boundary x=7

$7=3+3u$

$3u=4$ $u=4/3$ (discard this value)

Bottom boundary

$3=5+2u$

$-2=2u$, $u= -1$ (discard this value)

For upper boundary y=6

$6=5+2u$, $2u=1$, $u=1/2$

$x=3+3(1/2)$, $=3+1.5=4.5$

Record -P_2 (4.5, 6)

Now **Polygon edge EF** we have E (3, 5) F(6,2)

$x=3+3u$

$y=5-3u$, $0 \le u \le 1$

Left boundary x=4

$4=3+3u$, $u=1/3$

$y=5-3(1/3) = 4$ so record P_3(4,4)

Right boundary x= 7

$7=3+3u$, $u=4/3$ (discard it)

Bottom boundary y= 3

$3=5-3u$, $-2=-3u$,$u=2/3$

$x=3+3u$, $3+3(2/3) =5$, record - P_4(5,3)

For upper boundary y= 6

$6=5-3u$, $1=-3u$, $u=-1/3$(discard it)

Now Polygon edge FG F(6,2) G(6,7)

$x=6+u(0) = 6$

$y=2+u(7-2) =2+5u$

$x=6$, $0 \le u \le 1$ this line is parallel to y axis.

x=6, y=3, y=6 , P_5 (6, 3) and P_6 (6, 6)

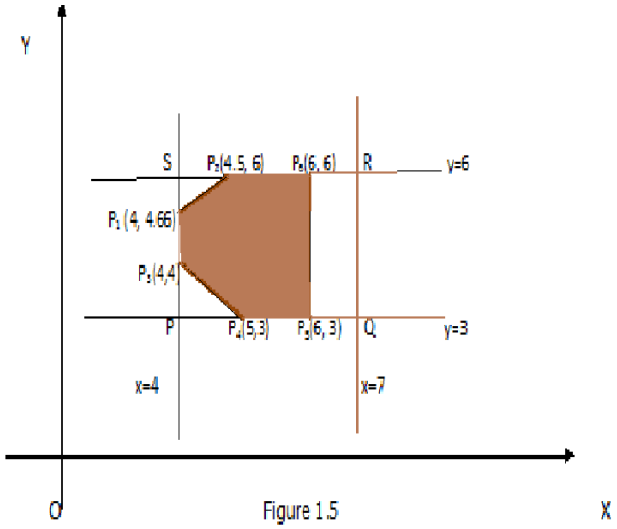

Figure 1.5

The polygon EGF clipping- the polygon is simply the collection of finite number of edges, every edge is a line, we can apply the same algorithm edge wise to clip the polygon. Here we have edges EG, EF and FG in this polygon shown in figure 1.5.

5. CONCLUSION

The proposed algorithm is based on parametric form of equation of line, which computes the point of intersection of the line with the boundaries of clip window with simple calculation. In other popular algorithms the complexity of calculation are very high but the proposed algorithm is quite simple and it is very easy to identify the clipping candidate line and their points of intersection with the boundaries of clip window. Further this algorithm can easily be modified for polygon clipping by considering each edge of polygon as a line segment.

6. ACKNOWLEDGEMENT

We wish to express our heartfelt gratitude and cordial thanks to Prof (Dr.) Kamal Ghanshala (President Graphic Era University) for his sincere and continual encouragement in preparing this paper. Thanks are also due to Dr.R. C. Joshi (Chancellor Graphic Era University, Ex HOD and Professor (EC and CS Deptt) IIT Roorkee) and Assistant Professor Ms. Neelam Kathait for their continual guidance, support and helpful discussion.

7. REFERENCES

[1] J. D. Foley, A. van Dam, S. K. Feiner and J. F. Hughes, "Computer Graphics: Principles and Practice", Addison-Wesley, (2nd edition), (1996).

[2] D. Hearn and P. Baker, "Computer Graphics with OpenGL", 3rd ed., Prentice Hall, (2004).

[3] D.F. Rogers. "Procedural elements for computer graphics". New York: McGraw-Hill, 1985. P.111–87.

[4] Y. D. Liang and B. A. Barsky, "A new concept and method for line clipping", ACM Transactions on Graphics, vol. 3, no. 1, (1984).

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

[6] T. M. Nicholl, D. T. Lee and R. A. Nicholl, "An efficient new algorithm for 2-D line clipping: its development and analysis", Computer & Graphics, vol. 21, no. 4, (1987).

[7] M. Dörr, "A new approach to parametric line clipping", Computers & Graphics, vol. 14, no. 3-4, (1990).

[8] N.C. Sharma, S.Manohar Line clipping revisited: two efficient algorithm basedon simple geometric observations. Computers and Graphics 1992;

[9] V. Skala, "An efficient algorithm for line clipping by convex polygon", Computers & Graphics, vol. 17, no. 4, (1993). 1992; 11(4): 241–5.

[10] V. Skala, "A new approach to line and line segment clipping in homogeneous coordinates", The Visual Computer, vol. 21, no. 11, (2005). [15] D. F. Rogers, "Procedural Elements for Computer Graphics", 2nd Edition, Tata McGraw-Hill, (2005).

[11] V. Skala,―O (lg N) Line clipping Algorithm in E , ‖Computers and Graphics, Vol. 18, No. 4, 1994, pp. 517-527.

[12] J. D. Day, "An algorithm for clipping lines in object and image space", Computers & Graphics, vol. 16, no. 4, (1992).

[13] L. Guodong , W.Xuanhui , P. Qunsheng "An efficient line clipping algorithm based on adaptive line rejection "Computers & Graphics 26 (2002) 409–415

[14] R. Andreev and E. Sofianska, "New algorithm for two-dimensional line clipping", Computers & Graphics, vol. 15, no. 4, (1991).

[15] M.S. Sobleow,P. Pospisil, Y.H .Yang,: A Fast Two Dimensional Line Clipping Algorithm via Line Encoding, Computers & Graphics, Vol.11, No.4, pp.459-467, 1987.