An Efficient Line Clipping Algorithm in 2D Space

Mamatha Elliriki[1], Chandrasekhara Reddy[2], and Krishna Anand[3]
[1]Department of Mathematics, GITAM University, India
[2]Department of Mathematics, Cambridge Institute of Technology-NC, India
[3]Department of Computer Science, Sreenidhi Institute of Science and Technology, India

Abstract: *Clipping problem seems to be pretty simple from human perspective point of view since with visualization a line can easily be traced whether it is completely inside and if not what portion of the line lies outside the window. However, from system point of view, the number of computations and comparisons for lines with floating point calculations are extremely large which in turn adds to inherent complexity. It needs to minimize the number of computations thereby achieving a significant increase in terms of efficiency. In this work, a mathematical model has been proposed for evaluating intersection points thereby clipping lines which decently rely on integral calculations. Besides, no further computations are found to be necessary for evaluating intersection points. The performance of the algorithm seems to be consistently good in terms of speed for all sizes of clipping windows.*

Keywords: *Ortho lengths, raster graphics system, line clipping, intersection points, geometrical slopes, rectangle window.*

Received July 23, 2015; accepted June 1, 2016

1. Introduction

The usage of line clipping in a wide variety of scientific applications seems to increase rather exponentially. Although the concept appears simple at the outset, its enormity of usage makes scientific research on the same an urgent requirement. With this perspective in mind, this work has shifted its focus to improving the efficiency of line clipping algorithms [4]. This in turn also provides an added impetus to perform improvement in clipping of polygons which has also been highlighted this work.

Perhaps, line clipping algorithms seems to be the second most important algorithms after line drawing algorithms in the rank point [10] of view in graphics. In general, the clipping window in raster scan system is usually associated with a rectangular window. However, window shapes including circular and polygonal [16, 26] can also be taken into be consideration. Rectangular windows are easy to be mapped to a computer screen using a transformation called 'window to viewport mapping' which in turn is composed of translation, scaling and rotation.

The primary goal of clipping in a two Dimensional computer graphics system is to remove objects, lines or line segments that are outside the viewing port window and clipping of lines is an elementary concept in the visualization process [9]. Real world objects can be represented relative to a reference world coordinate system [12]. It is difficult to view all the objects on computer screen at the same time in one screen shot since it usually occupies many places in the world coordinate system. As in human perception and cameras, one screen shot contains the images of some objects, parts of which are probably clipped. On identifying objects or parts of objects that need to be displayed on one screen shot, clipping algorithm needs to be executed on every object to determine whether it should appear completely or partially on the screen or to be clipped out. Clipping algorithms [7, 11, 23, 24, 27] usually have parameters describing the 'clipping window' in 2D worlds or 'clipping volume' in 3D worlds apart from the representation of the object being tested for clipping.

A large number of computer and mobile applications require powerful hardware coupled with efficient and fast algorithms [3, 13, 19] for their successful usage. Besides, the selection of right choice of tools helps in the usage of the application and provides an added degree of sophistication that arises in animations and graphical user interfaces [1]. In addition, hardware implementation provides a special significance for fast processing and accurate solutions.

The rest of this paper is organized as follows. Section 2, briefly explains about the existing algorithms for clipping lines in rectangle window. Section 3 presents the problem analysis and various possibilities of lines that exist in a two dimensional coordinate system. Computational formulae are derived in section 4 and these formulae greatly simplify the work of extraction of a line which needs to be saved in raster system. In detail study and development of algorithm is explained to crop a line from original line in section 5. In section 6 various algorithms are described for optimal utilization of resources for different cases. Section 7 concludes performance analysis and comparisons with previous algorithms.

2. Existing Algorithms

Research has shown that, computation with each and every intersection between line and edge of the window needs more cycles for Central Processing Units (CPU) since it incorporates floating point operations. Several algorithms are introduced to clip lines against rectangular [2, 15, 21] and non-rectangular [12, 28] windows. A reduction in number of intersections would lead to improvement in efficiency. In this paper, an efficient algorithm is proposed to minimize the number of intersections as much as possible.

Currently, two types of approaches are available to clip a line with respect to clipping window. One of these methodologies is to deal with a logical approach [18, 20, 25] where each and every line segment is clipped identically by dividing regions with logical code. Cohen Sutherland (CS) algorithm falls under this approach and deals by encoding endpoints of line segment. This algorithm is found to be highly effective when the entire line is found to be inside the window. For lines that are completely outside, unnecessary computations need to be carried out. The complexity of computations has been found to increase when the line is found partially inside the window [8]. The second approach deals with parameterization [15, 29]. Cyrus - Beck [5] and Liang Barsky (LB) [15] algorithms come under the second category and developed algorithms with the help of parametric equations. These methods are schematized based on the behaviour of line segment. However, it has been found to be inefficient in many scenarios.

A modified version for clipping has been tried out by Nicholl Lee Nicholl (NLN) [21] where they designed a hybrid parametric algorithm using both techniques. The principal advantage of this hybrid approach deals with minimization of time consuming operations like multiplication and division [14]. They are able to reduce the number of comparisons by two thirds with respect to Cohen- Sutherland algorithm and half with respect to Liang and Barsky [15] algorithm. However, the time consumed in creation of regions is significant as the algorithm needs to initially determine the position of endpoints. Besides, the amount of effort taken for applying geometric transformations and comparing slopes lie on the other higher side and saving these computations for further tests need more memory.

All the algorithms described herein did not cater to the speed requirement of the day to day needs. Keeping this viewpoint in perspective, it needs a much faster algorithm [17, 22, 30] for rectangular window and polygon space in the present world. However, it was observed that for some scenarios, the level of complexity was found to rise in leaps and bounds which in turn led to a serious loss in efficiency. In order to rectify the inherent deficiencies in these methods, a new clipping algorithm for a complex window has been proposed by Skala [24, 25] and Day has proposed an algorithm two dimensional small window [6]. The usage of binary search in this algorithm has led to a considerable decrease in the level of complexity. However, the gain observed could not be replicated on consideration of rectangular windows.

The direct procedure to clip a line is to evaluate all intersection points of the line with edges that lie on rectangular window. It is observed that clipping problem is trickier than evaluating juncture points of infinite lines. In clipping method, it is imperative to recognize whether a crossing point for a line lies inside or on edges or outside. The same method applies for all four edges of the rectangle window. Intersection points of a line segment that lie outside edges or outside endpoints are meaningless on saving line.

3. Computational Analysis and Algorithm Development

To compute intersection points of a line, it is worthwhile to denote the line segment connected by two end points (x_{start}, y_{start}) and (x_{end}, y_{end}) and the parameters of the rectangular clipping window are (x_{left}, y_{bottom}) and (x_{right}, y_{top}); the lower left and upper right corner coordinates expressed in real-world units. Screen viewport parameters are given in pixels in image space. Without loss of generality, (x_{start}, y_{start}) can be considered as starting point which is close to *x-axis* and (x_{end}, y_{end}) is the other end point of line segment. From this, it can be observed that if slope of the straight line is positive, then $x_{start} \leq x_{end}$. Otherwise, $x_{start} > x_{end}$. But for all slopes, y_{end} is always greater than or equal to y_{start} *i.e.*, $y_{start} \leq y_{end}$.

This information is highlighted in Figures 9 and 12. These inequalities are found to be helpful for reducing number of comparisons in inside/outside tests. Now, the line can be decided with one of the following tests.

- *Complete Acceptance*: Entire line lies within the clipping window. It implies that the whole line must be saved as no clipping takes place. The comparison shown in Equation (1) is founds to be sufficient. This provides an impetus to these acceptance criteria as shown in Figure 2.

$$x_{left} \leq \{x_{start}, x_{end}\} \leq x_{right} \ \& \ y_{bottom} \leq \{y_{st\ art}, y_{end}\} \leq y_{top} \quad (1)$$

- *Complete Rejection*: Entire line located outside and on one side of clipping window and will not intersect perpendicular edges, hence line can be rejected. The comparison shown in Equation (2) highlights the case as shown in Figure 3.

$$\{x_{start}, x_{end}\} < x_{left} \ or \ \{x_{start}, x_{end}\} > x_{right} \ or \ y_{end} < y_{bottom} \ or \ y_{start} > y_{top} \quad (2)$$

- *Partial Acceptance (Case 1)*: Both endpoints are placed exterior to the rectangular window and the line intersects perpendicular edges of the clipping window. In this case, if some portion of the line is inside of clipping window then the part of the line segment has to be extracted along with end points that lie on the clipping window. This case is the most complex one and presented in detail in sections 5.1. and 5.2.

- *Partial Acceptance (Case 2)*: One end point is inside the window while the other end is outside. It is essential to find out line intersection points with one of the four edges of the clipping window and line should be drawn from inside point to intersection point. This situation is considered as a part of the previous one and discussed in section 5.3. as a special case.

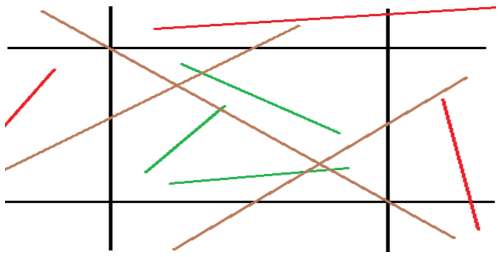
Figure 1. General line segments that are to be tested to clip in a rectangular window.

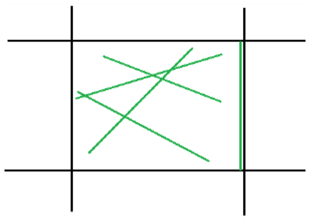
Figure 2. Set of line segments lie completely inside rectangular window (Test 1).

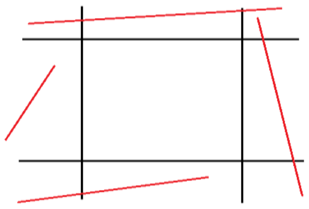
Figure 3. Completely rejected line segments that lie entirely outside clipping window (Test 2).

4. Ortho Length Formulae for Intersection Points

Formation of computational formulae is desirable for the development of algorithm and is considered to be significant work developed using fundamental geometry. As mentioned earlier, (x_{start}, y_{start}) and (x_{end}, y_{end}) are end points of a line and x_{edge} is either left or right edge dimension of the rectangular window. Similarly, y_{edge} is top or bottom edge dimension. Define, further $\Delta x = x_{end} - x_{start}$ and $\Delta y = y_{end} - y_{start}$ that are the distances in terms x directions and y directions respectively with end points of the line. From this, if slope is positive then the product of Δx and Δy is greater than or equal to zero $(\Delta x * \Delta y \geq 0)$ else product sign is negative. With the help of clipping dimensions and line segment end points, intersection points can be easily evaluated and these formulae are presented in subsequent part of this section.

4.1. Line Segment Intersection with $x = x_{edge}$ (δy is Unknown)

Let the line segment with end points $P(x_{start}, y_{start})$ and $Q(x_{end}, y_{end})$ and intersect $x = x_{edge}$ at the point $C(x_{edge}, R_y)$. R_y defined as ortho length, is the upright distance from *x-axis* to intersection point and plays strategic role to decide line segment. From geometry, the slope of a straight line for any two arbitrary points is same [8, 10, 11]. i.e.,

$$slope\{P(x_{start}, y_{start}), Q(x_{end} y_{end})\} = slope\{C(x_{edge}, R_y), Q(x_{end} y_{end})\} \quad (3)$$

$$\frac{y_{end} - y_{start}}{x_{end} - x_{start}} = \frac{y_{end} - R_y}{x_{end} - x_{edge}} \quad (4)$$

$$\frac{\Delta y}{\Delta x} = \frac{\delta y}{\delta x} \Leftrightarrow \delta y = \frac{\Delta y}{\Delta x}\delta x \quad (5)$$

where $\Delta x, \Delta y$ *and* $\delta x = x_{end} - x_{edge}$ are known constants. $\delta y = y_{end} - R_y$ is unknown constant to be evaluated.

From Equation (5), ortho length can be computed as

$$R_y = y_{end} - \frac{\Delta y}{\Delta x}\delta x \quad (6)$$

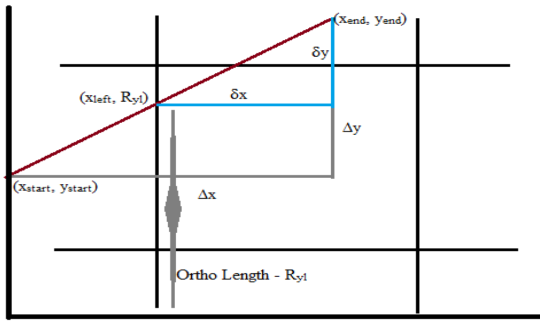

Figure 4. Ortho length Ryl in 2D-coordinate system representation.

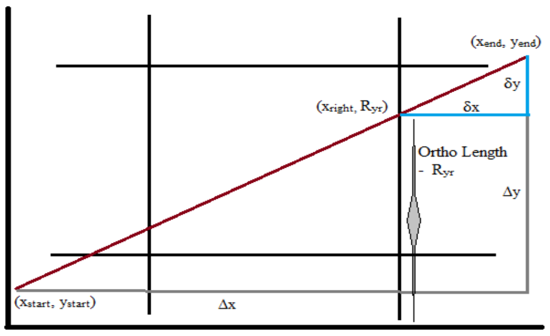

Figure 5. Graphical view of ortho length Ryr used to evaluate right edge intersection point.

Equation (6) holds good even for swapping end points of the line segment. Now, if the line segment intersects left edge of the clipping window at point *(x_{left}, R_{yl})*, (Figure 4) then Equation (6) becomes:

$$R_{yl} = y_{end} - \frac{\Delta y}{\Delta x}\delta x \text{ where } \delta x = x_{end} - x_{left} \quad (7)$$

Similar analysis can be given for the line segment that intersects right edge of the rectangle window. Let R_{yr} be the ortho distance from x-axis to intersection point of a line with right edge of the clipping window as presented in Figure 5. Then Equation (6) becomes

$$R_{yr} = y_{end} - \frac{\Delta y}{\Delta x}\delta x \text{ where } \delta x = x_{end} - x_{right} \quad (8)$$

4.2. Line segment intersection with y=yedge (δx is unknown)

Section 4.1. focussed on positive slope line segments. It could be observed that a suitable experimentation can be carried out with negative slopes also. A case is taken into consideration where the line PQ intersects straight line $y = y_{edge}$ at the point $D(R_x, y_{edge})$ where R_x is ortho distance from $(0, y_{edge})$ to (R_x, y_{edge}). Equation (9) deals with the formulation of slope. Equations (10) and (11) could be derived from slope Equation (9).

$$slope\{ P(x_{start}, y_{start}), Q(x_{end} y_{end})\} = slope\{ C(x_{edge}, R_y), Q(x_{end} y_{end})\} \quad (9)$$

$$\frac{y_{end} - y_{start}}{x_{end} - x_{start}} = \frac{y_{end} - y_{edge}}{x_{end} - R_x} \quad (10)$$

$$\frac{\Delta y}{\Delta x} = \frac{\delta y}{\delta x} \Leftrightarrow \delta x = \frac{\Delta x}{\Delta y}\delta y \quad (11)$$

$\delta x = x_{end} - R_x$ is unknown constant to be evaluated; Δx, Δy and $\delta y = y_{end} - y_{edge}$ are known constants can be evaluated with given parameters.

From Equation (11), ortho length is computed. The distance between y axis and the point of intersection on the clipping window could be computed as shown in Equation (12).

$$R_x = x_{end} - \frac{\Delta x}{\Delta y}\delta y \quad (12)$$

Rxb and Rxt are ortho lengths as mentioned in Figures 6, and 7 are distances from y-axis to bottom and top edges of clipping window respectively. Equations (13) and (14) can in turn be obtained from Equation (12).

$$R_{xb} = x_{end} - \frac{\Delta x}{\Delta y}\delta y \text{ where } \delta y = y_{end} - y_{bottom} \quad (13)$$

$$R_{xt} = x_{end} - \frac{\Delta x}{\Delta y}\delta y \text{ where } \delta y = y_{end} - y_{top} \quad (14)$$

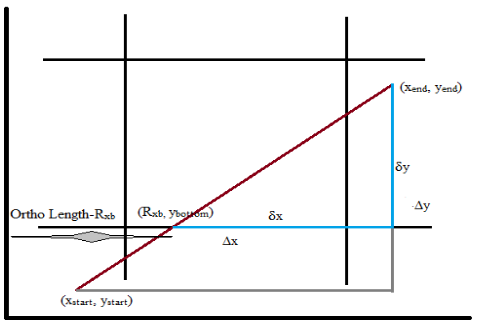

Figure 6. Ortho length Rxb evaluation for line segment that interest bottom edge.

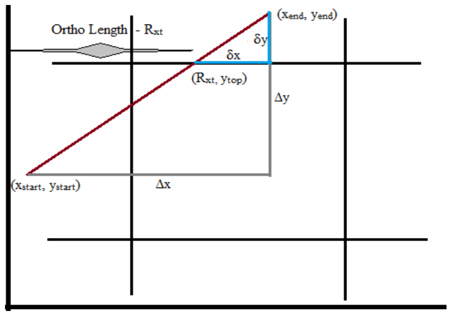

Figure 7. Graphical form of ortho length Rxt in 2D-coordinate system.

Equations (7), (8), (13), and (14), represent the intersection distance formulae for lines that clip the rectangular window. These formulae provide a precise indication of the presence of the line inside the rectangular window. Magnitudes Δx, Δy, δx and δy can be easily evaluated with help of given points of a line segment and clipping window dimensions. Ortho length dimension quantity formulae reveal equivalent even if end points are interchanged.

From this it is observed that for all slopes of a line segment, evaluation of distance from x-axis to intersection point of a line with left and right edges of clipping window has the same formula with small variation of δx and δy. These formulae further can be simplified by using roundup and floor functions in integral form as:

For left edge x = x_{left} of the clipping window

$$R_{yl} = y_{end} - \left\lfloor \frac{\Delta y}{\Delta x}\delta x + \frac{1}{2} \right\rfloor \quad (15)$$

$$R_{yl} = y_{end} - \left\lfloor \frac{2\Delta y \delta x + \Delta x}{2\Delta x} \right\rfloor \text{where } \delta x = x_{end} - x_{left} \quad (16)$$

Following similar procedure for right edge $x = x_{right}$ of the clipping window, we have:

$$R_{yr} = y_{end} - \left\lfloor \frac{2\Delta y \delta x + \Delta x}{2\Delta x} \right\rfloor \text{where } \delta x = x_{end} - x_{right} \quad (17)$$

For all slopes of a line segment the evaluation of integral distances from *y-axis* to intersection point of a line with bottom and top edges of clipping window are:

For bottom edge $y = y_{bottom}$ of the clipping window

$$R_{xb} = x_{end} - \left\lfloor \frac{2\Delta x \delta y + \Delta y}{2\Delta y} \right\rfloor \text{where } \delta y = y_{end} - y_{bottom} \quad (18)$$

For top edge $y = y_{top}$ of the clipping window

$$R_{xt} = x_{end} - \left\lfloor \frac{2\Delta x \delta y + \Delta y}{2\Delta y} \right\rfloor \text{where } \delta y = y_{end} - y_{top} \quad (19)$$

Where $\lfloor f(\omega) \rfloor$ is the floor function defined as integer part of the function $f(\omega)$.

These equation, in C compiler compute one third times faster than floating point computations. Majority of lines are eliminated by inequality tests. All remaining lines are evolved with the help of ortho lengths R_{yl}, R_{yr}, R_{xb}, and R_{xt} and these are the decision factors to choose whether the line is inside or outside of the clipping window. If inequality tests (1) and (2) fail for a line then the other possibilities for line are either both end points of line are outside which intersects perpendicular edges of the clipping window or one end of the line is inside. Now let us deal first problem as two special cases with assistance of slope of a line. For any line slope may be either positive or negative and can be decided as positive if the inequality $\Delta x * \Delta y \geq 0$ or in easy form $x_{end} \geq x_{start}$ holds good which is integral value computation, otherwise it is negative. Following sections deals in detail study for partially accepting or completely rejecting lines.

5. Line Clipping Tests

A vast majority of lines need few tests for determining the portion of line that remains inside the clipping window (Figures 1, 2, and 3). All ortholengths R_{yl}, R_{yr}, R_{xb} and R_{xt} of a line are first evaluated and the corresponding intersections have been computed. If all intersection points with ortho lengths are outside window then entire line is rejected. For the line which is partially inside clipping window, exactly two points are on the window edges and two points are outside. In this case, the line with two end points that are inside window is drawn and the remaining portion of line is discarded.

Evaluations of ortho lengths and all four intersection points could be avoided in case of invisible line through comparison tests thereby saving effort and time. All invisible lines can be eliminated by conducting a maximum of two comparisons. However, atmost four comparisons are needed for partially visible lines. For further computations, as earlier stated, first point is near to x-axis and end point of the line is above first point. From this, if $\Delta x * \Delta y \geq 0$, then slope is positive and line is drawn from left to right and/or diagonally above in the coordinate system, so that $x_{end} \geq x_{start}$ and $y_{end} \geq y_{start}$. The converse holds true for negative slopes. From this it can be accomplished that, irrespective of all slopes, authors have taken two end points in such a way that y_{end} is always greater than or equal to y_{start} ($y_{end} \geq y_{start}$).

This inequality constraint is used to reduce comparisons for partially visible and completely rejected lines.

The procedure for a line with positive slope is as follows. If $R_{yl} > y_{top}$, then entire line is outside clipping window and can be rejected (Figure 8). This can be visualized using simple analogy. Ryl is the distance from x-axis to intersection point of line with left edge of window and slope of the line is positive. Hence, the points of the line from staring point to (x_{left}, R_{yl}) fall to the left of the clipping window and outside. According to slope property, frequently used in NLN algorithm, if slope between first point and intersection point is greater than the slope between first point and top left vertex of window then entire line is outside, line is rejected. If $R_{yr} < y_{bottom}$ then line intersects right edge below clipping window and passes outside, hence rejected (Figure 8). In this case, the line with points (x_i, y_i) for all $y_i < R_{yr}$ is below clipping window and remaining portion of the line is right side to window.

Similar analysis can be carried out for negative slopes (Figure 11). For all remaining cases, a part of the line is inside clipping window and subsequent procedure is used to save part of line segment.

5.1. Straight line with Positive Slope

Initially, R_{yl} has been computed and checked for its range. If $y_{bottom} \leq R_{yl} \leq y_{top}$, then line enters the clipping window at the point (x_{left}, R_{yl}).

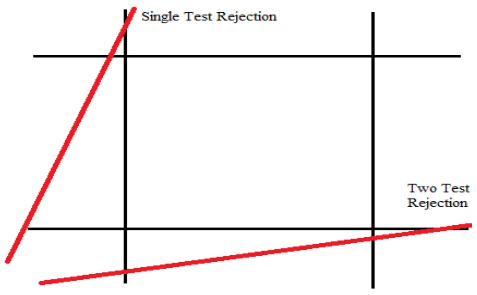

Figure 8. Line segments with positive slopes.

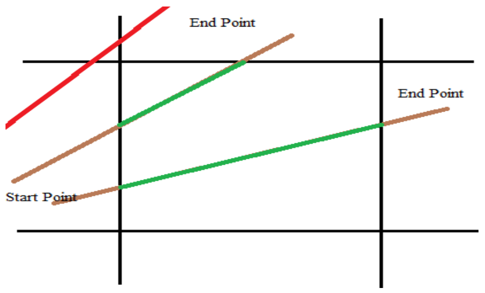

Figure 9. Line segment with positive slope.

Subsequently, the next end point has to be computed. This can be evaluated easily using ortho length formulae R_{yr} or R_{xt}. This in turn may lie either on the right or top edge of the clipping window. Figures 9, and 10 highlight the mentioned aspects.

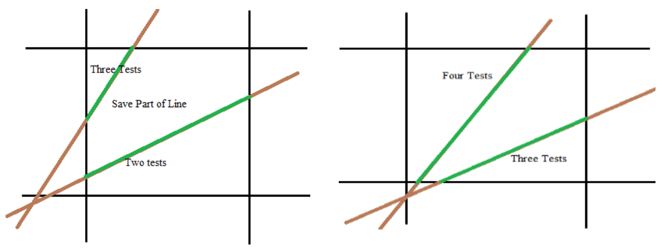

Figure 10. Segment lines enter into the clipping window with different possible cases.

5.2. Straight Line with Negative Slope

Lines with negative slopes have been found to be analogous with those of positive slope. In this case $x_{end} < x_{start}$. The values R_{yl} and R_{yr} that had been computed earlier have been retained for further computations. The following procedure helps in determination of the presence of that part of the line which is inside the window.

If $y_{bottom} \leq R_{yr} \leq y_{top}$, then line intersects the right edge of clipping window at the point (x_{right}, R_{yr}). The next end point evaluated may lie on left or top edge of clipping window. This can be evaluated using ortho length formulae Ryl or Rxt.

If $R_{yl} \leq y_{top}$, then last end point is (x_{left}, R_{yl}) else (R_{xt}, y_{top}) and part of visible line can easily be drawn using these two end points.

If $R_{yr} < y_{bottom}$ then line enters inside window at the point (R_{xb}, y_{bottom}) by intersecting bottom edge. If $R_{yl} \leq y_{top}$, the next intersection point is (x_{left}, R_{yl}). Otherwise the point is (R_{xt}, y_{top}).

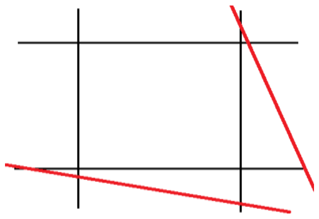

Figure 11. Discarded line segments with negative slopes.

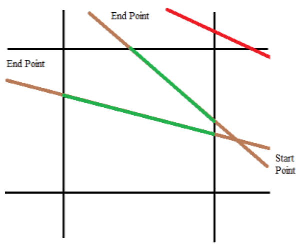

Figure 12. Part of line segments inside clipping window.

If $R_{yr} \leq y_{top}$, then the second end point is (x_{right}, R_{yr}). Otherwise it is (R_{xt}, y_{top}) as highlighted in Figure 12. The partially visible line could in turn be drawn using these two end points.

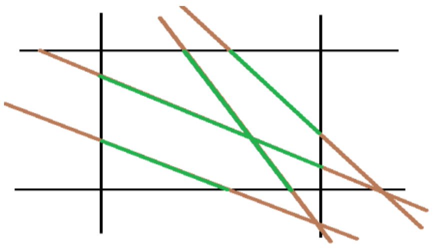

Figure 13. Line segments needing larger number of tests.

If $R_{yl} < y_{bottom}$ then line enter inside window at the point (Rxb, ybottom) by intersecting bottom edge and next point is (x_{right}, R_{yr}) if $R_{yr} \leq y_{top}$ else (R_{xt}, y_{top}).

5.3. One Line Edge is Inside Clipping Window

With small modifications as stated earlier, let (x_{in}, y_{in}) be one end point of line that lie inside clipping window and (x_{out}, y_{out}) be the other end point lying outside the window. As some part of the line lies inside the window, the possibility of complete rejection ceases to exist. For extracting the portion of the line, redefine $\Delta x = x_{out} - x_{in}$, $\Delta y = y_{out} - y_{in}$. The clipping window is now divided into four regions based on the location of the point lying inside the window. These regions are presented in Figure (14) and can be identified as:

1. Right top corner region: $\Delta x > 0$, $\Delta y \geq 0$
2. Left top corner region: $\Delta x \leq 0$, $\Delta y > 0$
3. Left bottom corner region: $\Delta x < 0$, $\Delta y \leq 0$
4. Right bottom corner region: $\Delta x \geq 0$, $\Delta y < 0$

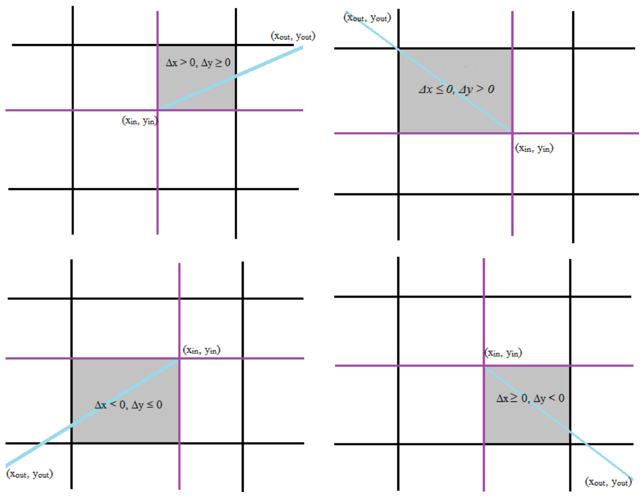

Figure 14. Various possibilities of straight lines exit from clipping window to external region.

Section of region to evaluate another end point that lie on edge will be decided with the help of positive and negative values of Δx and Δy. The various constraints that need to be checked in each case have been clearly highlighted in the various portions of Figure 14.

As defined earlier, ortho lengths are distances, from coordinate axis to intersection point between edges of clipping window and line.

$$R_x = x_{out} - \left\lfloor \frac{2\Delta x \delta y + \Delta y}{2\Delta y} \right\rfloor where\, \delta y = y_{out} - y_{edge} \quad (20)$$

$$R_y = y_{out} - \left\lfloor \frac{2\Delta y \delta x + \Delta x}{2\Delta x} \right\rfloor where\, \delta x = x_{out} - x_{edge} \quad (21)$$

δx and δy indicate horizontal and vertical distances from point that lies outside the window to corresponding intersection edge of clipping window. After identifying regions with the help of inequality conditions, corresponding ortho lengths R_x and/or R_y have been evaluated. For lines that lie in the top right corner region, the last point is (x_{right}, R_y) if $R_y \leq y_{top}$. Otherwise, it is (R_x, y_{top}). A similar procedure can be applied to compute other end point of a line for all remaining regions. In this case, it could be easily observed that a maximum of two computations R_x and R_y need to be performed. In the best case scenario, only one among R_x and R_y needs to be evaluated.

6. Proposed Algorithms

In order to consider the various cases of line clipping highlighted in section 5, a set of three mathematical algorithms have been designed. The choice of the set of algorithms to be applied depends on the constraints satisfied by the line. Magnitudes R_{xl}, R_{xr}, R_{yb}, and R_{yt} are the ortho lengths and calculated using the formulae described earlier.

Algorithm 1: Selection of algorithm and end points for the straight lines

```
Get clipping window edges $x_{left}, x_{right}, y_{bottom}$ and $y_{top}$
Collect end points of line segment $(x_1, y_1), (x_2, y_2)$
If $(y_1 < y_2)$
{
    Set $(x_{start}, y_{start}) \leftarrow (x_1, y_1)$
    Set $(x_{end}, y_{end}) \leftarrow (x_2, y_2)$
}
Else
{
    Set $(x_{start}, y_{start}) \leftarrow (x_2, y_2)$
    Set $(x_{end}, y_{end}) \leftarrow (x_1, y_1)$
}
Save entire line if it satisfies the inequalities
{
    $x_{left} \leq \{x_{start}, x_{end}\} \leq x_{right}$
    $y_{bottom} \leq y_{start}$ and $y_{end} \leq y_{top}$
}
Reject entire line if it satisfies the inequalities
{
    $y_{end} < y_{bottom}$ or $y_{start} > y_{top}$
    $\{x_{start}, x_{end}\} < x_{left}$ or $\{x_{start}, x_{end}\} > x_{right}$
}
If both end points are outside rectangular window
{   Apply second algorithm   }
Else
{   Apply third algorithm   }
```

Algorithm 2: Both End Points are Outside Window

```
Evaluate $R_{xl}, R_{xr}, R_{yb}$ and $R_{yt}$ at appropriate time
Case I: $x_{start} \leq x_{end}$ (Slope of the line is positive)
If $(R_{yl} > y_{top})$
{
    Reject line
}
Else If( $R_{yl} \geq y_{bottom}$)
{
    Set the first point $(x_{left}, R_{yl})$
    Compute next point $(x_{right}, R_{yr})$ or $(R_{xt}, y_{top})$
}
Else if $(R_{yr} < y_{bottom})$
{
    Reject line
}
Else
{
    Set the first point $(R_{xb}, y_{bottom})$
    If( $R_{yr} \leq y_{top}$)
    {   Set the last point $(x_{right}, R_{yr})$   }
    Else
    {   Set the last point $(R_{xt}, y_{top})$   }
}
Case II: $x_{start} > x_{end}$ (Slope of the line is negative)
If $(R_{yr} > y_{top})$
{
    Reject line
}
Else If $(R_{yr} \geq y_{bottom}$ )
{
    Set the first point $(x_{right}, R_{yr})$
    Compute the last point $(x_{left}, R_{yl})$ or $(R_{xt}, y_{top})$
}
Else If $(R_{yl} < y_{bottom})$
{
    Reject line
}
Else
{
    Set the first point $(R_{xb}, y_{bottom})$
    If $(R_{yl} \leq y_{top}$ )
    {
        Set the last point $(x_{left}, R_{yl})$
    }
    Else
    {
        Set the last point $(R_{xt}, y_{top})$
    }
}
```

Algorithm 3: One End Point of Line Inside Rectangular Window

```
Set $(x_{in}, y_{in})$ as inside point
Set $(x_{out}, y_{out})$ as outside point of line in clipping region
```

```
Save ($x_{in}$, $y_{in}$)
compute other end point with following procedure
If ( $\Delta x > 0$ && $\Delta y \geq 0$ && $R_{yr} \leq y_{top}$ )
{
    Set other end point ($x_{right}$, $R_{yr}$)
}
Else
{
    Set other end point ($R_{xt}$, $y_{top}$)
}
If ( $\Delta x \leq 0$ && $\Delta y > 0$ && $R_{xt} \geq x_{left}$ )
{
    Set other end point ($R_{xt}$, $y_{top}$)
}
Else
{
    Set other point ($x_{left}$, $R_{yl}$)
}
If ( $\Delta x < 0$ && $\Delta y \leq 0$ && $R_{yl} \geq y_{bottom}$ )
{
    Set other point ($x_{left}$, $R_{yl}$)
}
Else
{
    Set other point ($R_{xb}$, $y_{bottom}$)
}
If ( $\Delta x \geq 0$ && $\Delta y < 0$ && $R_{xb} \leq x_{right}$ )
{
    Set other point ($R_{xb}$, $y_{bottom}$)
}
Else
{
    Set other point ($x_{right}$, $R_{yr}$)
}
```

Once the end points are computed and if part of line segment is inside, then portion of the line can easily be drawn that lie inside clipping window. The selection of end points for a line in the third algorithm are slightly different and chosen such a way that one point is inside rectangular window and another point is outside anywhere in coordinate system.

7. Performance Analysis

The proposed algorithm has been compared with existing algorithms in graphics industry. A wide range of conclusions could be inferred.

- It has been observed that the proposed algorithm mostly depends on comparisons that need only one cycle of CPU apart from few arithmetic computations.
- Unlike other methods, no calculations are needed to evaluate end points of the saving line that lie inside clipping window.
- All presently existing algorithms are using floating point calculations whereas proposed algorithm depends only on integral values. It is observed that integral calculations are one third times faster than floating point computations.
- Time complexity does not change with the size of the window.
- LB algorithm may be straight forward but requires a large number of computations (multiplications and divisions) as compared to other methods.
- CS seems to be simplest and oldest algorithm but the number of computations grow exponentially for the points that lie on edge. Besides, there is ambiguity in the choice of upper or lower or side regions.
- NLN algorithm is developed based on slope comparisons that are stored for further computations. Hence, a huge quantity of memory is needed to handle the level of storage.
- In NLN algorithm, geometric transformations are used. This in turn leads to need for knowledge on the area initially for understanding and later for subsequent implementation. In NLN algorithm, first geometric transformation has to be applied to rotate point location and three slopes are compared. However, in the proposed algorithm, first R_{yr} is compared with inequalities $y_{bottom} \leq R_{yr} \leq y_{top}$ and the point is chosen as (x_{right}, R_{yr}).

Numerical computations are evaluated with three different cases.

1. Positive slope lines.
2. Negative slope lines.
3. Portion of line inside window apart from completely accepted and rejected lines.

Rectangular window dimensions, in this analysis are taken as (25, 25) and (75, 75). i.e., edges of clipping window are $x = 25$, $x = 75$, $y = 25$ and $y = 75$. Results are presented in tables.

Table 1 presents the results for positive slope lines, where these lines may be passed clipping window either by intersecting left or bottom clipping edges. Numerical results for negative slope lines are presented in Table 2 and clipping portion of the line and its end points are computed. Table 3 handles the results for the lines, where one point is inside clipping window and another end point is outside. Based on algorithm tests some of the lines are rejected and in some other lines a portion of the line is clipped.

Table 1. Positive slopes line segments.

S. No	**First Point**	**Last Point**	R_{yl}	R_{yr}	R_{xb}	R_{xt}	**Decision**	**Save Points**	
								First Point	**Last Point**
1	(15, 50)	(35, 150)	100	- -	- -	- -	Rejected	- -	- -
2	(00, 05)	(175, 40)	10	20	- -	- -	Rejected	- -	- -
3	(20, 35)	(100,75)	38	63	- -	- -	Accepted	(25, 38)	(75, 63)
4	(15, 30)	(85, 100)	40	90	- -	60	Accepted	(25, 40)	(60, 75)
5	(10, 10)	(110, 60)	18	43	40	- -	Accepted	(40, 25)	(75,43)
6	(20, 05)	(65, 95)	15	115	30	55	Accepted	(30, 25)	(55, 75)

Table 2. Negative slope line segments.

S. No	First Point	Last Point	R_{yr}	R_{yl}	R_{xb}	R_{xt}	Decision	Save Points	
								First Point	Last Point
1	(100, 30)	(65, 100)	80	- -	- -	- -	Rejected	--	--
2	(110, 05)	(00, 27)	12	22	- -	- -	Rejected	- -	- -
3	(90, 10)	(10, 90)	25	75	- -	- -	Accepted	(75, 25)	(25, 75)
4	(90, 20)	(50, 100)	50	150	- -	63	Accepted	(75, 50)	(63, 75)
5	(80, 00)	(00, 80)	5	55	55	- -	Accepted	(55, 25)	(25, 55)
6	(75, 00)	(20, 110)	0	100	63	38	Accepted	(63, 25)	(38, 75)

Table 3. One side of line segment is inside clipping window.

S. No	First Point	Last Point	R_y	R_x	Region	Save Points	
						In Point	Out Point
1	(50, 60)	(150, 80)	65	- -	I	(50, 60)	(75, 65)
2	(40, 60)	(70, 90)	95	55	I	(40, 60)	(55, 75)
3	(50, 40)	(00, 90)	65	- -	II	(50, 40)	(25, 65)
4	(50, 50)	(20, 110)	100	38	II	(50, 50)	(38, 75)
5	(50, 60)	(20, 30)	35	- -	III	(50, 60)	(25, 35)
6	(50, 50)	(30, 10)	00	38	III	(50, 50)	(38, 25)
7	(40, 60)	(90, 35)	43	- -	IV	(40, 60)	(75, 43)
8	(40, 50)	(80, 10)	15	65	IV	(40, 50)	(65, 75)

References

[1] Abdullah M., Pathan A., and Al Shaikhli I., "A Web and Software-Based Approach Blending Social Networks for Online Qur'anic Arabic Learning," *The International Arab Journal of Information Technology*, vol. 14, no. 1, pp. 80-90, 2017.

[2] Andreev R. and Sofianska E., "New Algorithm for Two-Dimensional Line Clipping," *Computers and Graphics*, vol. 15, no. 4, pp. 519-526, 1991.

[3] Bui D. and Skala V., "Fast Algorithms for Clipping Lines and Line Segments in E^2," *The Visual Computer*, vol. 14, no. 1, pp. 31-38, 1998.

[4] Chen X., Yong J., Wang G., Paul J., and Xu G., "Computing the Minimum Distance between a Point and a NURBS Curve," *Computer-Aided Design*, vol. 40, no. 10, pp. 1051-1054, 2008.

[5] Cyrus M. and Beck J., "Generalized Two-and Three-Dimensional Clipping," *Computers and Graphics*, vol. 3, no. 1, pp. 23-28, 1978.

[6] Day J., "A New Two Dimensional Line Clipping Algorithm for Small Windows," *Computer Graphics Forum*, vol. 11, no. 4, pp. 241-245, 1992.

[7] Day J., "An Algorithm for Clipping Lines in Object and Image Space," *Computers and Graphics*, vol. 16, no. 4, pp. 421-426, 1993.

[8] Dévai F., "Analysis of the Nicholl-Lee-Nicholl Algorithm," *in Proceedings of International Conference on Computational Science and its Applications*, Singapor, pp. 726-736, 2005.

[9] Foley J. and Van Dam A., *Fundamentals of Interactive Computer Graphics*, MA: Addison-Wesley, 1982.

[10] Gross M. and Pfister H., *Point-Based Graphics*, Morgan Kaufmann, 2011.

[11] Hearn D. and Baker P., *Computer Graphics*, Prentice Hall, 1997.

[12] Huang W., "Line Clipping Algorithm of Affine Transformation for Polygon," *in Proceedings of International Conference on Intelligent Computing*, Nanning, pp. 55-60, 2013.

[13] Hughes J. and Foley J., *Computer Graphics: Principles and Practice*, Pearson Education, 2013.

[14] Klette R. and Rosenfeld A., *Digital Geometry: Geometric Methods for Digital Picture Analysis*, Morgan Kaufmann, 2004.

[15] Liang Y. and Barsky B., "A new Concept and Method for Line Clipping," *ACM Transactions on Graphics*, vol. 3, no. 1, pp. 1-22, 1984.

[16] Liu Y., Wang X., Bao S., Gomboši M., and Žalik B., "An Algorithm for Polygon Clipping, and for Determining Polygon Intersections and Unions," *Computers and Geosciences*, vol. 33, no. 5, pp. 589-598, 2007.

[17] Lu G., Wu X., and Peng Q., "An Efficient Line Clipping Algorithm Based on Adaptive Line Rejection," *Computers and Graphics*, vol. 26, no. 3, pp. 409-415, 2002.

[18] Mamatha E., Reddy C., and Anand S., "Focal Point Computation and Homogeneous Geometrical Transformation for Linear Curves," *Perspectives in Science*, vol. 8, pp. 19-21, 2016.

[19] Mamatha E., Reddy C., and Prasad K., "Anti Aliased Digital Pixel Plotting for Raster Scan Lines Using Area Evaluation," *in Proceedings of Emerging Research in Computing, Information, Communication and Applications*, Singapore, pp. 461-468, 2016.

[20] Kumar M., Mamatha V., Reddy C., Mukesh V., and Reddy R., "Data hiding with Dual Based Reversible Image Using Sudoku Technique," *in Proceedings of International Conference on Advances in Computing, Communications and Informatics*, Udupi, pp. 2166-2172, 2017.

[21] Nicholl T., Lee D., and Nicholl R., "An Efficient New Algorithm for 2-D Line Clipping: its Development and Analysis," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 253-262, 1987.

[22] Reddy C., Janani1 B., Narayanan S., and Mamatha E., "Obtaining Description for Simple Images using Surface Realization Techniques and Natural Language Processing," *Indian Journal of Science and Technology*, vol. 9, no. 22, pp. 1-7, 2016.

[23] Rogers D. and Rybak L., "On an Efficient General Line-Clipping Algorithm," *Computer Graphics and Applications*, vol. 5, no. 1, pp. 82-86, 1985.

[24] Schneider P. and Eberly D., *Geometric Tools for Computer Graphics*, Morgan Kaufmann, 2002.

[25] Skala V., "A New Approach to Line and Line Segment Clipping in Homogeneous

Coordinates," *The Visual Computer*, vol. 21, no. 11, pp. 905-914, 2005.

[26] Slater M. and Barsky B., "2D line and Polygon Clipping Based on Space Subdivision," *The Visual Computer*, vol. 10, no. 7, pp. 407-422, 1994.

[27] Wang J., Cui C., and Gao j., "An Efficient Algorithm for Clipping Operation Based on Trapezoidal Meshes and Sweep-Line Technique," *Advances in Engineering Software*, vol. 47, no. 1, pp. 72-79, 2012.

[28] Weng L., Daman D., and Rahim M., "Shadow Casting with Stencil Buffer for Real-Time Rendering," *The International Arab Journal of Information Technology*, vol. 5, no. 4, pp. 102-110, 2008.

[29] Wu Q., Huang X., and Han Y., "A clipping Algorithm for Parabola Segments against Circular Windows," *Computers and Graphics*, vol. 30, no. 4, pp. 540-560, 2006.

[30] Zhang M. and Sabharwal C., "An Efficient Implementation of Parametric Line and Polygon Clipping Algorithm," *in Proceedings of the ACM Symposium on Applied Computing*, Madrid, pp. 796-800, 2002.

Mamatha Elliriki is working at GITAM University, Bangalore Campus. She is presently working on graphical models, digital geometry, performance of various processing systems. She is published good number of research articles in peer reviewed and indexed journals and international conferences.

Chandrasekhara Reddy has published several scientific papers in reputed journals and conferences. He is working as professor in Cambridge Institute of Technology–North Campus, Bangalore. His current areas of research include performance analysis of Markov models.

Krishna Anand is currently working as professor in the Department of Computer Science in Sreenidhi Institute of Science and Technology, Hyderabad. He has rich teaching experience in different universities apart from the industrial experience. His areas of interest include Simulation, Soft Computing, Graphics and Expert Systems.