A New Concept and Method for Line Clipping

YOU-DONG LIANG and BRIAN A. BARSKY
University of California, Berkeley

A new concept and method for line clipping is developed that describes clipping in an exact and mathematical form. The basic ideas form the foundation for a family of algorithms for two-dimensional, three-dimensional, and four-dimensional (homogeneous coordinates) line clipping. The line segment to be clipped is mapped into a parametric representation. From this, a set of conditions is derived that describes the interior of the clipping region. Observing that these conditions are all of similar form, they are rewritten such that the solution to the clipping problem is reduced to a simple max/min expression. For each dimension, the mathematics are discussed, an example is given, the algorithm is designed, and a performance test is conducted. The new algorithm is compared with the traditional Sutherland–Cohen clipping algorithm. Using randomly generated data, the new algorithm showed a 36 percent, 40 percent, and 79 percent improvement for two-dimensional, three-dimensional, and four-dimensional clipping, respectively. One of the advantages of this algorithm is the quick rejection of line segments that are invisible. In addition, this algorithm can be easily generalized for clipping against any convex viewing volume.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture Image Generation—*display algorithms*; I.3.4 [**Computer Graphics**]: Graphics Utilities—*software support*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*geometric algorithms, languages and systems*

General Terms: Algorithms

Additional Key Words and Phrases: Clipping, line clipping

1. INTRODUCTION

Clipping is the process of removing that portion of an image that lies outside a region called the *clip window* [5]. In the context of vector graphics, an image is composed of straight line segments, and thus the type of clipping considered in this paper is *line clipping.*

We develop a new concept and method for line clipping that describes clipping in an exact and mathematical form. The basic ideas form the foundation for a family of algorithms for two-dimensional, three-dimensional and four-dimensional (homogeneous coordinates) line clipping.

Authors' current addresses: Y.-D. Liang, Department of Mathematics, Zhejiang University, Hangzhou, Zhejiang, People's Republic of China; B. A. Barsky, Berkeley Computer Graphics Laboratory, Computer Science Division, University of California, Berkeley CA 94720.

The line segment to be clipped is mapped into a parametric representation. From this, a set of conditions is derived that describes the interior of the clipping region. Observing that these conditions are all of similar form, they are rewritten such that the solution to the clipping problem is reduced to a simple max/min expression.

One of the advantages of this algorithm is the quick rejection of line segments that are invisible. In addition, although this algorithm is presented for perspective viewing volumes, it should be noted that it is equally appropriate for nonperspective viewing volumes, and more generally, for any convex viewing volume.

2. TWO-DIMENSIONAL CLIPPING

2.1 Explanation

In two-dimensional clipping, lines are clipped against a two-dimensional region called the *clip window*. In particular, the interior of the clip window can be expressed by the following inequalities (Figure 1):

$$\begin{aligned} x_{\text{left}} \le x \le x_{\text{right}} \\ y_{\text{bottom}} \le y \le y_{\text{top}}. \end{aligned} \tag{2.1}$$

The line segment to be clipped can be mapped into a parametric representation as follows. Let the endpoints of the line segment be $\mathbf{V}_0$ and $\mathbf{V}_1$ with coordinates (x_0, y_0) and (x_1, y_1), respectively. Then the parametric representation of the line is given by

$$\begin{aligned} x &= x_0 + \Delta x\, t \\ y &= y_0 + \Delta y\, t \end{aligned} \tag{2.2}$$

where

$$\begin{aligned} \Delta x &= x_1 - x_0 \\ \Delta y &= y_1 - y_0. \end{aligned} \tag{2.3}$$

The parametric values corresponding to $\mathbf{V}_0$ and $\mathbf{V}_1$ are $t = 0$ and $t = 1$, respectively. As t varies from $t = 0$ to $t = 1$, the line segment $\mathbf{V}_0\mathbf{V}_1$ is traced out from $\mathbf{V}_0$ to $\mathbf{V}_1$; allowing $-\infty < t < \infty$ generates the line of infinite extent.

Substituting the parametric representation given by eqs. (2.2) and (2.3) into inequalities (2.1) yields the following conditions for the part of the extended line that is inside the interior of the clip window:

$$\begin{aligned} -\Delta x\, t &\le x_0 - x_{\text{left}} \quad \text{and} \quad \Delta x\, t \le x_{\text{right}} - x_0 \\ -\Delta y\, t &\le y_0 - y_{\text{bottom}} \quad \text{and} \quad \Delta y\, t \le y_{\text{top}} - y_0. \end{aligned} \tag{2.4}$$

Note that these conditions (2.4) are inequalities describing the *interior* of the clip window rather than equations defining its *boundary*.

Observe that inequalities (2.4) are all of similar form; thus, they can be rewritten as

$$p_i\, t \le q_i \qquad \text{for} \quad i = 1, 2, 3, 4 \tag{2.5}$$

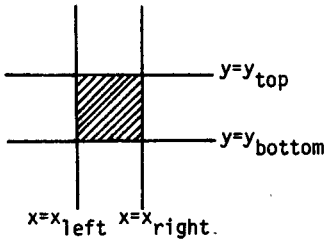
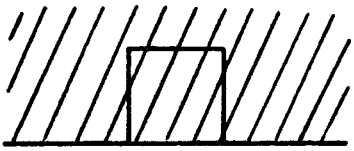


Fig. 1. The interior of the clip window.

Fig. 2. The visible side of a boundary line.



where the following notation is used:

$$\begin{aligned} p_1 &= -\Delta x, & q_1 &= x_0 - x_{\text{left}} \\ p_2 &= \Delta x; & q_2 &= x_{\text{right}} - x_0 \\ p_3 &= -\Delta y; & q_3 &= y_0 - y_{\text{bottom}} \\ p_4 &= \Delta y; & q_4 &= y_{\text{top}} - y_0. \end{aligned} \tag{2.6}$$

To establish a geometrical interpretation of these four inequalities, extend each of the four boundary line segments of the clip window to be a line of infinite extent. Each of these lines divides the plane into two regions; we define the *visible side* to be that side on which the clip window lies, as shown in Figure 2. In this way, the clip window can be defined as that region that is on the visible side of all the boundary lines. From this, it can be seen that each of the inequalities (2.5) corresponds to one of these boundary lines (left, right, bottom, and top, respectively), and describes its visible side. To be more specific, extend $\mathbf{V}_0\mathbf{V}_1$ to be a line of infinite extent. Then each inequality provides the range of values of the parameter t for which this extended line is on the visible side of the corresponding boundary line. Furthermore, the particular parametric value for the point of intersection is $t = q_i/p_i$. Also, the sign of q_i indicates on which side of the corresponding boundary line the point $\mathbf{V}_0$ lies. Specifically, if $q_i \geq 0$, then $\mathbf{V}_0$ is on the visible side of the boundary line (inclusive), and if $q_i < 0$, then $\mathbf{V}_0$ is on the invisible side.

Considering the p_i in eq. (2.6), it is clear that each can be negative, positive, or zero. If p_i is negative, the ith inequality becomes

$$t \geq q_i/p_i. \tag{2.7}$$

Observing that the range of parametric values for which the extended line is on the visible side of the corresponding boundary line is at a minimum at the point of intersection, the direction defined by $\overrightarrow{\mathbf{V}_0\mathbf{V}_1}$ is from the invisible side to the visible side of the boundary line.

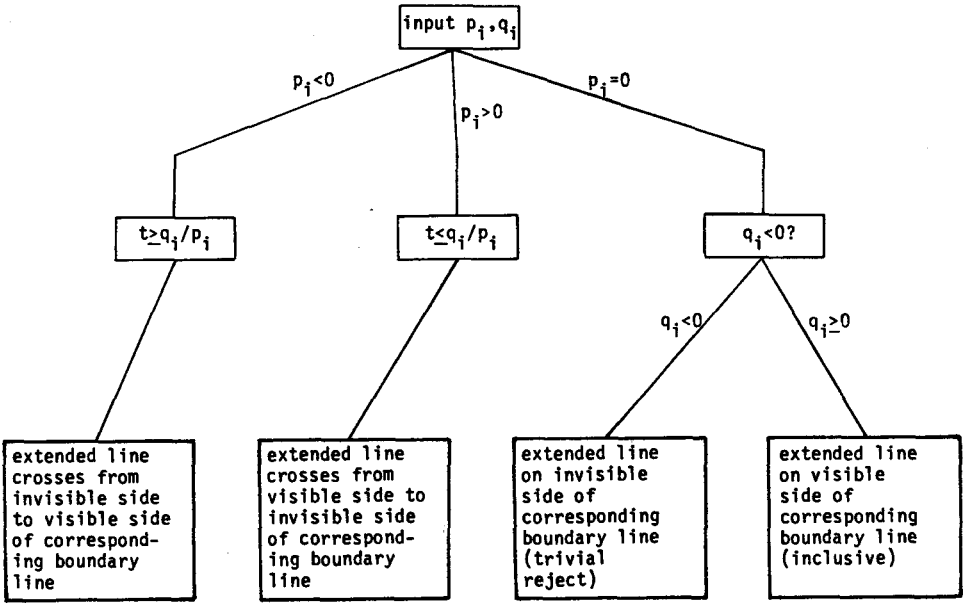


Fig. 3. Flowchart showing the geometrical interpretations for the p_i and q_i values.

Analogously, if p_i is positive, the ith inequality becomes

$$t \leq q_i/p_i. \tag{2.8}$$

Since the parametric range is at a maximum at the point of intersection, the direction defined by $\overrightarrow{\mathbf{V}_0\mathbf{V}_1}$ must be from the visible side to the invisible side of the corresponding boundary line.

Finally, if p_i is zero, the ith inequality becomes

$$0 \leq q_i. \tag{2.9}$$

Note that this is now independent of t. Thus, the inequality is satisfied for all t if $q_i \geq 0$, and there is no solution if $q_i < 0$. Geometrically, if p_i is zero, there is no point of intersection of the extended line with the corresponding boundary line; rather, they are parallel. Furthermore, if $q_i \geq 0$, then in this case the entire extended line is on the visible side of the boundary line (inclusive); if $q_i < 0$, then it is entirely on the invisible side of the line. In the former case, there may or may not be a visible segment depending upon where $\mathbf{V}_0$ and $\mathbf{V}_1$ lie on the extended line. However, the latter case cannot have a visible segment since the extended line lies outside the visible region, and hence this is a trivial reject case. All these cases are summarized in the flowchart shown in Figure 3.

Now, the intersection of the four inequalities gives the range of values of the parameter t for which the extended line is inside the clip window. However, the line segment to be clipped ($\mathbf{V}_0\mathbf{V}_1$) is only part of the extended line and is described by $0 \leq t \leq 1$. Hence, the solution to the two-dimensional clipping problem is equivalent to the inequalities (2.5) under the condition $0 \leq t \leq 1$.

Solving these inequalities is actually a max/min problem, which can be seen as follows. Recall that $t \geq q_i/p_i$ for all i such that $p_i < 0$, and also that $t \geq 0$. Thus,

$$t \geq \max(\{q_i/p_i \mid p_i < 0,\ i = 1, 2, 3, 4\} \cup \{0\}). \tag{2.10}$$

Analogously, $t \leq q_i/p_i$ for all i such that $p_i > 0$, and also $t \leq 1$. Thus,

$$t \leq \min(\{q_i/p_i \mid p_i > 0,\ i = 1, 2, 3, 4\} \cup \{1\}). \tag{2.11}$$

Finally, if $p_i = 0$ in the ith inequality for some i, then there are two possibilities. If $q_i \geq 0$, then there is no useful information to be gleaned from this inequality, and this inequality can be discarded. If $q_i < 0$, then this is a trivial reject case, and the clipping problem is solved with no further computation needed.

The right-hand sides of inequalities (2.10) and (2.11) are the values of the parameter t corresponding to the beginning and end of the visible segment, respectively (assuming there is a visible segment).

Denoting these parametric values as t_0 and t_1,

$$\begin{aligned} t_0 &= \max(\{q_i/p_i \mid p_i < 0,\ i = 1, 2, 3, 4\} \cup \{0\}) \\ t_1 &= \min(\{q_i/p_i \mid p_i > 0,\ i = 1, 2, 3, 4\} \cup \{1\}). \end{aligned} \tag{2.12}$$

If there is a visible segment, it correspnds to the parametric interval

$$t_0 \leq t \leq t_1. \tag{2.13}$$

Hence, a necessary condition for a line segment to be at least partially visible is

$$t_0 \leq t_1. \tag{2.14}$$

This is not a sufficient condition because it ignores the possibility of a trivial reject due to $p_i = 0$ with $q_i < 0$. Nonetheless, this yields a sufficient condition for rejection; specifically, if $t_0 > t_1$, then this is another reject case. The algorithm checks if $p_i = 0$ with $q_i < 0$ or if $t_0 > t_1$, in which case the line segment is immediately rejected without further computation.

In the algorithm, t_0 and t_1 are intialized to 0 and 1, respectively. Then each ratio q_i/p_i is considered successively. If $p_i < 0$, the ratio is first compared with t_1, and if it is greater, then t_0 will exceed t_1 and this must be a reject case; otherwise, it is compared with t_0, and if it is greater, then t_0 must be updated with this new value. If $p_i > 0$, the ratio is first compared with t_0, and if it is less, then this is a reject case; otherwise it is compared with t_1, and if it is less, then t_1 is updated. Finally, if $p_i = 0$ and $q_i < 0$, then this is a reject case. At the last stage of the algorithm, if the line segment was not rejected, the parametric values t_0 and t_1 are used to compute the corresponding points. However, if $t_0 = 0$, then the endpoint is $\mathbf{V}_0$ and need not be computed; similarly, if $t_1 = 1$, then the endpoint is $\mathbf{V}_1$.

The geometric meaning of this process is that the line segment is "squeezed" down by considering where the extended line intersects each boundary line. More specifically, each endpoint of the given line segment $\mathbf{V}_0\mathbf{V}_1$ is used as an initial value for an endpoint of the visible segment $\mathbf{V}_0'\mathbf{V}_1'$. Then the point of intersection of the extended line with each boundary line is considered. (This visibility

calculation with respect to each boundary line corresponds to an invocation of the *clipt* procedure in the algorithm.) If, for the current boundary line, the direction defined by $\overrightarrow{V_0V_1}$ is from the invisible side to the visible side of the line, this point of intersection is first compared with V_1'. If the point is further along the line, then V_1' (and thus $V_0'V_1'$) must be on the invisible side of the line, and thus the line segment is rejected. Otherwise, the point of intersection is compared with V_0'; if the point is further along the line, then V_0' is moved forward to this point. On the other hand, if the direction is from the visible side to the invisible side of the line, this point of intersection is first compared with V_0'. If V_0' is further along the line than the point, then V_0' (and thus $V_0'V_1'$) must be on the invisible side of the line, and thus the line segment is rejected. Otherwise, the point of intersection is compared with V_1', and if V_1' is further along the line than the point, then V_1' is moved back to this point. Finally, if the extended line is parallel to the boundary line and on its invisible side, the line segment is rejected. At the end of the algorithm, if the line segment was not rejected, V_0' and V_1' are the required endpoints of the visible segment.

On a final mathematical note, although it is not necessary for the algorithm, inequality (2.14) can be modified to be a necessary and sufficient condition for a line segment to be at least partially visible. This requires the inclusion of the $p_i = 0$ case (line segment parallel to the corresponding boundary line). This case can be combined with that for $p_i < 0$ if q_i/p_i is defined as follows:

$$\frac{q_i}{p_i} = \begin{cases} -\infty & \text{if } p_i = 0 \text{ and } q_i < 0, \\ +\infty & \text{if } p_i = 0 \text{ and } q_i \geq 0. \end{cases} \tag{2.15}$$

Then the necessary and sufficient condition is

$$t_0 \leq t_1^* \tag{2.16}$$

where

$$t_1^* = \min(\{q_i/p_i \mid p_i \geq 0,\ i = 1, 2, 3, 4\} \cup \{1\}). \tag{2.17}$$

2.2 Example

Consider the following example: let V_0 $(-1, -2)$ and V_1 $(2, 4)$ be the endpoints of a line segment and let $x_{left} = 0$, $x_{right} = 1$, $y_{bottom} = 0$, and $y_{top} = 1$ define the boundary lines of the clip window. Then

$$\Delta x = x_1 - x_0 = 3; \quad \Delta y = y_1 - y_0 = 6 \tag{2.18}$$

and

$$\begin{aligned} &p_1 = -\Delta x = -3; && q_1 = x_0 - x_{left} = -1; && q_1/p_1 = 1/3 \\ &p_2 = \Delta x = 3; && q_2 = x_{right} - x_0 = 2; && q_2/p_2 = 2/3 \\ &p_3 = -\Delta y = -6; && q_3 = y_0 - y_{bottom} = -2; && q_3/p_3 = 1/3 \\ &p_4 = \Delta y = 6; && q_4 = y_{top} - y_0 = 3; && q_4/p_4 = 1/2. \end{aligned} \tag{2.19}$$

Compute t_0 and t_1:

$$\begin{aligned} t_0 &= \max(\{q_i/p_i \mid p_i < 0,\ i = 1, 2, 3, 4\} \cup \{0\}) = \max(1/3, 1/3, 0) = 1/3 \\ t_1 &= \min(\{q_i/p_i \mid p_i > 0,\ i = 1, 2, 3, 4\} \cup \{1\}) = \min(2/3, 1/2, 1) = 1/2. \end{aligned} \tag{2.20}$$

Since $t_0 < t_1$, there is a visible segment having endpoints:

$$t_0 = 1/3: \quad x_0' = -1 + (3)1/3 = 0$$
$$y_0' = -2 + (6)1/3 = 0$$
$$t_1 = 1/2: \quad x_1' = -1 + (3)1/2 = 1/2$$
$$y_1' = -2 + (6)1/2 = 1.$$

2.3 Algorithm

The algorithm to perform two-dimensional line clipping is as follows:

```
(*clip 2d line segment with endpoints (x0, y0) and (x1, y1)*)
(*the clip window is xleft <= x <= xright and ybottom <= y <= ytop*)

procedure clip2d (var x0, y0, x1, y1: real);
var t0, t1: real;
  deltax, deltay: real;

function clipt ((*value*) p, q: real; var t0, t1; real): Boolean;
var r: real;
  accept: Boolean;
begin (*clipt*)
  accept := true;
  if p < 0 then begin
    r := q/p;
    if r > t1 then accept := false (*set up to reject*)
    else if r > t0 then t0 := r
  end else if p > 0 then begin
    r := q/p;
    if r < t0 then accept := false (*set up to reject*)
    else if r < t1 then t1 := r
  end else (*p = 0*) if q < 0 then accept := false; (*set up to reject*)
  clipt := accept
end (*clipt*);

begin (*clip2d*)
  t0 := 0;
  t1 := 1;
  deltax := x1 - x0;
  if clipt (- deltax, x0 - xleft, t0, t1) then
    if clipt (deltax, xright - x0, t0, t1) then begin
      deltay := y1 - y0;
      if clipt (- deltay, y0 - ybottom, t0, t1) then
        if clipt (deltay, ytop - y0, t0, t1) then begin
          if t1 < 1 then begin
            x1 := x0 + t1 * deltax;
            y1 := y0 + t1 * deltay
          end;
          if t0 > 0 then begin
            x0 := x0 + t0 * deltax;
            y0 := y0 + t0 * deltay
          end
        end
    end
end (*clip2d*);
```

2.4 Performance Test

The computational complexity of the algorithm is linear in the number of line segments to be clipped. The exact requirements are dependent upon the particular data. In general, the worst case occurs when a line segment is not trivially rejected, is not parallel to any of the boundary lines, and has both of its original endpoints outside the clip window. In this case, two intersection points must be computed. This requires a total of 12 additions/subtractions, 4 multiplications, and 4 divisions.

This algorithm was compared with the traditional Sutherland–Cohen two-dimensional line clipping algorithm. Pascal code for the latter was copied verbatim from [5, pp. 66–67]. Both were executed on 4 different sets of data, each containing 1000 randomly generated line segments. Each line segment was clipped 1000 times to reduce the effects of random variation, resulting in a total of 4 million clips. The new algorithm required an average of 333 seconds, while the traditional one used an average of 519 seconds (a 36 percent improvement) on a DEC VAX-11/780 with a floating point accelerator in Pascal under Berkeley UNIX.

3. THREE-DIMENSIONAL CLIPPING

3.1 Explanation

For three-dimensional clipping, lines are clipped against a *viewing pyramid.* In the *eye* coordinate system (observer at the origin), this volume is described by the following two pairs of inequalities:

$$\begin{aligned} -z_e \le \cot\left(\frac{\theta_x}{2}\right) x_e \le z_e \\ -z_e \le \cot\left(\frac{\theta_y}{2}\right) y_e \le z_e \end{aligned} \tag{3.1}$$

where θ_x and θ_y are the horizontal and vertical angles of view, respectively (see Figure 4). Note that either pair of inequalities (3.1) implies $-z_e \le z_e$ which implies $z_e \ge 0$.

From inequalities (3.1), a more natural coordinate system suggests itself for clipping. This transformation is given by

$$\begin{aligned} x &= \cot\left(\frac{\theta_x}{2}\right) x_e \\ y &= \cot\left(\frac{\theta_y}{2}\right) y_e \\ z &= z_e. \end{aligned} \tag{3.2}$$

Then the inequalities (3.1) become

$$\begin{aligned} -z \le x \le z \\ -z \le y \le z \end{aligned} \tag{3.3}$$

and again $z \ge 0$ is implied.

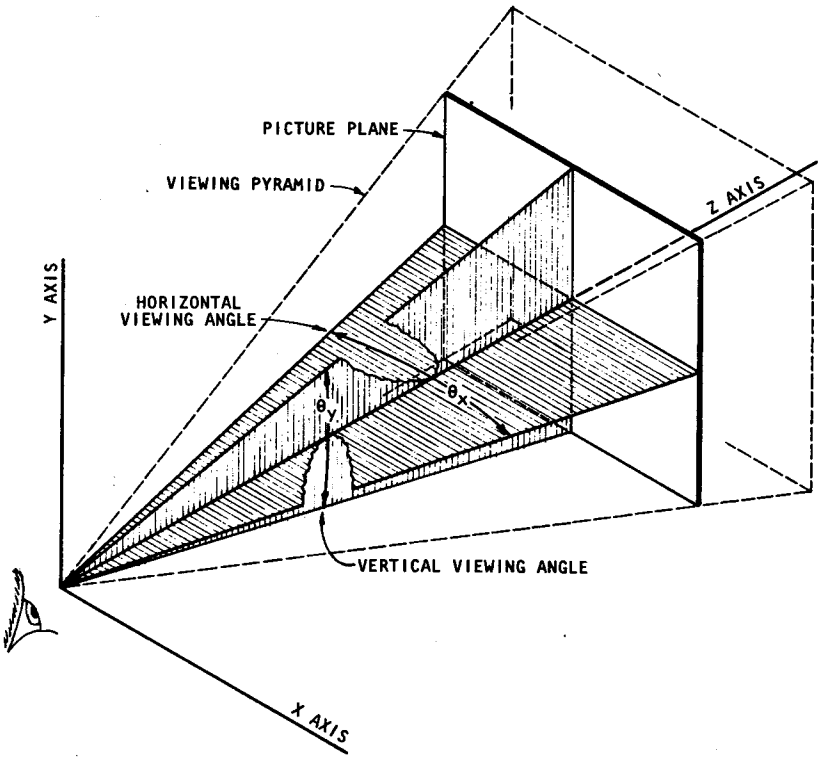


Fig. 4. The viewing pyramid in the eye coordinate system.

Geometrically, eq. (3.2) scales the viewing pyramid so that it becomes a right pyramid in the *clipping* coordinate system, as described by inequalities (3.3) (Figure 5).

Analogous to the method developed in Section 2, the line segment to be clipped can be mapped into a parametric representation. If the endpoints of the line segment are $\mathbf{V}_0$ and $\mathbf{V}_1$ with coordinates (x_0, y_0, z_0) and (x_1, y_1, z_1), respectively, then the parametric representation of the line is given by eqs. (2.2) and (2.3) with the addition of the following equation:

$$z = z_0 + \Delta z\, t \tag{3.4}$$

where

$$\Delta z = z_1 - z_0. \tag{3.5}$$

Substituting this parametric representation into inequalities (3.3) yields the following conditions for the part of the extended line that is inside the *volume* of the clip window:

$$\begin{aligned} -(\Delta x + \Delta z)t \le x_0 + z_0 \quad &\text{and} \quad (\Delta x - \Delta z)t \le z_0 - x_0 \\ -(\Delta y + \Delta z)t \le y_0 + z_0 \quad &\text{and} \quad (\Delta y - \Delta z)t \le z_0 - y_0. \end{aligned} \tag{3.6}$$

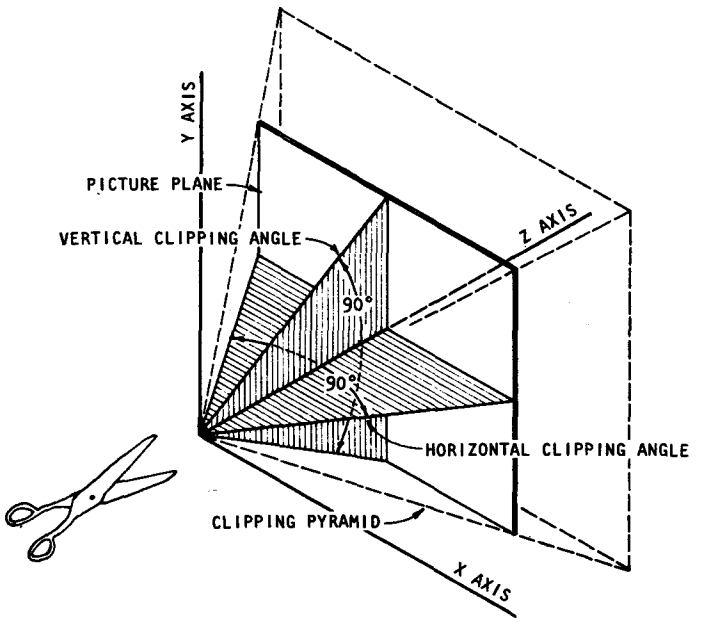


Fig. 5. The viewing pyramid in the clipping coordinate system.

The inequalities (3.6) can be rewritten in the form of inequalities (2.5). Specifically,

$$p_i\, t \le q_i \qquad \text{for} \quad i = 1, 2, 3, 4 \tag{3.7}$$

where the following notation is used:

$$\begin{aligned} p_1 &= -(\Delta x + \Delta z); & q_1 &= x_0 + z_0; & &\text{(left)} \\ p_2 &= \Delta x - \Delta z; & q_2 &= z_0 - x_0; & &\text{(right)} \\ p_3 &= -(\Delta y + \Delta z); & q_3 &= y_0 + z_0; & &\text{(bottom)} \\ p_4 &= \Delta y - \Delta z; & q_4 &= z_0 - y_0; & &\text{(top).} \end{aligned} \tag{3.8}$$

The geometrical interpretation of these four inequalities is analogous to that given for the two-dimensional case. Extend each of the four bounding sides of the viewing pyramid to be an (infinite) plane. Then, each of these bounding planes divides three space into two regions. The *visible side* is the side on which the viewing pyramid lies. In this way, the viewing pyramid can be defined as that volume that is on the visible side of all the bounding planes. From this, it can be seen that each of the inequalities (3.7) corresponds to one of these planes (left, right, bottom, and top, respectively) and describes its visible side. To be more specific, extend $\mathbf{V}_0\mathbf{V}_1$ to be a line of infinite extent. Then each inequality provides the range of values of the parameter t for which this extended line is on the visible side of the corresponding plane. Furthermore, the particular parametric value for the point of intersection is $t = q_i/p_i$. Also, the sign of q_i indicates on which side of the corresponding bounding plane the point $\mathbf{V}_0$ lies. Specifically, if $q_i \ge 0$, then $\mathbf{V}_0$ is on the visible side of the bounding plane (inclusive), and if $q_i < 0$, then $\mathbf{V}_0$ is on the invisible side.

Now, it is readily verified that three of the p_i coefficients are linearly independent since they satisfy only one linear relation:

$$p_1 + p_2 - p_3 - p_4 = 0.$$

This equation can be solved for any one of the coefficients in terms of the other three independent ones. Thus, there is no restriction on the sign of any of the four p_i.

If p_i is negative, the ith inequality becomes

$$t \geq q_i/p_i. \tag{3.9}$$

Since the range of parametric values for which the extended line is on the visible side of the corresponding plane is at a minimum at the point of intersection, the direction defined by $\overrightarrow{\mathbf{V}_0\mathbf{V}_1}$ is from the invisible side to the visible side of the plane.

Analogously, if p_i is positive, the ith inequality becomes

$$t \leq q_i/p_i \tag{3.10}$$

and the direction defined by $\overrightarrow{\mathbf{V}_0\mathbf{V}_1}$ is from the visible side to the invisible side of the plane.

Finally, if p_i is zero, the ith inequality becomes

$$0 \leq q_i. \tag{3.11}$$

Since inequality (3.11) is independent of t, it is satisfied for all t if $q_i \geq 0$, and there is no solution if $q_i < 0$. Geometrically, if p_i is zero, there is no point of intersection of the extended line with the corresponding plane; rather, the extended line is parallel to the plane. Furthermore, if $q_i \geq 0$, then in this case the entire extended line is on the visible side of the plane (inclusive); if $q_i < 0$, then it is entirely on the invisible side of the plane. In the former case, there may or may not be a visible segment depending on where $\mathbf{V}_0$ and $\mathbf{V}_1$ lie on the extended line. However, the latter case cannot have a visible segment since the extended line lies outside the visible region and hence this is a trivial reject case.

Now the intersection of the four inequalities gives the range of values of the parameter t for which the extended line is inside the viewing pyramid. However, the line segment to be clipped ($\mathbf{V}_0\mathbf{V}_1$) is only part of the extended line and is described by $0 \leq t \leq 1$. Hence, the solution to the three-dimensional clipping problem is equivalent to the inequalities (3.7) under the condition $0 \leq t \leq 1$. The algorithm to accomplish this is identical to that developed in Section 2, with the exception that the definitions of the p_i and q_i are given by eq. (3.8).

This can be extended to cover the case of a finite viewing volume. The viewing pyramid is truncated by *hither* and *yon* clipping planes to form a *frustum of vision* (Figure 6). Note that the location of these planes is completely independent of the position of the picture plane. This requires the addition of a constraint for z. If the hither and yon clipping planes are $z = h$ and $z = f$, respectively, then this constraint is

$$h \leq z \leq f. \tag{3.12}$$

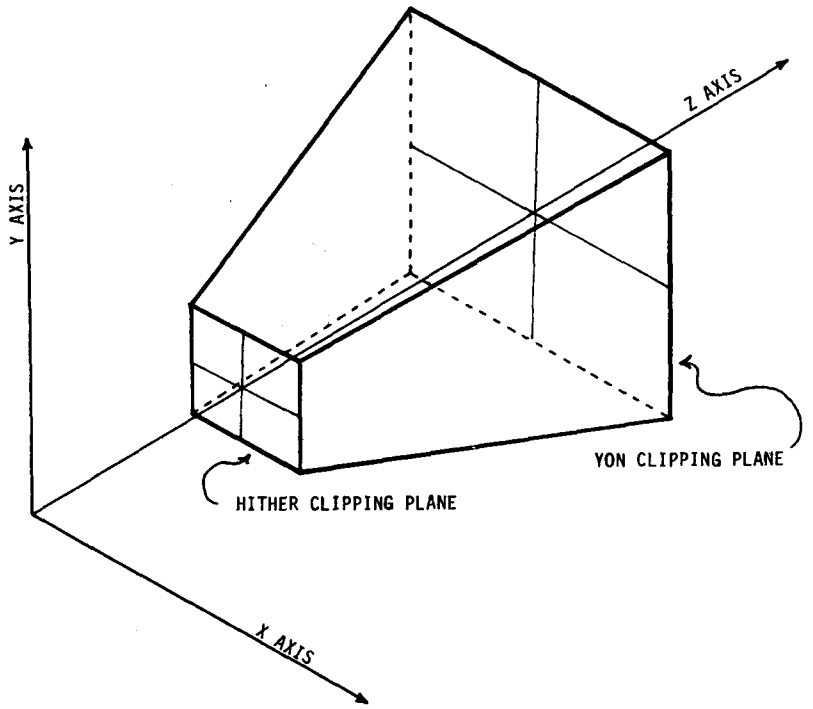


Fig. 6. The frustum of vision.

Substituting the parametric representation for z given by eqs. (3.4) and (3.5) yields the following additional two constraints:

$$-\Delta z\, t \leq z_0 - h \tag{3.13}$$

and

$$\Delta z\, t \leq f - z_0. \tag{3.14}$$

These inequalities can be rewritten in the form of inequalities (3.7) where

$$\begin{aligned} p_5 &= -\Delta z; \qquad q_5 = z_0 - h \\ p_6 &= \Delta z; \qquad q_6 = f - z_0. \end{aligned} \tag{3.15}$$

The geometric meaning of the three-dimensional clipping process is that the line segment is "squeezed" down by considering where the extended line intersects each bounding plane. More specifically, each endpoint of the given line segment $\mathbf{V}_0\mathbf{V}_1$ is used as an initial value for an endpoint of the visible segment $\mathbf{V}_0'\mathbf{V}_1'$. Then the point of intersection of the extended line with each bounding plane is considered. If, for the current bounding plane, the direction defined by $\overrightarrow{\mathbf{V}_0\mathbf{V}_1}$ is from the invisible side to the visible side of the plane, this point of intersection is first compared with $\mathbf{V}_1'$. If the point is further along the line, then $\mathbf{V}_1'$ (and thus $\mathbf{V}_0'\mathbf{V}_1'$) must be on the invisible side of the plane, and thus the line segment is rejected. Otherwise, the point of intersection is compared with $\mathbf{V}_0'$, and if the point is further along the line, then $\mathbf{V}_0'$ is moved forward to this point. On the

other hand, if the direction is from the visible side to the invisible side of the plane, this point of intersection is first compared with $\mathbf{V}_0'$. If $\mathbf{V}_0'$ is further along the line than the point, then $\mathbf{V}_0'$ (and thus $\mathbf{V}_0'\mathbf{V}_1'$) must be on the invisible side of the plane, and thus the line segment is rejected. Otherwise, the point of intersection is compared with $\mathbf{V}_1'$, and if $\mathbf{V}_1'$ is further along the line than the point, then $\mathbf{V}_1'$ is moved back to this point. Finally, if the extended line is parallel to the bounding plane and on its invisible side, the line segment is rejected. At the end of the algorithm, if the line segment was not rejected, $\mathbf{V}_0'$ and $\mathbf{V}_1'$ are the required endpoints of the visible segment.

3.2 Example

Consider the following example: let $\mathbf{V}_0$ (0, 2, 1) and $\mathbf{V}_1$ (−2, −3, 3) be the endpoints of a line segment to be clipped against a viewing pyramid. Then

$$\Delta x = x_1 - x_0 = -2; \qquad \Delta y = y_1 - y_0 = -5; \qquad \Delta z = z_1 - z_0 = 2 \tag{3.16}$$

and

$$\begin{aligned} p_1 &= -\Delta x - \Delta z = 0; & q_1 &= x_0 + z_0 = 1; & q_1/p_1 &= +\infty \\ p_2 &= \Delta x - \Delta z = -4; & q_2 &= z_0 - x_0 = 1; & q_2/p_2 &= -1/4 \\ p_3 &= -\Delta y - \Delta z = 3; & q_3 &= y_0 + z_0 = 3; & q_3/p_3 &= 1 \\ p_4 &= \Delta y - \Delta z = -7; & q_4 &= z_0 - y_0 = -1; & q_4/p_4 &= 1/7. \end{aligned} \tag{3.17}$$

Compute t_0 and t_1^*:

$$t_0 = \max(\{q_i/p_i \mid p_i < 0,\ i = 1, 2, 3, 4\} \cup \{0\}) = \max(-1/4, 1/7, 0) = 1/7$$

$$t_1^* = \min(\{q_i/p_i \mid p_i \geq 0,\ i = 1, 2, 3, 4\} \cup \{1\}) = \min(+\infty, 1, 1) = 1.$$

Since $t_0 \leq t_1^*$, there is a visible segment; its endpoints are

$$\begin{aligned} t_0 = 1/7: \quad x_0' &= 0 + (-2)1/7 = -2/7 & t_1^* = 1: \quad x_1' &= x_1 = -2 \\ y_0' &= 2 + (-5)1/7 = 9/7; & y_1' &= y_1 = -3 \\ z_0' &= 1 + (2)1/7 = 9/7; & z_1' &= z_1 = 3. \end{aligned}$$

3.3 Algorithm

The algorithm to perform three-dimensional line clipping is as follows:

```
(*clip 3d line segment with endpoints (x0, y0, z0) and (x1, y1, z1)*)
(*the viewing volume is a right pyramid with its apex at the origin*)
procedure clip3d (var x0, y0, z0, x1, y1, z1: real);
var t0, t1: real;
  deltax, deltay, deltaz: real;
begin (*clip3d*)
  t0 := 0;
  t1 := 1;
  deltax := x1 − x0;
  deltaz := z1 − z0;
```

```
  if clipt (−deltax − deltaz, x0 + z0, t0, t1) then
    if clipt (deltax − deltaz, z0 − x0, t0, t1) then begin
      deltay := y1 − y0;
      if clipt (−deltay − deltaz, y0 + z0, t0, t1) then
        if clipt (deltay − deltaz, z0 − y0, t0, t1) then begin
          if t1 < 1 then begin
            x1 := x0 + t1 * deltax;
            y1 := y0 + t1 * deltay;
            z1 := z0 + t1 * deltaz
          end;
          if t0 > 0 then begin
            x0 := x0 + t0 * deltax;
            y0 := y0 + t0 * deltay;
            z0 := z0 + t0 * deltaz
          end
        end
    end
end (*clip3d*);
```

where **function** *clipt* is as was defined in Section 2. Hither and yon clipping corresponds to the addition of the following two procedure calls in the algorithm:

```
clipt (− deltaz, z0 − h);
clipt (deltaz, f − z0);
```

3.4 Performance Test

Again, the computational complexity of the algorithm is linear in the number of line segments to be clipped with the exact requirements depending upon the particular data. In general, the worst case occurs when a line segment is not trivially rejected nor parallel to any of the bounding planes and has both of its original endpoints outside the viewing pyramid. In this case, two intersection points must be computed. This requires a total of 19 additions/subtractions, 6 multiplications, and 4 divisions.

This algorithm was compared with the traditional Sutherland–Cohen three-dimensional clipping algorithm. However, it was found that the Pascal code for the latter algorithm as it appears in [5, p. 345] sometimes enters an infinite loop. This is due to the fact that this program was based on an assumption that is not always true. Once it is recognized that the source of this problem is simply this erroneous assumption, then it is a straightforward matter to correct the program. This is done in the Appendix, and it is this corrected version that is used in the comparison test.

Both algorithms performed 4 million clips. As in the two-dimensional case, this consisted of four different sets of data, each containing 1000 randomly generated line segments which were each clipped 1000 times. The new algorithm required an average of 264 seconds, while the traditional one used an average of 440 seconds (a 40 percent improvement) on a DEC VAX-11/780 with a floating point accelerator in Pascal under Berkeley UNIX.

4. CLIPPING FOR HOMOGENEOUS COORDINATE SYSTEMS

4.1 Explanation

In the case of four-dimensional clipping with perspective depth transformation, a homogeneous coordinate system is employed. Clipping is performed after the

application of the perspective transformation, but prior to the perspective depth division.

The x and y coordinates are transformed by eq. (3.2) as they were in Section 3 so that the viewing pyramid is distorted to become a right pyramid. As in eq. (3.3), the x and y clipping limits are

$$\begin{aligned} -z_e \leq x \leq z_e \\ -z_e \leq y \leq z_e. \end{aligned} \tag{4.1}$$

This formulation can easily handle the case of a finite viewing volume, as was shown in Figure 6. This requires the following constraint for z:

$$h \leq z_e \leq f. \tag{4.2}$$

Since clipping is to be performed after the perspective transformation, inequalities (4.2) must be transformed to the corresponding clipping limit. In the eye coordinate system used in Section 3, the perspective transformation of the z coordinate involves a division by the depth z and is thus of the form

$$z^* = \frac{a\, z_e + b}{z_e} \tag{4.3}$$

where the asterisk indicates that the perspective division has been performed.

A particularly convenient limit which normalizes the depth values between the hither and yon clipping planes would be

$$0 \leq z^* \leq 1. \tag{4.4}$$

From this, eq. (4.3) becomes

$$z^* = \frac{f}{f-h}\left(1 - \frac{h}{z_e}\right). \tag{4.5}$$

Since the right-hand side of expression (4.5) is a monotonically increasing function that attains values of 0 and 1 at $z_e = h$ and $z_e = f$, respectively, the limit given in inequalities (4.4) is satisfied.

Using the homogeneous formulation to absorb the perspective division, eq. (4.5) can be rewritten as

$$z^* = \frac{z}{w} \tag{4.6}$$

where $z = [f/(f-h)](z_e - h)$ and $w = z_e$.

Substituting eq. (4.6) into inequalities (4.4) yields

$$0 \leq \frac{z}{w} \leq 1. \tag{4.7}$$

From eq. (4.6), w can be substituted into inequalities (4.1) yielding the x and y clipping limits

$$\begin{aligned} -w \leq x \leq w \\ -w \leq y \leq w. \end{aligned} \tag{4.8}$$

Inequalities (4.8) imply $w \geq 0$, and thus inequality (4.7) can be rewritten to yield the final clipping limit

$$0 \leq z \leq w. \tag{4.9}$$

The method of Section 3 can be easily extended to handle this case. Let the endpoints of the line segment be $\mathbf{V}_0\ (x_0, y_0, z_0, w_0)$ and $\mathbf{V}_1\ (x_1, y_1, z_1, w_1)$. The parametric representation of the line is given by eqs. (2.2), (2.3), (3.4), and (3.5) with the addition of the following equation:

$$w = w_0 + \Delta w\ t \tag{4.10}$$

where

$$\Delta w = w_1 - w_0. \tag{4.11}$$

Substituting this parametric representation into inequalities (4.8) and (4.9) yields the following conditions for the part of the extended line that is inside the frustum of vision:

$$\begin{aligned} -(\Delta x + \Delta w)t \leq x_0 + w_0 \quad &\text{and} \quad (\Delta x - \Delta w)t \leq w_0 - x_0 \\ -(\Delta y + \Delta w)t \leq y_0 + w_0 \quad &\text{and} \quad (\Delta y - \Delta w)t \leq w_0 - y_0 \\ -\Delta z\ t \leq z_0 \quad &\text{and} \quad (\Delta z - \Delta w)t \leq w_0 - z_0. \end{aligned} \tag{4.12}$$

Again, these conditions are inequalities describing the *interior* of the clipping volume rather than equations defining its *boundary*. The six inequalities (4.12) can be rewritten in the form of inequalities (3.7) with $i = 1, 2, \ldots, 6$. Specifically,

$$p_i\ t \leq q_i \quad \text{for} \quad i = 1, 2, \ldots, 6 \tag{4.13}$$

where

$$\begin{aligned} p_1 &= -(\Delta x + \Delta w); & q_1 &= x_0 + w_0; & &\text{(left)} \\ p_2 &= \Delta x - \Delta w; & q_2 &= w_0 - x_0; & &\text{(right)} \\ p_3 &= -(\Delta y + \Delta w); & q_3 &= y_0 + w_0; & &\text{(bottom)} \\ p_4 &= \Delta y - \Delta w; & q_4 &= w_0 - y_0; & &\text{(top)} \\ p_5 &= -\Delta z; & q_5 &= z_0; & &\text{(hither)} \\ p_6 &= \Delta z - \Delta w; & q_6 &= w_0 - z_0; & &\text{(yon).} \end{aligned} \tag{4.14}$$

It can be shown that there are exactly two linear relations satisfied by the p_i coefficients:

$$\begin{aligned} p_1 + p_2 - p_5 - p_6 &= 0 \\ -p_3 - p_4 + 2p_5 + 2p_6 &= 0. \end{aligned} \tag{4.15}$$

Thus, four of the coefficients are independent and hence there is no restriction on the sign of any of the six p_i.

Analogous to the method explained in Section 3, clipping using homogeneous coordinate systems is accomplished by solving the inequalities (4.13) under the condition $0 \leq t \leq 1$. The algorithm is similar to that given in Section 3, except

now there are six inequalities to be considered, and the p_i and q_i are defined by eq. (4.14).

4.2 Example

Consider the following example: let $\mathbf{V}_0$ (0, 2, 1, 1) and $\mathbf{V}_1$ (−2, −3, 1, 3) be the endpoints of a line segment to be clipped against a frustum of vision. Then

$$\begin{aligned} \Delta x &= x_1 - x_0 = -2 \\ \Delta y &= y_1 - y_0 = -5 \\ \Delta z &= z_1 - z_0 = 0 \\ \Delta w &= w_1 - w_0 = 2 \end{aligned} \tag{4.16}$$

and

$$\begin{aligned} p_1 &= -\Delta x - \Delta w = 0; & q_1 &= x_0 + w_0 = 1; & q_1/p_1 &= +\infty \\ p_2 &= \Delta x - \Delta w = -4; & q_2 &= w_0 - x_0 = 1; & q_2/p_2 &= -1/4 \\ p_3 &= -\Delta y - \Delta w = 3; & q_3 &= y_0 + w_0 = 3; & q_3/p_3 &= 1 \\ p_4 &= \Delta y - \Delta w = -7; & q_4 &= w_0 - y_0 = -1; & q_4/p_4 &= 1/7 \\ p_5 &= -\Delta z = 0 & q_5 &= z_0 = 1; & q_5/p_5 &= +\infty \\ p_6 &= \Delta z - \Delta w = -2; & q_6 &= w_0 - z_0 = 0; & q_6/p_6 &= 0. \end{aligned} \tag{4.17}$$

Compute t_0 and t_1^*:

$$t_0 = \max(\{q_i/p_i \mid p_i < 0,\ i = 1, \ldots, 6\} \cup \{0\}) = \max(-1/4, 1/7, 0, 0) = 1/7$$

$$t_1^* = \min(\{q_i/p_i \mid p_i \geq 0,\ i = 1, \ldots, 6\} \cup \{1\}) = \min(+\infty, 1, +\infty, 1) = 1.$$

Thus, the visible segment is defined by the parametric interval $1/7 \leq t \leq 1$ that corresponds to the following endpoints:

$$\begin{aligned} t_0 = 1/7: \quad x_0' &= 0 + (-2)1/7 = -2/7 & t_1^* = 1: \quad x_1' &= x_1 = -2 \\ y_0' &= 2 + (-5)1/7 = 9/7 & y_1' &= y_1 = -3 \\ z_0' &= 1 + (0)1/7 = 1 & z_1' &= z_1 = 1 \\ w_0' &= 1 + (2)1/7 = 9/7 & w_1' &= w_1 = 3. \end{aligned}$$

4.3 Algorithm

The homogeneous line clipping algorithm is as follows:

```
(*clip 4d line segment with endpoints (x0, y0, z0, w0) and (x1, y1, z1, w1)*)
(*the viewing volume is a frustum of vision in three-space*)

procedure clip4d (x0, y0, z0, w0, x1, y1, z1, w1: real);
var t0, t1: real;
  deltax, deltay, deltaz, deltaw: real;

begin (*clip4d*)
  t0 := 0;
  t1 := 1;
```

```
  deltax := x1 − x0;
  deltaw := w1 − w0;
  if clipt (− deltax − deltaw, x0 + w0, t0, t1) then
    if clipt (deltax − deltaw, w0 − x0, t0, t1) then begin
      deltay := y1 − y0;
      if clipt (− deltay − deltaw, y0 + w0, t0, t1) then
        if clipt (deltay − deltaw, w0 − y0, t0, t1) then begin
          deltaz := z1 − z0;
          if clipt (− deltaz, z0, t0, t1) then
          if clipt (deltaz − deltaw, w0 − z0, t0, t1) then begin
            if t1 < 1 then begin
              x1 := x0 + t1 * deltax;
              y1 := y0 + t1 * deltay;
              z1 := z0 + t1 * deltaz;
              w1 := w0 + t1 * deltaw
            end;
            if t0 > 0 then begin
              x0 := x0 + t0 * deltax;
              y0 := y0 + t0 * deltay;
              z0 := z0 + t0 * deltaz;
              w0 := w0 + t0 * deltaw
            end
          end
        end
    end
end (*clip4d*);
```

where **function** *clipt* is as was defined in Section 2.

4.4 Performance Test

As was the case in Section 3, the computational complexity of the algorithm is linear in the number of line segments to be clipped. The exact requirements are dependent upon the particular data, and the worst case requires a total of 25 additions/subtractions, 8 multiplications, and 6 divisions.

This algorithm was compared with the traditional Sutherland–Cohen homogeneous line clipping algorithm. Pascal code for the latter was copied verbatim from [5, p. 360]. Both algorithms performed 4 million clips, using data identical to that which was generated for the three-dimensional case, except transformed to the homogeneous formulation by eq. (4.6). The new algorithm required an average of 278 seconds, while the traditional one used an average of 1350 seconds (a 79 percent improvement) on a DEC VAX-11/780 with a floating point accelerator in Pascal under Berkeley UNIX.

5. CONCLUSION

A new concept and method for line clipping has been developed. The basic ideas have been embedded in a family of algorithms for two-dimensional, three-dimensional, and four-dimensional (homogeneous coordinates) line clipping.

For each dimension, the mathematics were derived along with geometrical interpretations, and an example was shown. Then the algorithm was designed, and a performance test was conducted. The new algorithm was compared with the traditional Sutherland–Cohen [5] clipping algorithm. Both were executed on randomly generated data. The new algorithm showed a 36 percent to 79 percent improvement, as illustrated in Table I.

Table I. Comparison of the Performance of the Algorithms

Dimension	Sutherland–Cohen (seconds)	Liang–Barsky (seconds)	Improvement (percent)
2	519	333	36
3	440	264	40
4	1350	278	79

One of the advantages of this algorithm is that it quickly determines whether a line segment is to be rejected. In the Sutherland–Cohen algorithm, a line segment is trivially rejected only if both of its endpoints lie on the same (invisible) side of the window. On the other hand, there are two different categories of line segments that will be trivially rejected by the Liang–Barsky algorithm. The first category comprises those line segments that are parallel to and outside one of the window boundaries; this is detected when $p_i = 0$ and $q_i < 0$. The second kind are those that do not intersect the window boundary at all; this occurs when the value about to be stored in t_0 exceeds t_1, or when the value about to be stored in t_1 is less than t_0.

Although a line segment in the second category is quickly rejected by the new algorithm, the Sutherland–Cohen algorithm may perform a significant amount of computation before rejecting it. As an example, consider a diagonal line segment that lies outside the window passing through the lower right region. The Sutherland–Cohen algorithm will clip this line segment so that it begins at the top boundary of the window and will then execute another pass through the loop, clipping the line segment yet again so that it now begins at the right boundary of the window. It should be noted that each of these clip operations requires the computation of an intersection point. The Liang–Barsky algorithm would simply compute t_1 to be the parametric value at which the line segment intersects the left boundary. Then, considering the right boundary, it would be in the process of computing t_0 when the reject case would show that the value about to be stored in t_0 already exceeds the computed value of t_1. This information is sufficient to conclude that the line segment bypasses the window and the algorithm determines that the line segment should be rejected. This early determination of reject cases results in a computational savings for the Liang–Barsky algorithm.

Since the structure of the algorithm is such that the conditions describing the interior of the clipping region are all of similar form, these visibility calculations could be processed in parallel.

Although this algorithm was presented for perspective viewing volumes, it should be noted that it is equally appropriate for nonperspective viewing volumes, and more generally, for any convex viewing volume. Since the algorithm computes visibility by taking the intersection of the visible sides of a set infinite planes, this approach is applicable to any convex volume.

APPENDIX

When You're Behind, You Can Be Left of Left and Still Be Right of Right

The Sutherland–Cohen three-dimensional clipping algorithm as coded in [5, p. 345] is based on an assumption that is not always true. Specifically, it assumes that if an endpoint lies to the left of the left bounding plane of the

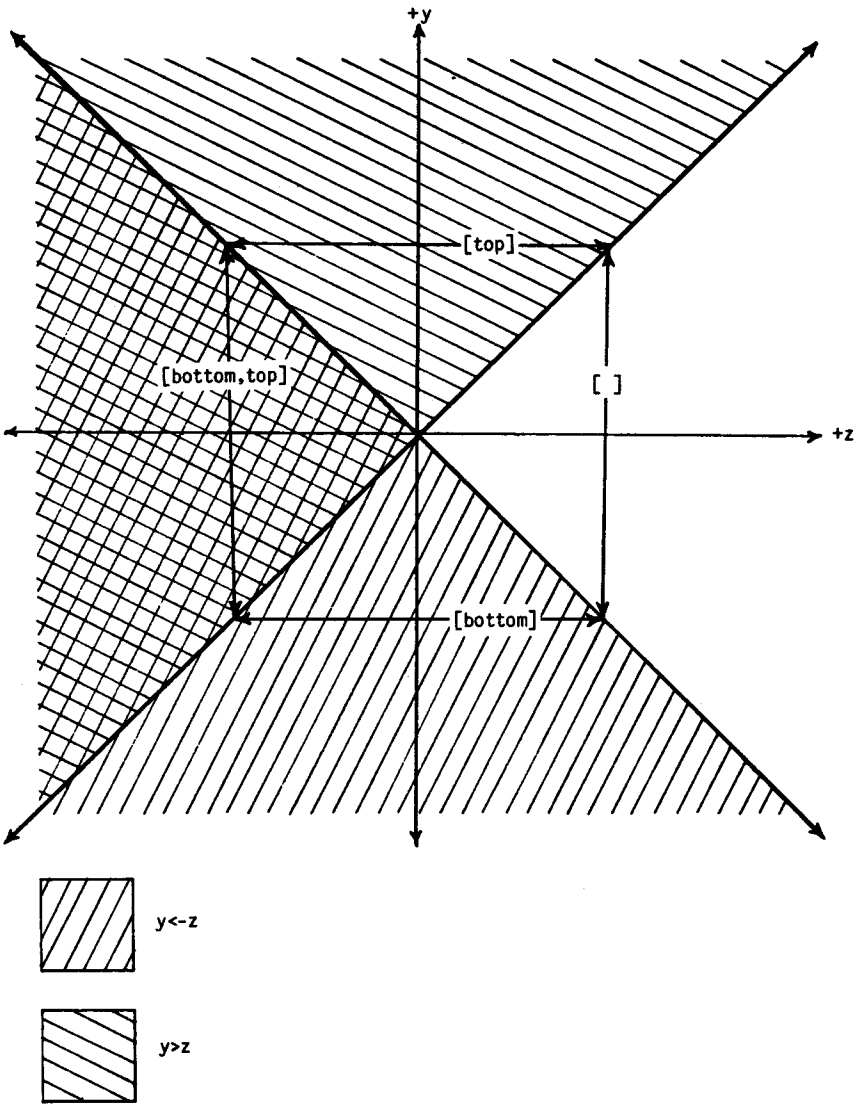


Fig. 7. The four regions and their bottom/top codes.

viewing pyramid, then it cannot lie to the right of the right bounding plane, and similarly for the bottom and top bounding planes. Although this may seem to be obviously correct, a point can be both "left of left" and "right of right," or both "below bottom" and "above top," if it lies *behind* the observer, an important case in three-dimensional clipping.

To see this for the bottom/top case, consider a slice through three-space along the vertical plane $x = 0$, as shown in Figure 7. The region that is "below bottom" ($y < -z$) is shown hatched in one direction, while the "above top" area ($y > z$) is shown with a different hatching. Note that these two regions have a nonempty intersection; thus, points in this area are *both* above top and below bottom.

By ignoring this possibility, the codes that are generated by the algorithm as written in [5] are not always correct. If one of these codes contains *left*, it cannot contain *right*, and similarly with *bottom* and *top*, even though there are cases where it should. Hence, only 9 of the 16 possible codes can be constructed with this algorithm, and thus some endpoints will have incorrect codes.

When a code is incomplete, a pair of endpoints that both lie strictly outside one of the bounding planes may not be recognized as such. Therefore, there are some line segments that should be trivially rejected, but will instead put the program into an infinite loop.

As an example, consider the line segment having endpoints **A** (0, 8, 0) and **B** (8, 2, −7). The code for **A** is [*top*]; however, the code for **B** will be incorrectly constructed as [*right, bottom*] instead of [*right, bottom, top*] as it should be. This line segment should be trivially rejected because it is "above *top*." However, since *top* is absent from the code for **B**, the codes for the endpoints have an empty intersection and hence the line will be clipped. Since the code for **A** is [*top*], it is computed to become (−64, 56, 56), the code for which is [*left*]. The next iteration through the loop recomputes **A** as (0, 8, 0), the same as its initial value, thereby demonstrating the existence of an infinite loop. The code for **A** will alternate between [*top*] and [*left*] and will always have an empty intersection with the incomplete code of [*right, bottom*] for **B**.

Once it is recognized that the source of this problem is simply the erroneous assumption that an endpoint cannot lie to the right of the right bounding plane if it lies to the left of the left bounding plane, and similarly for the bottom and top bounding planes, then it is a straightforward matter to correct the program.

The following procedure corrects the procedure *Code* as given in [5]:

```
procedure Code (x, y, z: real; var c: outcode);
begin (*Code*)
  c := [ ];
  if x < −z then c := [left];
  if x > z then c := c + [right];
  if y < −z then c := c + [bottom];
  if y > z then c := c + [top]
end; (*Code*)
```

where, as in the book, the following type declaration is implicit:

```
type edge = (left, right, bottom, top); outcode = set of edge;
```

ACKNOWLEDGMENTS

The authors are grateful to Shinichiro Haruyama and Pauline Ts'o of the Berkeley Computer Graphics Laboratory for their programming assistance.

Appreciation is expressed to Walter N. Brown of the Engineering/Physics Department, Santa Rosa Junior College, and to Caryn Dombroski of the Computer Science Division, University of California Berkeley, for their excellent work in preparing the figures for this paper.

The authors would also like to thank the referees for providing many valuable criticisms.

REFERENCES

1. BLINN, J.F., AND NEWELL, M.E. Clipping using homogeneous coordinates. In *SIGGRAPH '78 Conference Proceedings* (Atlanta, Ga., Aug. 1978). ACM, New York, 1978, pp. 245–251.
2. LIANG, Y.-D., AND BARSKY, B.A. An analysis and algorithm for polygon clipping. *Commun. ACM 26*, 11 (Nov. 1983), 868–877.
3. LIANG, Y.-D., AND BARSKY, B.A. Introducing a new technique for line clipping. In *Proceedings of the International Conference on Engineering and Computer Graphics* (Beijing, Aug. 27–Sept. 1, 1984), 548–559; also in *J. Zhejiang Univ.* Special Issue on Computational Geometry (1984), 1–12.
4. LIANG, Y.-D., AND BARSKY, B.A. Three new algorithms for two-dimensional line clipping. In preparation.
5. NEWMAN, W.M., AND SPROULL, R.F. *Principles of Interactive Computer Graphics*, 2nd ed. McGraw-Hill, New York, 1979.
6. SPROULL, R.F., AND SUTHERLAND, I.E. A clipping divider. In *Proceedings of the Fall Joint Computer Conference*, vol. 33. AFIPS Press, Reston, Va., 1968, pp. 765–776.

Received December 1982; revised January 1984; accepted February 1984