Computer generated graphic segments in a raster display

by RICHARD A. METZGER
Rome Air Development Center
Rome, New York

INTRODUCTION

The increased use of computer graphics to enhance the man-machine interface has resulted in many and varied systems and devices to meet a multitude of needs. One type of display that is receiving new emphasis as a computer output device is the "raster format" display (of which standard television is a particular type). Among the reasons for using this type of display are: (1) the relative simplicity of the display device, (2) the ease of remote operation for multiple station users, (3) the low cost per station, (4) capability for mixing output with standard television sources, and (5) good position repeatability for computer generated data.

However, to utilize a raster display (either standard 525 line TV or other line standard) as a computer output device, a conversion from digital to video data must be performed. The device utilized to perform this function is commonly referred to as a Digital-to-Video (D/V) Converter. It accepts digital data from the data processing system and converts it to a video signal compatible with the raster scan display. The converter, being a digital device, often becomes complicated since it is forced to operate within the timing constraints of the Raster Scan Display System (RSDS). A problem also arises in the subjective appearance of the display, since all data must be generated within the line-by-line structure of the raster. In the case of alphanumerics, a fixed-size matrix of dots can be used to generate very acceptable symbology. However, the generation of graphic segments is not as easily accomplished.

Graphic segments and figures vary greatly in terms of size, shape, orientation, and complexity. If the scan lines forming the raster together with the discrete points on each scan line are considered as a Cartesian grid, it can be seen that in general, graphic segments will not fit exactly within the constraints of this grid. In addition, graphic figures require a set of defining equations, each valid for a given domain. Thus one of the simplest means of generating complex figures is by combination of graphic segments (lines, circles, arcs, etc.), into the higher order figures.

The particular problem to be addressed here is the generation of graphic segments (straight and curved lines) within the constraints of a raster format display. Algorithms are developed to allow computer generation of selected graphic segments of arbitrary length (constrained by screen size) at any desired screen location. The raster will be considered as an $N \times M$ Cartesian grid, where $N$ is the number of scan lines and $M$ is the number of addressable points on a line. All operations will be performed within the constraints of this address grid. By treating the grid dimensions as variable, the algorithms are immediately usable for any line standard raster. The software approach to development of the algorithms was adopted since this allows usage with any of the standard D-V converters, whereas a hardware implementation is particular as to type. However, there is nothing within the adopted approach which prohibits hardware implementation.

Prior to considering the algorithms for generation of graphic segment, a brief discussion of Digital-to-Video conversion as a display technique will be presented.

Digital-to-video conversion

The generation of data within a raster formatted display is governed by the timing of the raster sweeps. To generate a "dot" at a given point it is necessary to unblank the beam at the instant it traverses the specified address. As stated previously, the Raster Format of the display can be considered as an address coordinate grid where the individual scan lines represent the ordinate values (Y-Address) and the points along the line represent the abscissa values (X-Address).
By considering the relation between the scan rate of the electron beam and the Cartesian address grid, it can be seen that associated with any given picture element there is a corresponding X-Y coordinate address and vice versa. Thus by taking into account the number of lines preceding the addressed line and the number of elements on this line preceding the addressed element, a given time interval after the beginning of each frame can be associated with each picture element. In this way D-V conversion can be considered as a position to time conversion.

**Bit per element techniques of D-V conversion**

One means of implementing a digital-to-video converter to provide the above type of position to time conversion is the “Bit Per Element” converter. The general functions of such a converter are outlined in Figure 1. Input data is accepted from the data source by the interface control in a word serial, bit parallel form. This data consists of an X and Y address(es), character code and control bits. Data format and parity are checked and if proper, the data is transferred to the buffer memory and process control. The buffer memory is a small high speed digital memory (usually core) which temporarily stores the character, vector, and control data in the same form as received by the interface unit. By means of the process control, the address bits are separated and transferred to the sync and comparator section while the character/vector data is held in the buffer memory. If the data supplied from the data source is not sorted (in X-Y address), then the process control has the additional function of sorting the address data in ascending orders of Y and X within each Y.

Since the raster timing must provide overall control of the conversion as well as display process, all digital and display functions must be timed to the raster synchronization pulses or harmonics thereof. The sync and comparator section provides this control interface as well as provide the necessary timing signals internal to the converter. There are two primary ways of providing the control interface. In the first, the raster synchronization pulses are fed from the display device to the sync and comparator section and the internal timing for the converter derived from it. In the second mode, the sync and comparator section of the converter contains a crystal controlled clock which operates at a harmonic of raster timing, with the latter being derived from the crystal.

Under control of the sync and comparator section, the data is transferred from the buffer memory to the character generator where a dot pattern of the alpha-numeric symbol, etc. . . . to be displayed is formed. This dot pattern is then transferred to a section of the video memory determined by the display coordinate address. In the case of a graphic segment, a single dot is generated at each of a series of coordinate addresses with the group or sequence of dots forming the graphic figure.

The video memory contains one bit of digital storage (1 = unblanked electron beam, 0 = blanked electron beam) for each picture element on the display surface. Thus by loading the dot pattern of the character at a
memory address corresponding to the display address, the contents of the video memory bear a one-to-one correspondence to the generated display. To maintain display continuity and eliminate presentation of partial characters caused by loading sections of characters during free memory cycles, loading of the video memory is performed during retrace of the electron beam when it is blanked from the display surface. The memory is read out in synchronization with the beam, i.e., every bit is read out as the electron beam traverses the corresponding point on the display surface. To attain the output speed required, it is necessary to perform multiple word read out, multiplex several tracks or lines, or use very long word lengths. In each case the data is read into a register for parallel to serial shifts. When the serial bit stream is inserted into the synchronization and blanking interval, the video signal results.

Real time techniques of D/V conversion

An alternative method for implementation of D/V converters are the “Real-Time” type represented by Figure 2. The primary difference is the absence of the large (digital) video memory to recirculate the “bit per element” display at the 30 frame/sec refresh rate.

The data is transferred from the data source through the interface unit to the buffer memory in a manner analogous to the previous example. In this case, however, the buffer memory is of sufficient capacity to store a complete frame in computer word form, e.g., to present 1000 characters, each defined by three computer words requires a 3000 word memory. The addressed portion of the stored words are continuously compared to the position of the scanning beam. This is accomplished by use of two counters controlled by the raster synchronization pulses. One counter is advanced by the horizontal synchronization pulse and indicates which scan line (Y-Address) is being written. A second, higher speed, counter advances by M for each horizontal sync pulse, until a number corresponding to the number of picture elements per line is attained. In this way the second counter indicates the picture element (X-Address) being scanned. By continuously sampling both counters, the screen address for any specified X-Y Address can be obtained.

A given interval prior to coincidence (sufficient to account for propagation delay) the synchronization and comparator circuits transfer the character data from the buffer memory to the character/vector generator. A dot pattern of the character/vector is generated at a bit rate sufficient for insertion directly into the raster. Thus the data is transferred directly from the character/vector generator to the display device by means of high speed register without the requirement for a digital video memory. This sequence is performed at a 30 frame per second rate since no digital video memory is available for display refresh. Erasure of the data is accomplished by inhibiting the data output from the character/vector generator and entering new data into the digital memory. In the use of the bit per element converters, a specific erase function (which amounts to entering the complemented character dot pattern) must be provided to remove the displayed dot pattern from the digital video memory.

Straight line generation

The complexity involved in generation of graphic segments in a raster format can be seen by considering the straight line vector as shown in Figure 3. It is desired to generate a line connecting points \((X_1, Y_1)\) and \((X_2, Y_2)\) where the Y values represent raster lines and
the X values represent picture elements within a raster line. If the lines were horizontal, vertical, or 45 degrees, the generation would be trivial. Without loss of generality, the analysis which follows is for a line directed down to the right at an angle, $\alpha < 45^\circ$ as shown in Figure 3. (A similar type of analysis could be performed for any other octant.)

To form a trace at an angle $\alpha < 45^\circ$, the address of the generated dot must be increased by a given number of units (called the DELTA value), in the X direction, prior to a one-unit increase in the Y address. For a line at an angle $\alpha > 45^\circ$, the address is incremented DELTA units in the Y direction prior to a one-unit increase in the X address. The key to generation of the proper line thus lies in the choice of the “DELTA” value. If DELTA is set equal to the slope, then the error is equal to the remainder (R) of the integer division.

If the DELTA is set equal to $Q$ rounded to the nearest integer, then the error equals R if $R < \Delta Y/2$ and equals $(\Delta Y - R)$ if $R \geq \Delta Y/2$. In general, if the slope $Q$ is used to determine the DELTA value, then to obtain a zero end point error requires R increments of DELTA equal to $(Q + 1)$ and $(\Delta Y - R)$ increments DELTA equals to $Q$.

This yields

$$\Delta X = (Q + 1) R + (\Delta Y - R) Q = Q\Delta Y + R \quad (2)$$

which is the condition for zero error. However, it is a formidable task to obtain a line with acceptable linearity and no end point error when computations are based only end point data, since the technique for intermixing the two different length DELTA segments resulting in a zero error is dependent on the particular end points in question. Thus the approach adopted is based upon a “best fit” technique minimizing the error with respect to the desired line, introduced with an incrementation in either the X or Y direction or both.

The main functions to be performed by any straight line algorithms are:

1. determination of direction, right or left, which controls whether incrementation or decrementation, respectively of the X address coordinate is required.
2. determination of whether the angle of inclination is greater than, less than, or equal to 45°.
3. determination of the DELTA segment length.
4. generation of a “dot” at the sequence of points between $(X_1, Y_1)$ and $(X_2, Y_2)$ within the constraints previously listed and in accordance with the above data.

The DELTA value is based upon a determination of whether incrementing the current address $(X_n, Y_n)$, in the X direction, Y direction or both will result in the smallest deviation from the desired line. By summing the number of unit incrementations in a given direction, X or Y (which corresponds to the number of iterations prior to sign change), the DELTA length can be determined.

Let the coordinate grid of Figure 4a represent the raster coordinate grid in the vicinity of the start point $X_s, Y_s$. The actual trace represents the sequence of points which most closely approximates the desired line connecting $X_s, Y_s$, and $X_l, Y_l$ (not shown). The inclination with respect to the 45° line (determined from $\Delta X$ and $\Delta Y$) imposes a limitation upon the degrees of freedom of movement. Referring to the case depicted in Figure 4b, it can be seen that the possible new addresses are point 1 $(X_s + 1, Y_s)$ or point 2...
Figure 4B—First move determination in grid structure

If the error associated with each move is represented by A or B respectively, the geometric relationships allow the ratio of the error values to be determined as

\[ \frac{B}{A} = \frac{\Delta X - \Delta Y}{\Delta X} \]  

(3)

If the condition of equal error is chosen as the discriminant point, a value \( R_1 \) can be defined such that

\[ R_1 = 2\Delta Y - \Delta X \]  

(4)

where the sign of \( R_1 \) indicates the minimum error move to point 1 or point 2. (In Figure 4b the minimum error is to point 2.)

Referring to Figure 4c to consider the second move determination involving error distances C and D, the same analysis yields an \( R_1 \) discriminant value

\[ R_t = R_1 + 2\Delta Y - 2\Delta X \]  

(5)

If the initial address modification from \( X_s, Y_s \) had been to point 1 rather than point 2, the \( R_t \) for the subsequent move, Figure 4d discriminant, would assume the form

\[ R_t = R_1 + 2\Delta Y \]  

(6)

In both cases the sign of \( R_t \) would determine the minimum error move.

If this analysis is pursued to the general case including inclinations both greater and less than 45°, the following results are obtained:

a. After the first move from point \( X_s, Y_s \), all subsequent moves reduce to one of the two above cases.
b. The form of the error equation is dependent upon

\[ R_1 = 2\Delta Y - \Delta X \]  

(7)

and

\[ R_{t+1} = \begin{cases} R_t + 2\Delta Y - 2\Delta X & \text{if } R_t \geq 0 \\ R_t + 2\Delta Y & \text{if } R_t < 0 \end{cases} \]  

(8)

b. If the angle of inclination is >45° the roles of \( X \) and \( Y \) are interposed and we obtain

\[ R_1 = 2\Delta X - \Delta Y \]  

(10)

and

\[ R_{t+1} = \begin{cases} R_t + 2\Delta X - 2\Delta Y & \text{if } R_t \geq 0 \\ R_t + 2\Delta X & \text{if } R_t < 0 \end{cases} \]  

(12)
The resulting equations (Equations 7, 8, 9, 10, 11, and 12) can be employed to determine the length of the DELTA segments by noting the number of iterations performed on \( R_i + 1 \) between sign changes. Proper initialization based upon inclination with respect to the 45° line allow use of the following equations which are independent of line inclination:

\[
R_1 = 2T[2] - T[1] \tag{13}
\]

\[
R_{ref} = \begin{cases} 
R_i + 2T[2] - 2T[1] & \text{if } R_i \geq 0 \tag{14} \\
R_i + 2T[2] & \text{if } R_i < 0 \tag{15}
\end{cases}
\]

Each iteration of \( R_i \geq 0 \) increases the current address by one in both the X and Y directions. Each iteration of \( R_i < 0 \) increases the X Address by one, while holding the Y Address constant. Summation of the iterations \( R_i < 0 \) prior to each \( R_i \geq 0 \) yield the DELTA value.

This algorithm was verified by means of a computer program to generate straight lines connecting any two points within a 525-line raster grid.

The constraints placed on the straight line generator routine were the following:

1. All vectors will be generated in the direction of increasing Y address with the origin of the coordinate system at the upper left-hand corner of the display. (Addresses need not be sorted for input.)
2. All vectors will be processed (although not specified) as one of the following:
   a. Down to the left at an angle <45°
   b. Down to the left at an angle >45°
   c. Down to the right at an angle <45°
   d. Down to the right at an angle >45°
   e. Horizontal line
   f. Vertical line
   g. Down to the right/left at an angle = 45°

Typical results are depicted in Figure 5. The routine required approximately 370 instructions in a machine with a 64 instruction repertoire. The maximum error from any computed point to the desired line is less than 0.5 units.

**Circular line generation**

In a manner similar to generation of a straight line, the generation of a circular trace consists of determining the X-Y Address points which minimize the error distance from the desired trace to the selected point. By examining a circular trace overlaid on a Cartesian grid, many of the properties of the circular symmetry are observable. A circle of radius 7 is depicted in Figure 6, together with the appropriate X-Y Address sequence to generate the trace within the constraints of the grid structure, where the Y-Address corresponds to scan lines and X-Address to picture elements along that line. The display trace would be generated by a “dot” (the area of which is equal to a unit square) at each arrowhead.

It can be seen from Figure 7a (and is equally true for any radius) that the address sequence from 1 to A is exactly equivalent to those from 1 to D, 2 to A, 2 to B, 3 to B, 3 to C, 4 to C, and 4 to D. Thus by determining the proper address sequence for the first octant, the address sequence for all other octants (refereed to the center of the circle) have been obtained. It should also be noted that no point is more than one unit removed in each direction from the previous or subsequent point. Two possible address modifications can be determined for each octant, the first of which modifies only one address (X or Y) by one, while the second modifies both X and Y addresses by one. The octant in question determines whether the address modification is an incrementation or a decrementation. The possible address sequence for each octant are de-
picted in Figure 7b. Using these address sequences, it is apparent that each move encompasses a point exterior to the circle and a point interior to the circle. Table I represents the various possible moves for each octant, where \((X_n, Y_n)\) represents the point from which the move is being made and the columns are the resultant addresses after the move is made.

<table>
<thead>
<tr>
<th>Octant #</th>
<th>Exterior X-Address</th>
<th>Exterior Y-Address</th>
<th>Interior X-Address</th>
<th>Interior Y-Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(X_n)</td>
<td>(Y_n - 1)</td>
<td>(X_n - 1)</td>
<td>(Y_n - 1)</td>
</tr>
<tr>
<td>2</td>
<td>(X_n + 1)</td>
<td>(Y_n)</td>
<td>(X_n + 1)</td>
<td>(Y_n + 1)</td>
</tr>
<tr>
<td>3</td>
<td>(X_n - 1)</td>
<td>(Y_n)</td>
<td>(X_n - 1)</td>
<td>(Y_n + 1)</td>
</tr>
<tr>
<td>4</td>
<td>(X_n)</td>
<td>(Y_n - 1)</td>
<td>(X_n + 1)</td>
<td>(Y_n - 1)</td>
</tr>
<tr>
<td>5</td>
<td>(X_n)</td>
<td>(Y_n + 1)</td>
<td>(X_n + 1)</td>
<td>(Y_n + 1)</td>
</tr>
<tr>
<td>6</td>
<td>(X_n - 1)</td>
<td>(Y_n)</td>
<td>(X_n - 1)</td>
<td>(Y_n - 1)</td>
</tr>
<tr>
<td>7</td>
<td>(X_n + 1)</td>
<td>(Y_n)</td>
<td>(X_n + 1)</td>
<td>(Y_n - 1)</td>
</tr>
<tr>
<td>8</td>
<td>(X_n)</td>
<td>(Y_n + 1)</td>
<td>(X_n - 1)</td>
<td>(Y_n + 1)</td>
</tr>
</tbody>
</table>

The problem then is to determine in each case, whether a move to an exterior or an interior point represents the minimum error. This can be accomplished by comparing the differences between the actual radius \((R_o)\) and the external and internal radii \((R_e\) and \(R_i\)) respectively, as shown in Figure 7b. If it is determined that

\[
(R_o - R_i) > (R_e - R_o) \tag{16}
\]

then the minimum error move is to the next exterior point. Likewise if

\[
(R_o - R_i) < (R_e - R_o) \tag{17}
\]
then the interior point represents the minimum error. From Figure 7b it can be seen that

\[ R_\Delta = \sqrt{A_n^2 + B_n^2} \quad (18) \]
\[ R_\times = \sqrt{A_n^2 + (B_n + 1)^2} \quad (19) \]
\[ R_\odot = \sqrt{(A_n - 1)^2 + (B_n + 1)^2} \quad (20) \]

Substituting Equations (19) and (20) into Equation (16) yields

\[ 2R_\times > \sqrt{A_n^2 + (B_n + 1)^2} + \sqrt{(A_n - 1)^2 + (B_n + 1)^2} \quad (21) \]

By means of the Triangular Inequality this yields

\[ 4R_\times > (A_n - 1)^2 + (2B_n + 2)^2 \quad (22) \]

In a similar manner, Equations (17), (19), and (20) yield

\[ 4R_\Delta < (2A_n - 1)^2 + (2B_n + 2)^2 \quad (23) \]

Thus if we define a quantity \( Q \) such that

\[ Q = (2A_n - 1)^2 + (2B_n + 2)^2 - 4R_\times \quad (24) \]

then the sign of \( Q \) determines whether the minimum error move is exterior or interior, with \( Q < 0 \) denoting an exterior point and \( Q \geq 0 \) denoting an interior point. Due to the symmetry of the trace, it is only necessary to determine the move pattern for the first octant and use this result to perform the address modification for all octants.

This algorithm was implemented and found to yield circular traces (Figure 8) in which no computed element is in error by a distance of more than 0.5 address units in either direction, i.e., interior or exterior. The generation of circular arcs is accomplished by the same algorithm by merely specifying data to define the end points.

It should be noted that for circles with sufficiently small radii (<4 units) the appearance of uniform curvature diminishes. This is due not to a failure of the routine but to the fact that for those radii, insufficient points within the grid structure exist to properly define the curvature. It can be shown that the algorithm itself is valid for all radii greater than 1.25, with this lower limit being dictated by the lowest value of \( R \) for which the use of the triangular with Equation 21 remain valid.

\[ X^2 = 4PY \quad (25) \]

Similarly for selected values of \( X \), one can compute values of \( Y \) according to the Equation

\[ Y^2 = 4PX \quad (26) \]

The difference in method of generation arises in that the trace on the graph paper is obtained by connecting
the points defined by the computation, whereas within the raster the increments for the independent variable are chosen to be one line width apart and thus by merely generating a "dot" at the computed X-address points, a continuous curve results.

A typical X-Y address sequence for generation of a parabola is depicted in Figure 9. The actual trace would be created by generating a "dot" at each arrowhead. Although the address sequence will vary depending upon the focus value (P) several important factors can be deduced from the figure. The move pattern is up with increments to the right and left. The number of increments in a given direction is dependent upon the square root for a value of independent variable in relation to the square root for the previous value of the same variable. For each increment in the Y direction if

1. \( 0 < \sqrt{4Py_n} - \sqrt{4Py_{n-1}} \leq \frac{1}{2} \)
   then \( X_n = X_{n-1} \)

2. \( \frac{1}{2} < \sqrt{4Py_n} - \sqrt{4Py_{n-1}} \leq 1 \)
   then \( X_n = X_{n-1} + 1 \)

3. \( 1 < \sqrt{4Py_n} - \sqrt{4Py_{n-1}} < \infty \)
   then \( X_n = X_{n-1} + M \)

The third case depicted will occur near the base of the parabola with case (1) prevailing as the parabola approaches its asymptote. Since the parabola approaches an asymptote which tends toward infinity, the trace is continued until a screen address is exceeded in one direction at which point the computation stops. The proper incrementation address sequence is dependent upon the form of the equation, the sign of P, and the relative values of the square root for succeeding values of independent variable. The incrementation conditions for all cases are depicted in Table II where \((X_n, Y_n)\) is the point being computed, while the value along axis of symmetry is incremented by one unit.

The implementation of an algorithm for performing the above arithmetic and manipulative function is not complex. However, the computation time becomes very large for all cases except where the length of the axis of symmetry is kept small. This is necessitated by the iterative methods required for computation of the square root. When implemented on a computer with a six (6) microsecond cycle time, running times in excess of 30 seconds were not uncommon. However, the accuracy obtained placed each computed point within one half display element, since the X (Y) address points are computed for each Y (X) address. The question immediately arises that if the accuracy specification is relaxed and an approximation to the parabola (as it approaches the asymptote) is acceptable, can an appreciable decrease in running time be obtained?

If, for example it is assumed that beyond a given point \((X_p, Y_p)\) (Figure 10) the parabola can be approximated by the straight line \((X_p, Y_p), (X_s, Y_s)\) a maximum error \(E\) would result at the point the trace reaches the maximum address or edge of the screen. In addition, to maintain the trace uniformity, the slope of the approximating line is chosen to be the same as the slope of the parabola at the point \((X_p, Y_p)\). The slope of the parabola at \((X_p, Y_p)\) is

\[
\frac{dy}{dx} \bigg|_{yp} = \frac{X_p}{2P} \tag{27}
\]

Since this also represents the slope of the approximating line,

then

\[
\frac{X_p}{2P} = \frac{\Delta y}{\Delta X} \tag{28}
\]

which yields

\[
\Delta X = \frac{2P}{X_p} (Y_2 - Y_p) \tag{29}
\]
The value for $Y_p$ is the STOP address of the axis variable from which $X_p$ can be computed. The approximation can be completed by drawing straight lines between the points $(X_p, Y_p)$ and $(X_2, Y_2)$ where $X_2 = X_1 + E$.

A similar analysis can be carried out for a parabola satisfying the Equation

$$Y^2 = 4PX$$

yielding a defining Equation for $Y_p$,

$$Y_p = [(Y_2 + E) \pm \sqrt{(Y_2 + E)^2 - 4PX_2}]$$

The inclusion of the approximation as a subroutine, to be entered if the allowable error is $E > 0$, results in greatly reduced running times proportional to the error allowed. In addition, the subjective appearance is maintained (Figure 11) by retaining the desired curva-

\[ \begin{align*}
X_p &= X_2 + E \left( \frac{Y_2 - Y_p}{X_p} \right) \\
E &= X_2 - X_p
\end{align*} \]
Table II—Address modification pattern for parabolic trace

<table>
<thead>
<tr>
<th>Defining Equation</th>
<th>Axis of Symmetry</th>
<th>P</th>
<th>$X^2 = 4PY$</th>
<th>$Y^2 = 4PX$</th>
<th>Left Side</th>
<th>Right Side</th>
<th>Top</th>
<th>Bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X^2 = 4PY$</td>
<td>Pos. Y</td>
<td>&gt; 0</td>
<td>&gt; 0, ≤ 1/2</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y^2 = 4PX$</td>
<td>Pos. X</td>
<td>&gt; 0</td>
<td>&gt; 0, ≤ 1/2</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X^2 = 4PY$</td>
<td>Neg. Y</td>
<td>&gt; 0</td>
<td>&gt; 0, ≤ 1/2</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y^2 = 4PX$</td>
<td>Neg. X</td>
<td>&lt; 0</td>
<td>&gt; 0, ≤ 1/2</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
<td>$X_{n-1}^+X_{n+1}^-X_{n-1}^-X_{n+1}^+$</td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ture at the vertex and use of the straight line approximation as the parabola approaches the asymptotic value.

Many other graphic figures could be generated by using the "error minimization" technique above; however, their utility would depend upon the system in question. The figures developed thus far (straight line, circle, circular arc, parabola, and parabolic arc) form the basis for a rudimentary system. The more important aspect from a user point of view would be the combination of the various segment generators for complex figures.

**Raster graphic system considerations**

To effectively utilize the algorithms thus far defined, they must be incorporated as part of an overall graphic system or subsystem which includes the Graphic Segment Generators as well as a Graphic Operating System.

In configuring the combined Segment Generator, it is assumed that the Operating System would provide the following data in addition to the segment type designator:

- **a. Straight Line**—Start Point ($X_1$, $Y_1$), Stop Point ($X_2$, $Y_2$)
- **b. Circle**—Center Point ($X_c$, $Y_c$), Radius ($R$)
- **c. Circular Arc**—Center Point ($X_o$, $Y_o$), Radius, Start Point (Octant, $Y$ Add), Stop Point (Octant, $Y$ Add)
- **d. Parabola**—Vertex ($X_v$, $Y_v$), Focus ($P$), Orientation (H or V), Axis Stop Point
- **e. Parabolic Arc**—Vertex ($X_v$, $Y_v$), Focus, Orientation, Side, Axis Start, Axis Stop

The Generator would be entered from a Graphic Operating System and upon completion of its processing function would return control to the operating system. In addition, all data points would be entered through the Graphic Operating System which would control storage and output of the calculated points as well.

The routine is entered at the same point for any of the segments and branches to the desired section (Straight line, Circle, etc.), based upon the mode bits of the data words.

The "move discriminant" functions are the same as previously derived and must remain separate entities in the combined program. In each case, a determination is made to increment or decrement the variables $X$ and $Y$. In general, $N$ incrementation or decrementation will be performed upon one variable for each increment or decrement of the other variable, where $N$ is calculated by the respective discriminant function.

Typical program length on a six-microsecond, 64-Instruction Set computer, by function:
The size of the output block set aside for storage of the calculated points depends upon the system in question. For example, if the converter being driven requires three computer output words to describe a symbol, then 1200 storage locations would describe 400 symbols to the converter. The size of this storage block is one reason for considering hardware methods of graphics generation as an alternative to software.

The running time for generation of any given segment is directly proportional to the dimensions of that segment since the relative size determines the number of iterations required and all computer instructions execute in equal time.

In the construction of complex figures from simple graphic segments many geometric constructions (Tangents, Normals, etc.) appear often enough to warrant inclusion in the Operating system. In general, this amounts to calculating a new set of defining parameters based upon the segment (or segments) already generated and new segment to be generated.

The following have been investigated and found to produce desired results with minimum software (Figure 12).

1. Head-to-Tail straight lines.
2. Parallel lines.
3. Perpendicular line from a Point \( X_p, Y_p \) on a line \( L \).
4. Perpendicular line from a Point \( X_p, Y_p \) to a line.
5. Tangent and Normal to a circle.
6. Tangent and Normal to a parabola.

It has been found that if the desired segments are defined in terms of critical parameters much of the manipulative software developed for direct writing systems is applicable, with the segments defined by the manipulated points being generated by the special algorithms.

The entire question of number of segments, hardware vs software generation, amount of manipulative capa-

The references are as follows:

1. F G Stockton
   X-Y move plotting
   C A C M Vol 6 No 4 April 1963
2. Beckenback & Bellman
   An introduction to inequalities
   Random House New York 1961
3. J H Wilkenson
   Rounding errors in algebraic processes
   Prentice Hall Series in Automatic Computation
   Englewood New Jersey 1963
4. K E Iverson
   A programming language
   K Wiley and Co New York 1962
5. F Gruenberger
   Computer graphics
   Thompson Book Co Washington D C 1967
6. L L Briner
   A graphic information processing system
   IBM TR-21 197 March 1966
7. I E Sutherland
   Sketchpad: A man machine graphical communication system
   Technical Report # 296 Lincoln Labs MIT Jan 1963
8. A D Falkoff & Iverson
   The APL/360 terminal system
   IBM Research Report RC-1922 Oct 16 1967
9. A D Falkoff & Iverson
   The APL terminal system: instructions for operation
   IBM Research Yorktown Heights New York 1966