THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 51, No. 4, April, 1972
Printed in U.S.A.

Perspective Drawing of Surfaces With Hidden Line Elimination

By N. Y. GRAHAM

(Manuscript received February 18, 1972)

An efficient computer algorithm is described for the perspective drawing of a wide class of surfaces. The class includes surfaces corresponding to single-valued, continuous functions which are defined over rectangular domains. The algorithm automatically computes and eliminates "hidden lines." The number of computations in the algorithm grows linearly with the number of sample points on the surface to be drawn. An analysis of the algorithm is presented, and extensions to certain multi-valued functions are indicated. The algorithm is implemented and tested on two different computers: a large central computer with hard-copy capability, and a small laboratory computer affording interactive use. Running times are found to be exceedingly efficient on both machines. Interactive implementation of the algorithm, with on-line scope display and view-point control, enables effective and rapid examination of a surface from many perspectives.

I. INTRODUCTION

The general problem of efficiently and meaningfully displaying three-dimensional objects in two dimensions is central to computer graphics. In particular, the "hidden line problem" of drawing objects in perspective while eliminating line segments not visible from the viewing point, is one of long standing. In recent years various algorithms dealing with this problem have appeared in literature.[1-7] Some of these algorithms entail prohibitively long computation time (relative to the number of data points) or have large storage requirements; some have both of these undesirable characteristics.

In this paper a detailed description will be given of a highly efficient algorithm for the perspective drawing of an arbitrary surface which corresponds to a single-valued continuous function defined over a rectangular domain, with elimination of hidden lines. The vantage point from which the surface is viewed may be any point not on the

surface. Some generalizations of the algorithm to nonrectangular domains and to multiple-valued functions will be indicated.

This algorithm has been implemented on both a large and a small computer (Honeywell 6070, DDP 516) with considerably shorter computation time than that based upon previous algorithms for drawing similar surfaces. A very useful feature of the implementation on the DDP 516 (a 16K laboratory machine with 16-bit word length) is that the program may be run interactively, allowing the user to select varying vantage points and rapidly display different views of a given surface.

II. STATEMENT OF THE PROBLEM

Given a three-dimensional surface S defined over a rectangular domain R, and a vantage point V not on S (Fig. 1), we may assume that R is centered about the origin of the x, y-plane and is oriented so that its sides are parallel to the x-axis and y-axis, respectively. (This may be accomplished without loss of generality by a suitable translation and rotation of R, together with V). The plane P, containing the perspective image S' of S, is chosen to be perpendicular to the line joining V and the origin. A rectangular coordinate system with x', y'-axes is chosen on P which preserves the original vertical direction

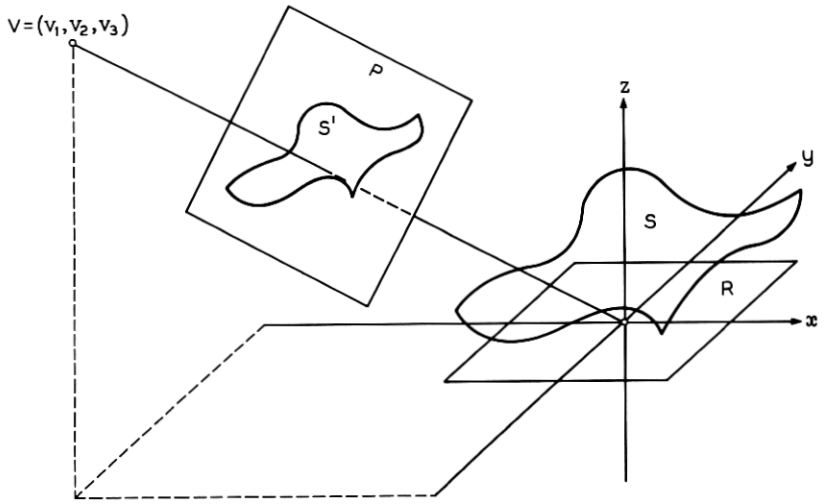

Fig. 1—Perspective projection of surface S on plane P.

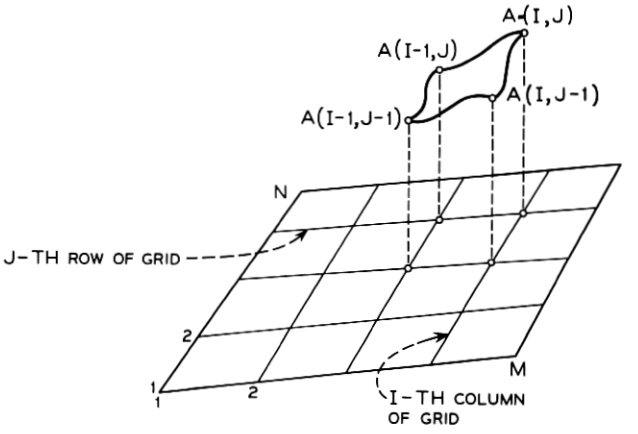

Fig. 2—Surface patch of one grid subrectangle.

of S. That is, the y'-axis on P must be the projection of the original z-axis. This latter choice is necessary, as we shall see later. (Relative to the rectangular system on P, the perspective image on P of each point of S has a well-defined pair of coordinates. The evaluation of these coordinates will be given in the Appendix.)

The surface S will first be quantized to an $M \times N$ rectangular grid of points on the domain R. Figure 2 shows the grid lines on R, partitioning R into subrectangles. The part of S defined over one subrectangle is shown. It will be referred to as a "surface patch" of S. After quantization, only the four points of the patch defined at the four vertices of the subrectangle are known, and linear interpolation of the function will be assumed between adjacent grid points, that is, between adjacent points in the same row and adjacent points in the same column of the grid. (Thus, S need not be explicitly defined by a mathematical function; the data for S may be given by a rectangular array of points corresponding to a rectangular grid of points on the domain.) The behavior of S in the interior of each subrectangle of the grid will be ignored. Thus, each surface patch of S will be represented by its four linearly interpolated edges in three-dimensional space. The image of these edges on the projection plane is a four-sided polygon (Fig. 3). For each surface patch of S, only the visible line segments of its edges will be drawn.

Intuitively, the surface should be thought of as an opaque elastic membrane stretched over a rigid frame consisting of all the linearly

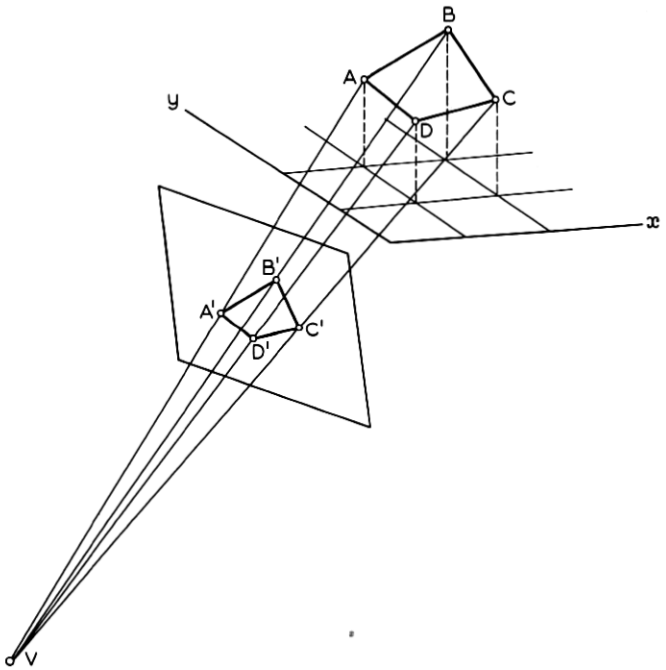

Fig. 3—Projection of linearly interpolated surface patch.

interpolated edges of the surface patches. Then the problem is to give a perspective drawing of the frame which eliminates line segments of the frame hidden by the opaque membrane from the vantage point. (Note that the vertical projection of the frame onto the x, y-plane coincides with the grid lines on R.)

III. BASIC IDEAS OF THE ALGORITHM

The algorithm for determining the visibility of any given edge is based upon two ideas. The first one consists of choosing a particular ordering of the surface patches of S so that no part of any patch occurring earlier in the ordering is obscured from the vantage point by any part of a patch occurring later. The surface is to be drawn according to such an ordering, one patch at a time. Then, a line segment on any edge of a patch under consideration is hidden from the vantage point if and only if its image on the projection plane lies inside a region of the plane already covered by the images of earlier patches.

This particular ordering of the surface patches has the further

property that, for a surface S which corresponds to a single-valued continuous function over a rectangular domain, the region of the projection plane being covered by the emerging image of S will grow contiguously as each patch in the ordering is drawn. And because of the preservation of the vertical direction of the surface, at each stage of the drawing of S, this "hidden" region of the plane can be precisely bounded between two piecewise linear continuous functions. The determination of the visibility of any edge of a surface patch is thus reduced to the problem of deciding which part of its image on the projection plane lies between these two functions. This delineation of the hidden region of the projection plane, after each patch is drawn, by means of two piecewise linear continuous functions, constitutes the second idea behind the algorithm.

Remark: If the y'-axis on the plane of projection is not chosen to be the perspective image of the original z-axis, then the image of the surface may be "tilted" so that it will be impossible to delineate its boundary by using only *two* functions.

IV. ORDERING OF THE SURFACE PATCHES

The fact that it is possible to order the surface patches of S so that earlier ones are not obscured by later ones depends on the following observation:

Suppose A and B are any two points of the surface such that A obscures B from V. Geometrically, this means that A, B, and V lie in a straight line, and the distance from A to V is less than the distance from B to V. Then it can be easily shown that the vertical projections A_0 , B_0 , V_0 of the points A, B, V, respectively, onto the x, y-plane satisfy this relationship also. That is, they are collinear, and A_0 is closer than B_0 to V_0 .

This observation is the key to the ordering of the surface patches. First, consider the case in which the vertical projection $V_0 = (v_1 , v_2 , 0)$ of the vantage point onto the x, y-plane is southwest of the domain, as shown in Fig. 4. [That is, $v_1 \leqq x_1$ and $v_2 \leqq y_1$, where $(x_1 , y_1 , 0)$ is the lower left-hand corner of the domain.] An immediate consequence of the above observation is that if $A = (a_1 , a_2 , a_3)$ obscures $B = (b_1 , b_2 , b_3)$ from V, then $a_1 < b_1$ and $a_2 < b_2$; i.e., A_0 must be southwest of B_0 (given the southwest location of V_0). This suggests the following rule for determining the relative order of points of S, given the southwest location of V_0 : Let A and B be any two points of S. If one of them, say A, is southwest of the other (more precisely, if A_0 is south-

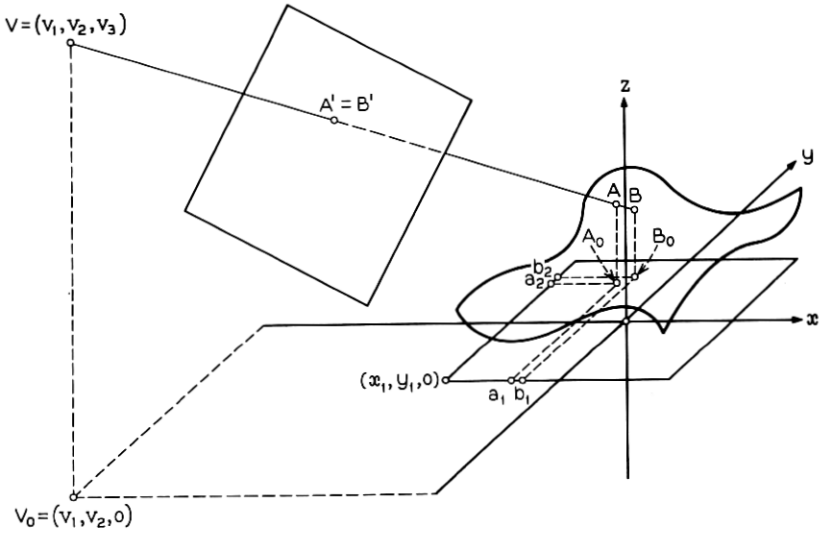

Fig. 4—Points of the surface with A obscuring B.

west of B_0), then A must precede B. Otherwise, A and B may be ordered independently of each other.

There are various ways to order the surface patches of S so that this rule for the relative order of points is satisfied, and hence, so that earlier patches are not obscured by later patches. Figure 5 illustrates one possible ordering of the subrectangles of R so that the induced

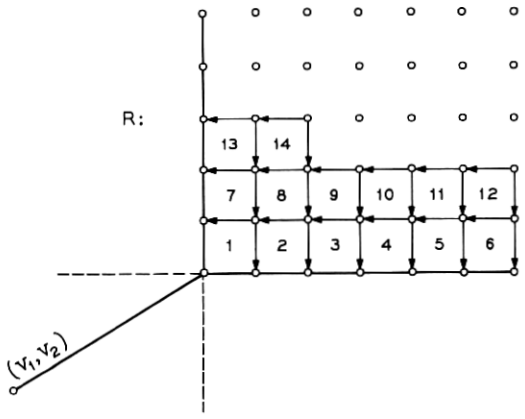

Fig. 5—One possible ordering of subrectangles.

ordering of the patches of S has this property; i.e., they are ordered by rows, from left to right in each row, beginning with the front row (relative to the southwest location of V_0). It is obvious that, under this ordering of the patches, no point of an earlier patch will be obscured by a point of a later patch, since no point of a later patch is southwest of any point of an earlier patch.

For other "corner" locations of V_0 (i.e., northwest, northeast, and southeast of the domain) analogous orderings of the surface patches may be made. Instead of discussing suitable orderings of the surface patches for other locations of V_0 , we shall proceed to describe the algorithm in detail for the special case in which V_0 is southwest of the domain, and then show how the other cases may be reduced to cases involving only "corner" locations of V_0 .

V. THE ALGORITHM IN DETAIL

Let S be a surface defined over a rectangular domain R of the x, y-plane. Assume S is to be viewed in perspective from a vantage point V whose vertical projection V_0 onto the x, y-plane is southwest of R. Given an $M \times N$ rectangular grid of points on R, for $I = 1, 2, \cdots, M$ and $J = 1, 2, \cdots, N$, let $A(I, J)$ denote the "sample point" on S defined at the grid point on R located at the Ith column and Jth row of the grid. For $I = 2, \cdots, M$ and $J = 2, \cdots, N$, let $S(I, J)$ denote the surface patch with vertices $A(I, J)$, $A(I, J - 1)$, $A(I - 1, J)$, $A(I - 1, J - 1)$. (See Fig. 2.) By above considerations, the patches may be ordered as follows, and drawn according to this ordering, one at a time:

$$S(2, 2), S(3, 2), \cdots, S(M, 2),$$
$$S(2, 3), S(3, 3), \cdots, S(M, 3),$$
$$\vdots$$
$$S(2, N), S(3, N), \cdots, S(M, N).$$

Now, consider the sequence of points $A(1, J)$, $J = 1, \cdots, N$, defined along the first column of grid points. The line segments joining these points in consecutive order will be called the "leading left edge" of S. Similarly, the line segments joining the sequence of points $A(I, 1)$, $I = 1, \cdots, M$ in consecutive order will be called the "leading front edge" of S. As a consequence of the observation made at the beginning of Section IV, both leading edges are entirely visible from V. Thus, we may begin by drawing all the line segments of the leading edges.

Then the drawing of each successive patch $S(I, J)$ in the ordering may be completed by adding the visible segments of just two of its four edges, namely its "back edge," joining $A(I, J)$ and $A(I - 1, J)$, and its "right edge," joining $A(I, J)$ and $A(I, J - 1)$, since the other two edges will have already been considered in connection with earlier patches or with one of the leading edges.

Figure 6a gives an example of a surface defined over a 5×3 grid of points. The 5×3 array of sample points on the surface gives rise to two rows of patches, four in the front row and four in the back row. Figure 6b shows the conjunction of the leading left edge and the leading front edge of this same surface.

Figures 7a through 7h show successive patches being added to the drawing of the surface. With each patch that is being added, L_1 denotes its "back" edge, and L_2 denotes its "right" edge. Dotted lines indicate hidden segments which are not drawn.

As indicated earlier, the determination of the visibility of each new edge under consideration will involve two piecewise linear continuous functions. Intuitively, one of these functions, Max, will be used to define the path of the upper perimeter of the region of the projection plane thus far covered, and the other function, Min, will be used to define the path of the lower perimeter. At this point, it would be useful to identify the grid of the plotting device with a rectangle in the projection plane which circumscribes the perspective image of the surface. (This corresponds to a scaling and translation of the coordinates of the image points so that they fall within the dimensions of the plotting grid.) Then, in order to optimize the precision in delineating the hidden region of the projection plane, or equivalently, the hidden region of the plotting grid, the number of breakpoints of each of the functions Max and Min is chosen to coincide with the number of horizontal units of the plotting grid.

Before plotting begins, we artifically set Max (k) = Min (k) = -1,

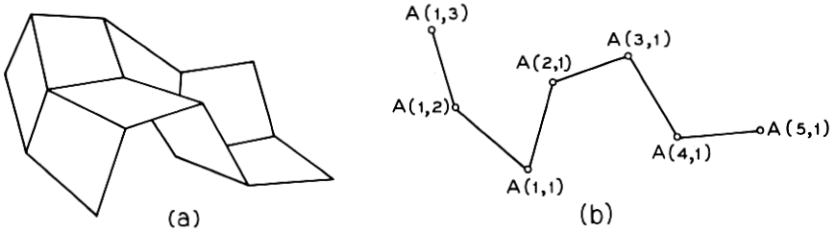

Fig. 6—Surface and its leading left and front edges.

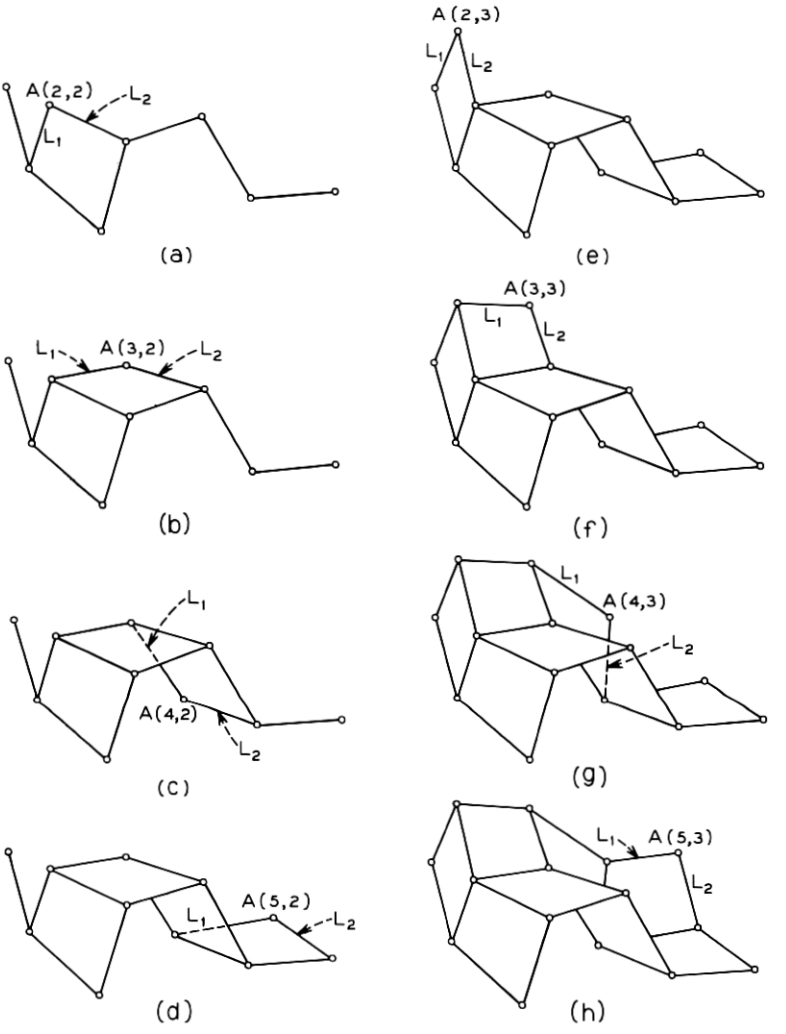

Fig. 7—Successive patches of surface being drawn.

for $k = 1, 2, \cdots, K_x$, where K_x is the horizontal dimension of the plotting grid. Then, after each line segment is drawn, Max or Min (possibly both) will be modified to reflect the change in the shape of the emerging image of S. We begin with the line segments on the leading edges of S. Let A and B be any two consecutive "sample points" of the leading left edge or of the leading front edge, that is, any two

consecutive points in the following sequence (see Fig. 2):

$$A(1, N), A(1, N-1), \cdots, A(1, 1), A(2, 1), \cdots, A(M, 1).$$

Let A_x , A_y be the (scaled and translated) plotting coordinates of A, and B_x , B_y the plotting coordinates of B. First, the line segment joining the points (A_x , A_y) and (B_x , B_y) is drawn. Then both Max and Min are redefined between A_x and B_x by setting

$$\text{Max}\,(A_x) = \text{Min}\,(A_x) = A_y$$

$$\text{Max}\,(B_x) = \text{Min}\,(B_x) = B_y$$

and linearly interpolating both Max and Min between A_x and B_x . (In Fig. 8, K_x and K_y refer to the dimensions of the plotting grid, in the horizontal and vertical directions, respectively.)

After the above procedure is carried out for every pair of consecutive points in the sequence, the leading edges will be completely drawn, and their path will be defined by Max and Min between the two endpoints. Note that the region bounded between Max and Min has area zero, reflecting the fact that no patch has yet been drawn.

We now proceed to draw the surface patches. Each $S(I, J)$ in the ordering, beginning with $S(2, 2)$ will be uniformly processed as follows: Let L_1 denote the edge between $A(I, J)$ and $A(I-1, J)$, L_2 denote the edge between $A(I, J)$ and $A(I, J-1)$. For each of these edges, we must first determine its visible segments, then draw only those segments, and modify Max or Min to reflect the change in the shape of the hidden region (the entire edge may be hidden, of course).

First, let $L = L_1$ and $A = A(I, J)$, $B = A(I-1, J)$, the endpoints of L_1 . The following criterion is used to determine the visibility of any point C on L: If C_x , C_y are the plotting coordinates of C, then

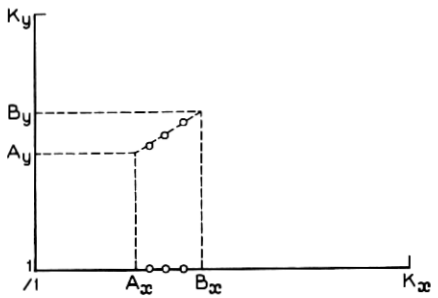

Fig. 8—Linear interpolation of Max and Min.

C is visible if and only if $C_y \geqq \text{Max}\ (C_x)$ or $C_y \leqq \text{Min}\ (C_x)$. This visibility criterion is applied to each of the endpoints A and B. (Let A_x , A_y be the plotting coordinates of A; B_x , B_y those of B.) There are four possibilities:

(*i*) A, B are both visible. Then all of L is assumed to be visible. (This assumption is made to reduce computation time. It may happen that some segments of hidden lines will be incorrectly drawn if the surface has sharp "spikes" in the foreground. This problem can be circumvented by starting with a finer grid on R.) The line segment joining (A_x , A_y) and (B_x , B_y) is drawn. If $A_y \geqq \text{Max}\ (A_x)$, Max is linearly interpolated between A_x and B_x by setting $\text{Max}\ (A_x) = A_y$, $\text{Max}\ (B_x) = B_y$. Otherwise, we must have $A_y \leqq \text{Min}\ (A_x)$, and Min is similarly modified between A_x and B_x , with $\text{Min}\ (A_x) = A_y$, $\text{Min}\ (B_x) = B_y$.

(*ii*) A and B are both hidden. Then for the same reasons as in case (*i*), we assume all of L is hidden and no line segment is drawn. Max and Min are left unchanged, and we proceed to the next edge.

(*iii*) A is hidden and B is visible. A search is made along the line joining A and B, starting at A, for the first visible point C (discrete steps are taken in the horizontal direction of the plotting grid). The segment joining C to B is assumed to be entirely visible and is drawn. If $C_y \geqq \text{Max}\ (C_x)$, Max is linearly modified between C_x and B_x with $\text{Max}\ (C_x) = C_y$, $\text{Max}\ (B_x) = B_y$. Otherwise, Min is modified between C_x and B_x .

(*iv*) A is visible and B is hidden. A search is made along the line joining A and B for the first visible point C, starting from the hidden point B. The rest is analogous to case (*iii*).

For each surface patch $S(I, J)$, this procedure is carried out with $L = L_1$ and $B = A(I - 1, J)$; then the procedure is repeated with $L = L_2$ and $B = A(I, J - 1)$. This completes the description of the algorithm for the special case in which V_0 , the vertical projection of the vantage point onto the x, y-plane, is southwest of the domain R.

If V_0 is in any of the other "corner" regions of the x, y-plane (indicated by NW, NE, and SE in Fig. 9a), we may rotate the data defining the surface (either the mathematical function or the rectangular array of sample points) by 90, 180, or 270 degrees, respectively, and apply the algorithm as described for V_0 in the SW region of the x, y-plane. Another approach is to change the ordering of the surface patches and to define new leading edges to suit the other locations of V_0 .

If V_0 is in any of the regions directly west, north, east, or south of R

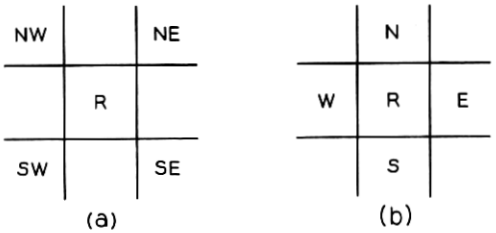

Fig. 9—Locations of vertical projection of vantage point.

(Fig. 9b), we may partition R into two subrectangles R_1 and R_2 (Fig. 10a), and apply the algorithm to each of R_1 , R_2 , since V_0 is in a "corner" location relative to each subrectangle. The two parts of the surface, corresponding to R_1 and R_2 , will have to be "pieced together" along the line of partition, and this may be done without "flaws" by augmenting the sample points of the surface with the set of points defined at the intersection of the partition line with the grid lines on R.

Finally, if V_0 lies inside the domain R, we may partition R into four subrectangles (Fig. 10b) and apply the algorithm to each of R_1 , R_2 , R_3 , R_4 , with V_0 in a "corner" location.

Remark: Note that under the assumption that the plane of projection P be perpendicular to the line joining V and the origin, if V_0 coincides with the origin, then the projection of the z-axis from V onto P is a point. In this case, the y'-axis on P must be chosen to be the image of some other line. See the Appendix.

Figures 11, 12, and 13 are sample drawings generated on the Honeywell 6070 computer with the Stromberg–Carlson 4060 microfilm plotter. Each surface was evaluated at a 64 × 64 grid of points, for a total of 4096 data points. The average run time was 9 seconds for each surface. (Part of the 9 seconds was used in computing the function values and

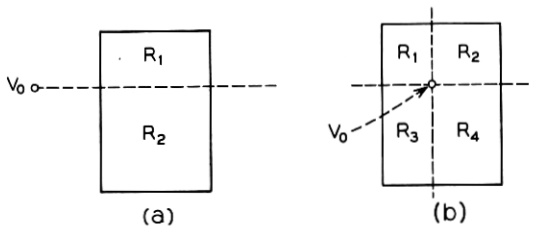

Fig. 10—Partitioning of domain for non-corner locations of V_0.

plotting coordinates at each point of the 64 × 64 grid. In fact, these computations were performed twice, once for determining the scale factors and again for the actual plotting of the surface.) Total storage for generating microfilm output of each surface was 16K.

It is clear that using this algorithm, computation time should vary linearly with the number of data points, since each data point is processed by evaluating its plotting coordinates IX, IY, and then comparing IY with just two values, Max (IX) and Min (IX).

Economically generated movies have also been produced on both the Honeywell 6070 and the DDP 516 computers. These movies show surfaces in rotation and surfaces undergoing continuous change in shape. The average run time on the Honeywell 6070 for making such a movie with a 32 × 32 grid and viewed from 240 distinct vantage points was 7.4 minutes, or about 1.85 seconds per frame. The corresponding run time on the DDP 516 was 7.5 seconds per frame. (For these movies, the scale factors were precomputed so that function values and plotting coordinates were evaluated only once for each frame of the movie.)

For a rough comparison of computation times for hidden-line elimination drawings based upon other algorithms (and implemented on other

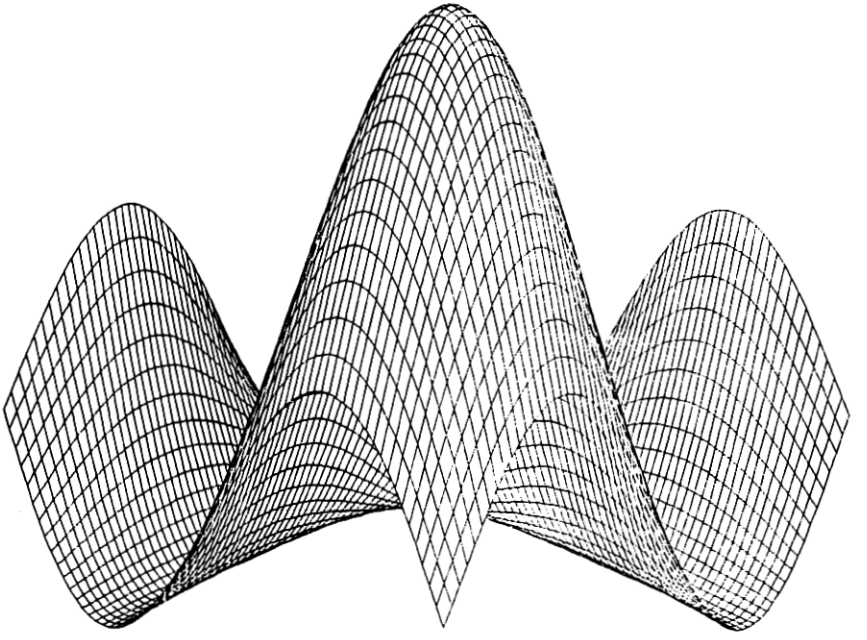

Fig. 11—$f(x, y) = (\sin(x) - 1)(\sin(y) - 1)$, $-\pi \leqq x, y \leqq \pi$.

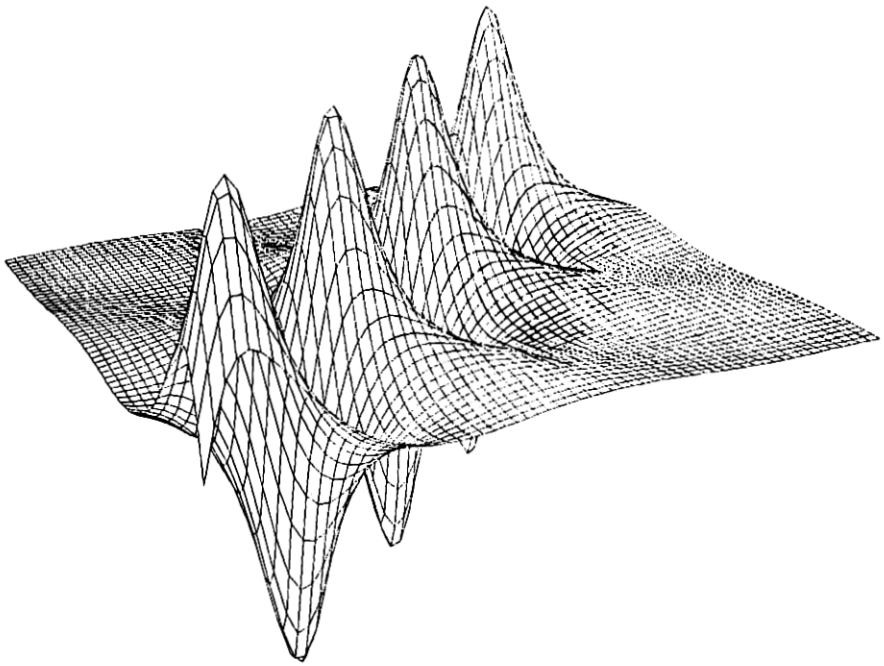

Fig. 12—$f(x, y) = \sin(x + y)/(1 + (x - y)^2)$, $-2\pi \leqq x, y \leqq 2\pi$.

computers) see Refs. 1 through 7. In particular, see Refs. 2 and 3 for a comparison with two other algorithms which deal with single-valued functions defined over a rectangular domain. Using the algorithm described in Ref. 2, computation time increases quadratically with the number of data points, since the visibility of each data point is determined by an exhaustive comparison with every other data point. Although computation time increases linearly using the algorithm of Ref. 3, the storage required is large in order to obtain high-quality drawings. This is because the "plotting page" is subdivided into rectangles, and, of all possible line segments of the surface whose images lie in any given rectangle, only the one closest to the vantage point is drawn in that rectangle. Thus, in order to avoid a "sketchy" drawing, a fine subdivision of the "plotting page" is necessary and a large array is required to store the information in each rectangle of the subdivision. (The algorithm of Ref. 3, however, is applicable to a larger class of surfaces than the algorithm described in this paper. For example, it can be used to draw surfaces corresponding to functions which are not single-valued, e.g., intersecting cylinders.)

VI. GENERALIZATIONS

6.1 *Convex Domains*

The algorithm may be generalized to an arbitrary surface S defined by a single-valued continuous function over a *convex* domain R, by artificially extending the defining function f to a rectangular domain R' containing R. The ideas of the algorithm are then applied to the surface S' defined by the extended function f' over R', with the following modifications: First, the sample points along the leading edges of S' must be replaced by a subset of points defined at the intersection of the boundary of R with the grid lines on R', and special consideration must be given to drawing the lines connecting these points. Second, a test must be made of sample points of S' to determine if they belong to S, so that only line segments that are part of S are drawn. One way to accomplish this is by setting $f'(x, y) = z_0$ for all (x, y) in $R' - R$, where z_0 is a number outside the range of values of f on R. (Convexity of the domain is necessary in order to bound the emerging images at each stage of the drawing between *two* functions.)

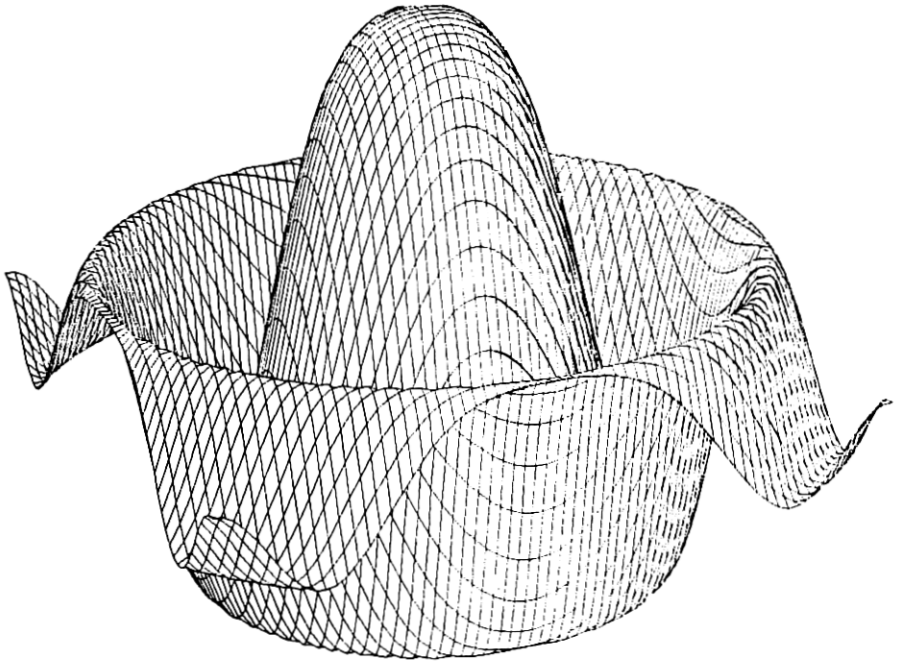

Fig. 13—$f(x, y) = \cos(x^2 + y^2)/e^{0.2(x^2+y^2)}$ $-2.5 \leqq x, y \leqq 2.5$.

6.2 *Multi-Valued Functions*

The algorithm may also be extended to certain multi-valued functions. For example, if the surface is a sphere, it is first partitioned by a horizontal plane into two single-valued pieces, and if the vantage point is above the horizontal plane, the algorithm is first applied to the upper hemisphere, and then, without reinitializing Max or Min, the algorithm is applied to the lower hemisphere.

6.3 *Bivariate Histograms*

With minor modifications the algorithm may be adapted to the perspective drawing of bivariate histograms, with elimination of hidden lines. Figures 14a through 14d show four different perspectives of a bivariate histogram generated by the same data. The computation time on the Honeywell 6070 was 1.6 seconds for each of the four drawings.

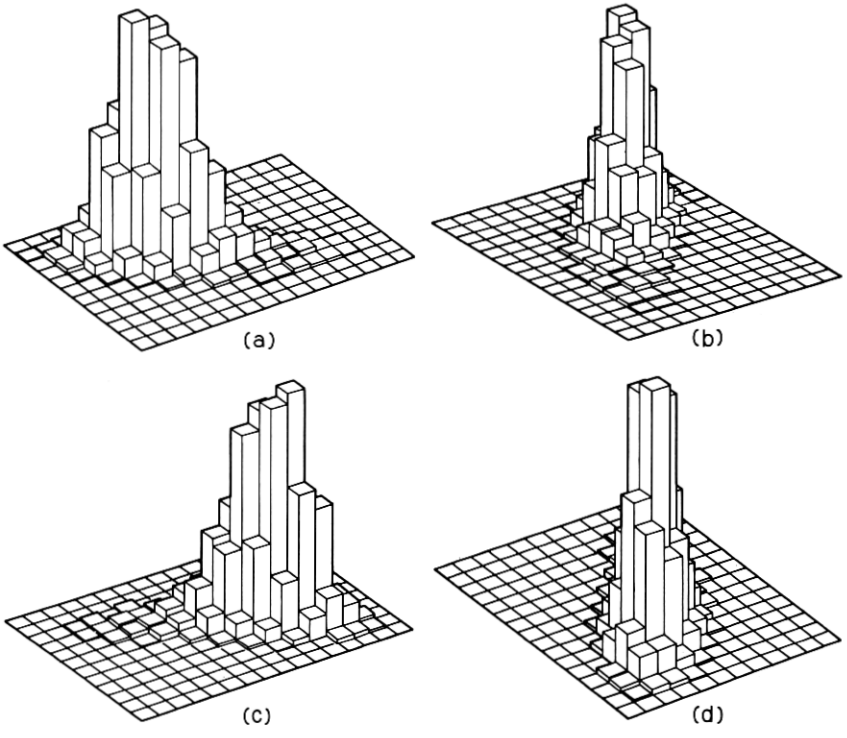

Fig. 14—Bivariate histograms: (a) V_0 in SW corner. (b) V_0 in SE corner (c) V_0. in NE corner. (d) V_0 in NW corner.

APPENDIX

Evaluation of Rectangular Coordinates of Image Points

Let S be a surface defined over a domain centered at the origin of the x, y-plane, and let $V = (v_1 , v_2 , v_3)$ be a point not on S. In order to simplify the calculation of the coordinates of image points and hence reduce computation time, a special plane P^*, normal to $\bar{V}$ and passing through the origin, will be used as the plane of projection. (In Fig. 1, a plane P parallel to P^* is shown. Note that the image of S on P differs from the image of S on P^* only by a scale factor.)

Let $A = (a_1 , a_2 , a_3)$ be any point on S, and let $B = (b_1 , b_2 , b_3)$ denote its image point on P^*. The plotting coordinates of A depend on the two-dimensional coordinates (u, v) of B with respect to an orthogonal system on P^*. If $\bar{X} = (x_1 , x_2 , x_3)$ and $\bar{Y} = (y_1 , y_2 , y_3)$ are the positive unit vectors of the coordinate system on P^*, then the scalars u and v must satisfy the following relationship:

$$u\bar{X} + v\bar{Y} = \bar{B}. \tag{1}$$

Since B is collinear with the points A and V, then

$$\bar{B} = \bar{V} + r(\bar{A} - \bar{V}), \quad \text{for some scalar } r. \tag{2}$$

Thus, eq. (1) may be replaced by

$$u\bar{X} + v\bar{Y} = \bar{V} + r\bar{A} - r\bar{V}. \tag{3}$$

Taking the dot product of both sides of eq. (3) with $\bar{X}$, we have

$$u\bar{X}\cdot\bar{X} + v\bar{Y}\cdot\bar{X} = \bar{V}\cdot\bar{X} + r\bar{A}\cdot\bar{X} - r\bar{V}\cdot\bar{X}.$$

Since $\bar{X}\cdot\bar{X} = 1$ and $\bar{Y}\cdot\bar{X} = \bar{V}\cdot\bar{X} = 0$ ($\bar{V}\cdot\bar{X} = 0$ because $\bar{X}$ on P^* and P^* normal to $\bar{V}$), we have

$$u = r\bar{A}\cdot\bar{X}. \tag{4}$$

Similarly, taking the dot product of both sides of eq. (3) with $\bar{Y}$, we have

$$v = r\bar{A}\cdot\bar{Y}. \tag{5}$$

The scalar r is determined by taking the dot product of both sides of eq. (2) with $\bar{V}$. Since $\bar{B}$ lies on P^*, then $\bar{B}\cdot\bar{V} = 0$, and we have

$$r = \bar{V}\cdot\bar{V}/\bar{V}\cdot(\bar{V} - \bar{A}). \tag{6}$$

It remains to determine the coordinates of the unit vectors $\bar{X}$, $\bar{Y}$.

As mentioned in Section II, the y'-axis on P^* must be the perspective

image of the original z-axis. Note, however, that if V is on the z-axis, then the perspective image from V of the z-axis is a single point, which of course cannot be used as an axis line. In this case, we must compromise by either moving the vantage point slightly off the z-axis or choosing the y'-axis on P^* to be the image of a slightly "askew" z-axis. For the rest of this discussion we assume that V is not on the z-axis.

Let $Z = (0, 0, 1)$. Then $\bar{V}$, $\bar{Y}$, $\bar{Z}$ lie on the same plane, because $\bar{Y}$ is the image from V of some scalar multiple of $\bar{Z}$. Thus $\bar{Z}$ is a linear combination of $\bar{V}$ and $\bar{Y}$. Since $\bar{X} \cdot \bar{V} = 0$ and $\bar{X} \cdot \bar{Y} = 0$, then $\bar{X} \cdot \bar{Z} = 0$. The normalization of any vector $\bar{X}'$ satisfying the equations $\bar{X}' \cdot \bar{V} = 0$ and $\bar{X}' \cdot \bar{Z} = 0$ will be a unit vector on the x'-axis of P^*. $\bar{X}' = \overline{(-v_2, v_1, 0)}$ satisfies these two equations. Let $d_1 = \sqrt{v_1^2 + v_2^2}$. Then

$$\bar{X} = \overline{(-v_2/d_1 , v_1/d_1 , 0)}$$

is the positive unit vector on the x'-axis of P^*. The coordinates of $\bar{Y}$ are similarly determined from the equations $\bar{Y} \cdot \bar{X} = 0$ and $\bar{Y} \cdot \bar{V} = 0$. Let $d = \sqrt{v_1^2 + v_2^2 + v_3^2}$. Then

$$\bar{Y} = \overline{(-v_1 v_3 , -v_2 v_3 , d_1^2)/dd_1}$$

is the positive unit vector on the y'-axis of P^*. Substituting these values of $\bar{X}$ and $\bar{Y}$ back into eqs. (4) and (5), we have

$$u = r(a_2 v_1 - a_1 v_2)/d_1 \tag{7}$$

$$v = r(-a_1 v_1 v_3 - a_2 v_2 v_3 + a_3 d_1^2)/(dd_1), \tag{8}$$

where $r = d^2/(d^2 - (a_1 v_1 + a_2 v_2 + a_3 v_3))$, as defined in eq. (6).

The above expression for v is algebraically equivalent to the following one, which involves half as many arithmetic operations:

$$v = (v_3 + r(a_3 - v_3)) \cdot d/d_1. \tag{9}$$

Thus, given a point A on S, the rectangular coordinates (u, v) of its image point are evaluated according to eqs. (7) and (9).

REFERENCES

1. Roberts, L. G., "Machine Perception of Three-Dimensional Solids," Technical Rep. No. 315, Lincoln Laboratory, M.I.T., (May 1963).
2. Kubert, B., Szabo, J., and Giulieri, S., "The Perspective Representation of Functions of Two Variables," J. Assn. for Computing Machinery, *15*, No. 2, (April 1968).
3. Kubert, Bruce R., "A Computer Method for Perspective Representation of Curves and Surfaces," Technical Rep. No. TR-0200-2, Aerospace Corporation, (December 1968).

4. Galimberti, R., and Montanari, U., "An Algorithm for Hidden Line Elimination," Commun. Assn. for Computing Machinery, *12*, No. 4, (April 1969).
5. Loutrel, P., "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," IEEE Trans. Computers, *C-19*, No. 3, (March 1970).
6. Encarnacao, J. L., "A Survey of and New Solutions for the Hidden-Line Problem," Symp. on Interactive Computer Graphics, (October 1970).
7. Weiss, Ruth A., "BE VISION, A Package of IBM 7090 Fortran Programs to Draw Orthographic Views of Combinations of Plane and Quadric Surfaces," J. Assn. for Computing Machinery, *13*, No. 2, (April 1966).