Three-Dimensional Data Input by Tablet

IVAN E. SUTHERLAND, MEMBER, IEEE

Abstract—A large-area multiple-pen tablet system for three-dimensional data input is described. The large tablet area provides space for simultaneous use of several views of the three-dimensional object being digitized. The multiple pens enable the user to indicate a single point simultaneously in two such views, thus defining the three-dimensional position of the point.

Five significant techniques are outlined. First, the large-area digitizing surface with multiple pens has proved to be an instrument very different from the more familiar single-pen small tablets. Second, a pair of two-dimensional positions is converted into a four-dimensional space and then back to three dimensions. Third, the specification of view areas, viewing directions, view positions, and coordinate axis is accomplished by giving examples directly in the viewing space rather than by specifying abstract viewing parameters. Fourth, an attitude about coordinate conversion using the inverse of a basis matrix is used throughout which automatically compensates for any tilt in the views on the tablet surface and any nonperpendicularity of the tablet axis. Fifth, the mathematics of converting from pairs of perspective views or pairs of photographs, while not new, is formulated simply in an appendix with several examples.

Introduction

The art of producing pictures of three-dimensional objects by computer has advanced remarkably in the past several years. The pictures now being produced are in perspective with hidden surfaces removed, and are often quite realistically colored. The objects portrayed may present a smoothly curved appearance; in fact, a degree of realism has been reached which makes it difficult to identify such pictures as computer output. The pictures included in the paper by Resch in this issue [7] and in Fig. 6 of this paper give some idea of what is possible.

Whereas the art of producing such pictures has advanced well, the art of getting suitable data for them into the computer has remained fairly primitive. This paper addresses the problem of preparing three-dimensional data for computer production of such pictures or for whatever other purpose one might want to present such data in computer-readable format. We have assembled a data input system which makes it quite easy and convenient to enter three-dimensional data from multiple two-dimensional views of the object to be digitized. With our system one can easily build a computer model of some object for which one has drawings or photographs.

Our system uses a large-area tablet to provide for simultaneous use of multiple views of the object. The user indicates points in three dimensions by pointing to them in two views simultaneously with two pens, one sometimes mounted in a stand (see Fig. 1). From the positions of the two pens, our programs can deduce which views are being used, and thus can apply the correct transformations to derive the three-dimensional position of the point indicated. The result is a system that makes it easy to construct computer models of objects for which multiple views are available.

The mathematics of going from two two-dimensional positions to the three-dimensional position of a point can account for either the use of orthographic projections (mechanical drawings) or perspective views (photographs). If photographs are used, of course, additional complexity is introduced, but in either case the principles of conversion are the same. While the mathematics involved are not new, we hope that our presentation of the two-dimensional to three-dimensional conversion process may be useful. We especially hope that our notions of defining the axes in each view and then inverting the axis matrix may prove a useful model for other tablet users, for it eliminates any need to align the views being digitized with the tablet and compensates for any nonperpendicularity of the tablet axes.

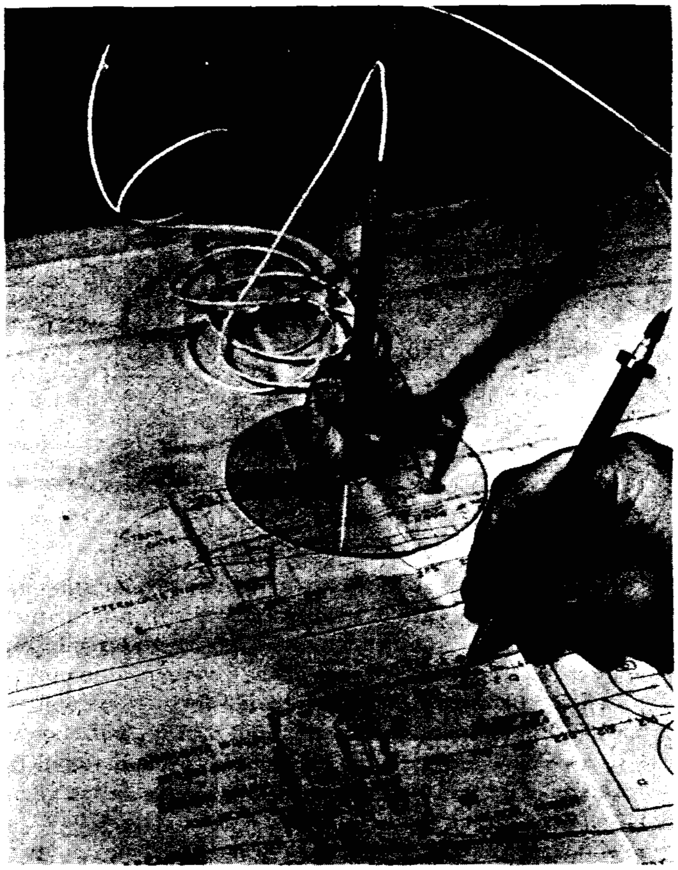

Fig. 1. Closeup of two digitizing pens. The small stand is often used to hold one pen if digitizing in a plane surface. Two views of a Navy ship are being digitized.

Tablet Input

The tablet we are using provides for digitizing the position of up to seven pens simultaneously on a tablet surface approximately 3 ft by 4 ft. The digitizing surface is embedded in the top of a formica-covered table which forms the main working surface for the operator. The position sensors are ordinary ball-point pens connected to an amplifier and filter unit by small coaxial cable. The tablet may be seen in Fig. 2. It is more fully described elsewhere [10].

Manuscript received July 9, 1973; revised October 29, 1973. The work reported in this paper was supported by the Office of Naval Research under Contract N000-14-72-C-0346.

The author is with Evans & Sutherland Computer Corporation, Salt Lake City, Utah 84112.

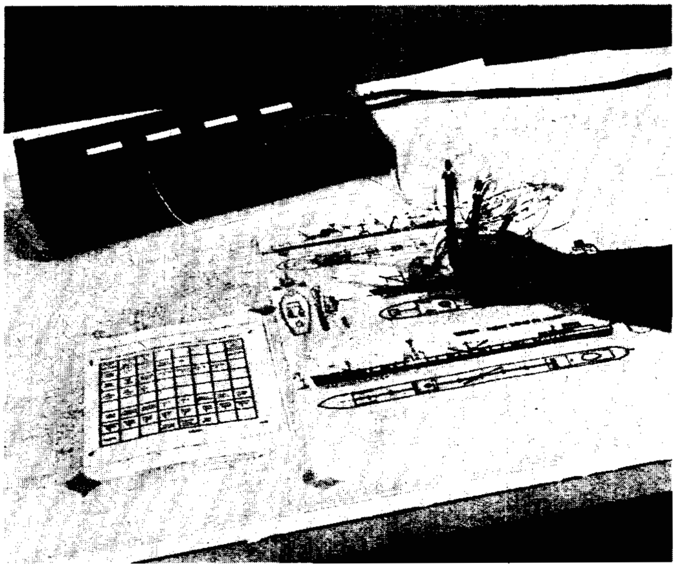

Fig. 2. The E&S tablet system. The tablet surface covers virtually the entire area of the table itself. The two pens are connected by coaxial cable to the dual-pen amplifier and filter unit seen near the top of the picture. Sensitivity of the "down" detection circuitry may be adjusted with the knobs on the tabletop unit. The meters indicate received signal strength. Sketches of ships are shown being digitized. Sketches of this quality can be used to produce objects which are identifiable though not, of course, dimensionally precise.

The tablet system provides the computer with information about the position of each of the pens. Coordinate information is given as a 15-bit number for the X position and a 15-bit number for the Y position of each of the several pens in operation. These coordinate numbers provide for a resolution of about 0.003 in and an accuracy of about 0.01 in. The coordinates of all pens are gathered simultaneously and stored in integrated-circuit registers to be made available to the computer on demand.

Height information about each pen is provided in two height bits which indicate roughly how far from the tablet surface the pen has been raised. Because these height bits are controlled by a proximity sensor for each pen rather than by a switch, the pen tip is firmly fastened to the pen barrel and the pens have a firm feel for easy writing. The first of the height bits, the "down" bit, is commonly adjusted to operate when the pen is within about 0.05 in of the tablet surface and is indistinguishable in operation from the "down" switch often contained in tablet pens. The other height bit operates when the pen is above some other threshold, typically one-half inch or so above the tablet surface. Above this threshold, coordinate data are ignored.

The large area of the tablet makes is possible for us to use it quite differently from small tablets, as shown in Fig. 2. We are free to devote areas of the tablet surface to various functions, being limited more by the length of our arm than by the tablet area available. It is a real luxury to assign an area to control functions and tape down a "menu" specifying the various functions, secure in the knowledge that there is plenty of area left for digitizing data. The tablet area lost to the pen control box and to reference books, papers, etc., holds about the same importance in a user's mind as similar area lost on the top of his desk. Because people are used to the space-management problems of paper work on desk-sized areas and because the large tablet surface eliminates the need to husband tablet area, it provides the user with a very comfortable working environment.

The multiple-pen input of the tablet makes it easy to think about digitizing simultaneously from several views. Of course, one could move a single pen back and forth between two views, assuming there was enough room for two views on a "conventional" tablet, but one's thinking process is assisted by actually pointing simultaneously to the two views. Input of two points which are logically related is actually simultaneous, making their association clearer to the user than would be possible with sequential association using a single pen. Moreover, with two input pens, it is substantially easier to keep track of where one is in the digitizing process, because the pens provide a center of attention in each of the views being used, thus guiding eye motion from one view to another.

Specifying View Areas and Axis Positions

For the digitizing system to know to which view the user is pointing, the user must initially specify the tablet area occupied by each view. The user makes this specification by pointing to the lower left corner and the upper right corner of each view area. The view areas are assumed to be rectangular spaces aligned with the tablet axis, although as we shall see, the views themselves which lie within these areas need not be aligned in any particular way. View areas may overlap; for example, an inset view may occupy a part of another view area provided the inset was specified first.

Before specifying the locations of viewing areas, the user specifies the location of two special control areas devoted to "menus" of command functions. The system treats the command areas exactly the same as other view areas, except that positions digitized from them are used to cause control functions rather than as data output. Because the command areas are specified first, they may overlap the areas of other views. Thus the two command areas may be placed wherever convenient on the tablet. Printed plastic overlays with holes in which to place the pen are sometimes used to define the command functions. A drawing of the command menu as shown in Fig. 2 will serve equally well.

The user specifies the viewing parameters of each view by pointing to the two-dimensional location of predefined three-dimensional points in each view. Thus, for example, the user points to the origin of the three-dimensional coordinate system as seen in each of the views. For orthographic projections, it is sufficient that he point also to some point on each of the three-dimensional axes as seen in each view he is using. By knowing where the three-dimensional axes lie in each view, the system can ascertain the kind of view involved. For example, if Z represents vertical motion, then the Z axis in a top view will lie at the same place as did the three-dimensional origin; noticing that the Z axis and the origin are colocated tells the system that this is a top view.

Moreover, if the view is not aligned with the tablet axes, the position of three-dimensional axes as seen in that view will inform the system of the misalignment. For example, if the X axis as seen in some view is not only to the right, but also slightly above the origin of coordinates as seen in that view, then the system can automatically compensate for the slight counterclockwise tilt of the view.

Giving, in each view, an example of a point at the three-dimensional origin and an example in each view of a point on each axis is a very simple and powerful way of specifying what kinds of views are being used. We call this kind of view specification "exemplary" specification of views because examples are given of what the viewing transformation did to certain known points. Exemplary specification is also used for perspective views, but at least six points are required rather than the four required for orthographic projections. In either case, the exemplary points chosen must not be coplanar in three dimensions.

Exemplary specification of a transformation appeared earlier in the 1968 design of the "clipping divider" [8]. The "window" to "viewport" transformation was specified by giving the locations at which the left, right, bottom, and top edges of a viewing space should lie on the screen. Similarly, subpictures were placed on a drawing by specifying where key points in them should lie rather than by specifying the transformation directly. It is often easier to make an exemplary specification than to know the parameters of the transformation directly. Moreover, as we shall see, the exemplary specification provides simply and cleanly for different types of views, relieving the user of any need to understand the mechanics of the transformations involved.

Selecting Coordinates from Multiple Views

With the two-pen digitizing system we can indicate a point's position in three dimensions by pointing simultaneously to its projection in each of two views. Because we have previously defined the tablet area occupied by each of the views, the system can easily determine which pen is pointing to which view, thus leaving the user free to change from one pair of views to another at will. Because we have also previously given an exemplary specification of the transformation involved in each view, the system can "undo" the transformation involved and compute the three-dimensional coordinates of the point in question. We have found it particularly convenient to have many views of an object available and to move pens freely from one view to another as convenience of digitizing dictates.

There are, of course, four measured tablet coordinates, two for each pen, but only three output coordinates. One must therefore be prepared somehow to eliminate one degree of freedom in going from the two views to the three-dimensional position of a point. For example, if one is digitizing mostly horizontal surfaces, such as the decks of a ship, from a plan (or top) view and an elevation (or side) view, one may want to ignore the horizontal position measured from the elevation view, because one is thus freed of the need to make the horizontal position of the elevation pen correspond to that of the plan pen, and an entire deck can be digitized with the elevation pen stationary anywhere at the right level for the deck in question. If, on the other hand, one is digitizing a mostly vertical surface such as the facade of a building, one may want to ignore one axis in the plan view and use both axes of the elevation view. One could, of course, obtain the third output coordinate as the average of two coordinates measured from the tablet, but we have found it preferable to ignore one kind of pen motion because one can then be less careful in positioning the second pen.

Unfortunately, the pen motion to be ignored depends on the particular views being used and the intent of the user. For example, in digitizing a ship model we discovered that whereas both coordinates should be taken from the plan view for decks, both should be taken from the elevation view when digitizing the ship's cranes. When the decks are finished and one is ready to start on the cranes, one must either redefine which coordinate to ignore or suffer very inefficient operation in which lateral positions for a surface are selected from an edge view of it.

Because users like to concentrate their attention mostly in one view, the second pen is often mounted in a small stand (Fig. 1), which will keep it upright and in position between the relatively infrequent times it is moved. Thus information about which pen is pointing to which view can tell the system which view shall have two axes used and which view only one axis used: pen A, without the stand, will always yield two coordinates; pen B, with the stand, will always yield only one coordinate. This arrangement works out quite well, except when digitizing from pairs of perspective views or photographs for which the directions of the axes are not clear.

When digitizing from pairs of photographs, one indicates the exact location of a point in each of the two views. The three-dimensional coordinate should presumably be computed in a symmetric way from the data thus measured, that is, the same three-dimensional value should be computed if the pens were interchanged. In the case of perspective views, it is not clear which motion of which pen should be ignored, and so the redundant degree of freedom in the pair of two-dimensional specifications is removed by computing a least mean-squares fit as outlined in the Appendix.

Specifying the Axis to Be Ignored

In order to explicitly indicate which axis is to be ignored, our system provides for output in four dimensions rather than three. The fourth dimension, which we refer to as W,[1] will be ignored when only three-dimensional output is desired, but it is given an explicit exemplary specification in each view just as are the other three. Thus if the user specifies the location of the W axis as horizontal in some view, he is effectively saying that in that view horizontal motion of the pen should be ignored. Thus the exemplary specification of five points (an origin and some point on each of four axes, for example) specifies everything about the view including any pen motions to be ignored.

Because one may sometimes want to use a pen motion in a view and sometimes ignore it, the exemplary specification of viewing parameters is made separately for each pen. Thus, whereas horizontal motion in some view may be important if detected with pen A (which the user is holding in his dominant hand), the same horizontal motion may properly be ignored if detected with pen B (which is mounted in the stationary holder). Thus a complete exemplary view specification includes the location of the origin and a point on each of the four axes as detected by each pen in each view. Entering such a specification requires digitizing nine points for each view, an operation which requires only a few seconds.

Undoing the Viewing Transformations

It might seem quite a hard task to "undo" the viewing transformations, given only an exemplary specification of them. This is not so, as we hope to show, if one thinks of the transformation as a matrix and has chosen a simple set of exemplary points. It is slightly more difficult if the transformation is more complex or the set of points not simply chosen, but still quite straightforward as the Appendix indicates.

If we think of the transformation between tablet coordinates and four-dimensional output coordinates in matrix form, the task of "undoing" the viewing transformation

[1] This use of four dimensions and the W coordinate must be distinguished from the use of homogeneous coordinates which will be seen later in the paper. Here W is a "real" coordinate which will simply be ignored in the final three-dimensional output. In homogeneous coordinates, the fourth coordinate is a scale factor which must be divided out, not ignored.

amounts to performing a matrix multiplication of the tablet data by some "magic" matrix M, which will undo the transformation.

$$\begin{array}{ccc} \text{Tablet Data} & \times\ \text{Matrix} = & \text{4-D Output} \\ [X_a \quad Y_a \quad X_b \quad Y_b] \times & [M] & = [X \quad Y \quad Z \quad W]. \end{array} \qquad (1)$$

Of course, M is not initially known, but the exemplary specification tells what it does to the example points:

$$\begin{array}{ccc} \text{Measured Example} & \times\ \text{Matrix} = & \text{4-D Example} \\ \text{Points} & & \text{Points} \end{array}$$

$$\begin{bmatrix} X_{ax} & Y_{ax} & X_{bx} & Y_{bx} \\ X_{ay} & Y_{ay} & X_{by} & Y_{by} \\ X_{az} & Y_{az} & X_{bz} & Y_{bz} \\ X_{aw} & Y_{aw} & X_{bw} & Y_{bw} \end{bmatrix} \times [M] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (2)$$

That is to say, whatever the magic matrix is, it must transform the exemplary points [seen in the leftmost matrix in (2)] into their correct positions in four dimensions [seen in the rightmost matrix in (2)]. The magic matrix M must also transform the exemplary origin point into the point $[0\ 0\ 0\ 0]$, a fact which is not shown in (2).

Now, if we express all tablet coordinates as relative to the origin location in their view, any matrix will transform the origin correctly, for $[0\ 0\ 0\ 0][M] = [0\ 0\ 0\ 0]$. Moreover, we can also notice that the right-hand side of (2) is the identity matrix, and it follows that M is the inverse of the matrix formed by the relative tablet coordinates of the exemplary points:

$$[M] = \begin{bmatrix} X_{ax} & Y_{ax} & X_{bx} & Y_{bx} \\ X_{ay} & Y_{ay} & X_{by} & Y_{by} \\ X_{az} & Y_{az} & X_{bz} & Y_{bz} \\ X_{aw} & Y_{aw} & X_{bw} & Y_{bw} \end{bmatrix}^{-1}. \qquad (3)$$

Having measured the relative tablet coordinates of the exemplary points, we form a matrix of them, invert it, and use the result to transform all other points digitized from those views. This simple process will account not only for the viewing direction of the view we are using but also for any tilt of the view on the tablet, and for any skew in the tablet axis.

Of course, whenever the user changes from one pair of views to another, a new M matrix must be computed. During the view definition process, the relative tablet coordinates of the exemplary points in each view for each pen are stored. Whenever a new view pair comes into use, these values are retrieved, used to form a matrix, inverted, and installed in place of the former value of M. This process takes only a few milliseconds and thus passes unnoticed to the user. The user blissfully points to locations in three dimensions in whatever views he chooses and the system seems to "understand" what he means.

```
COLOR:BROWN
AG1 AG4 AG3 AG2
AG1 AG2 AB3 AB4
AG2 AG3 AF1 AB3
AG3 AG4 AF2 AF1
AG4 AG1 AB4 AF2
/       TEETH
COLOR:WHITE
/RIGHT
AE1 AH1 AI1
AH1 AH2 AI1
AH2 AE1 AI1
/LEFT
AE2 AJ1 AK1
AJ1 AJ2 AK1
AJ2 AE2 AK1
/       EARS
COLOR:YELLOW
/LEFT
AL1 AL2 AL3
AM1 AM2 AB2
AM1 AB2 AL2 AL1
AB2 AM2 AL3 AL2
AM2 AM1 AL1 AL3
/RIGHT
AN1 AN2 AN3
AO1 AB1 AO2
AO1 AO2 AN2 AN1
AO2 AB1 AN3 AN2
AB1 AO1 AN1 AN3
/       EYES
COLOR:WHITE
AP1 AP2 AP3 AP4
COLOR:BROWN
AQ1 AP2 AP1
AQ1 AQ2 AP3 AP2
AQ2 AP4 AP3
COLOR:WHITE
AR1 AR2 AR3 AR4
COLOR:BROWN
AS1 AR2 AR1
AS1 AS2 AR3 AR2
AS2 AR4 AR3                         } POLYGONS
POINTS:
AS2    281    500    2540
AS1    143    506    2528
AR4    331    850    2275
AR3    290    843    2521
AR2    150    843    2525
AR1    93     853    2287
AQ2    -100   500    2525
AQ1    -243   512    2518
AP4    -56    856    2278
AP3    -103   846    2521
AP2    -243   843    2521
AP1    -287   856    2300
AO2    -396   15     2775
AO1    -509   281    3175          } POINTS
```

Fig. 3. Example of output from the digitizing system. The three-dimensional coordinates of points are indicated. Polygons are defined by mentioning the names of their vertices in sequence. Only a part of the object is shown in this example.

Different scale factors for different views or different axes of different views can easily be accommodated. If the exemplary points given for different views lie at different distances from the origin, then the matrix M obtained by inverting the exemplary points will contain automatic scale factor compensation. In fact, M transforms the position given in the example into a location on one of the four output axes at unit distance from the origin. Thus, if one is using a scale of 1 in/ft and wants output coordinates in feet, the example point in the 1-in-scale view will be 1 in from the origin, that being the place into which the four-dimensional example point located 1 ft from the origin is transformed by whatever the viewing transformation is.

Because the scale factors of views being digitized are often the same, one may want to indicate merely the direction of an axis and not a scale factor, maintaining instead a constant scale factor. This is easily accomplished by normalizing the corresponding row of the exemplary point matrix, i.e., making the length of that row vector unity. In fact, we have found that this is such a common requirement that introduction of a scale factor by *not* normalizing the rows of the measured example point matrix is considered the exception rather than the rule.

Defining Polygons

In building models for production of computer pictures, our objective is to define a collection of polygons which form the "skin" of the model. The vertices of these polygons are points whose three-dimensional coordinates have been defined by tablet input. The polygons themselves are recorded as the names of the sequence of vertices involved so that vertices may be shared between polygons. Thus, for example, $A1$, $B5$, $B6$, $A8$, $B9$ is a specification for a pentagon connecting the indicated vertices, The programs we have built for data input by tablet record not only the coordinates of the points digitized, but also the polygons defined by the user as the "skin" of his model. A typical output form is shown in Fig. 3.

Our first attempts at defining "skin" involved a simple two-step process. First, the user pointed accurately to all of the points he felt necessary for defining the object, and the system recorded the three-dimensional coordinates of these points. Having defined all of the points, the user then went back and "touched" points in sequence to define each polygon on the "skin." During this second process, the system would define no new points, but would rather use as the vertex of the polygon whatever already defined point was closest in three dimensions to the place indicated by the dual-pen specification. A user could define the locations of points using one view pair and point to them through another view pair.

The notion of defining the coordinates of all points first and then returning to define the polygons proved slow and cumbersome because each point was used several times. In defining a rectangular block, for example, the user had to touch each point four times: first to define its coordinates, and then three additional times—once during the definition of each of the three faces associated with it. With this system, it took a user 4 s to define a point or select a predefined point, and thus about 2 min to define both corner locations and polygon topology for a rectangular block.

Because 2 min seems a long time to define a block, we built an improved system which proceeds simultaneously with the definition of point coordinates and polygonal skin. The user thinks only in terms of defining the skin; any points that he uses in doing so are entered into a point table. If a point already exists in the point table at almost the same coordinates as the new point, the existing point will be used and the new point will not enter the table. The geometric tolerance required to define a new point is, of course, adjustable. The user gets feedback as to which choice was made because the system makes a slightly different noise when a new point is defined than when an existing point is referenced.

Defining Sets of Polygons

The ability to define and use points simultaneously reduces the work of defining a surface somewhat, but still fails to address the real problem of defining collections of polygons quickly and easily. We have observed that for most useful objects, the polygons often share common boundaries, and thus might possibly be defined in sets, using the boundary of one polygon as a part of the boundary of the next. Accordingly, we have included in our system the ability to define a "row" of polygons by defining the points that lie on each side of the row. As shown in Fig. 4, the four upper polygons might be defined by describing the A points in sequence and the B points in sequence. The program could then automatically generate polygons beginning $A1$, $A2$, $B2$, $B1$ and ending $A4$, $A5$, $B5$, $B4$. Subsequent definition of the points in the C row (in order) could permit the system to define the lower row of polygons.

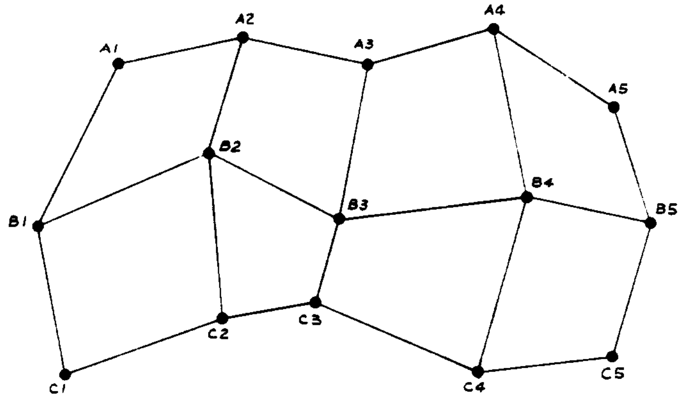

Fig. 4. A mesh of points may be used to define many polygons simultaneously. The points labeled $A1$ through $A5$ are input in sequence, followed by the points labeled $B1$ through $B5$. The system automatically defines and outputs the correct polygon definitions for the entire upper row of polygons.

In our improved system, the user is provided with two working sets of points, called V (visible) and I (invisible), into which he may place collections of points. Actually, the V and I sets contain the names of points previously defined, if such points exist, or if not, the names of points newly defined at the time of entry. Suitable editing controls are provided to enter new points into either set, delete single points from either set, clear either set, or treat the two sets as a push-down stack. Other controls are provided to use the V and I sets for defining a collection of polygons.

There are three polygon definition controls and a fourth control which is a combination of the three. Two of the basic controls each output a single polygon, namely, that corresponding to the V or the I set of vertices, respectively. The polygon that is output eliminates any duplication that may be in the V or I set, including duplication of the first point as the last in the set. Thus if the V set contains the points $A1$, $A2$, $A3$, $A3$, $A4$, $A5$, $A1$, the output polygon will be $A1$, $A2$, $A3$, $A4$, $A5$.

Output from the I set is in the inverse order from that of the V set. This inversion of order is intended to make it easy to define all polygons in a clockwise (or counterclockwise) direction as seen from *outside* the body being defined. The user defines all polygons in a clockwise direction as he sees them in his drawing. If the polygon is visible, he puts it into the V set and its clockwise order will be preserved. If the polygon is invisible, he puts it into the I set and its direction will automatically be reversed to correct for the fact that it was defined as seen through the object of which it is a face.

The third polygon output control generates a number of output polygons which form a skin connecting the V set of points with the I set of points. The polygons of this skin are usually quadrilaterals, as shown in Fig. 4, but may be triangles under special circumstances. The skin generated forms an open channel between the V set and the I set of points; a closed tube will be obtained if the first point in I is repeated as the last point in I and the first point in V is repeated as the last point in V. Notice that such a duplication would have no effect on the output polygons corresponding to I and V themselves, but will cause the skin between them to form a closed tube. If the number of points in I and V do not correspond, the skin between them is not defined and an error message is generated.

The "no duplication" function already mentioned provides for making skins between sets of points unequal in point count. Suppose that one wishes to form a skin between one section of a ship defined with eight points and another section defined with only seven points. One can do this by outputting six quadrilaterals and one triangle. What one does, in fact, is to enter the point that is to be the vertex of the triangle into either I or V twice, as a duplicate, thus in effect putting out a quadrilateral which has a repeated vertex. Because the system refuses to output such a repeated vertex, the output becomes the desired triangle. Thus whenever the number of points per section is changed along the hull of a ship, for example, one indicates which part of the skin is to be made of triangles by entering a duplicate point into V or I.

The V and I sets of points are organized as a push-down stack to facilitate defining long skins between consecutive sections as, for example, of a ship hull. One can prepare a new set of points corresponding to the next section and then "push" it into I or V, causing the old value of I to move into V or vice versa. One can thus progress from section to section down the length of a hull, generating the polygons that connect each section with the previous one by skinning in between V and I at each stage. One pushes into I or into V consistently during the definition of any one hull depending on whether one is working forward or aft, which in turn depends on the sense of the coordinate axes being used. The sense of the polygons output during the skinning between two sections corresponds to that of the V and I output polygons themselves.

Appendix

Defining a Three-Dimensional Point from Two Perspective Views

With the two-pen digitizing system, we can indicate a position in three dimensions by measuring its position in a pair of two-dimensional views. As we have already seen, it is easy to construct the three-dimensional coordinates of the point if the two views are the ordinary orthographic views common in mechanical drafting practice, and if the axes of the views are aligned with the axes of the tablet. It is more difficult, however, to see how to compute the three-dimensional coordinates of the point indicated if the two views are arbitrary perspective projections, e.g., photographs, casually fastened at an angle to the tablet axis. It is the purpose of this Appendix to describe the mathematical transformations required. We think in terms of the general perspective projection in order to be able to discover the three-dimensional positions of points captured in photographs.

Of course, by handling the general perspective case, we automatically take care of the simpler orthographic projections of mechanical drawing, and a host of other special projections such as single-point perspective, two-point perspective, cabinet projection, etc. Moreover, we also automatically take care of any misalignment between the view and the tablet axis, because such misalignment shows up as a simple rotation which can be included in the general perspective projection. Any nonperpendicularity of the tablet axes themselves is also automatically accounted for in the perspective projection.

It is less obvious, but nonetheless true, that the perspective projection will account properly for digitizing devices that detect angles from two fixed locations rather than detecting position in a linear fashion, provided the total angle over which measurements are made is small. Alternatively, if the angle of measurement is large, one must use its tangent as the measured coordinate to get proper operation of the perspective projection mathematics.

The ability to function with angular measurements, provided the angle is small, follows from the fact that perspective projections do not preserve parallelism of lines. A perspective projection which projects the centers of the angular measurement to infinity will convert such angular measures into the more familiar linear measures of a Cartesian coordinate system. Because a compounding of perspective projections results merely in another perspective projection, this final projection can easily be accommodated in the mathematics described below. The fact that angular measurements are acceptable leads to the conclusion that angular scanners such as the "twinkle box" [2] fit into this same mathematical framework.

There are two parts to converting from pairs of two-dimensional views to the three-dimensional coordinates of a point. First, one must discover the numerical values which characterize the particular perspective projection in use; second, one must use the pairs of two-dimensional data points to determine the three-dimensional coordinates. To discover the numerical values which characterize a particular projection we will measure the final projected positions of some example points whose three-dimensional coordinates are known. To compute the position in three dimensions of some new point whose position in the two views has been measured, we will use the projection parameters to establish the positions of planes in three dimensions, planes whose intersection must be the three-dimensional location of the point. Unfortunately, measuring both horizontal and vertical locations in each of two separate views gives us four planes which may not intersect in a single point and thus some kind of data fitting will be required.

For those not familiar with homogeneous coordinate representation, a word about our notation is probably in order. In the notation used here, we will represent a point in three dimensions whose real coordinates are X, Y, and Z, either as the four-component row vector $[X, Y, Z, 1]$ where 1 is the homogeneous term, or as any multiple of that vector, e.g. $[wX, wY, wZ, w]$, where w is arbitrarily chosen. When we are thinking of a point in homogeneous terms we may represent it with lower-case notation as $[x, y, z, w]$, meaning exactly the same point as above, because X, Y, and Z can always be found by dividing out w, the homogeneous term: $X=x/w$, $Y=y/w$, $Z=z/w$. If this form of notation is not familiar, read this paragraph again, for it contains nearly all you need to know about homogeneous coordinates.

The general perspective projection can be represented in homogeneous coordinates as a four-by-four transformation matrix. Thus if $[X, Y, Z, 1]$ is some point in a three-dimensional space, we are free to represent it homogeneously as $[wX, wY, wZ, w]=[x, y, z, w]$, and to transform it by a four-by-four matrix into some other space $[X', Y', Z', 1]$:

$$[x, y, z, w] \times \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{bmatrix} = [x', y', z', w'] \quad (4)$$

$$X' = x'/w' \qquad Y' = y'/w' \qquad Z' = z'/w'. \quad (5)$$

Because the division so essential to perspective is implicit in the homogeneous notation, we often omit its explicit representation, saying that the transformation matrix T represents the perspective projection. Indeed, the 16 numbers of the matrix do account for all of the possible variations of perspective that may occur.

There are several things to note about this perspective projection. First, the perspective projection is a projection from one three-dimensional space to another three-dimensional space! We can actually compute a perspective depth as well as the positions of objects on the screen! Surprisingly, cameras produce such three-dimensional images, as evidenced by the need to focus, i.e., the need to place the film plane in that part of the three-dimensional image in which one is most interested. Second, notice the delightful symmetry of the X', Y', and Z' computations. The perspective depth computation is no different from the lateral position computations! Third, notice that the matrix T itself may be arbitrarily scaled, because the division will remove the effects of any such scaling.

In a two-dimensional view we have dropped the depth coordinate Z' and no longer have it available to us. Therefore, the projection becomes:

$$[X, Y, Z, 1] \times \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \\ T_{41} & T_{42} & T_{43} \end{bmatrix} = [x', y', w'] = w'[U, V, 1]. \quad (6)$$

In this transformation we have assumed w to be unity, which we are free to do because the scale factor does not matter, and to distinguish the two-dimensional measurements clearly in what follows we have called them U and V, where $U=x'/w'$ and $V=y'/w'$. Notice that there are 12 elements in the 4 by 3 transformation matrix. Notice also that the tablet measurements of the photographs or other view are not the values x', y', and w' shown in (6), but the ratios $U=x'/w'$ and $V=y'/w'$. In fact, we have no way of knowing the value of w', and different input points $X\ Y\ Z$ may give different values of w'.

It is worth writing out the separate equations represented in the matrix expression in (6). There are three:

$$T_{11}X + T_{21}Y + T_{31}Z + T_{41} = w'U$$
$$T_{12}X + T_{22}Y + T_{32}Z + T_{42} = w'V$$

and

$$T_{13}X + T_{23}Y + T_{33}Z + T_{43} = w'. \quad (7)$$

By substituting the value of w' found in the third equation into the first two equations and grouping terms, we can get to the heart of the transformation:

$$(T_{11} - T_{13}U)\,X + (T_{21} - T_{23}U)\,Y + (T_{31} - T_{33}U)\,Z + (T_{41} - T_{43}U) = 0$$
$$(T_{12} - T_{13}V)\,X + (T_{22} - T_{23}V)\,Y + (T_{32} - T_{33}V)\,Z + (T_{42} - T_{43}V) = 0. \quad (8)$$

Equations (8) can be considered in three ways, depending on what is known and what is not. First, if we know the transformation T_{ij} and the position of the point in three space, $X\ Y\ Z$, then each equation is an equation in one unknown, namely, U or V, and we can compute the position of the point in the perspective picture.

$$U = \frac{T_{11}X + T_{21}Y + T_{31}Z + T_{41}}{T_{13}X + T_{23}Y + T_{33}Z + T_{43}}$$
$$V = \frac{T_{12}X + T_{22}Y + T_{32}Z + T_{42}}{T_{13}X + T_{23}Y + T_{33}Z + T_{43}}. \quad (9)$$

Second, if we know the transformation T_{ij} and the position of the point in the picture $U\ V$, then the terms of (8) in parentheses become computable constants, and we can think of the equations as being of the form:

$$a_1X + b_1Y + c_1Z + d_1 = 0$$
$$a_2X + b_2Y + c_2Z + d_2 = 0. \quad (10)$$

When seen in this form, it is obvious that each equation represents a plane in three space on which the point must lie, and each pair defines a "ray" extending out from the camera through the point. With known (different) transformations T_{ij} for two views and measured two-dimensional coordinates $U\ V$ in each view, we have four equations in the three unknowns $X\ Y\ Z$, and can solve for the position of the point in three dimensions by some sort of best fit method.

Third, if we know the location of several points in three dimensions $X\ Y\ Z$, and know their measured positions in the view $U\ V$, then (8) are two equations per point in the twelve unknowns T_{ij}. In the case of eight points, subscripts a through h, for example:

$$\begin{bmatrix}
X_a & Y_a & Z_a & 1 & 0 & 0 & 0 & 0 & -U_aX_a & -U_aY_a & -U_aZ_a & -U_a \\
X_b & Y_b & Z_b & 1 & 0 & 0 & 0 & 0 & -U_bX_b & -U_bY_b & -U_bZ_b & -U_b \\
X_c & Y_c & Z_c & 1 & 0 & 0 & 0 & 0 & -U_cX_c & -U_cY_c & -U_cZ_c & -U_c \\
X_d & Y_d & Z_d & 1 & 0 & 0 & 0 & 0 & -U_dX_d & -U_dY_d & -U_dZ_d & -U_d \\
X_e & Y_e & Z_e & 1 & 0 & 0 & 0 & 0 & -U_eX_e & -U_eY_e & -U_eZ_e & -U_e \\
X_f & Y_f & Z_f & 1 & 0 & 0 & 0 & 0 & -U_fX_f & -U_fY_f & -U_fZ_f & -U_f \\
X_g & Y_g & Z_g & 1 & 0 & 0 & 0 & 0 & -U_gX_g & -U_gY_g & -U_gZ_g & -U_g \\
X_h & Y_h & Z_h & 1 & 0 & 0 & 0 & 0 & -U_hX_h & -U_hY_h & -U_hZ_h & -U_h \\
0 & 0 & 0 & 0 & X_a & Y_a & Z_a & 1 & -V_aX_a & -V_aY_a & -V_aZ_a & -V_a \\
0 & 0 & 0 & 0 & X_b & Y_b & Z_b & 1 & -V_bX_b & -V_bY_b & -V_bZ_b & -V_b \\
0 & 0 & 0 & 0 & X_c & Y_c & Z_c & 1 & -V_cX_c & -V_cY_c & -V_cZ_c & -V_c \\
0 & 0 & 0 & 0 & X_d & Y_d & Z_d & 1 & -V_dX_d & -V_dY_d & -V_dZ_d & -V_d \\
0 & 0 & 0 & 0 & X_e & Y_e & Z_e & 1 & -V_eX_e & -V_eY_e & -V_eZ_e & -V_e \\
0 & 0 & 0 & 0 & X_f & Y_f & Z_f & 1 & -V_fX_f & -V_fY_f & -V_fZ_f & -V_f \\
0 & 0 & 0 & 0 & X_g & Y_g & Z_g & 1 & -V_gX_g & -V_gY_g & -V_gZ_g & -V_g \\
0 & 0 & 0 & 0 & X_h & Y_h & Z_h & 1 & -V_hX_h & -V_hY_h & -V_hZ_h & -V_h
\end{bmatrix} \times \begin{bmatrix} T_{11} \\ T_{21} \\ T_{31} \\ T_{41} \\ T_{12} \\ T_{22} \\ T_{32} \\ T_{42} \\ T_{13} \\ T_{23} \\ T_{33} \\ T_{43} \end{bmatrix} = 0. \quad (11)$$

Because the equations of (11) are homogeneous, of course, the solutions T_{ij} will contain an arbitrary scale factor if any such solutions exist. We may prefer to choose T_{43} to be unity, move the last column of the matrix to the other side of the equation, and solve for the remaining 11 T's by a mean-square fit method. To do so, of course, requires at least 11 equations or $5\frac{1}{2}$ measured data points.

Now the equations of (11) get at the heart of finding the transformation, given example points whose coordinates in three dimensions $X\ Y\ Z$ are known, and whose locations in two dimensions $U\ V$ are also known. In addition to knowing the example points, if we also know that the view is orthographic (i.e., a view from infinity but not necessarily a view from along an axis), then we know in advance that T_{13}, T_{23}, and T_{33} are zero, and only eight unknowns remain, namely, the first two columns of T's, assuming T_{43} to be unity. Thus to specify an orthographic projection, one need define only the locations of four points, e.g., the origin and a point on each of the axes.

In an earlier section of the paper we chose to transform from a pair of two-dimensional views into a four-dimensional space rather than directly into three dimensions. To do this we must expand (6) to include another row:

$$[X, Y, Z, W, 1] \times \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \\ T_{41} & T_{42} & T_{43} \\ T_{51} & T_{52} & T_{53} \end{bmatrix} = w'[U, V, 1]. \tag{12}$$

In the orthographic projection we were using, the final column of (12) is zero, except for T_{53} which may be assumed to be unity, and thus ten T_{ij} remain to be found and five points will be required to define the transformation. We used the origin and a point on each of the four axes.

Burton's "twinkle box" [2] is a system which measures only a single coordinate from each view, requiring at least three views of a point before its three-dimensional coordinate can be computed. For this case we rewrite (6) and omit one of the measurements:

$$[X, Y, Z, 1] \times \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ T_{31} & T_{32} \\ T_{41} & T_{42} \end{bmatrix} = w'[U, 1]. \tag{13}$$

In this case the measured coordinate U of a three-dimensional point $X\ Y\ Z$ provides only a single equation defining the T_{ij}, and seven such points must be measured to discover the transformation involved.

For those who may want to program some of the ideas given here, the following technique computes a mean-squares fit. Given a set of equations in matrix form:

$$AX = B \tag{14}$$

where the A matrix has too many rows for the number of degrees of freedom in X, one can get the least mean-squared error fit by solving

$$A^TAX = A^TB \tag{15}$$

where A^T is the transpose of A, and of course A^TA is square and of the right dimension to suit X. Should no solution to (15) be computable, it means that the conditions imposed in (14) are redundant and no one solution represents the least error condition. For your reference we have computed A^T and A^TA corresponding to (11) to be:

$$\begin{bmatrix}
X_a & X_b & X_c & X_d & X_e & X_f & X_g & X_h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
Y_a & Y_b & Y_c & Y_d & Y_e & Y_f & Y_g & Y_h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
Z_a & Z_b & Z_c & Z_d & Z_e & Z_f & Z_g & Z_h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X_a & X_b & X_c & X_d & X_e & X_f & X_g & X_h \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Y_a & Y_b & Y_c & Y_d & Y_e & Y_f & Y_g & Y_h \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Z_a & Z_b & Z_c & Z_d & Z_e & Z_f & Z_g & Z_h \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-U_aX_a & -U_bX_b & -U_cX_c & -U_dX_d & -U_eX_e & -U_fX_f & -U_gX_g & -U_hX_h & -V_aX_a & -V_bX_b & -V_cX_c & -V_dX_d & -V_eX_e & -V_fX_f & -V_gX_g & -V_hX_h \\
-U_aY_a & -U_bY_b & -U_cY_c & -U_dY_d & -U_eY_e & -U_fY_f & -U_gY_g & -U_hY_h & -w_aY_a & -V_bY_b & -V_cY_c & -V_dY_d & -V_eY_e & -V_fY_f & -V_gY_g & -V_hY_h \\
-U_aZ_a & -U_bZ_b & -U_cZ_c & -U_dZ_d & -U_eZ_e & -U_fZ_f & -U_gZ_g & -U_hZ_h & -V_aZ_a & -V_bZ_b & -V_cZ_c & -V_dZ_d & -V_eZ_e & -V_fZ_f & -V_gZ_g & -V_hZ_h
\end{bmatrix}$$

$$\begin{bmatrix}
\sum X_i^2 & \sum X_iY_i & \sum X_iZ_i & \sum X_i & 0 & 0 & 0 & 0 & -\sum U_iX_i^2 & -\sum U_iX_iY_i & -\sum U_iX_iZ_i \\
\sum Y_iX_i & \sum Y_i^2 & \sum Y_iZ_i & \sum Y_i & 0 & 0 & 0 & 0 & -\sum U_iY_iX_i & -\sum U_iY_i^2 & -\sum U_iY_iZ_i \\
\sum Z_iX_i & \sum Z_iY_i & \sum Z_i^2 & \sum Z_i & 0 & 0 & 0 & 0 & -\sum U_iZ_iX_i & -\sum U_iZ_iY_i & -\sum U_iZ_i^2 \\
\sum X_i & \sum Y_i & \sum Z_i & 8 & 0 & 0 & 0 & 0 & -\sum U_iX_i & -\sum U_iY_i & -\sum U_iZ_i \\
0 & 0 & 0 & 0 & \sum X_i^2 & \sum X_iY_i & \sum X_iZ_i & \sum X_i & -\sum V_iX_i^2 & -\sum V_iX_iY_i & -\sum V_iX_iZ_i \\
0 & 0 & 0 & 0 & \sum Y_iX_i & \sum Y_i^2 & \sum Y_iZ_i & \sum Y_i & -\sum V_iY_iX_i & -\sum V_iY_i^2 & -\sum V_iY_iZ_i \\
0 & 0 & 0 & 0 & \sum Z_iX_i & \sum Z_iY_i & \sum Z_i^2 & \sum Z_i & -\sum V_iZ_iX_i & -\sum V_iZ_iY_i & -\sum V_iZ_i^2 \\
0 & 0 & 0 & 0 & \sum X_i & \sum Y_i & \sum Z_i & 8 & -\sum V_iX_i & -\sum V_iY_i & -\sum V_iZ_i \\
-\sum U_iX_i^2 & -\sum U_iX_iY_i & -\sum U_iX_iZ_i & -\sum U_iX_i & -\sum V_iX_i^2 & -\sum V_iX_iY_i & -\sum V_iX_iZ_i & -\sum V_iX_i & \sum L_i^2X_i^2 & \sum L_i^2X_iY_i & \sum L_i^2X_iZ_i \\
-\sum U_iY_iX_i & -\sum U_iY_i^2 & -\sum U_iY_iZ_i & -\sum U_iY_i & -\sum V_iY_iX_i & -\sum V_iY_i^2 & -\sum V_iY_iZ_i & -\sum V_iY_i & \sum L_i^2Y_iX_i & \sum L_i^2Y_i^2 & \sum L_i^2Y_iZ_i \\
-\sum U_iZ_iX_i & -\sum U_iZ_iY_i & -\sum U_iZ_i^2 & -\sum U_iZ_i & -\sum V_iZ_iX_i & -\sum V_iZ_iY_i & -\sum V_iZ_i^2 & -\sum V_iZ_i & \sum L_i^2Z_iX_i & \sum L_i^2Z_iY_i & \sum L_i^2Z_i^2
\end{bmatrix}$$

$L_i^2 = U_i^2 + V_i^2$.

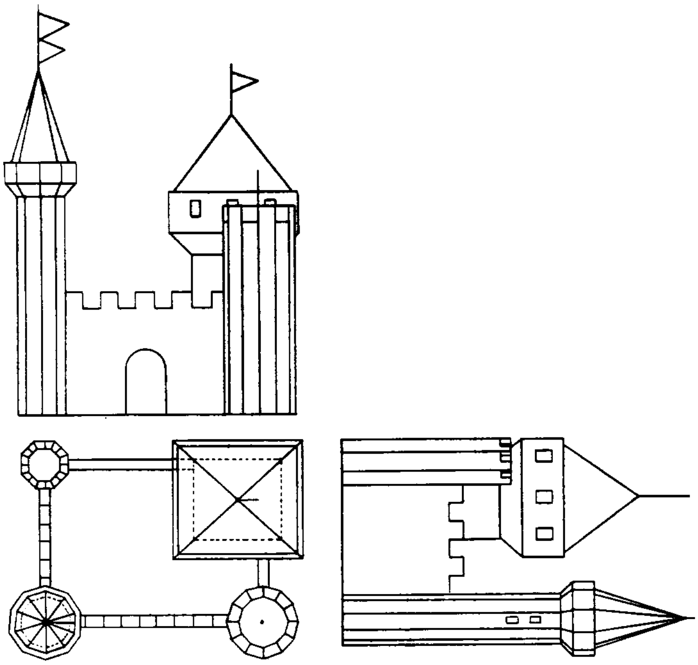

Fig. 5. An "engineering" drawing of a castle ready for digitizing. The completed castle may be seen in Fig. 6.

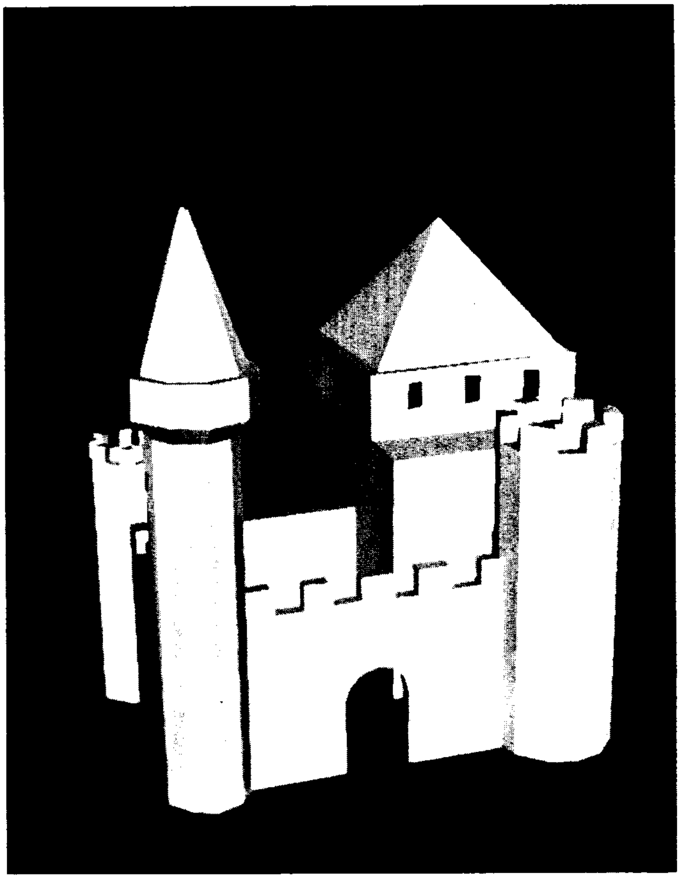

Fig. 6. A castle digitized using the tablet system. Artist's drawing of the castle is shown in Fig. 5.

Figs. 5 and 6 are a direct illustration of the mathematical principles described in the Appendix.

References

[1] R. P. Burton, "Real-time determination of multiple three-dimensional positions," Ph.D. dissertation, Dep. of Elec. Eng., Univ. of Utah, Salt Lake City, June 1973. Filed as Tech. Sci. Rep. CSC-72-122.

[2] R. P. Burton and I. E. Sutherland, "Twinkle box: A three-dimensional computer input device," to be published.

[3] E. E. Catmull and F. I. Parke, "A system for computer-generated movies," in *Proc. Ass. Comput. Mach., Annu. Conf.* (Boston, Mass., Aug. 1972), vol. I, pp. 451–457.

[4] A. M. Noll, "Man-machine tactile communication," *SID J.*, p. 6 ff., July/Aug. 1972.

[5] ——, "Real-time interactive stereoscopy," *SID J.*, vol. I, pp. 14–22, Sept./Oct. 1972.

[6] F. I. Parke, "Computer-generated animation of faces," Master's thesis, Univ. of Utah, Salt Lake City, UTEC-CSC-72-120, June 1972.

[7] R. D. Resch, "Portfolio of computer art," this issue, pp. 496–502.

[8] R. F. Sproull and I. E. Sutherland, "A clipping divider," in *Spring Joint Computer Conf., AFIPS Conf. Proc.*, vol. 33, pt. I, 1968, pp. 765–776.

[9] I. E. Sutherland, *Operation and Maintenance Manual for the E&S Tablet System*, Evans and Sutherland Computer Corporation, Salt Lake City, Utah, Nov. 1, 1972.

[10] ——, "The E&S tablet system: A large-area, multiple-pen, computer input device," to be published.

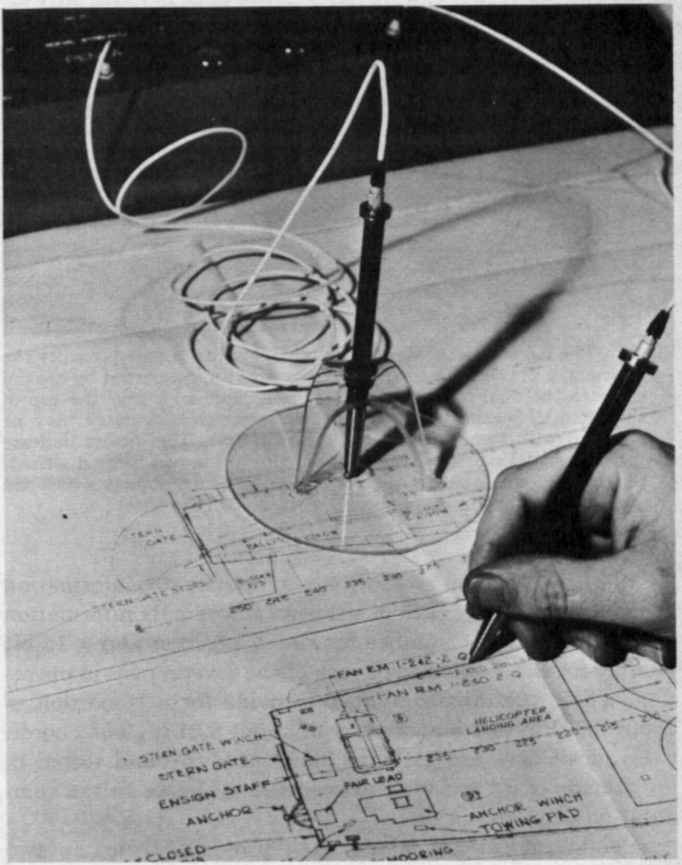

Fig. 1. Closeup of two digitizing pens. The small stand is often used to hold one pen if digitizing in a plane surface. Two views of a Navy ship are being digitized.

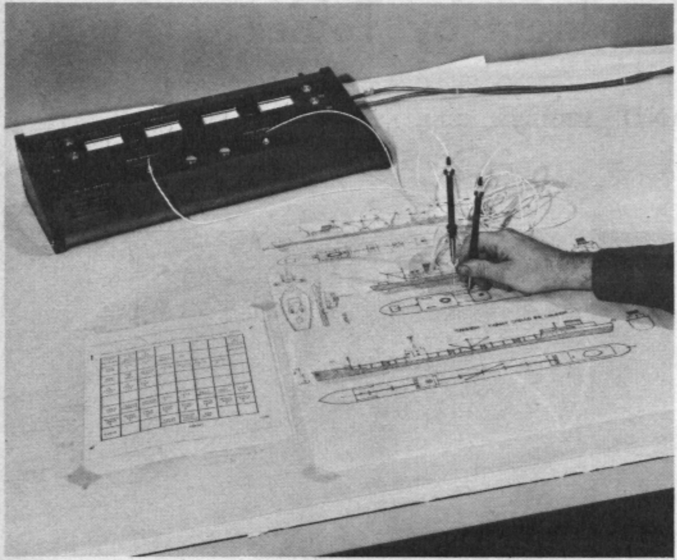

Fig. 2. The E&S tablet system. The tablet surface covers virtually the entire area of the table itself. The two pens are connected by coaxial cable to the dual-pen amplifier and filter unit seen near the top of the picture. Sensitivity of the "down" detection circuitry may be adjusted with the knobs on the tabletop unit. The meters indicate received signal strength. Sketches of ships are shown being digitized. Sketches of this quality can be used to produce objects which are identifiable though not, of course, dimensionally precise.

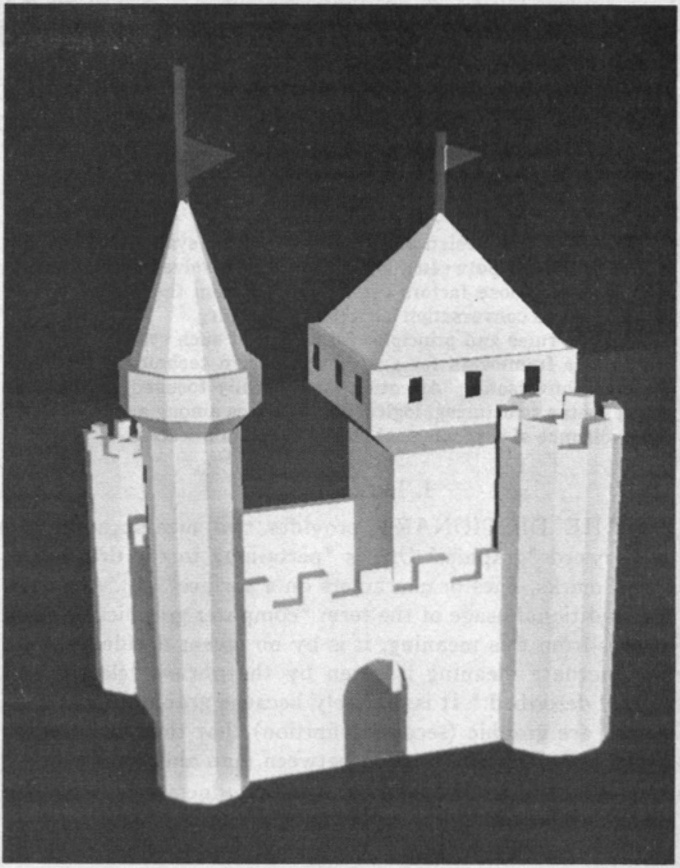

Fig. 6. A castle digitized using the tablet system. Artist's drawing of the castle is shown in Fig. 5.