Computer Methods in Image Analysis

Edited by
J.K. Aggarwal
Professor of Electrical Engineering and Computer Science
The University of Texas at Austin
Richard O. Duda
Staff Scientist
Stanford Research Institute
Azriel Rosenfeld
Research Professor, Computer Science Center
University of Maryland

A volume in the IEEE PRESS Selected Reprint Series,
prepared under the sponsorship of the
IEEE Computer Society.

IEEE PRESS

The Institute of Electrical and Electronics Engineers, Inc. New York

MACHINE PERCEPTION OF THREE-DIMENSIONAL SOLIDS *

L. G. Roberts

Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Massachusetts

I. Introduction

The problem of machine recognition of pictorial data has long been a challenging goal, but has seldom been attempted with anything more complex than alphabetic characters. Many people have felt that research on character recognition would be a first step, leading the way to a more general pattern recognition system. However, the multitudinous attempts at character recognition, including my own, have not led very far. The reason, I feel, is that the study of abstract, two-dimensional forms leads us away from, not toward, the techniques necessary for the recognition of three-dimensional objects. The perception of solid objects is a process which can be based on the properties of three-dimensional transformations and the laws of nature. By carefully utilizing these properties, a procedure has been developed which not only identifies objects, but also determines their orientation and position in space.

Three main processes have been developed and programed in this report. The input process produces a line drawing from a photograph. Then the three-dimensional construction program produces a three-dimensional object list from the line drawing. When this is completed, the three-dimensional display program can produce a two-dimensional projection of the objects from any point of view. Of these processes, the input program is the most restrictive, whereas the two-dimensional to three-dimensional and three-dimensional to two-dimensional programs are capable of handling almost any array of planar-surfaced objects.

* This report is based on a thesis of the same title submitted to the Department of Electrical Engineering at the Massachusetts Institute of Technology on 10 May 1963, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

In order to implement the three-dimensional processing of pictures, perspective effects must be considered. For this reason, a four-dimensional, homogeneous system of coordinates will be used. In this system a single 4×4 matrix can modify a position vector by a linear transform, a translation, and a perspective transformation. Although many books discuss this homogeneous system of coordinates, their presentations are either incomplete or too involved for our purposes.[1] Therefore, the system is explained in Appendix A. Without the notational simplicity provided by using homogeneous transformations, most of the following analysis would not have been accomplished.

II. Background

A. Pattern Recognition by Computer

There have been numerous attempts to recognize simple patterns by machine. There is the work with neuronlike nets of threshold elements which divide the set of all input patterns into a number of classes by correlating a set of adaptive weights with some functions of multiple input cells. For this type of system there may not be many output classes and the transformations of the patterns must be minimal or nonexistent. Because of these restrictions, the patterns worked on so far have been those which, although complex, are not subject to much transformation such as characters and spoken digits. My paper on character recognition is typical and gives the other references.[2] This type of system would be of no value for multiple object recognition, except perhaps for finding the lines originally.

In the work by Selfridge and Neisser *et al.*,[3] a more useful set of tests is made on the input, whereas the output processing is similar to that of the system described earlier. That is, computation routines are developed to extract the useful information from the input, and their outputs are weighted to determine the most likely output class. Here again a small set of outputs is expected and characters were the patterns tested. One problem with both these methods is that they were intended for specific groups of abstract patterns, such as characters, and not for the well-defined geometry of photographs. They are better suited for looking at my resultant list structure of objects and deciding whether a group of objects is a chair or a table.

The closest that any researcher has come to the problem I propose is Leo Hodes in his work on processing line drawings.[4] His main result was to produce a list of lines and vertices by following out lines. Then he suggested a few simple tests which might be made on this list to find triangles, etc. Although his main purpose was to study abstract line patterns, he did describe a working line follower which was of value to me in this part of my effort.

The only work I know of on machine depth perception is that on binocular images. Julesz has reported a procedure which shifts the binocular pictures to find the areas at different depths.[5] This procedure uses only texture, not edges, to develop the depth information and shows that the binocular information alone is sufficient for depth perception. This work is similar in goal but completely different in procedure from mine. Other work in machine photograph processing has mainly been in the field of information reduction for bandwidth compression and my paper in this area summarizes this work.[6]

B. Psychophysical Theory

There has been a large volume of psychophysical research on human depth perception and shape recognition. From all this I have tried to isolate the ideas and theories which are used to explain our monocular perception of a three-dimensional world. It will be apparent, however, that the work of Gibson is dominant in my mind, since his book is both clear and complete.[7]

Of all the monocular depth cues perhaps the most written about is that of known object size. Ittelson reports experiments in which only one object could be seen, thus eliminating other depth cues.[8] Although the assumed size of objects such as playing cards tended to vary, the subjects would judge the depth reasonably well for normal-sized cards and proportionately shorter for jumbo cards. Thus he, for one, showed that the size of familiar objects is a good relative depth cue and fair for absolute depth. Gibson points out, however, that this type of distance perception is rarely used in the everyday world, since we look at arrays of objects rather than at single objects and can use more general depth cues.

Gibson's favorite cue is that of texture gradient. This is the effect of perspective on the grain or fine structure of large surfaces. As these surfaces recede, the apparent grain becomes finer. Another gradient cue is the illumination variation which puts curved surfaces in relief. This shows us the surface depth variations. The final depth cues are those of aerial perspective or blur with depth, and the angular upward position of objects toward the horizon which is a depth measure in most outdoor scenes.

Recognition of forms, shapes, and objects is often discussed from the Gestalt point of view, where shadowy forms and plane geometry figures are the forms to be recognized. Attneave and Arnoult spend many pages explaining random shape generation and the useful procedures for analyzing them.[9] They discuss contour following, differentiating pictures, and some of the simple measures of shape complexity. If they were discussing character recognition, it might be reasonable to use these tools; however, they say they are investigating "natural forms." This preoccupation with the abstract projected form is strongly attacked by Gibson. He feels that the visual world of objects and surfaces should be studied rather than the visual field on the retina. A perspective transformation does not reduce a solid object to a shadowy form. Rather, it defines the set of shapes which go with a single perception.

Perspective variations in a cube were tested by Langdon in an experiment on three-dimensional solids.[10] He found that perspective plays only a minor part in the perception of the size and depth and that the subjects always saw a cube, even when it was badly distorted by the perspective transform. The continual perception of a cube, even when transformed, is consistent with Gibson's idea that shape perception is and must be invariant under perspective transformation. My idea of models also follows from this, since each model represents an invariant percept, and can be identified with any projection of itself.

III. Depth Perception

The perception of depth in a monocular picture is based completely upon the assumptions of the observer. Some of the assumptions are about the nature of the real world and some are based on the observer's familiarity with the objects. Without these assumptions the picture is just another two-dimensional image, whereas with them the human is rarely confused about the depth relationships represented in the picture. Since humans agree so closely on their depth impressions, it is fair to assume that their major assumptions are the same, and are therefore subject to identification and analysis. The following is an attempt to set down some of the likely assumptions and derive what depth information can be obtained if they were used.

A. Transformation of Real World

The first assumption is that the picture is a view of the real world recorded by a camera or comparable device and therefore that the image is a perspective transformation of a three-dimensional field. This transformation is a projection of each point in the viewing space, toward a focal point, onto a plane. The transformation will be represented with a homogeneous, 4×4 transformation matrix P such that the points in the real world are transformed into points on the photograph (see Appendix A for an explanation of homogeneous coordinates). The transformation depends on the camera used, the enlargement printing process, and, of course, the coordinate system the real world is referred to. Let us fix the real world coordinates by assuming that the focal plane is the $x = 0$ plane and that the focal point is at $x = f$, $y = 0$, $z = 0$. In order that the picture not be a reflection, we choose the focal plane in front of the camera. Then the objects seen will be in the x half-space. Thus the focal plane is really the plane of the print, not of the negative. Figure 1 shows this arrangement.

A particular camera will have some focal distance f. We shall consider the square on the focal plane which was enlarged to create the print. The center of this square will be at some coordinates y_0, z_0, and the size of the square from the center to an edge will be some distance S. The actual size to which the square is enlarged is unimportant, since we shall measure the print with a normalized system which has the origin at the center and the

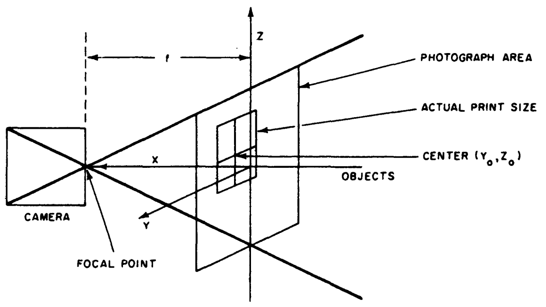

FIG. 1. Camera transformation.

edges at $y, z = \pm 1$. Normally, the whole film area is printed, and in this case $y_0 = z_0 = 0$ and S is equal to half the film size. For standard cameras without special lenses, the ratio of focal distance to film size is usually the same since this corresponds to a fixed viewing angle. Thus the ratio S/f is fairly constant at about $\frac{1}{4}$. In my case, the ratio is known if the camera is known and could be supplied with the photo.

It is not necessary to know the variables y_0, z_0, f, and S since they can be computed from the picture, given other assumptions later on. However, for the sake of simplicity we shall assume that S/f is known and that $y_0 = z_0 = 0$. The numerical values of S and f alone are not necessary, since this just affects the scale of the real world. Thus we can assume that $S = 1$ and with $r = S/f$ obtain a simple transformation P,

$$P = \begin{bmatrix} 1 & & & -r \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

If $\bar{v}$ is a point in real space, then $\bar{v}P$ is a point in a perspective space such that its Y and Z coordinates are the original point's projection in the picture plane. The X coordinate of $\bar{v}P$ is also obtained and will be useful for hidden line computation during display of three-dimensional objects.

Thus a transformation from the real world to a picture has been described, and to go the other way simply requires the inverse transformation P^{-1}. Of course, the x component of the real-world points will not be known in this case.

B. Objects Observed

We shall further assume that the three-dimensional field observed consists of a set of solid objects which occupy a definite region of space. Since we realize that it is usually possible to pick out the lines which define the boundaries of the objects and their surfaces, we shall assume that this has been accomplished and that the picture has been reduced to a line drawing. Because the objects are solid, we do not expect to see the boundaries which are hidden from the focal point by another solid.

Second, we shall assume that the objects seen could be constructed out of parts with which we are familiar. That is, either the whole object is a transformation of a preconceived model, or else it can be broken into parts that are. The models could be anything from a cube to a human body; the only requirement is that we have a complete description of the three-dimensional structure of each model.

The transformation from the model to the real world object will be a suitably restricted homogeneous transformation matrix R. We must allow an arbitrary rotation and translation of the model in order to position it properly in space. We should also like to allow three degrees of freedom for size change of the model so that a cube model can represent any parallelepiped. So far we have allowed nine degrees of freedom. The 4×4 matrix R can allow fifteen degrees of freedom since it has 16 elements and the total scale of the matrix is arbitrary in the homogeneous coordinate system. The last six degrees of freedom represent skew and perspective deformations. Skew deformations are size changes in the x, y, and z directions after the model has been rotated and will change the sides of a cube to parallelograms. A perspective deformation is most easily visualized as a compression of one end of the model. Objects that have been deformed in either way are not usually considered to be simple instances of the model. Furthermore, objects deformed in these ways could be constructed from smaller parts, so it is not necessary to allow skew and perspective deformations.

We cannot allow perspective deformation and still obtain a unique transform R from the picture; therefore, we require the top three elements in the last column of R to be zero. Skew variations can be allowed if we maintain very high accuracy in our computations, so our derivation will allow them, but later on they will be eliminated.

Now R transforms a model into an object and P transforms the object onto the picture so that if

$$H = RP$$

H transforms the model points into picture points. Therefore, in order to identify a group of points and lines in the picture with a particular model, we must find out if there is any transformation H which will take the model's points and lines into those of the picture. If such a model and transform are found, it can be said that the object represented in the picture could be that model under the transformation $R = HP^{-1}$.

C. Model Identification

Let us say that we are given a picture of a parallelepiped, and it has been reduced to a line drawing. We can then find the interior polygons which correspond to the surfaces of the object. There will normally be three quadrilaterals visible. These polygons all come together at one point which can be used for a reference point. If we look through our list of models, we find that a cube and perhaps other models have three quadrilaterals about one point. Therefore, we can pick a point in the cube model which has the proper polygons around it, pick a polygon from both the cube and the picture as starting points, and proceed to list topologically equivalent point pairs. When we have finished, we have a list of seven three-dimensional points from the model and a corresponding list of seven two-dimensional points from the picture. By adding a homogeneous coordinate $w = 1$ to each point vector, we obtain a 4×7 matrix of model points A and a 3×7 matrix of picture points B.

Now by means of the similarity test derived in Appendix B, we obtain the best transform H which will take A into B. We also obtain a mean-square error which indicates whether or not the model chosen really can fit the picture. We can then choose the model that causes the least error. For the parallelepiped, the cube model should fit with very little error. The transform obtained is a 3×4, since no depth data accompanied the picture points. Since we know P and thus P^{-1}, we can start to obtain R, the real space transform of the model. Since we have required that the three perspective components of R should be zero, we can specify the top three components of R in the first column as $(-1/r)$ times the corresponding elements of the last column of H.

$$R = HP^{-1}$$

$$H = \begin{bmatrix} y_1 & z_1 & w_1 \\ y_2 & z_2 & w_2 \\ y_3 & z_3 & w_3 \\ y_4 & z_4 & w_4 \end{bmatrix} \qquad P^{-1} = \begin{bmatrix} 1 & & & r \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$R_0 = \begin{bmatrix} -w_1/r & y_1 & z_1 & 0 \\ -w_2/r & y_2 & z_2 & 0 \\ -w_3/r & y_3 & z_3 & 0 \\ 0 & y_4 & z_4 & w_4 \end{bmatrix}$$

The lower left element of R still is not known since it is the x position of the whole object, and if an object grows in size as it moves away from us, it can maintain the same projection on the focal plane. Thus this depth value must be found some other way. For the present, we can call it zero and label the matrix R_0.

At this point, by assuming that an object in a picture is a transformation of a known model and by utilizing our knowledge of perspective geometry, we have been able to find the model and transformation which best represent the object. We know the precise orientation and position of the object except for one depth variable. We also know all the dimensions of the object relative to its total size. We also should know the skew deformation of the object, since we have obtained eleven variables. However, compression or expansion in the x direction produces only a slight change in the picture due to the perspective, since we are looking along the x axis. Thus the determination of the x skew can easily be in error. This problem of skew error in derived transformations was not realized until the computer program implementing these ideas began to produce distorted transformations of models. Even though the program could match every point in the picture with model points within one part in 4000, the x skew of the transformation might be off by a factor of 2. Thus it is clear that the x skew is not really obtainable from a picture, even though it can be derived mathematically.

If x skew deformations of the model must be restricted, it is logical to eliminate all skew variations for consistency. To restrict skew it is necessary to ask that the top three rows of R be orthogonal. Since the introduction of this requirement into the similarity test requires solving second-order equations, it is simplest to modify R after it has been computed. Thus the top three rows of R are forced to be orthogonal by modifying the first column, which is the one with the worst error. This means that the focal ratio r and w_1, w_2, and w_3 are not needed for the computation of the model transformation. Now in fact, r may be computed from the ratio of the new first column to the w_j.

Thus, by assuming that the objects seen in a picture are nondeformed transformations of known models, we can find the model and transformation without knowing the camera characteristics. Even if the picture is an orthogonal projection, as is almost the case with long telephoto lenses, we can compute the proper transformation. This would be impossible if we did not eliminate skew since r, w_1, w_2, and w_3 would all be zero. Thus the process accounts for, but does not depend on, perspective information.

The information required to obtain a transformation is obtained from the points in the line drawing. These points have two dimensions each and we need to determine eight degrees of freedom in the transformation. Therefore, at least four points from the picture must be used. These points cannot all lie in the same plane of the object or the equations will be degenerate. If more than four points are available, the mean-square error will indicate whether they are consistent with the model, and therefore help in the selection of the proper model.

D. Depth Information

After the matrix R_0 is obtained for an object or object part, it is still necessary to obtain the x translation or depth. Here we must resort to another assumption. There are several possibilities which I have chosen to ignore. If we were to assume that we know the models so well that we knew their size, this would fix their depth also; or, if we wished to interpret shadows, this might determine the depth if shadows existed. The fact that one object is partly in front of another supplies depth information, but only in the form of inequalities. Lastly, the various gradients—intensity, blur, and texture—might be useful for determining the depth gradients of each surface, but this information has already been found through the use of models. All these cues may be useful to humans, but each one is restricted in its generality and only useful in special cases.

The one depth perception concept which is suitably general, and sufficiently accurate to position all objects properly, is the use of a support theorem. We assume that each object must be supported somehow, either by another object or by a ground plane. This assumption allows us to project each object back in the x direction until it hits the ground plane or another object. While it is being projected back, it must be expanded so as to maintain the same image on the focal plane. The slope of the ground plane can be determined by examining each object for parallel planes, choosing a plane which goes under the focal point and is as parallel to the $z = 0$ plane as possible. When such a plane is found, we know only its slope, not its distance, from the origin. However, this single variable can be set arbitrarily, since it affects only the total scale of the picture. Actually it can be guessed rather accurately for the majority of pictures just by assuming or knowing the distance the camera was held from the ground and the focal ratio r. Since r may be computed, we could assume that the camera was held five feet above the floor, and now we can state the dimensions of each object in feet.

For compound objects, we know the pieces should fit together, so their relative depths are determined and the compound object can then be treated as one object and projected onto the ground plane. The whole procedure is relatively simple so long as the ground plane is really planar. If the ground curves, this could be taken care of by computing the curvature from the slopes of several objects. If there are breaks in the floor's slope such as walls, the breaks will be seen as lines and walls treated as objects. Thus the support assumption enables us to properly place all the objects in space.

To review the depth perception assumptions and results, we assumed that the picture was, in fact, a perspective view of the real world, that the objects shown in the picture could be described by means of one or more transformations of known models, and that all objects were supported by others or by a ground plane. The transformations allowed were restricted to rotation, translation, and size changes. Then, from a single picture, each object which has four or more points showing can be described in terms of the models and positioned in a three-dimensional space. The scale of this space in feet can even be determined if the distance of the camera from the floor can be supplied. The whole representation in three dimensions should be accurate, except for a simplification of hidden details and occasional problems due to the breakdown of the assumptions. However, humans have the same problems.

IV. Picture Input and Reduction to Lines

Pictures are presently being entered into the computer by means of a facsimile scanner, although many types of optical scanners would be suitable. The facsimile scanner, however, was already connected to the computer for some of my previous experiments.[6] A 4 × 5 photographic print is placed on the drum of the scanner and the computer made ready. Then, during each rotation of the drum, a photomultiplier output scans a line of the picture. An analog-to-digital converter samples the photomultiplier output at about 600 cps and sends the computer ten-bit digital intensity values. Thus, in about three minutes, a 256 × 256 raster of intensity samples can be read into the computer. Each sample is compressed to eight bits in the computer, so the storage of one picture requires about half a million bits of memory. Thus four pictures can be stored in the TX-2 memory. Figure 2*a* and *b* shows a picture before and after computer sampling.

When the scanning is completed, the picture is processed with a local differential operator to produce a new raster which has the appearance of a line drawing. The choice of a differential operator is very critical and many variations were tried. Three main criteria can be used to judge such an operation. The edges produced should be as sharp as possible, the background should produce as little noise as possible, and the intensity of the lines produced should correspond closely to a human's ability to perceive the edge in the original picture. Edge sharpness depends upon the number of samples used by the differential operator. Background noise seems to be reduced by using operators symmetric in x and y. In order to make equally apparent edges have equal derivatives, the intensity values of the picture can be subjected to a gamma change so as to make intensity differences proportional to a human's ability to perceive them. According to psychophysical theory, the square root of the intensities should be used in order to achieve the desired effect.[11]

Therefore, after a picture is read in, a differential picture is created according to the functions

$$y_{i,j} = \sqrt{x_{i,j}}$$
$$z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2}$$

where $x_{i,j}$ is the initial intensity value, $z_{i,j}$ is the computed derivative value, and i and j are the coordinates in the two dimensions. The resulting z values indicate the probability of a line through that cell. Even though the square-

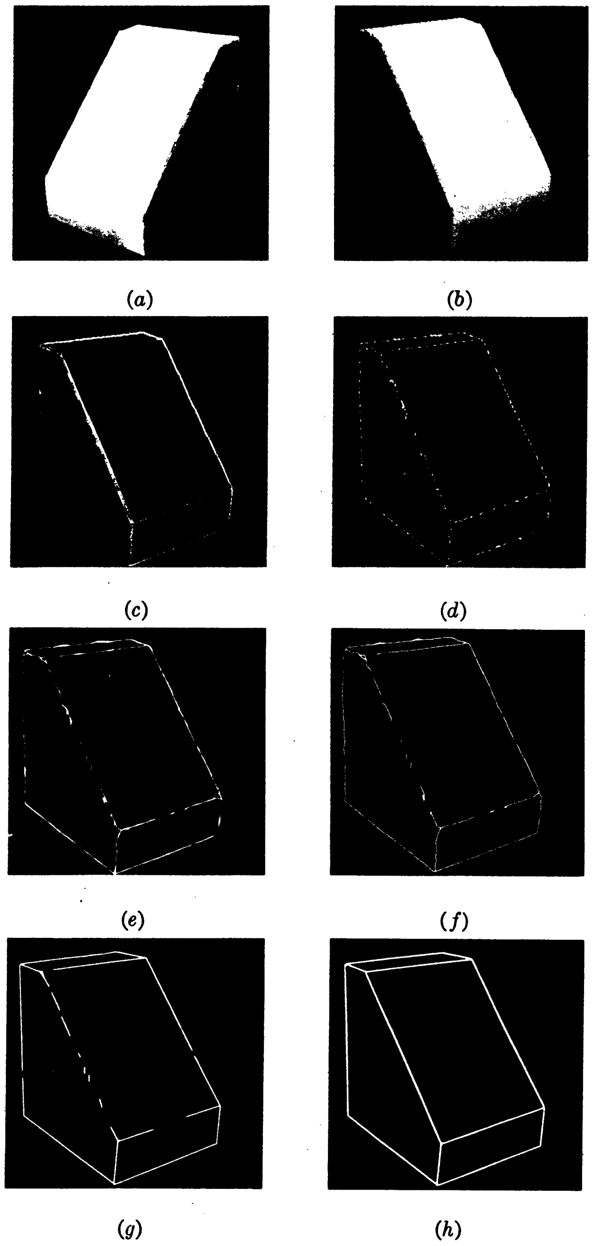

FIG. 2. Picture to line drawing. (*a*) Original picture. (*b*) Computer display of picture (reflected by mistake). (*c*) Differentiated picture. (*d*) Feature points selected. (*e*) Connected feature points. (*f*) After complexity reduction. (*g*) After initial line fitting. (*h*) Final line drawing.

root operation involved takes more time than several simpler operations which were tried, the extra line sharpness and background noise reduction obtained are well worth the additional time. Figure 2c is an example of the result of this differentiation.

After obtaining the differential picture, the problem is to determine a set of lines and end points which correlate well with the raster. It is no simple chore to obtain a list of lines and their end points from a half-million-bit array of data. A simple procedure might be to choose a clip level and start tracing out lines which correspond to a string of adjacent cells in the raster, all of whose values are above the clip level. The hopelessness of this procedure is easily seen when one looks at typical pictures and considers them as a three-dimensional surface where the z values are used as the height. Even a very clean input picture when viewed in this way looks like a bumpy, hilly landscape, with a broken-down stone wall representing the lines, and where some hills are higher than the top of other stone walls. If we imagine the clip level as a flood over this landscape, there is no water level which covers all the hills and yet does not submerge some stone walls. In fact, even by adjusting the water level to be optimal for a particular area, a line will look like stepping stones in a rock-strewn brook rather than a smooth dam. Thus it can be seen that the problem of mapping the walls is not a simple one.

The procedure I have developed for finding the lines from the differential picture makes mistakes in complex pictures and is a complex special-purpose program demonstrating very few general concepts. However, it does manage to produce accurate line drawings from a sufficient group of pictures so that the transformation techniques can be tested on data from real photographs. The description of this procedure will be rather general in nature.

The over-all concept was to look at local features first and build up to the determination of long lines in a series of steps. There are two reasons for this procedure. First, at each step the complexity of the processing goes up since more data are being considered, but at the same time the number of features to process is being reduced at each step. Thus the total time to process the picture is much less than if a one-step process were used. Such a one-step process was tried by Leo Hodes in an attempt to find the lines in black and white line drawings.[4] For each possible line a correlation with the raster was made, and then the line position was corrected and recorrelated. This type of procedure becomes more and more time consuming as the raster size, line length, and accuracy required increase, whereas a multistep process need not consume additional time.

The second reason for a multistep process is that the initial local feature extraction can take into consideration the local noise level and thus detect lines which an over-all clip level would miss.

The first step of the process looks for points in the differential picture which seem to be on a line. The maximum sample in each 4 × 4 square of raster points is considered as a possibility if its value exceeds a small threshold. This threshold is low enough to include all the lines and much of the noise but it does eliminate further processing of smooth areas of the picture. Its value has been determined by experiment to be about one-tenth of the maximum intensity in the original picture. After a point has been chosen in this way, four correlations are performed to find the direction of a line through the point which best fits the data. Four lines, having a length of five samples with slopes of 0, 1, ∞, −1, are correlated with the data around the point, and the ratio of the best to the worst fit is taken. If the ratio is greater than a second threshold (about 3), the point and the best direction are recorded. In this way a set of feature points is obtained, along with approximate line directions through them. The procedure can be considered to be a ridge detector which locates points along each ridge with limits on the height and width of the ridge. The number of feature points obtained is usually between 100 and 1000. Figure 2*d* shows a short line at each feature point, pointing in the recorded direction.

After the feature points are found, the next step is to connect lines between neighboring points. The line directions of the points are used to limit cross connections between adjacent lines. Specifically, a pair of points are connected if they are in touching 4 × 4 squares and if the line's direction will be within ±23° of the direction recorded for either point. Any points left unconnected are eliminated, thus filtering out most of those created by noise. The result is a preliminary line drawing composed of many short lines. There are two major problems with this line structure: sections of a long line may be missing, and sections may be complicated by multiple interconnections. The interconnections are most obvious at corners but also appear along a line as extra width. Figure 2*e* gives a more graphic illustration. To reduce these small networks of lines to a single line or neat corner, two reduction techniques are used. Their important property is that they do not change the over-all connectivity and topology of the line structure. First, the longest side of each triangle is deleted. A triangle is defined here as any three lines in a loop. This cleans out most of the unwanted lines; however, there are a few quadrilaterals left. Therefore, each group of four lines connected in a loop is compressed along its shortest diagonal; that is, the two closest, nonadjacent points are merged. There may still be a few pentagons left but most of the networks have been reduced. The last step in smoothing out the line structure is to remove all small tails or spurs which are unconnected at one end and connect to more than one line at the other end. These smoothing operations are all on short lines so no major features are changed. They do, however, limit the resolution of the input system to about four to eight samples out of 256. This restriction is on resolving short, close lines, not on the accuracy of longer lines. The result of smoothing Fig. 2*e* appears in Fig. 2*f*.

Now that a structural outline of the lines in the picture has been obtained, longer lines can be fitted to sections, missing segments filled in, and extra segments removed. The segments along the path of a true straight line may weave in and out but each point was obtained by the ridge detection technique and is accurate to about one sample width if it was caused by the edge. A sequence of singly connected points with no intersections will probably be caused by the same edge or sequence of edges. By a least-mean-square error-line-fitting routine, a straight line can be put through the points and result in a very high accuracy line. Curve segments could also be fitted to the points if this were desired; the main additional problem would be the choice of the type of curve to try—circle, parabolic, cubic, etc. It becomes more and more obvious, as one considers fitting curves to the picture edges, that it is advantageous to have a set of points already determined, through which to put a curve, instead of having to correlate various curves with the picture data.

To fit a straight line to a sequence of points, a sequential least-mean-square error-fitting routine is used. The problem is to find the best coefficients (a, b, c) for the line equation

$$ax + by = c$$

The data are supplied sequentially in the form of points (x, y) and it is desirable to recompute as little as possible each time a point is added. However, upon each addition of a point, the coefficients (a, b, c) and the new mean-square error E should be available. It is sufficient to keep a history of five numbers, the cumulative sums

$$\sum x, \qquad \sum y, \qquad \sum xy, \qquad \sum x^2, \qquad \sum y^2$$

and the number of entries made n. Then, after each addition of a new point (x, y), the coefficients are computed as

$$\begin{aligned} a &= \sum x \sum y^2 - \sum y \sum xy \\ b &= \sum y \sum x^2 - \sum x \sum xy \\ c &= \sum x^2 \sum y^2 - \sum xy \sum xy \\ E &= c(nc - a \sum x - b \sum y)/n(a^2 + b^2) \end{aligned}$$

Since these equations represent the mean-square best reduction of the unnormalized error $E(a^2 + b^2)$, they do not always produce the least-mean-square distance error as represented by E. However, the solution is much easier than the complete form and just as good in almost all cases.

The procedure for fitting a line to a series of connected points starts by choosing any small line segment as a starting place and moving in one direction until a point is reached with other than two line segments attached, or until the mean-square error exceeds a threshold. When the error threshold is reached, a bend in the true lines has probably been passed, so the procedure is to back up until the angle between the little line segments and the computed line has decreased by a factor of 2. Usually this condition will occur at the bend sought for, since on the other side of the bend the angles must be negative. This procedure works very well and needs no threshold adjustment because of the backup procedure.

When the line has been finished in one direction from the starting point, the other direction is investigated in the same manner, still modifying the same computed line. The method of using cumulative sums as the only history for the line computation allows points to be subtracted out during backup, eliminating the need for large tables of past history. Also, the procedure is fairly independent of the starting point since, during the second direction's backup, the starting point can be passed, thus creating a line totally to one side of the starting point. This will occur when a break point appears just after the starting point, and as the break point is passed in the forward direction, the error does not build up to threshold.

Thus the line-fitting procedure replaces groups of small line segments with longer, more accurate lines. The ends of these new lines are at the intersections of several lines, at break points as detected by the error criteria, or are free and unconnected. Each time a long line is computed, the points at its ends are moved onto the line since the line is more accurate than the points. If several long lines meet at a point, the point's coordinates are computed to be the intersection point of the two longest nonparallel lines. Thus the points become as accurate as the lines connected to them. A special case may come up due to the incomplete removal of a network of small lines: Two lines may be constructed between the same end points. These must be merged and the lines connected to the false intersections reprocessed into one line, if possible.

When all the lines have been fitted to the small segments, the representation of the picture consists of a set of lines and end points mapping the edges in the picture. There still may be sections of lines missing and extra segments near intersections. Figure 1*g* shows the result at this point. Now line filling and merging are done to complete the line drawing. Each line is considered for modification. If the line is of significant length, the nearest points to both ends of the line, which are within about three raster units of the extended line, are considered as possibilities for new end points. A new line is correlated with the differential picture between an end point and the new point and if the average value along the line is greater than a threshold, the line is put in. When a line is very short, it is not extended but is considered for merging or elimination. If the line's end points both connect to one other line, then the end points are merged, otherwise the line is deleted. After extending, merging, or deleting the proper lines, the whole line structure is again processed with the mean-square line-fitting program in order to eliminate extra joints which may have been created. The resulting line drawing is the finished version as shown in Fig. 1*h*.

The entire picture-to-line-drawing process is not optimal but works for simple pictures. It has several useful parts; the differentiation, the feature point extraction, and the mean-square line fitting are the best parts. In the future, I hope to recombine these sections to produce a more general system.

V. Construction of Three-Dimensional Objects from Line Drawing

The program described in this section starts with a planar line drawing and produces a three-dimensional description of the objects shown in the drawing in terms of models and their transformations. The line drawing may be one generated by the picture input process or some other computer program such as the three-dimensional display program (Sec. VI). The main restriction on the lines is that they should be a perspective projection of the surface boundaries of a set of three-dimensional objects with planar surfaces. Any line drawing produced by the three-dimensional display program is acceptable as an input to this program, and since this program's output is an input to the display program, the two programs can be used to check each other. The models used for construction can be any set of three-dimensional building blocks which seem useful so long as all their surfaces are planar. Since the models can be put together so that their joint lines disappear, almost any complex object can be constructed with a very few models. There are only three models presently used in the program: a cube, a wedge, and a hexagonal prism (Fig. 6, in Sec. VI illustrates these models). Section III describes the general procedure used for this two-dimensional to three-dimensional transformation, whereas we now wish to develop the specific mathematics and techniques used in the program.

A. Polygon Recognition

The line drawing which is produced by the three-dimensional display program is just a list of end-point pairs, one pair for each line. This type of input is specially processed to put it in the form wanted. Each line is assigned to a line block in a line list and each point to a point block. Each point indicates which lines are connected to it and each line block points to its end points. Thus, for each end-point pair in the input list, a line block is created and the point list searched for point values close to the end points. If the points already exist, the line is just tied into them; otherwise, a new point block is created. Upon completion of this phase, each point is checked against all lines to see if it lies on a line but is not connected to the line. If this occurs, the line is broken in two and both new ends are tied to the point. The list format produced is a good form for topology processing and is the same format as that produced by the picture input program.

The first problem is to find the polygons described by the lines. In order to trace the polygons easily, the lines tied to each point are ordered by their angle of exit. This allows us to start with any line, choose an adjacent line at one end point and continue around the polygon to the first line, without ever getting off the polygon. This procedure can be made to go clockwise or counterclockwise around polygons, and thus record two polygons for each line. A list of polygon blocks is prepared, each tied to the lines that compose it. The lines also point to their two polygons. As each polygon is produced, the exterior angle at each vertex is computed and the sum of these angles is kept for the polygon. These angles are computed in semicircles, so they are between $+1$ and -1. The sum will therefore be $+2$, if the polygon was hollow on its inside, but if the polygon was really an exterior boundary of an object, the "hollow" part is outside and the sum will be -2. Figure 3 shows the polygons of a cube projection and their exterior angles.

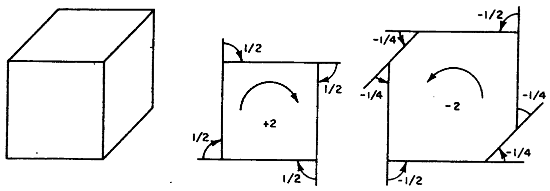

FIG. 3. Exterior angles. The exterior boundary of an object is separated from the normal polygons by the sum of the exterior angles, taking the unconnected direction as the inside. The direction of travel and the sum of the exterior angles appear in the center of (*b*) and (*c*). Angles are in semicircles. (*a*) Complete cube. (*b*) Interior polygon. (*c*) Exterior boundary.

It may not be apparent that exterior boundaries are difficult to separate from the real polygons, but as the computer traces a polygon, it has no concept of the inside; it just traces a closed path with all connecting lines on one side of the boundary. It must expect some negative angles because the polygon may be concave. Thus the sum of the exterior angles is necessary, if the computer is going to separate exterior boundaries from real polygons.

Some further information is obtained as the polygon is traced out: the number of sides of the polygon, the number of negative angles encountered, and the number of near-zero exterior angles. One or more negative angles indicate that the polygon is concave, whereas the zero angles indicate collinear joints which most likely were produced by another object partially hidden behind this one. Thus a first guess at the number of sides the surface really has is the number of lines minus the number of zero angles. The polygon is then marked as complete and convex if there are no negative angles, it is not an exterior boundary, and it does not include a point where a zero angle was observed on another polygon. The last condition eliminates from initial consideration polygons that are most likely partially hidden by another object. Figure 4 shows an example where each complete and convex polygon is labeled with its first-guess number of sides.

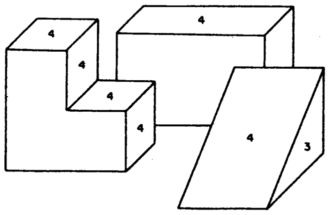

FIG. 4. Complete convex polygons. The polygon selection procedure would select the numbered polygons as complete and convex. The number indicates the probable number of sides. A polygon is incomplete if one of its points is a collinear joint of another polygon.

The reason concave polygons are not considered at first is that the models presently being used in the program are all convex. However, if a model with concave polygons is included, the appropriate concave polygons from the line drawing should be considered. In this connection, the rule is that both the number of positive angles and the number of negative angles of a polygon are invariant under any perspective transformation. Thus the only polygons which need be considered at this point are those which have the same number of plus and minus angles as some model surface. With the models included in the present program, the acceptable polygons are further restricted to have 3, 4, or 6 sides.

A comment on incomplete polygons may be useful at this point. A perfect projection of solid objects with visible width and no two-dimensional markings is being assumed for the input. Since such projections are completely composed of surface boundary projections, they will never contain any points connected to only one line, unless the points are on the boundaries of the picture. If a special external boundary square is included, there will be no incomplete polygons as is assumed. The three-dimensional display program will always generate acceptable input, but the picture input process could miss lines, generate false lines, or include two-dimensional markings. These cases could be taken care of by further checking the picture and other techniques, but at present this task was ignored. The program eliminates such problems by deleting all lines not connected in a closed polygon. Also any isolated polygon is considered to be a two-dimensional marking and is deleted. Thus all letters except *B* are deleted as well as most other simple two-dimensional markings. When the process is completed, these markings could be transferred to the appropriate surface, if desired, but this was not done.

One further computation is performed on each polygon even though it is used only to order the investigation of the polygons. The area of each polygon is found as the program moves clockwise around the points (x_j, y_j),

$$A = \tfrac{1}{2} \sum (x_{j+1}y_j - x_jy_{j+1})$$

This procedure, which is really summing the signed areas of the triangles formed by each line and the origin, is very simple and works for any shape polygon. If the motion is counterclockwise instead of clockwise, as is the case for external boundaries, the area will come out negative. This formula for the area of a polygon was derived by the author since no suitable formulation could be found easily.

B. Model Matching

The first tool used to match the polygon structure to the models is topology. Basically, we wish to find points in the line drawing which fit a transformation of some model. The polygon structure is used to find a suitable model with a set of topologically equivalent points. Then the mean-square error technique is used to find out whether the point positions are related by a simple transformation or not. Topology matching proceeds in four steps. First, each point is examined to see if it is completely surrounded by approved polygons. When such a point is found, the number of polygons of each type is counted. At present, since the only approved polygons are those with 3, 4, or 6 sides, three counts are obtained. A list of triads corresponding to distinct points on the models is maintained so that a quick search will indicate which model points are surrounded by the same polygon structure as the picture point. For example, if a point has three quadrilaterals around it, the list will specify a particular point on the cube model. The other points on the cube need not be listed because they are all similar to each other. Once a model point has been chosen, the program cycles around the picture and model points to line up the order of the polygons. If the orders cannot be matched, other listed model points are tested; however, if they are matched, a list of equivalent point pairs is constructed.

The computation of the optimal transformation matrix from the point pairs is presented in Appendix B. Besides producing the transformation, the procedure generates the mean-square error. A threshold is placed on this error to eliminate models which fit the picture topologically, but do not fit exactly without being deformed. Models having acceptably small error can now be transformed to produce the lines and points which were not part of the fitted area. The points are checked against the picture to make sure they do not fall outside the object's external boundary. If a point does exceed this boundary, the model must be discarded since it would produce new lines not in the picture. Models that pass this test, however, could represent at least part of the object, and are accepted. If a transformed model completely accounts for a group of connected lines, the transform and model are used to represent that object; however, if some lines are left, the procedure described under object construction must be used. Figure 5*a* through *d* shows the processing of a photograph in which a single cube model was used to describe the three-dimensional object, whereas Fig. 6*a* through *d* illustrates a situation in which two models were needed.

The examination of all points surrounded by polygons is only the first step of the topology matching. When all the points are tested, lines are examined for approved polygons on both sides. A second list of model information is searched for any models with such a pair of adjoining polygons. When a line and model are found, the polygons are aligned and a list of point pairs produced. From here on the transformation procedure is the same as before.

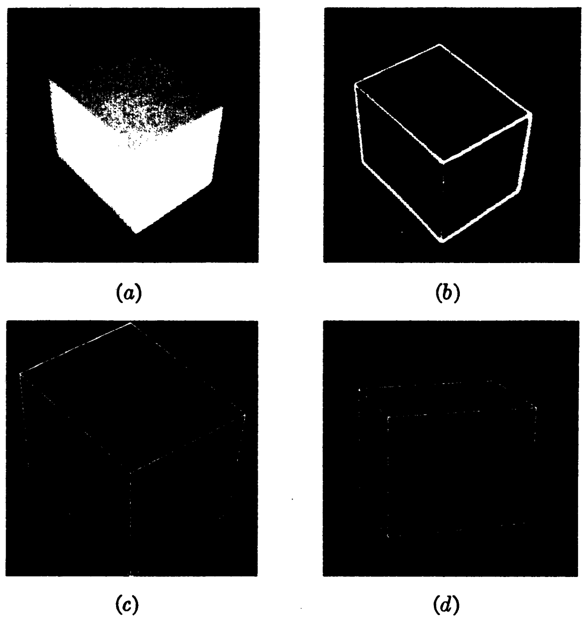
FIG. 5. Single model: reduction of photograph to line drawing and display of three-dimensional construction from another viewpoint. (*a*) Original picture. (*b*) Differentiated picture. (*c*) Line drawing. (*d*) Rotated view.

The third step, after all lines have been examined, is to test each remaining approved polygon. The polygon must have a line attached to one vertex. The model information lists each distinct model, polygon, and vertex for each type of polygon, and the point pair list is easily prepared. The fourth step, if necessary, is to take each point with three lines out of it and test these four points against each distinct vertex and line orientation in each model. This test is a last resort because there are so many model possibilities and in every case a transformation can be found which makes the mean-square error zero. Only finding a point outside the boundary can eliminate these cases.

The four steps are ideal for the cube model since the number of points found decrease by one with each step. Almost any visible piece of a cube will be processed with the maximum number of points possible. Even though this is not true for all models, the four steps are a good approximation to a comprehensive topology search. Topology tests and matching are very difficult to implement on a computer, even with the aid of a good list structure. Computer languages seem to be far superior at numerical and symbolic manipulation than the decision-loaded searching associated with topology. The inherent limitations of the serial processing of a line struc-

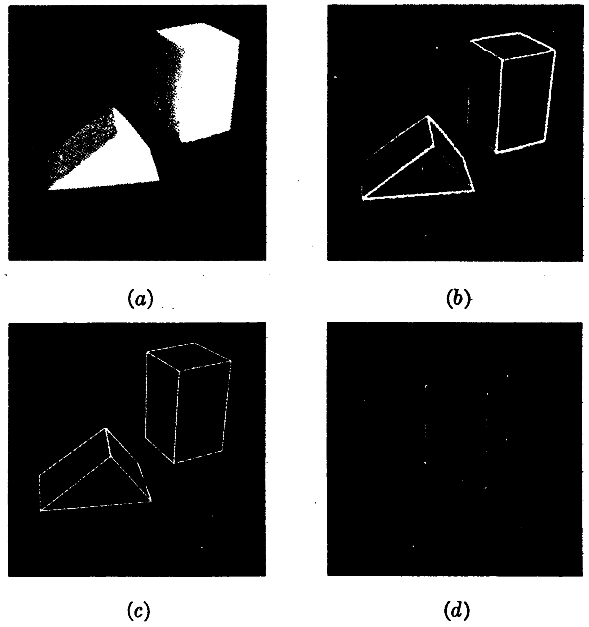

FIG. 6. Multiple objects: reduction of photograph to line drawing and three-dimensional construction, involving ground plane determination of depth. (*a*) Original picture. (*b*) Differentiated picture. (*c*) Line drawing. (*d*) Rotated view.

ture, without some global picture of what one is looking at, make specific tests such as the four steps far easier to achieve than any general procedure. Even to accomplish these steps without continual searching and backtracking, the list structure must be organized in a very special manner. At each vertex the connecting lines are ordered by angle of exit in a counterclockwise ring. Thus any time the program arrives on one line, it can easily exit on an adjacent line. Further, the lines in each polygon are ordered in a clockwise ring, with the end points of each line and the two polygons of the line specifically related. This is important because, upon arrival at an end point of a line, the program can immediately identify the right and left polygons and proceed around either one. Both the model structure and the line drawing are organized in this way. The definition of clockwise and counterclockwise in a three-dimensional model structure is not obvious but, for compatibility with the projections of a solid model, all such orderings are made while looking in from the outside.

C. Compound Object Construction

A compound object is a single solid object which is not a transformation of a single model, but must be formed by piecing together several models. Whenever two models are fitted together such that they have a frontal plane in common, the three-dimensional display program will eliminate any piece of line which touches both models and is in that plane. Since these joint lines do not reflect any surface discontinuity, it is expected that they will also be missing in the line drawing input. If joint lines happen to appear in the input, the resultant structure will probably be the same as if they were missing; therefore, the display will not include them. As models are found which fit a part of an object, their lines are projected back onto the drawing and the joint lines found. Thus the complex unapproved polygons are cut up by these joint lines into smaller polygons until each piece is approved. The following rules are for deleting an accepted model from an object and reforming the picture to make ready for the next model identification. The expression "*T* joint" refers to a vertex at which two collinear lines and one other line, the "stem," meet. During the process, parts of the drawing are deleted, but these changes are never allowed to modify the external boundary polygon since it must be used to test new models. The concept of a "visible" model line or point refers to the points and lines on the frontal surfaces of the model.

1. Each visible point of the transformed model is projected on the drawing.
   a. Any new point pairs disclosed by the proximity of model and picture points are used to recompute the transformation for better accuracy.
   b. If a model point falls on a picture line, the line is cut in two and the point inserted.

2. All the model lines and points are added to the picture if they are not already there. Any picture point which lies on a model line, but not on either end, is separated from the model line structure. Thus a picture line that ends in a T-joint stem may be extended to its proper end point, and the collinear T-joint lines unified into one line. Joint lines are those visible model lines which were not in the picture and which divide a picture polygon.

3. Each visible model point in the picture which does not connect to any nonmodel lines is now marked "used." Also, all points on polygons with more than two "used" points are marked "used." The joining polygon between parts is the polygon which includes the joint line but is not divided

by it, and all the points on such a polygon must be unmarked. Now all "used" points are deleted along with their attached lines and polygons. Also, any line in a joining polygon which is the stem of a T joint at both ends should be deleted.

4. All remaining model lines should be marked as unnecessary. If all lines left connected to them are unnecessary or if no lines are left connected, the object has been finished.

Each time a model is stripped from an object in this way, its transform and model name are saved as part of an object block. For each model, a point which was connected to the remainder of the object is remembered, so that the depth relations between the parts may be computed. When an object is finished, there will be a string of points connecting its parts. An object transformation is set up to position the depth of the whole object; hence, the first model can be assumed to have the correct depth. Then each model whose point connects to the first has its transformation modified so that the points have the same depth. Then the models connected to those are updated and so on.

To correct a transformation R_0 when a known point $\bar{v}$ should be equal to a point $\bar{p} = \bar{q}R_0$, the new lower-left element x of R_0 is computed as

$$x = \frac{q_4 v_1 w - p_1 v_4}{q_4(v_4 - r v_1)}$$

Here w is the present lower-right element of R_0 and the new $w' = w + rx$. Note that the focal ratio r for the picture must be known at this point. However, the accuracy of r will not affect the accuracy of positioning the parts with respect to each other.

Eclipsed objects, or objects partially hidden from view by other objects, are automatically taken care of by the construction rules. One case, however, needs further attention. When an object is so well hidden that a dimension cannot be determined, this dimension must be estimated. An example of this case would be when only the top of a building is visible over another. The first assumption we make is that the object is supported by the ground plane. But a second assumption is needed to place the object, and the program assumes that the hidden object just touches the object in front. This is not a very good assumption, but there are no good assumptions.

Figure 7 illustrates the construction of a compound object. The original line drawing appears in A1 and includes a compound object and a partially hidden object. Since there are no points surrounded by acceptable polygons, we must look for a line with good polygons on both sides. There is only one such line to which a model can be fitted and this is in the upper object. Both the cube and wedge models fit this object; however, the cube is always tested first to avoid splitting cubes into wedges. The lines of the cube model are then projected onto the line drawing as in A2, and the transformed model is entered into the three-dimensional structure as displayed in A3. After a model has been identified, the "used" points and lines are deleted, thus producing the line drawing in B1. Now a new search for fitting models is made. The lower-right quadrilateral and the bottom line adjoining it are found to fit a cube model, resulting in B2 and B3. Next, a cube is fitted at

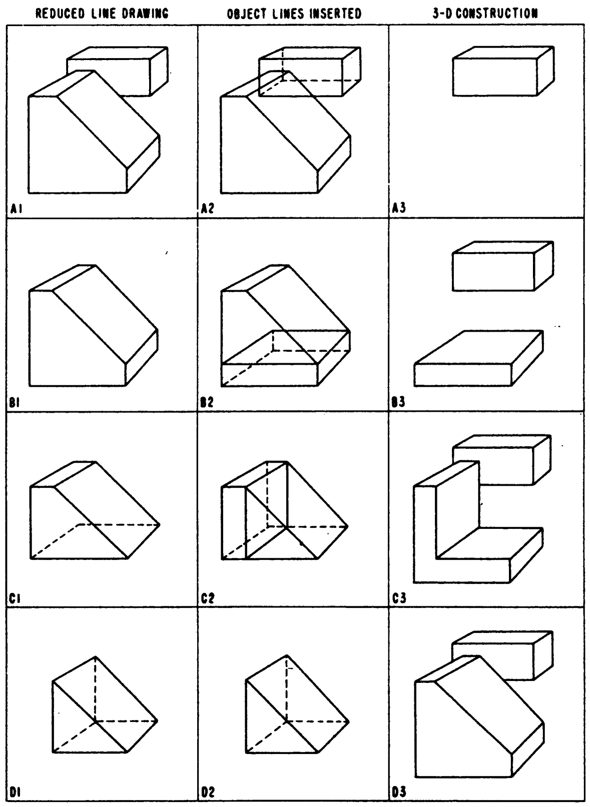

FIG. 7. Compound object construction: original line drawing in A1 is processed to obtain three dimensions in D3 by sequential recognition and deletion of four models in steps A, B, C, and D.

the left, in C1, producing C2 and C3. Finally, just a wedge is left in D1 and since all the back lines have already been determined, D2 appears the same. When this model is added to the three-dimensional structure, the result is a complete description of the objects and can be displayed as in D3 or from any other point of view. Figure 8*a* through *d* shows the computer processing of a similar compound object from a photograph. The collection of models describing the three-dimensional object can then be rotated as in Fig. 8*e* through *h*.

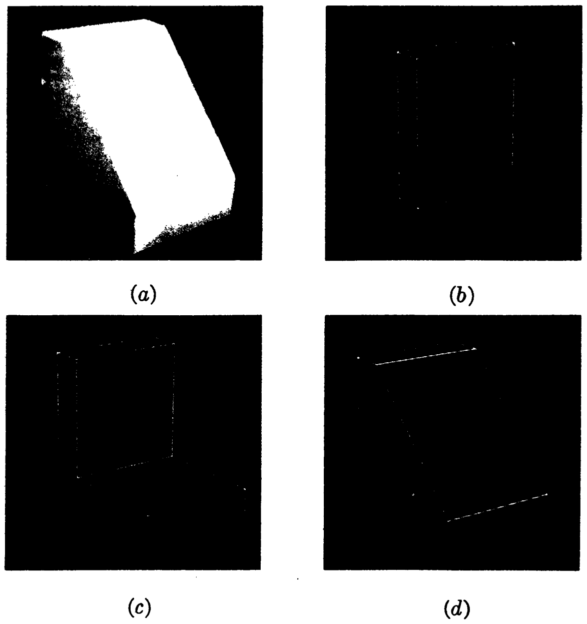

FIG. 8. (*a–d*) Compound object construction: processing of Fig. 2 to obtain three-dimensional description of compound object. (*a*) Original picture. (*b*) First construction model. (*c*) Two construction models. (*d*) Complete three-dimensional object.

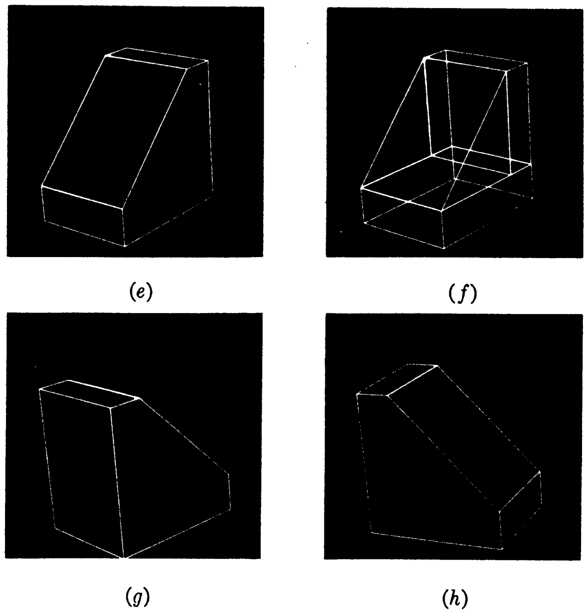

FIG. 8. (*e–h*) Compound object construction: rotated views of object obtained in Fig. 8(*d*). (*e*) Rotated view. (*f*) All lines of Fig. 8(*a*). (*g*) Second rotated view. (*h*) Third rotated view.

D. Ground Plane Assumption

When all the objects in the picture have been constructed, they still need their object transformations adjusted for depth. If an object consisted of a single model, the transformation listed is R_0. If the object was compound, an identity matrix was set down as the object transform and the model transforms all relate to it. In both cases each object has one free variable related to depth and size. The support theory would place each object on top of another or on a ground plane. In order to simplify the present program, the ground plane is the only support assumed. However, the addition of object support will not be very difficult.

Finding the ground plane is the most difficult part of the depth computation. Each object could be examined for possible support planes and all objects compared for a common plane. An object could be supported by anything from three points to a full plane in contact with the ground. To simplify this chore, it was assumed that the picture was upright and each object had a full plane in contact with the ground. A simple test of the slope of each plane of an object is used to determine if that plane is the bottom. This test merely asks that the bottom is not visible, that it faces the downward z direction, and that the tilts in the y and x directions are moderate. The best such plane is chosen from each object, and all are expected to agree. The slope of this plane is the only information available from the incomplete transformations, so the distance of the plane below the origin or focal point must be assumed. This distance just sets the numerical scale of all distances, so it might as well be unity. Thus a ground plane is determined and all the object bottoms are now made to lie in this plane.

With each model surface is stored its plane equation vector for the use of a three-dimensional display program. The dot product of this vector with any position vector is zero if the position is on the surface, and positive if the position is inside the solid. These plane vectors should be transformed by R^{-1} to become plane vectors of the transformed model. If the plane vector of an object plane is transformed by R_0^{-1}, it has the correct slope but not necessarily the correct length. In other words, its first three components are correct. These are used to find the ground plane. Now with the ground plane distance equal to one, we must find the depth x for each R. The lower-right component of R_0 we shall call w, and the same component in the final R, w'. We first set $x_1 = -w/2r$ and $w_1 = w/2$ to obtain R_1. If we transform the plane vector by R_0^{-1} and R_1^{-1}, we obtain $\bar{v}_0$ and $\bar{v}_1$. Normalizing these vectors so that the sum of the squares of the first three components is 1, we obtain normalized fourth components p_0 and p_1. The plane vector for R has a fourth component which is a linear combination of p_0 and p_1. Setting this combination equal to one and solving for the depth x, we obtain

$$x = \frac{w(p_0 - 1)}{r(1 + p_1 - 2p_0)}$$

$$w' = w + rx$$

Here again the focal ratio r is needed, and this time if r is zero, the computation of x blows up. This just reflects the fact that an infinite projection has infinite depths; however, it restricts r to nonzero for this procedure.

VI. Three-Dimensional Display

After a list of three-dimensional objects has been obtained in some manner, it should be possible to display them from any point of view. The sections of objects behind other objects should not be seen, nor should the back lines and construction lines of individual objects. The three-dimensional display program will do all this and more. It allows macrolike instances of objects so that a single object construction can be used many times with different transformations. It allows structures of models to be built up by the use of the knobs, push buttons, and light pen. Any object can be duplicated, deleted, or transformed. These extras make possible the construction of test cases for the two-dimensional to three-dimensional program to process. However, the most significant feature of this program is the mathematical technique which makes possible the hidden line removal.

A. Storage Structure

A good method of storing three-dimensional data is extremely important. The structure used is the basis for both the display program and the three-dimensional construction process. Therefore, the data necessary for hidden line removal must be quickly available and at the same time the topological structure must be suitable for model matching.

The list structure used is a list of tied blocks connected in rings. Ring list structures were developed for the TX-2 computer by Sutherland for his Sketchpad system.[12] Sketchpad allows a user to draw two-dimensional line drawings on the computer display with the aid of the light pen, knobs, and push buttons. An extension of this work to three dimensions is currently being completed by Johnson.[13] These two systems use ring list structure and, in order to be compatible with them, the same format is used in the three-dimensional display program. However, the exact block form used is different because of the different data requirements. In the ring structure, a block of registers is used for each item and contains pairs of ties to other blocks. Each pair of ties is part of a ring which allows the program to move from block to block around the ring in either direction.

The basis of all three-dimensional forms is the set of models. Each model block is tied to lists of its points, lines, and surfaces. The point blocks are tied to the lines connected to them and include a four-component position vector. Since a homogeneous vector can be normalized without changing the point, each component of the vector can be represented by a fixed-point, 36-bit number. The line blocks are tied to two point blocks and two surface blocks, since two planes determine a line. The surface blocks are tied to a ring of lines which represent the surface polygon and also include a plane vector. This detailed structure is needed for models only, since the objects are to be composed out of transformations of models. The models are always in the list structure and must be referred to by instances in order to be displayed. An instance is an intermediate block between a picture and either a model or another picture. Each instance includes a 4 × 4 transformation matrix and also the inverse of this matrix. The picture blocks may be referred to by any number of instances and have as their parts any number of instances. Thus each picture represents an object or a collection of objects and is composed of transformations of other pictures or models by means of instances. One picture is the current picture being displayed, and it has only one instance containing the picture transformation. Figure 9 shows a possible structure leading from the models to the current picture. The instances have been compressed to ties with matrices on them for simplicity.

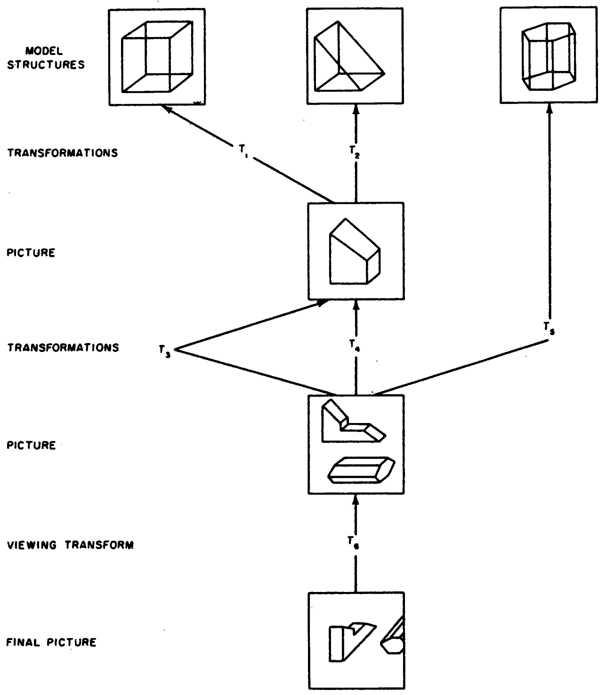

FIG. 9. List structure formation: representation of list structure formation of compound pictures. Each "picture" is composed of transformed versions of models and other pictures. There may be as many levels of pictures as necessary.

B. Display Generation

In order to display the picture represented by the current storage pattern, a recursive procedure is used, starting with the current picture. Each picture block has a temporary storage area to store the effective transformation and its inverse at that picture level. One recursion involves taking the transformation at one picture level, premultiplying by the matrix of one of the instances the picture refers to, and placing that transformation in the picture block referred to by the instance. The inverse matrices are carried along also, but postmultiplication must be used for them. If the instance refers to a model, the matrices obtained are put in a transform block along with the model name. Each time a model is reached, the process backs up to the last picture block and proceeds to the next instance, or if all instances have been processed, it backs up to the preceding picture block. In this way a list of transform blocks is obtained. Each transform block is now processed by tying to it a list of point-pair blocks obtained by transforming the end points of each line in the selected model. Each position vector is postmultiplied by the transformation in the transform block to obtain the new position vector. Also, the plane vectors of the model are transformed by the inverse matrix and collected as the columns of a volume matrix.

Thus a list of transform blocks is obtained, each block of which has a volume matrix and a list of point pairs. The point pairs represent all the lines in the display and can now be processed to eliminate the hidden lines. The display coordinates of a point are obtained from the four-component position vector $\bar{v}$ as

$$x = v_2/v_4$$
$$y = v_3/v_4$$

C. Hidden Line Elimination

Three steps are required to prepare a line for display. First, it is trimmed off at the edges of the display. Next, the back lines of each model are deleted. Third, the sections of each line which are hidden by other models are removed. It is the third part which is difficult and time-consuming. As far as I know, no one has ever devised a procedure for determining hidden line segments. One can imagine brute force methods such as calculating all the line intersections on the focal plane and then computing which lines were in which polygons and tracing out the frontal lines. But procedures such as this are hard to make complete for all cases, and the processing time could be fantastic. Therefore, a new mathematical method was conceived which utilizes volume inequality matrices to find out whether a point is inside or outside a volume. This test can then be extended by linear inequality solutions to tell which segment of a line is behind a volume. This is why the inverse transformations, plane vectors, and volume matrices are needed.

Since they are available, they can be used to advantage in the first two steps.

The volume matrix V of a model is a $4 \times n$ matrix with the n-plane vectors as columns. In the homogeneous coordinate system, a dot product of a position vector and a plane vector produces a measure of the distance of the point from the plane. The plane vectors of the models have had their signs adjusted so that the dot products will be negative if the point is outside the solid and positive if the point is inside. Thus, when a position vector $\bar{v}$ is postmultiplied by a volume matrix, the resulting vector $\bar{v}V$, since it is a collection of dot products, will have at least one negative term if the point is outside the volume. If any terms are zero, the point is on the model's surface, and if all are positive, the point is inside. In order that this test may work, the models must be convex. This is a small restriction, since the objects constructed with the models need not be convex,

$$(\bar{v}V \geq 0) \Rightarrow \bar{v} \text{ inside volume } V$$

In the expression above, the inequality sign means that all components of the vector should be nonnegative. Expressions and formulations of inequality matrices have been used in the field of linear programing to express the interior region of convex polyhedral cones where the general problem was the optimization of a function in such a region. However, literature in this field does not seem to cover the geometrical type of problem we are concerned with.

A volume matrix can be used for the first step of the line reduction: the elimination of lines from the display. As well as cutting off lines at the display edge, we wish to cut off lines in front of the focal plane. Thus a volume matrix can be designed which has planes at $y = \pm 1$, $z = \pm 1$, and $x = 0$. These planes form a semi-infinite box, and we wish to find the section of each line inside the box. First, we shall define the volume matrix V_0,

$$V_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

The set of all lines is the list of point-pair blocks. We shall want to use a line equation of the form

$$\bar{v} = \bar{s} + t\bar{d}, \qquad 0 \leq t \leq 1$$

where t is a running variable, moving the position $\bar{v}$ from $\bar{s}$ to $\bar{s} + \bar{d}$. A point-pair block has two points, $\bar{s}$ and $\bar{r}$. If the fourth component of these vectors is made to agree by normalizing one vector, the difference vector $\bar{d} = \bar{r} - \bar{s}$ can be formed. With this line representation we can proceed to find the values of the variable t for which the following inequalities are true:

$$\bar{s}V_0 + t\bar{d}V_0 \geq 0$$

This set of inequalities requires that the line be inside V_0 and we can solve for the minimum and maximum t. If we define

$$\bar{p} = \bar{s}V_0, \qquad \bar{q} = \bar{d}V_0$$

then for all j,

$$p_j + tq_j \geq 0$$

For $q_j > 0$, $t \geq -p_j/q_j$. For $q_j < 0$, $t \leq -p_j/q_j$. From these relations it is fairly obvious that we can obtain a minimum and maximum value for t and thus define the segment of line inside the volume. This segment then replaces the old point pair.

The second step of the line elimination is the removal of all the lines hidden by their own volume. Although this would be done automatically if we went right on to the third step and included the line's own volume among all the others, it is faster to eliminate these back lines ahead of time. The volume matrix associated with a line contains the plane vectors of the model, and two of these planes intersect to form the line. In each point-pair block we keep track of which columns of V are the line's plane vectors. The first component of a plane vector is the x component and will be negative if the plane faces toward the observer. Thus the second step is very simple: a line should be deleted if both the x components of its plane vectors are positive.

The third method of eliminating a line is to see if it is hidden by some other volume. For this test, each remaining line is tested against every volume matrix except its own. The procedure is similar to that of step one except that a two-variable inequality solution must be found this time. The segment of line found is the part to be deleted, and this may be an end, the center, or the complete line.

A volume matrix produces inequalities which tell if a point is in the volume. In step one we used a point with one degree of freedom t along a line. Now the line may be behind the volume so a second degree of freedom must be used to move the point forward through the volume. If a point on the line is moved forward in x in this transformed space, it will have to move through the volume, if it is hidden. The variable point will thus be represented as before with a variable t along the line, but also with a variable α, in the $\bar{x} = [1, 0, 0, 0]$ direction.

Variable point:	$\bar{v} = \bar{s} + t\bar{d} + \alpha\bar{x}$
Inequalities:	$\bar{v}V \geq 0$
Thus:	$\bar{s}V + t\bar{d}V + \alpha\bar{x}V \geq 0$

Now $\bar{x}V$ is just the top row of V, which we can call $\bar{w}$, and $\bar{s}V$ and $\bar{d}V$ can be computed as before.

Define:	$\bar{p} = \bar{s}V, \quad \bar{q} = \bar{d}V, \quad \bar{w} = \bar{x}V$
Thus for all j:	$p_j + tq_j + \alpha w_j \geq 0$
Where:	$0 \geq \alpha \geq 1, \quad 0 \geq t \geq 1$

These inequalities must be solved for the minimum and maximum t for any nonnegative α. A few simple tests allow the removal of equations which are always satisfied or the termination of the test if an equation will never be satisfied. Ignore equation:

$$(q_j > 0) \wedge (p_j \geq 0) \wedge (p_j + w_j \geq 0) \tag{1}$$

$$(q_j < 0) \wedge (p_j > 0) \wedge (p_j + q_j \geq 0) \wedge (p_j + q_j + w_j \geq 0) \tag{2}$$

Quit, leave line:

$$(q_j > 0) \wedge (p_j < 0) \wedge (p_j + q_j \leq 0) \wedge (p_j + q_j + w_j \leq 0) \tag{3}$$

$$(q_j < 0) \wedge (p_j \leq 0) \wedge (p_j + w_j \leq 0) \tag{4}$$

These tests speed up the process considerably since only a few planes of each volume are really involved in the determination of t. To solve the remaining equations, many methods are possible and a very simple one was chosen. Each inequality is considered as an equality and all intersections between these equalities are found. Also, the intersections of these equalities with the boundaries, $\alpha = 0, 1$ and $t = 0, 1$, are found. Each intersection results in a pair (α, t), and these pairs are tested in the inequalities. Any pair satisfying all inequalities and the boundary conditions is used to compute a minimum and maximum t. Because the actual minimum and maximum t must occur at intersections, this is a complete solution. It is also fairly fast since there are usually very few equations. If no pair will satisfy all the inequalities, then the line is not hidden by the volume and is left intact. Otherwise, the section of line corresponding to the solution area, between minimum and maximum t, is deleted.

One case of interest can be modified during this process. We want to eliminate joint lines between touching, tangent objects. A joint line, when processed with the touching solid, is detectable since the inequality caused by the plane through the line will have $q_j = p_j = 0$. If this occurs, the solution space is limited to $\alpha = 0$. Thus the joint line can have a proper solution and will be eliminated. However, this also occurs for joints between objects where the planes are not tangent. We want these lines and must act to preserve them. Therefore, whenever $q_j = p_j = 0$, we make a special test to find out if either of the planes through the line from its own volume is parallel to the plane being tested. We also ask that the tangent planes face the same direction. If parallel planes facing the same way are not found, the line under consideration is kept intact and the rest of the test skipped.

This completes the testing of one line with one volume. The line is then tested against the other volumes. The final result of the complete hidden line removal is a modified list of point pairs. These lines are then displayed. The complete process tends to be time consuming for complicated displays, but the time does not seem to go up as the square of the number of objects, as might be expected. This is because objects far away from the line being processed are quickly disposed of by conditions (3) and (4). Thus the time consumed is on the order of one second per object for displays of up to thirty objects. It is not necessary to do the complete hidden line removal, though, if a faster moving display is required. The complete process can be reserved until the desired view is obtained.

D. Display Construction

Besides being able to display the list structure, the display program has provisions for modifying the list. The picture transformation in the first instance can always be changed by a rotation about each of three axes, a translation in three directions, or a size change. All transformation changes are obtained through the use of four shaft encoder knobs on the computer console. The function of these knobs is selected by means of push buttons. In addition to modifying the picture transformation, the light pen can be used to point out any instance transformation for modification. The pen is pointed at the object to be modified, and a level register indicates which instance level of the object to modify. This method is somewhat crude but does allow any instance to be modified. The transformation changes allowable for objects include rotation, translation, three size components, three skew components, and an over-all size factor. Beyond these transformation controls, any object can be deleted or duplicated. New instances of models can be generated and instances made of present pictures. These controls allow the construction of any list structure possible or the modification of any existing structure. Thus test pictures may be generated to facilitate the testing of this program and the three-dimensional construction program. Some sample pictures generated on the computer appear in Figs. 10*a* through *h*. These photographs illustrate the complexity of three-dimensional arrays which can be constructed on the computer in a few minutes.

VII. Conclusion

In the past, research in the pattern recognition field has been limited to the identification of two-dimensional shapes, mainly because it was thought that any three-dimensional analysis would be more difficult. The idea seems to have been that the two-dimensional work would pave the road for future three-dimensional work. However, progress has been slow and it may well be that the study of three-dimensional projections is an easier step. The human visual field is the result of a projective transformation, and the shapes perceived are independent of this transform. Thus it makes sense to utilize this transformation, since our goal is to recognize the same similarity classes which humans do.

The mathematics necessary to go from a photograph to an object list have been described. A set of transforms is found which takes a set of models

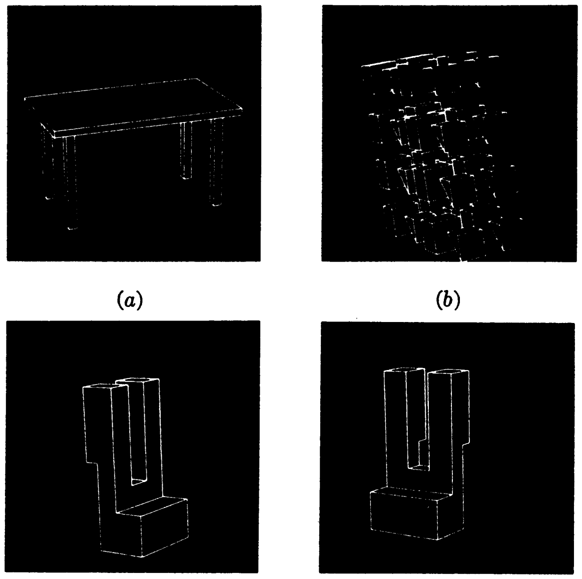

FIG. 10. (*a*–*d*) Three-dimensional displays: pictures constructed with three-dimensional display program. (*a*) Table. (*b*) Array made with instances. (*c*) Compound object. (*d*) Rotated view of object.

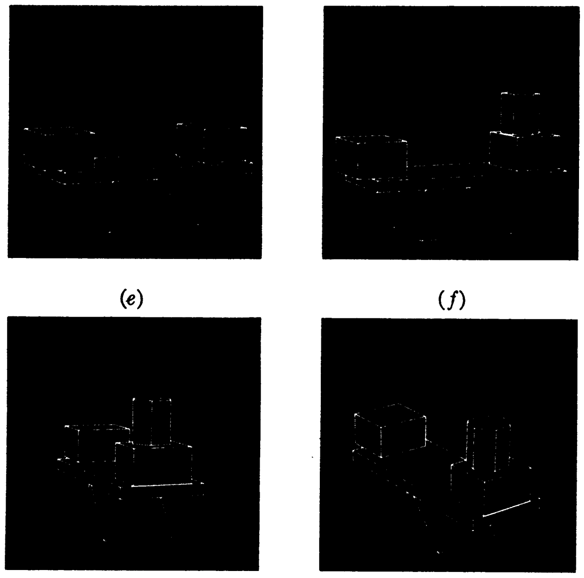

FIG. 10. (*e*–*h*) Three-dimensional displays: pictures constructed with three-dimensional display program showing hidden line elimination. (*e*) Seesaw. (*f*) With hexagonal prism. (*g*) Second view. (*h*) Balanced.

into the shapes observed. The models are then the invariant shapes which we perceive as the same object from any point of view. Individually or in specific groups, they could be given names.

The process of creating an object list from a picture is mainly mathematical, based on the natural laws of the space around us. It is based on the assumptions of model construction and object support and shows the theoretical implications of these concepts. Four or more points on an object must be seen in order to find the correct model and its transformation. Further, the depth relationships between objects depend on the focal ratio which can either be assumed as a camera constant, or calculated from the picture if the data are accurate enough. If the focal ratio used is wrong, the effect will be a contraction or expansion of the depth dimension. When both the focal ratio and the distance from the camera to the ground are known, the exact size of each object can be calculated.

The two-dimensional to three-dimensional and three-dimensional to two-dimensional programs are completely general, as long as the assumptions of model construction and support are not violated. The display program can handle any structure made up of transformations of models. The construction program will always produce a three-dimensional structure which projects to the given line drawing except that it will eliminate any two-dimensional markings, isolated polygons, and superfluous joint lines. Any drawing produced by the three-dimensional display program will be correctly reconstructed into the three-dimensional structure if the objects are properly supported and sufficiently visible.

The programs are all written for the TX-2 computer at Lincoln Laboratory. The TX-2 is ideal for this type of work because of its large memory for picture storage and its special input-output equipment. During the course of developing these programs, I designed a vector-drawing display for the computer which can draw line and circle segments. This display enables the computer to display a line drawing continuously and still spend most of its time computing new data. Thus it is possible to display rotating objects and have them move fairly smoothly.

The input program has about 5000 instructions and uses over 40,000 registers of data storage for its pictures and lists. It takes about one minute to process a picture into a line drawing of which half is for differentiation. The three-dimensional construction and display programs are each about 3000 instructions and use from 5000 to 40,000 registers of data storage depending upon the number of objects. Both construction and display take about one second per object. All told, a rotated view of the objects in a photograph might be obtained in two minutes.

I foresee at least two uses for this type of picture handling system. First, it could be used for an information reduction system to aid in the transmission of pictorial information. However, the necessity of an ultrahigh-speed computer will probably limit this use. Second, the computer programs will be useful input-output tools for future investigations of three-dimensional processes. The biggest benefit of this investigation, however, is an increased understanding of the possible processes of visual perception.

APPENDIX A
Homogeneous Coordinates

In a homogeneous coordinate system, a fourth coordinate, or scale factor, is used in such a way that the total scale of a vector is unimportant. That is,

$$k\bar{v} \equiv \bar{v}$$

I am using $\equiv$ to indicate that the same point is represented even though the individual components may not be equal. The above form is achieved by defining the point's coordinates, X, Y, Z, in terms of the homogeneous components x, y, z, w as below,

$$X = x/w, \qquad Y = y/w, \qquad Z = z/w$$

When new points are introduced into the system, w may be assigned to any nonzero value and these equations used to find x, y, and z. When points are to be displayed, the same equations are used to find X, Y, and Z. An added advantage for a fixed point computer is gained by using a homogeneous system: w may always be chosen so as to keep the numbers normalized.

A plane is represented by l, m, n, p such that on the plane

$$lx + my + nz + pw = 0$$

I have chosen to represent points by row vectors; therefore, to transform a set of points $\bar{v}_i$, each is postmultiplied by a 4×4 matrix H,

$$\bar{v}_i' = \bar{v}_i H$$

The advantage of homogeneous coordinates is that a single transform H can accomplish a full projective transformation. Normally, it is convenient to separate the various functions provided by a transform until they are needed and then multiply them. Below is a breakdown of a transform H, consisting of a rotation by a standard 3×3 R matrix, a translation by a vector $\bar{v} = (x, y, z, w)$, a perspective transform from a focal point at f on the x axis, a translation after perspective to a center (y_0, z_0), and a total picture scale factor S. This sequence might represent the transform made by taking a picture with an arbitrary camera orientation and making an enlargement of a section. The transform would take the real space points X, Y, Z into a Y', Z' on the print. An X' will be formed which can be used for eliminating object overlap.

$$H = \underset{\text{Rotation}}{\begin{bmatrix} & & & 0 \\ & [R] & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{\text{Translation}}{\begin{bmatrix} w & & & \\ & w & & \\ & & w & \\ x & y & z & w \end{bmatrix}} \underset{\text{Perspective}}{\begin{bmatrix} f & & & -1 \\ & f & & \\ & & f & \\ & & & f \end{bmatrix}} \underset{\text{Translation}}{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & y_0 & z_0 & 1 \end{bmatrix}} \underset{\text{Scale}}{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & S \end{bmatrix}}$$

It should be noted that a set of points $\bar{v}_i$ can be written as the successive rows of a matrix V and transformed by one matrix equation, $V' = VH$.

Also, a plane equation can be expressed as a scalar product in terms of $\bar{n} = (l, m, n, p)$,

$$\bar{v} \cdot \bar{n} = 0$$

A plane normal n can be transformed along with the points of a space,

$$\bar{v}' = \bar{v}H, \qquad \bar{n}' = \bar{n}(H^{-1})^T, \qquad \bar{v}' \cdot \bar{n}' = 0$$

Of course, if the transform is orthogonal, $(H^{-1})^T = H$.

APPENDIX B
Similarity Test

We are given a matrix A of n points (x, y, z, w) from a model and want to find a transform H that will most nearly fit n points (y, z, w) in a matrix B. Thus we hope for

$$AH \cong B$$

However, we cannot write an equality sign above without introducing a diagonal scale matrix D which will allow the w_i values of AH to differ from the w_i of B,

$$AH = DB$$

We now have 12 unknowns in H (3×4) and n unknowns in D $(n \times n)$. Matrix A is $4 \times n$ and matrix B is $3 \times n$, creating $3n$ equations. Therefore, $n \geq 6$ should produce a complete solution. We shall use a minimum square-error technique to solve the equations. Thus we wish to minimize the squared error in each equation. We shall use A' to indicate the transpose of A,

$$\frac{\partial}{\partial h_{lm}} \left[\sum_i^n \sum_k^3 \left(\sum_j^4 a_{ij}h_{jk} - d_i b_{ik} \right)^2 \right] = 0$$

$$\sum_j^4 h_{jm} \sum_i^n a_{il}a_{ij} = \sum_i^n d_i a_{il} b_{im}$$

or $A'AH = A'DB$. Thus

$$H = (A'A)^{-1}A'DB$$

Now we must find D,

$$\frac{\partial}{\partial d_l} \left[\sum_i^n \sum_k^3 \left(\sum_j^4 a_{ij}h_{jk} - d_i b_{ik} \right)^2 \right] = 0$$

$$\sum_j^4 a_{lj} \sum_k^3 h_{jk} b_{lk} = d_l \sum_k^3 b_{lk}^2$$

Thus the diagonal terms of AHB' equal those of DBB'. Substituting for H and making the definition

$$G = A(A'A)^{-1}A' - I$$

we obtain a matrix $GDBB'$, which has zero diagonal terms.

$$\text{Define: } Q = BB'$$

If we now multiply the terms of G by those of Q', term by term, we get a new $n \times n$ matrix S,

$$S_{ij} = g_{ij}q_{ji}$$

$$\text{Define: } \vec{d} = d_1 \cdots d_n$$

Now the vector $\vec{d}$ or diagonal terms of D can be found by solving

$$Sd = 0$$

This equation requires S to be singular, with degeneracy at least one. If the degeneracy is one, the problem is solved since the common scale factor of D and H is unimportant. A degeneracy more than one means that too few equations were used ($n < 6$) or that the picture had no perspective. However, by assuming a value of unity for each undefined d_j, an accurate, but not complete solution, can be found.

When D is found, H can be found,

$$H = (A'A)^{-1}A'DB$$

It should be noted that for $n = 4$, A^{-1} will probably exist and in this case the best solution obtainable is one with no perspective,

$$H = A^{-1}B$$

If solutions without perspective are expected, the matrix D is unnecessary and the ordinary minimum square-error solution holds,

$$H = (A'A)^{-1}A'B$$

An error criteria can be found to indicate the mismatch of model and picture. An error matrix E is found,

$$E = AH - DB \quad \text{or} \quad E = GDB$$

Now if the sum of the squares of the components of E is taken, this number can be used to indicate the error magnitude. If one row of E contributes the main error, this point of B probably should not be mapped to the model A.

Acknowledgments

I am indebted to Professor P. Elias for his valuable suggestions and criticisms in his capacity of Thesis Supervisor. His advice, as well as that of Professors Claude Shannon, Murray Eden, and Thomas Stockham, has been very helpful in encouraging and guiding this work. I also wish to thank Ivan Sutherland for his help and list structure subroutines; Leonard Hantman, who programed the major part of the three-dimensional display process; and the M.I.T. Lincoln Laboratory staff for their cooperation in the use of the TX-2 computer.

References

1. Somerville, D. M. Y., *Analytical Geometry of Three Dimensions,* Cambridge University Press, 1959.
2. Roberts, L. G., "Pattern Recognition with an Adaptive Network," *IRE Intern. Conv. Record,* Pt. 2, 66–70 (1960).
3. Selfridge, O. G., and U. Neisser, "Pattern Recognition by Machine," *Sci. Am., 203,* 60–68 (August 1960).
4. Hodes, L., *Machine Processing of Line Drawings,* Report 54G-0028[U,] Lincoln Laboratory, Massachusetts Institute of Technology (March 1961).
5. Julesz, B., "Toward the Automation of Binocular Depth Perception," *Proceedings of the I.F.I.P. Congress,* Munich (1962).
6. Roberts, L. G., "Picture Coding Using Pseudo-Random Noise," *IRE Trans., IT-8,* 145–154 (1962).
7. Gibson, J. J., *The Perception of the Visual World,* Houghton Mifflin Co., Boston, 1950.
8. Ittelson, W. H., "Size As a Cue to Distance," *Am. J. Psychol., 64,* 54–67 (1951).
9. Attneave, F., and M. D. Arnoult, "The Quantitative Study of Shape and Pattern Perception," *Psychol. Bull., 53,* 452 (1956).
10. Langdon, J., "The Perception of 3-D Solids," *Quart. J. Exptl. Psychol., 7,* 19–36 (1955).
11. Stevens, S. S., "The Psychophysiology of Vision," *Sensory Communication,* W. Rosenblith, Ed., M.I.T. Press, Cambridge, and John Wiley & Sons, Inc., New York, 1961, p. 13.
12. Sutherland, I. E., *Sketchpad: A Man-Machine Graphical Communication System,* Technical Report No. 296[U], Lincoln Laboratory, Massachusetts Institute of Technology, 30 January 1963.
13. Johnson, T., *Sketchpad III, 3-D, Graphical, Communication with a Digital Computer,* S.M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1963.