MASSACHUSETTS INSTITUTE OF TECHNOLOGY
A.I. LABORATORY

Artificial Intelligence
Memo No. 263

June 1972

A HETERARCHICAL PROGRAM FOR RECOGNITION OF POLYHEDRA

Yoshiaki Shirai

ABSTRACT

Recognition of polyhedra by a heterarchical program is presented. The program is based on the strategy of recognizing objects step by step, at each time making use of the previous results. At each stage, the most obvious and simple assumption is made and the assumption is tested. To find a line segment, a range of search is proposed. Once a line segment is found, more of the line is determined by tracking along it. Whenever a new fact is found, the program tries to reinterpret the scene taking the obtained information into consideration. Results of the experiment using an image dissector are satisfactory for scenes containing a few blocks and wedges. Some limitations of the present program and proposals for future developments are described.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0003.

1. INTRODUCTION

We do not know how to make a program to recognize objects visually as well as a human being. One of the shortcomings of many computer programs is, as Minsky has pointed out[1], their hierarchical structure. A human may recognize objects in the context of the environment. The environment may be recognized based on his a priori knowledge. The recognition procedure is, however, well programmed so that the simple obvious parts are recognized first and the recognition proceeds to the more complicated details based on the previous results.

The work in this paper studies an example of a heterarchical program to recognize polyhedra with an image dissector. Most previous works begin by trying to find feature points in a entire scene and make a complete line drawing. It is very difficult to get a complete line drawing without knowledge about a scene. If the line drawing has some errors, the recognition by a theory based on the assumption of the complete line drawing such as Guzman's[2] might make still more serious mistakes. Our work is an attempt to recognize objects step by step, at each time making use of the previous results.

We assume in this paper that the difference in brightness between objects and the background is large enough to detect the boundary approximately. At present, this program works for recognizing moderately complicated configurations of blocks and wedges. The limitations and proposals for future development are described later.

2. GENERAL STRATEGY

2.1 Priority Of Processing

For convenience, we define the edges of the objects in a scene as falling into 3 classes. A line formed at the boundary between the bodies and the outer background is a <u>contour line</u> of the bodies. In Fig.1, lines AB, BC, CD, DE, EF, FG, GH, HI, IJ, JK, KL, LM, MN, NO, AO, VW, WX, XY, YZ and ZV are contour lines. A <u>boundary line</u> is a line on the border of an object. Contour lines are boundary lines. In Fig.1, the boundary lines are the contour lines and lines on the boundary between two bodies, i.e. CP, PE, IQ, QR, and RH. An <u>internal line</u> occurs at the intersection of two planes of the same body. Lines JS, LS, QS, PT, NT, AT, BU, GU, DU and XV are internal lines.

The global strategy is shown in Fig.2. At first, the contour lines are extracted (because we assume a priori enough contrast between the objects and the background). If more than one contour is found, as in Fig.1, one contour for bodies B1, B2, B3 and another for body B4, then the boundary lines and internal lines are searched one by one for each contour. The global strategy in block 2 in Fig.2 is as follows.

A) Find boundary lines before finding internal lines because boundary lines often give good cues to guess internal lines. Note that to find boundary lines implies to find bodies.

B) In searching for lines, different situations require examination of larger or smaller areas. In our strategy, the smaller the area required to search lines, the higher priority we give to that search.

In Fig.1, for instance, to determine the existence of a extension of line IC, it is enough to search a small area whose center is on the extension of the line. To find line IQ, however, we should consider all possible directions of a line between IP and IJ. Thus the former search has priority over the latter

The priority to extract the most obvious information first is the following order.

(1) If two boundary lines make a concave point (such as point B in Fig.3), try to find the extension of them. If only one extension is found, track along this line. Most of such cases are like in Fig.3 (b) where one body hides the other. We can determine to which side of body this line belongs.

(2) If no extensions of two concave lines are found, try to find another line which starts from the concave point. If only one line is found, track along this line. Most of these cases are as in Fig.3 (c) where it is not clear locally to which body this line belongs. In Fig.3 (c), line BD belongs to the upper body, but this is not always true. That is the lines AB, BC, BD are not sufficient to decide the relation.

(3) If both extensions of two lines are found at a concave point, try to find a third one. If only one line is found, track along this line. This is the case as shown in Fig.3 (d) where the third line is the boundary line.

Whenever tracking terminates, an attempt is always made to connect the new line to the other lines that were already found. If more than one line segment is found in (1), (2) or (3), the tracking of those lines is put off hopefully to be clarified by the results of knowledge obtained in simpler cases. Fig.4 illustrates two extensions found at concave point P. The interpretation of the two lines is put off to treat simpler cases first. That is, one would continue examining the contour and lines AB and CD might be found next; then, by a circular search at points B (which is explained later), line BP would be found. At this stage it is easier to interpret lines AB and BP as boundary lines which separate two bodies. Then line DP might be found similarly and interpreted correctly.

(4) If an end of a boundary line is left unconnected as PQ in Fig.5, try to find the line starting from the end point (Q in this example) by circular search. If multiple lines are found, try to decide which line is the boundary. If a boundary line is determined, track along it. In Fig.5, the dotted lines are found by circular search and the arrows show the boundary lines to be tracked.

(5) If no line is found in the case (4) as stated above, extend the line (PQ in this example) by a certain length and test if the line is conected to other lines. If not, then apply circular search again as in (4). This is necessary because the termination point of the tracking is not always precise.

Note that this process can be repeated until successful (that is either the line is connected to other lines or line segments are found by circular search).

(6) If the boundary lines of a body are known, select the vertices of the boundary that might have internal lines starting at them. The selection of vertices is based on heuristics such as seleting upper right vertex rather than lower right vertex. At each vertex, try to find an internal line which is nearly parallel to other boundary lines. If one line is found, track along it. In Fig.1, for example, internal line JS is parallel to the boundary line KL or IQ, and QS is parallel to Rl or IJ. Line FU is parallel to ED and XV is parallel to XZ. Thus it is often useful to find internal lines parallel to boundary lines of the same body. Note that search for parallels has small area.

(7) If no line is found in (6), try to find one by circular search between adjacent boundary lines. When one line is found, track along it. In Fig.6, circular search between BA and BC is necessary to find the internal line BF.

(8) If two internal lines meet at a vertex, try to find another internal line starting at the vertex. This process is used in two cases. One is where no internal line was found in (7) because of little difference in brightness between adjacent faces. Suppose in Fig.1, that the internal line SJ was not found at vertex J, but that LS and QS were found. Then try to find an internal line starting at S toward J. If there is enough contrast near S, a line segment is found. The other case is where a body is partly hidden by other bodies. In Fig.6, the triangular prism is partly hidden. After EF and CE are found, EF is searched for. In both cases, the direction of the line is sometimes predictable and sometimes not. If it is predictable, then try that direction. If it is unpredictable or if the predicted direction failed, then apply circular search between the two internal lines. If one line is found, track along it.

(9) If an end of an internal line is not connected to any line, try to find lines starting from the end by circular search. If lines are found, track along them one by one.

(10) If no line is found in (9), extend the line by a certain length as in (5) and test if it is connected to other lines. if not connected, try circular search again as (9). This process can also be repeated until successful. Fig.7 illustrates this process. In Fig.7 (a), line MN′ is not connected to others at N′, thus step (9) is tried at N′ and fails. The line is extended to P_1 and (9) is again applied. This process is repeated until the line is connected to line KL at N. Fig.7 (b) shows that line HI is extended by this process to P_2 where a new line is found by circular search. Similarly line CG is extended to P_4. This process is useful so as to not miss a new body sitting on an obscure edge.

At each stage when an above step is finished, the obtained information is interpreted as shown in Fig.2 (block 3). For instance, if tracking along a line terminates, a test is made whether the line is an extension of other lines and/or the line is connected to other lines at a vertex. If a boundary line is connected to another boundary line, the body having the lines is split into two bodies and the properties of both lines and vertices are stored in an appropriate structure. In Fig.8, for example, line N'P' is obtained by tracking starting at point N. This line is interpreted as an extension of HN, and HN and N'P' are merged into one straight line using the equations of these two lines. Then, it is connected to CO and Fig.8 (b) is obtained. Before the line was connected to CO, there were two bodies B1 and B2 as in Fig.8 (a). Now body B2 is split into two bodies B2 and B3. We can interpret line NO as the boundary of B3 which hides a part of B2. The other properties of lines and vertices are obtained similarly at this stage.

2.2 Example

We illustrate the entire line-finding procedure with the aid of the example shown in Fig.9. At first, the contour lines AB, BC, CD, DE, EF, FG, GH, HI, IJ, JK and KA are obtained as shown in Fig.9 (a). Step (1) described in the previous section is tried for the concave points G and J. In this example, the position of G is not precise enough to find the extension of FG. On the other hand, a line segment is found as an extension of the line KJ. KJ is extended by tracking as far as L. Because there is no other point to which step (1) is applicable, step (2) is tried for point G. One line segment is found and extended till tracking terminates. Thus a line G'M' is obtained as in Fig.9 (b). This line is interpreted as an extension of FG and connected to JL. Then the position of point F, G, L are adjusted to as shown in Fig.9 (c). Now two bodies B1 and B2 are created by the boundary lines GL and JL. It is important to notice this, for it means that step (1) is again applicable (to point L) at this stage. Thus line FL is extended as far as M in Fig.9 (d) (Note that line MN' has not yet been found). LM is interpreted as an extension of FL but the end point M is not connected to any other lines. Thus vertices F, G, L and end point M are adjusted considering the new line LM. Here neither step (1), (2) nor (3) is applicable, so that (4) is now applied to M. Three lines are found by circular search as Fig.9 (d). MN' is determined as a boundary line and extended by tracking. When it terminates, the line is connected to boundary line BC at vertex N as in Fig.9 (e). Body B1 splits into body B1 And B2. It is known at this stage that B1 is hidden by B3 and B2 is hidden by partly B3 and partly by B1. Next, step (6) is applied to each body one by one at each time selecting the easiest body for proposing the internal lines(in this example, the order is B3, B1, B2 because B3 hides B1 which hides B2). Internal lines CO and MO are found and connected at vertex O, but no line segment is found using step (6) and (7) applied to vertex E (this stage is shown in Fig.9 (e)). Step (8) is applied to vertex O and a line segment toward E is found. This is extended by tracking as far as E' as in Fig.9 (f). Line OE' fails to be connected to any other lines which activates step (10). After a few trials, OE' is extended to connect to vertex E. Similarly, internal line AN is obtained for body B1 and line IP is obtained for B2. When every step has finished, three bodies are known together with the relationships between them.

3. ALGORITHMS

This section describes the details of the algorithms that are used in finding contours and in the steps stated in section 2. Some of them, such as tracking and circular search are used in more than one step. An algorithm used in more than one step may be slightly different in each step but its essential part is not changed. In the tracking algorithm, for example, some changes occur depending on whether tracking is used for boundary lines or internal lines.

3.1 Contour Finding

Fig.10 shows the outline of the procedure to find contour lines. The picture data obtained with an image dissector usually consists of a large number of points (say about 100,000) each of which represents light intensity level. To speed up the processing, one point for every 8 x 8 points is sampled. This compressed picture data consists of 1/64 the number of points in the original picture. To find the contour, this data is scanned till a contour point is found. The judgement of contour point is based upon the simple assumption that there is enough contrast between the background and objects. It is then checked whether or not the point is a noise point. If it is a real contour point, trace along the contour. Thus a set of contour points are found. Then, the picture data is again scanned until a new contour point is found. This process is repeated for all the picture data. When all the sets of contour points have been found, each set is separately analysed.

Suppose a certain set of contour points is to be analysed. We now return to the original high-resolution picture. We can guess approximately the position of the boundary point in the original picture data which corresponds to the first point found in the sampled picture. The precise boundary point is searched for near this point. A set of contour points is obtained by tracing from this point in the same way as in the sample picture. A polygon is formed after we connect contour points one by one. To classify the points of this 'curve' into segments, the 'curvature' of the polygon is used. This curvature in a digital picture is defined here for convenience as shown in Fig.11. Each cell in the figure represents a contour point. The curvature of a point P is defined to be the difference in angle between PR and PQ (α), where Q and R are a constant number of points away from P (6 points in this case). If we plot the curvature along the contour as shown in Fig.12, we can tell what part is near a vertex and what part lies in a straight line.

Note that curvature is not very sensitive to noise or digitization error. If we integrate the curvature in part of a straight line, the result is nearly zero despite the effect of noise. If we sum up the curvature of consecutive points whose absolute value is greater than some threshold, we can determine the existence of a vertex. That is if this sum of the curvature of such points exceeds a certain threshold, there is a vertex near those points. Thus every contour point is classified to be either in the straight part of a line or near a vertex. Using points which belong to the straight part of a line, the equation of the line is calculated. Then each vertex is decided as an intersection of two adjacent lines.

3.2 Line segment Detection

A line Segment is detected given its direction and starting point. This procedure is used in most of the steps stated in section 1. The procedure consists of two parts. One is to detect the possible feature points which are to be regarded as elements of the line. The other is to test whether or not obtained feature points make a line segment.

In detecting feature points, we should consider various types of edges. Herskovits and Binford classified the light intensity profiles across an edge into 3 types, namely step, roof and edge-effect, and proposed 3 types of boundary detectors. In this paper, a roof type detector is not considered because roof type edges can be detected by a step detector or an edge-effect detector. In addition, most roof type edges are accompanied by step or edge-effect types. We set up local Cartesian coordinates U-V such that U is the direction of the line segment to be detected. Let I(u,v) denote the light intensity at point (u,v), and define the contrast function $F_u(v)$ at (u,v) as

$$F_u(v) = \sum_{j=1}^{j_m} \sum_{i=-i_m}^{i_m} \left\{ I(u+i, v+j) - I(u+i, v-j) \right\}$$

Suppose we have an intensity profile as shown in Fig.13 (a), $F_u(v)$ at P in Fig.13 (b) is the difference of summed intensity between area A_2 and A_1. $F_u(v)$ for a typical step type profile (Fig.13 (a)) is shown in Fig.13 (c) in which the edge is detected as the peak. The typical profiles of $F_u(v)$ for other types are shown in Fig.14 where the edge is detected as the middle point between positive and negative peaks.

The basic procedure, therefore, is to detect the peak of $F_u(v)$ and its position. The necessary properties for a peak are as follows (see Fig.15).

(a) If $F_u(v)$ ranges from v_ℓ to v_r, there must exist the maximum of $F_u(v)$ at v_m other than v_ℓ or v_r.

(b) $F_u(v_m) > f_m$ where f_m is threshold

(c) There must exist a minimum of $F_u(v)$ at v_1 between v_ℓ and v_m and a minimum of $F_u(v)$ at v_2 between v_m and v_r such that

$F_u(v_m) - F_u(v_1) > f_d$

$F_u(v_m) - F_u(v_2) > f_d$ where f_d is a threshold

If such v_m is found, the left of the peak (v_3) and the right of the peak (v_4) are determined as the intersection of $F_u(v)$ with the line $F_u(v) = f_t$ as shown in Fig.15. The value of f_t depends on $F_u(v_m)$ and is represented as

$$f_t = c_1 F_u(v_m) + c_0$$

where c_1 and c_0 are constant and $0 < c_1 < 1$, $c_0 > 0$

The position of the peak v_0 is obtained as the middle of v_3 and v_4. If more than one peak is found between v_ℓ and v_r, the point v_0 which is nearest to the middle of v_ℓ and v_r is adopted. A negative peak is similarly detected. A feature point for an edge-effect or roof is obtained as the middle of the positive and negative peaks, if both are found, (although the threshold f_m is not the same as in the simple positive or negative peak detection). This method for the detection of peaks and positions is not appreciably affected by noise.

The other part of the line segment detection is a test of the co-linearity for the detected feature points. Suppose concave boundary lines L_0 and L_1 meet at P_0 and suppose the line segment extending L_0 is tested as shown in Fig. 16. Feature points are detected in a rectangular search area with given length and width whose direction is equal to that of $L_0 (= U)$, at an appropriate place where the detection of feature points is not affected by the edge corresponding to L_1. Feature points are detected along the direction v at the center points $P_1, P_2, \ldots, P_m$ sequentially. If positive peaks are found at $M_1, M_2, \ldots, M_n$ as shown in the figure, the linearity of the points are tested as follows.

(a) The number of the feature points must exceed a threshold number n .

(b) The deviation δ^2 of the points in line fitting with the least square method should be less than a threshold δ_t^2.

(c) let U' denote the direction of line segment obtained by line fitting,

$|U' - U| < U_d$

where $|U' - U|$ denotes the diffrence in directions U' and U

Similar tests are made for the different types of feature points. If more than one type of line segment is found, the selection depends on the following criteria.

(a) If an edge-effect type is found, then it is selected.

(b) For the line segment with δ^2 and U', let the criterion function C be

$C = \delta^2 + w_a |U' - U|$ where w is a constant

The line segment selected is the one wiht smaller C.

3.3 Circular Search

Circular search is used to search for lines starting at a given point. The direction of the lines to be searched for is not known. The range of directions in circular search depends upon the particular case. Suppose two known lines L_1 and L_2 meet at P as in Fig.17 (a) and suppose we wish to search for lines lying between them. The search range α is between two lines L_1' and L_2' whose directions are slightly inside of L_1 and L_2 respectively. If lines starting at point P of line L_0 are searched for as in Fig.17 (b), L_1' and L_2' are similarly set inside of L_0. The center point P of the circular search is not always precisely determined, especially when tracking along a line has terminated at point P as shown in Fig.17 (b). Therefore circular search should not be too sensitive to the position of the center point.

It might be natural to try to detect feature points, as defined in section 3.2, based upon $F_u(v)$ along arcs around the center. The difficulty with this search is the classification of feature points into line segments if there is more than one as shown in Fig.18. To avoid this difficulty, a simple algorithm is used in this paper. Its basic method is to apply line segment detection successively in various directions. This is illustrated in Fig.19, where successive line segment detections toward u_1, u_2 and u_3 are applied. The step of direction change and search area (A_1, A_2 and A_3 in the figure) are determined so that line segments of any direction near the center point can be found. Thus successive circular search along a line as shown in Fig.7 can find lines starting at points between two adjacent center points (e.g. line L in Fig.20 starting between P_1 and P_2). The algorithm for line segment detection is the same as described in 3.2 except with respect to thresholds and search area. Because the search areas for different directions overlap each other, the same line segment may be found in different searches. Each time a line segment is found by line segment detection, a check should be made whether or not it is the same as the one obtained by the previous detection.

If the center point of circular search has not been determined precisely, it is not always possible to find all the lines starting at the given point. In Fig.21, for example, line L_2 might be missed in circular search at P_0. To avoid this inconvenience, when line segments are found (such as L_1 and L_3 in the figure), a new center point P_1 is calculated based on the known line (L_0) and the obtained line segments (L_1 and L_3). Then circular search is applied again at P_1.

3.4 Tracking

Tracking is used when a line segment is given, to track along it until it terminates. The requirements for a tracking procedure are 1) the line should not be lost due to the effect of other lines or noise, and 2) the procedure should terminate as precisely as possible at the end of the line. These requirements are contradictory in that the termination condition should be strict to satisfy the second requirement which makes it difficult to satisfy the first. The following algorithm is a compromise between these requirements.

The basic procedure is to predict the location of a feature point and to search for it near the point using line segment detection. The result of the search is classified into the following 4 cases.

(a) there is no feature point.

(b) a feature point is on the line.

(c) A feature point is not on the line.

(d) It is not clear whether or not a feature point is on the line.

In case (a), the detection of a feature point is similar to line segment detection except that the type of edge is already known so that the thresholds stated in 3.2 can be adjusted based on the average peak of $F_u(v)$. The decision between cases (b), (c) and (d) is made using the distance d between the point and the line. That is

If $d \leq d_1$ then case (b)

If $d > d_2$ then case (c)

If $d_1 < d \leq d_2$ then case (d)

The threshold d_1 changes depending on the state of tracking. The state of tracking is represented by two integers m_1 and m_2 which are set initially to 0. The value of m and m are changed for each case (a), (b), (c) and (d) as follows.

(a) $m_1 = m_1 + 1$

(b) If $m_1 > m_2$, $m_1 = m_1 - 1$, (where m is a constant)

Otherwise, $m_1 = 0$, $m_2 = 0$, and classify those feature points into (b) which have been clasified into case (d) in the previous steps of tracking. Adjust the equation of the line with these points and the present feature point

(c) If $d \leq d_3$ and $m_1 > m_a$, $m_1 = m_1 - 1$
(where d is a constant)
Otherwise no change

(d) $m_2 = m_2 + 1$, and if $m_1 > m_a$, $m_1 = m_1 - 1$

The threshold d_1 is represented as

$d_1 = d_0 + w_m m_2$ (where d_0 and w_m are constants)

This procedure is repeated and tracking proceeds step by step extending the line until the termination condition is satisfied. The termination condition of tracking is either

$m_1 > m_n$ or $m_1 + m_2 > m_t$ (where m_n and m_t are constants)

The terminal point is defined as the last point classified into case (b). Fig.22 illustrates how this algorith works. In Fig.22 (a), two lines cross at P_0. Tracking might finish at some point beyond P_0 (P_m in the figure) which satisfies the termination condition. The terminal point of tracking is, however determined more precisely near P_0 (P_1 or P_2). In Fig.22 (b), P_1, P_2, P_3, P_4 are classified into case (d) increasing the value of m_2 which classifys P_5 into case (b). Then the line is adjusted with these points which are now classified into case (b) and tracking proceeds.

Fig.22 (c) and (d) illustrate that even if a part of the intensity profile is disturbed by noise or other lines, tracking does not terminate there. In Fig.22 (d), however, if the light intensity of the right side of L_0 changes across L_1, the type of feature points might change across L_1. Thus feature points $P_3, P_4, \ldots$ might not be obtained and tracking might terminate at P_1. When tracking terminates, the line segment detection is applied at the extension of the line to see if another type of line segment is found. If found, we adjust the line equation and tracking proceeds. If not found, tracking finally terminates at point P_1 and the position of I_1 is adjusted with the line equation. The above procedure often extends the line across other lines when it terminates temporarily at their crossing as in Fig.9 (b) where tracking along G'M' crosses many vertical lines.

4. EXPERIMENTAL RESULTS AND COMMENTS

To test the program, experiments are made with cubes and wedges having relatively uniform white surfaces placed on a black background. The image dissector camera , used as an input device, dissects the scene onto a 20000 x 20000 (octal) grid. In this experiment, one point for every 8 x 8 block of grid elements is sampled. Thus, the scene is represented by 1024 x 1024 grid points. Objects occupy only a part of the scene. In the typical scene, the rectangular area which includes the objects of interest may consist of about 400 x 400 points. This area is divided into blocks each of which is made of 64 x 64 points and stored in disk memory. When a light intensity at some point is required, a block containing the point and adjacent blocks are stored in core memory. The core memory is accessed for the input of the light intensity until a point outside of those blocks is refererenced.

Video input is at first converted into a 10 bit digital number which is an inverse linear measure of the light intensity. It is again converted into 10 bit logarithmic measure. Some intensity level resolution is lost in the logarithmic conversion. In this experiment the light intensity is represented by a little less than 100 levels. The input data for a clear bright edge in the dark background is blurred due to some limitations (mostly defocusing). If the intensity change is a step function, there is a transient area in the input data about 10 points wide. Thus the resolution of the picture is regarded as 10 points. The parameters used in line segment detection and tracking are based upon this resolution. Features of the picture involving resolution of less thean 10 points are not usually found.

Some results are shown in Fig.23. The difficulty or processing time of the recognition depends not only on the complexity of the object but also on the information extracted at each stage. In Fig.23 (c), for example, boundary lines SJ, KS and QS are easily proposed as the extension of contour lines. On the other hand, it is not easy to find boundary lines KM or LM in Fig.23 (c). That is, after DK and HL are found, circular search is necessary at K and L respectively. Circular search is less reliable in finding a line segment, and more time consuming. Once the boundary lines are determined, all the internal lines are proposed in both cases. But tracking along VW in Fig.23 (c) and EN in Fig.23 (c) terminates in the middle. Then step (10) stated in section 1.1 is applied. This is the most time consuming process (about 10 times more than the simple tracking process).

Some examples of the result of a hierarchical program are shown in Fig.24. Hierarchical programs may look at the whole scene homogeniously and pick up feature points. Lines are found with those feature points obtained in the previous stage. It is very difficult to determine a priori the various thresholds for detection of feature points, line fitting and connection of lines. In this heterarchical program, it is possible to adjust various thresholds with the context of the information obtained previously. Furthermore the algorithm itself can be modified case by case. (For instance, tracking algorithm is changed depending on whether the line is a boundary or internal.) The results of experiments with moderately complex scenes are mostly satisfactory. Because of the many checks for consistency of lines and vertices, the program has small probability of finding false lines.

However, there are some limitations of this program at present. One of them is that bodies may be missed in some cases. A simple example is shown in Fig.25. The boundary lines AB and BC in Fig.25 (a) are not proposed though the other contour lines and internal lines are found, because the resulting regions are so 'neat' that no conceive vertices activate step (1). In such a case when bodies are neatly stacked, it is necessary to search for boundary lines which start from some points on the boundary line. In Fig.25 (b) body B2 is not found. To find a body that is included in a face of another body, it is necessary to search for line segments inside the region. Though these two kind of search (search along the boundary line and search in the region) are required to find all the bodies in the scenes as shown in Fig. 25, they are still more effective than the exhaustive search in the entire scene. Besides, it is simpler to interpret the scene when a line is found by those searches. This procedure, however, is left to future work.

The other limitation of the present program is, as stated in the introduction, that it is not always applicable to concave objects. Fig.26 (a) shows a simple example. Line BD is found as an extension of line CB. If all the bodies are convex, line ED is interpreted as the boundary line as shown in Fig.26 (b). This does not hold for concave bodies. In this program, line ED is regarded as a boundary line, and then line DE can be found by circular search at E. At this stage, however

DE should be interpreted as an internal line of the same body instead of the boundary line which seperates the body into two. If DE is interpreted correctly, then line BD can be determined as an internal line. This procedure should also be implemented in the present program.

CONCLUSION

A heterarchical program to recognize polyhedra is presented. The program is based upon the strategy of recognizing objects step by step, at each time making use of the previous results. The order of the lines to be detected is 1) contour lines (boundary of bodies and the background), 2) boundary lines which are the boundary between two bodies, 3) internal lines (intersection of two faces of the same body. Among boundary lines or among internal lines, the 'most plausible lines' are proposed at each stage and an attempt is made to find the line. To find a line, the range where a line segment may exist is proposed and it is detected in a suitable way for the proposed range. If a proper line segment is found, the end of the line is determined by tracking along the line. When the line is determined, the program tries to understand the scene taking this line into consideration. Because lines are mostly proposed instead of found by exhaustive search in the scene, the program is relatively effective. Results of the experiment using an image dissector are satisfactory for scenes including a few blocks and wedges. Although the present program has limitations, some of them may be overcome by developments proposed here for future work.

REFERENCES

1) M. Minsky and S. Papert, 'Proposal to ARPA for research on artificial intelligence at M.I.T., 1970-1971', MIT Project MAC Artificial Intelligence Memo No. 185, 1970.

2) A. Guzman, 'Computer Recognition Of Three-Dimensional Objects In A Visual Scene', MIT MAC-TR-59 (Thesis), 1968.

3) A. Herskovits and T. Binford, 'On Boundary Detection', MIT Project MAC Artificial Intelligence Memo No. 183, 1970.

4) B. Horn, 'The Image Dissector 'eye' ', MIT Project MAC Artificial Intelligence Memo No. 178, 1969.

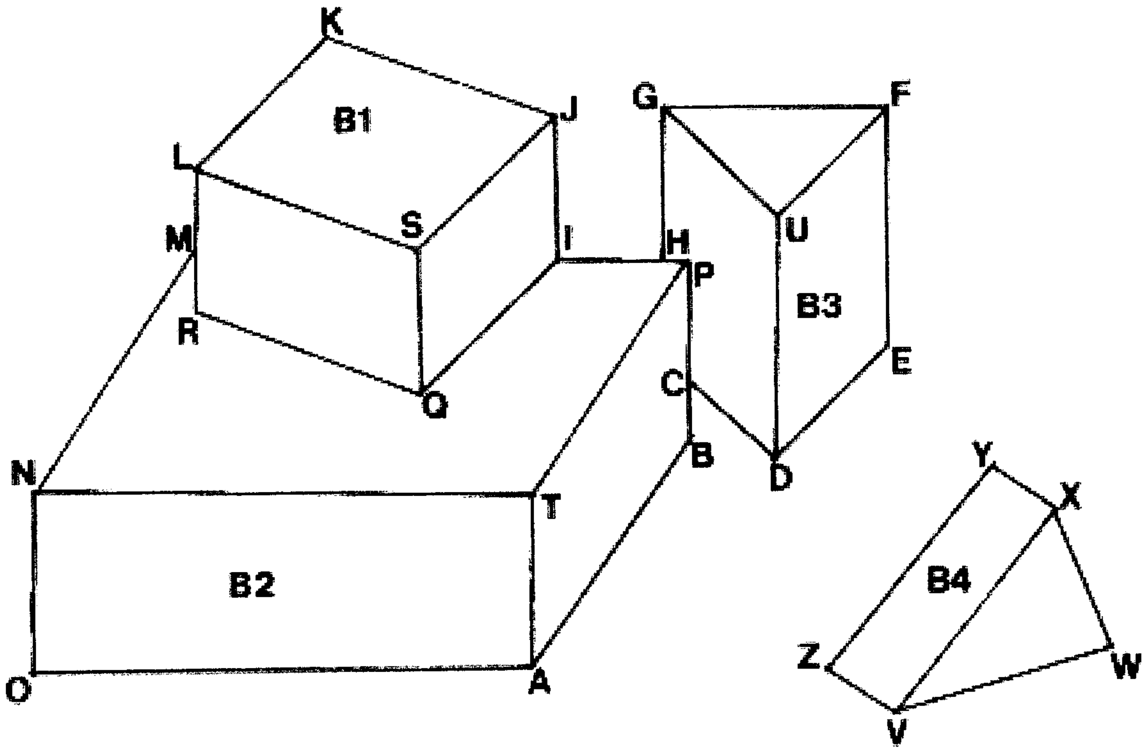


Fig. 1. Example of Scene

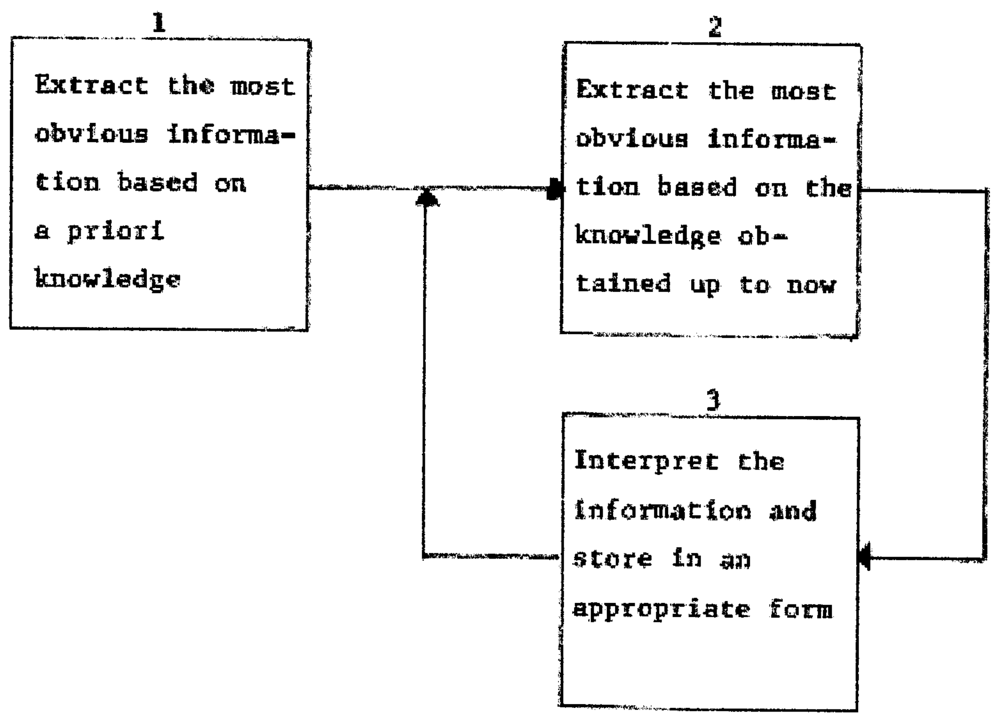


Fig. 2. Schematic diagram of the strategy

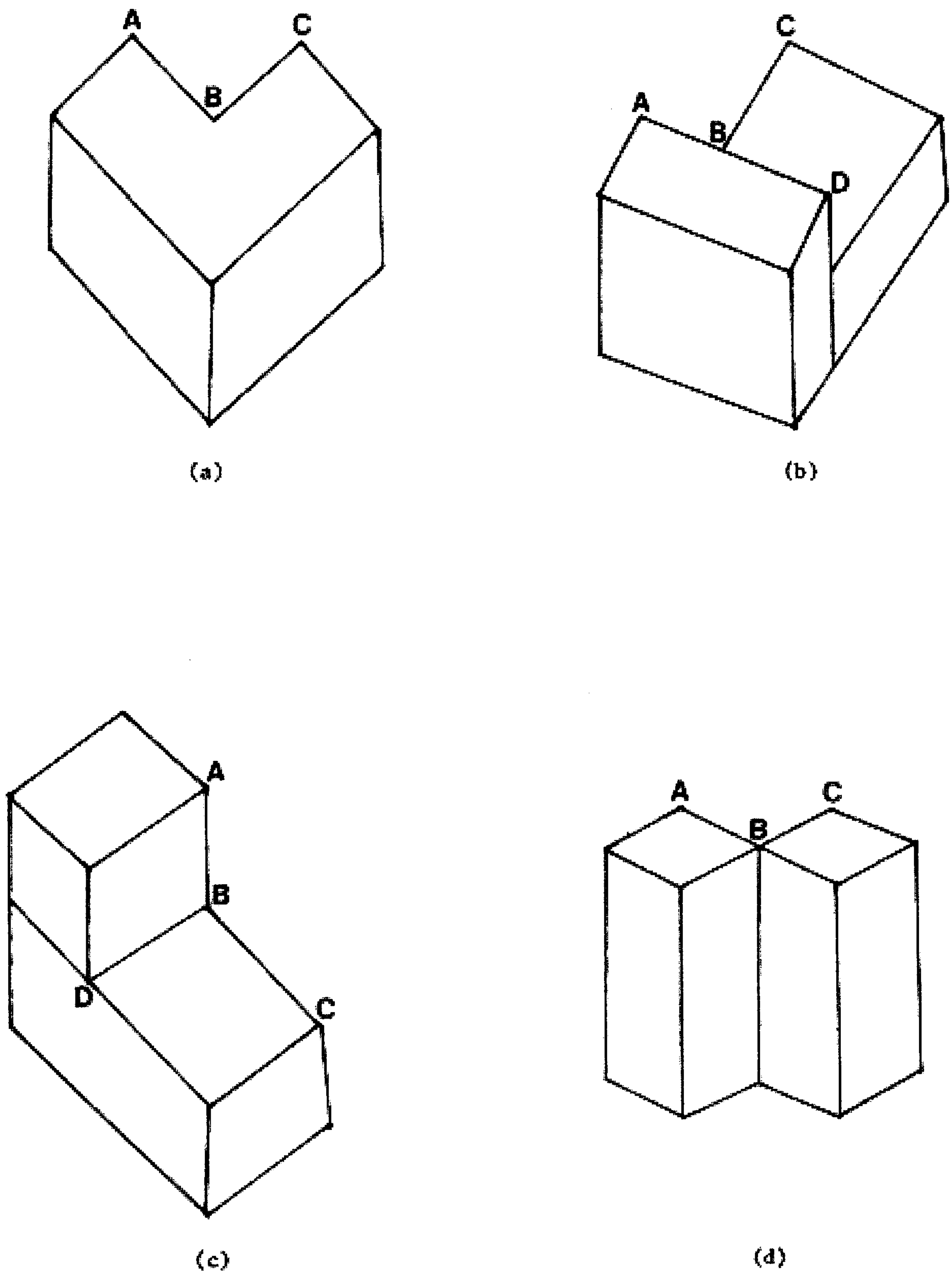


Fig. 3. Examples of concave boundary lines

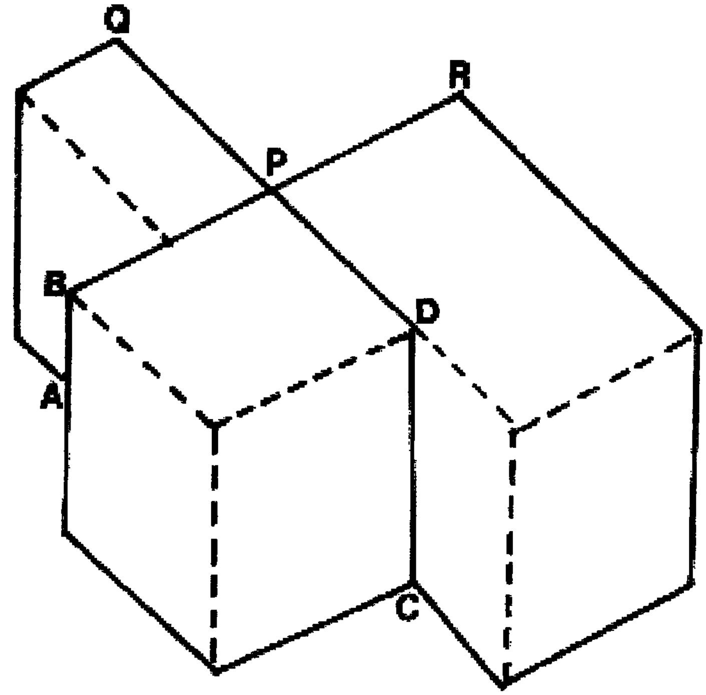


Fig. 4. Illustrates two extension lines at concave point P.

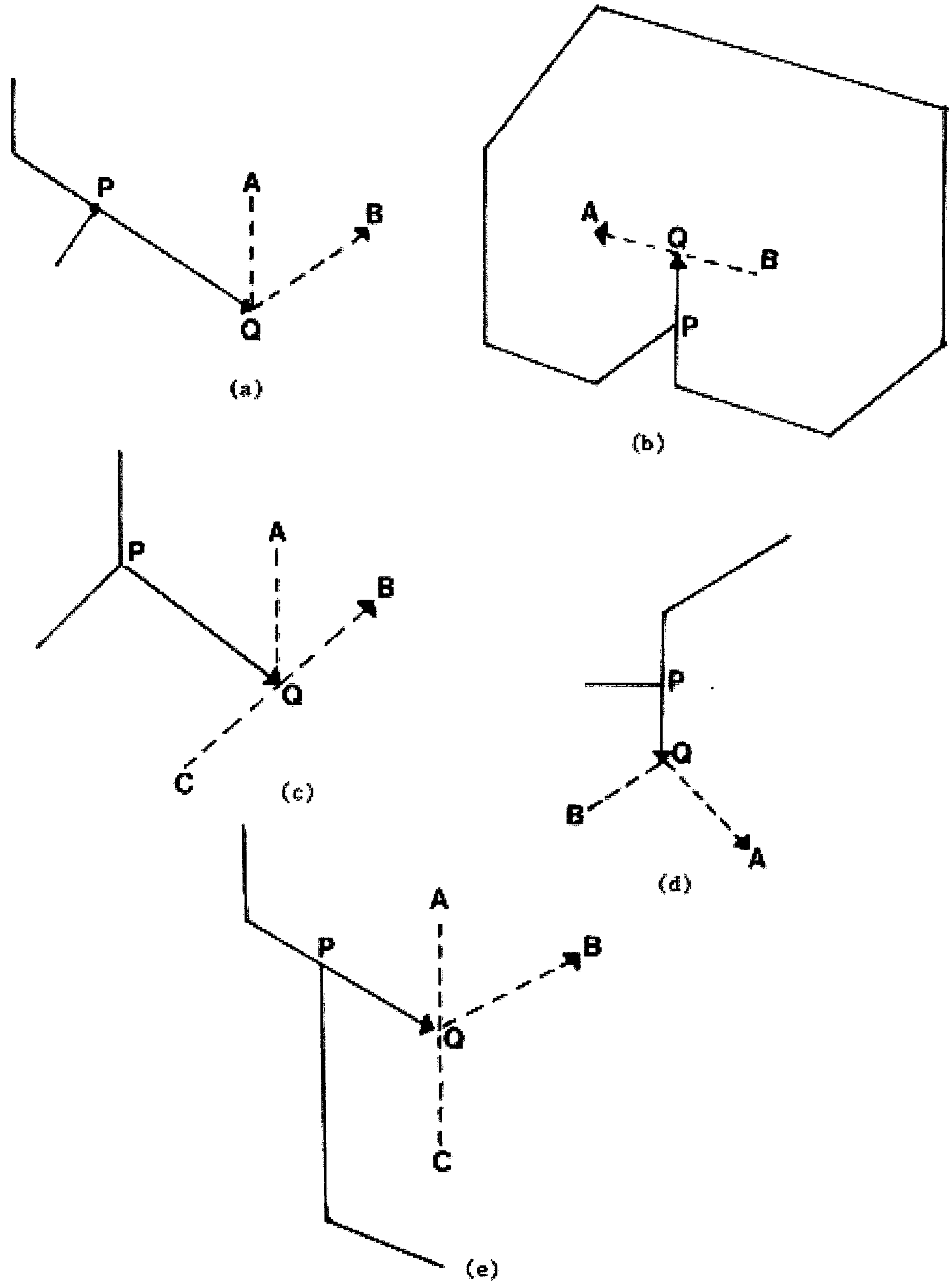


Fig. 5. Examples of line configuration found by circular search.

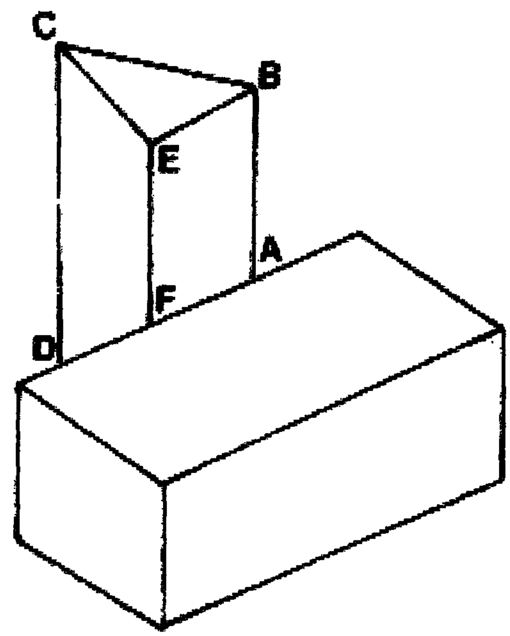
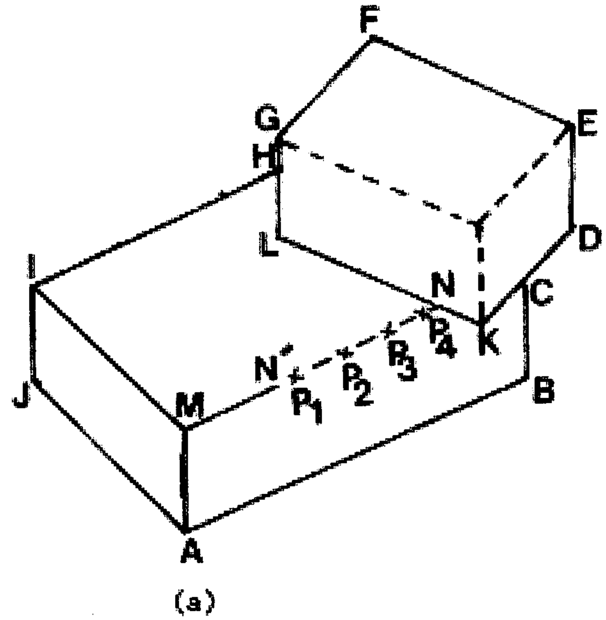
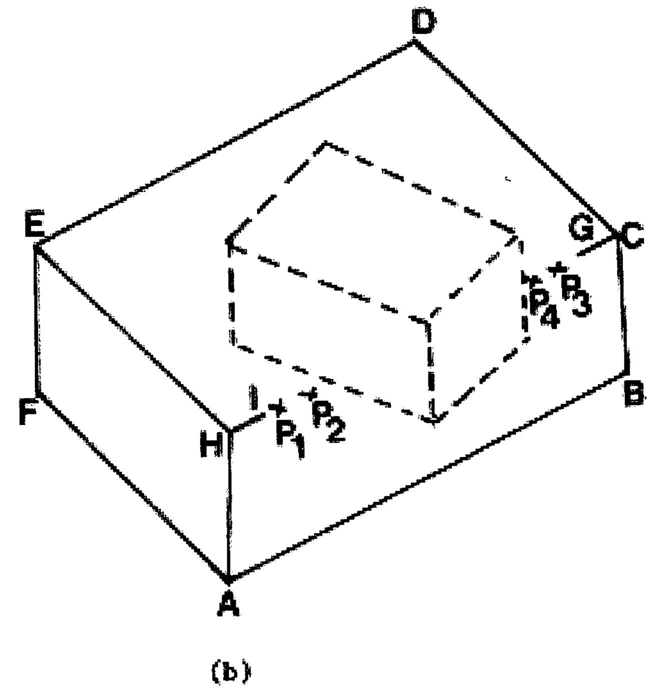


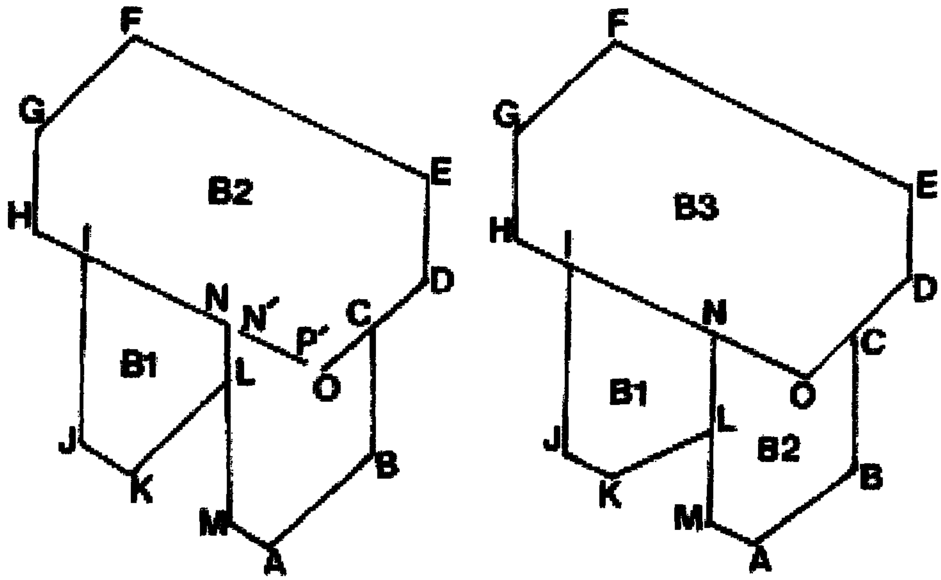
Fig. 6. Example of circular search for internal lines.





(dotted lines are not yet found in this stage)

Fig. 7. Examples of line verifying by circular search.



(a) (b)

Fig. 8. Illustrates the process after tracking

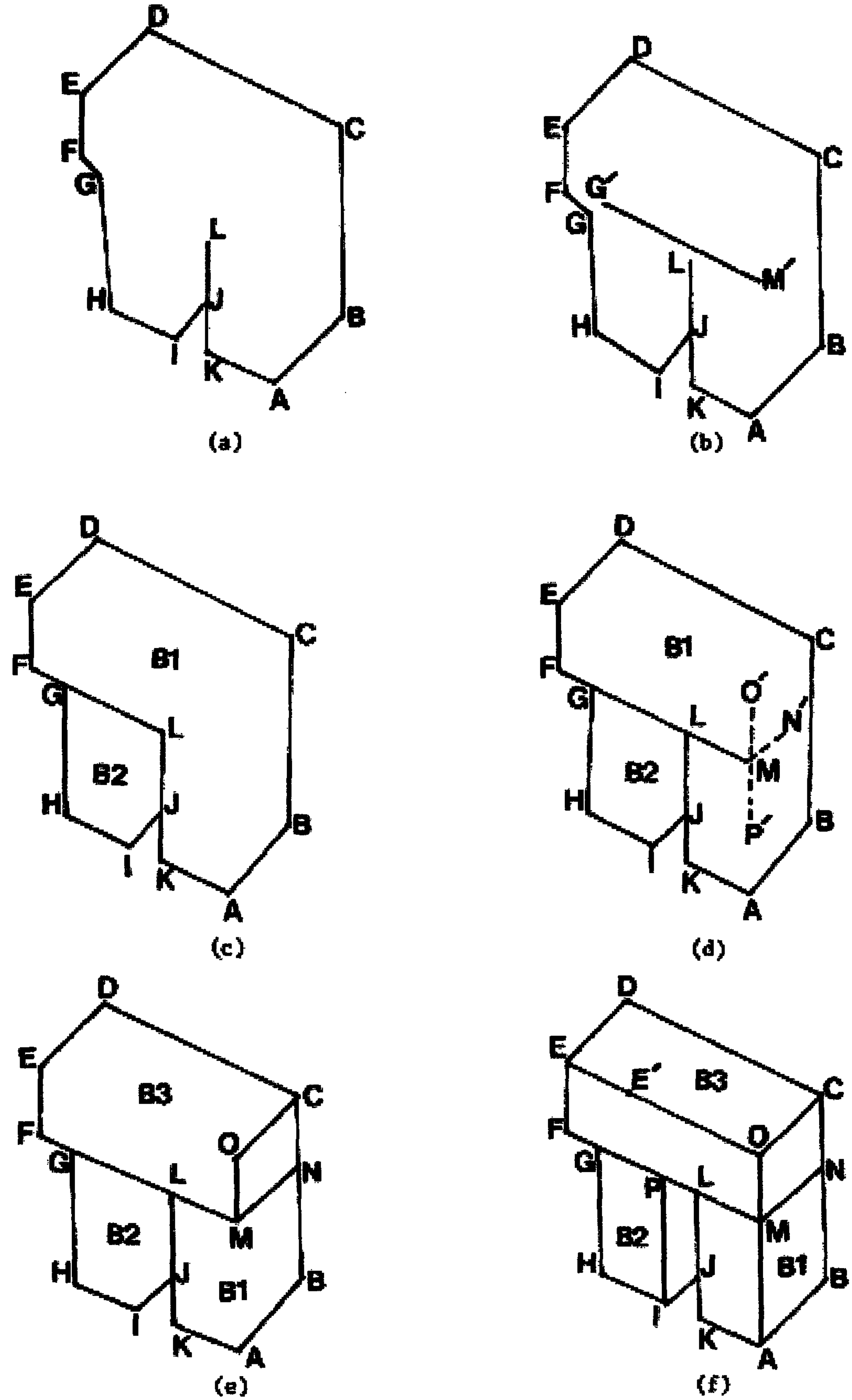


Fig. 9. Illustrates the procedure to find lines.

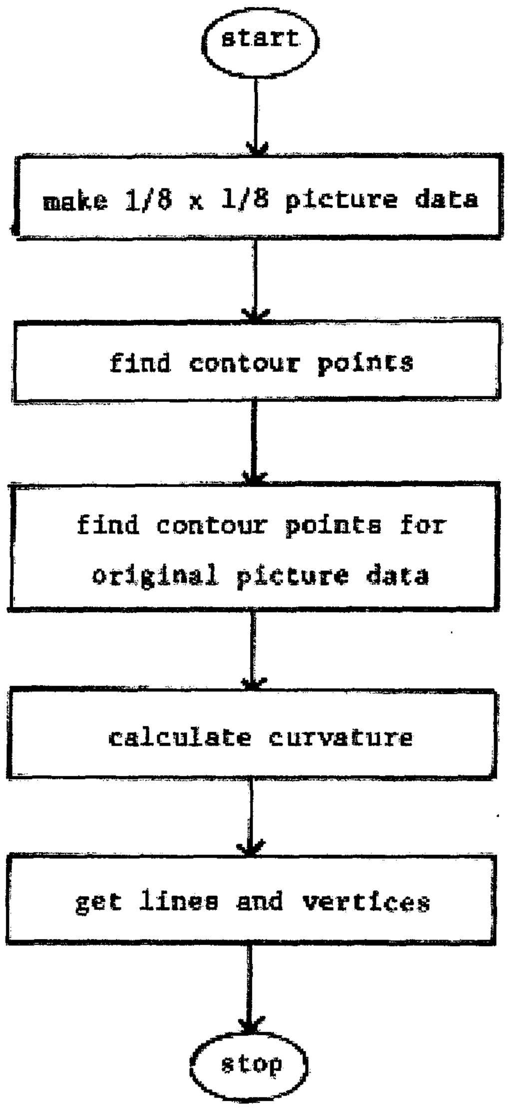


Fig. 10. Flow chart of contour finding.

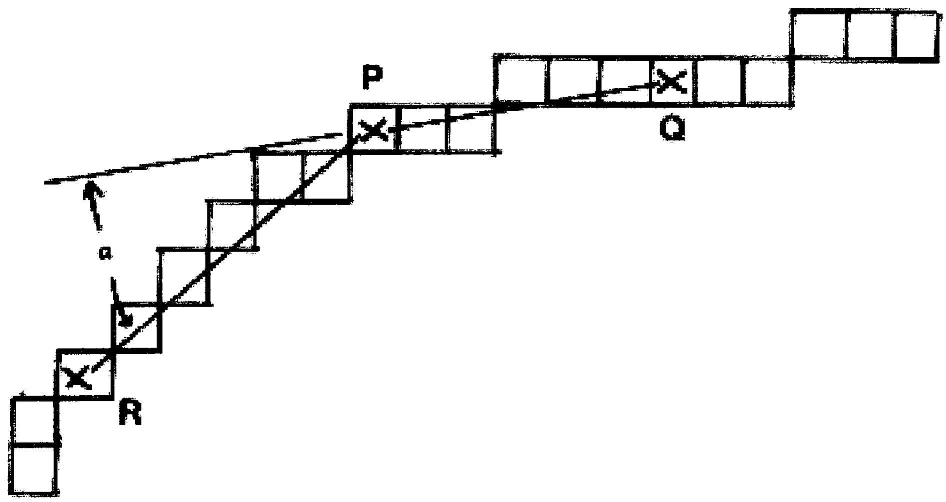


Fig. 11. Illustrates the definition of curvature

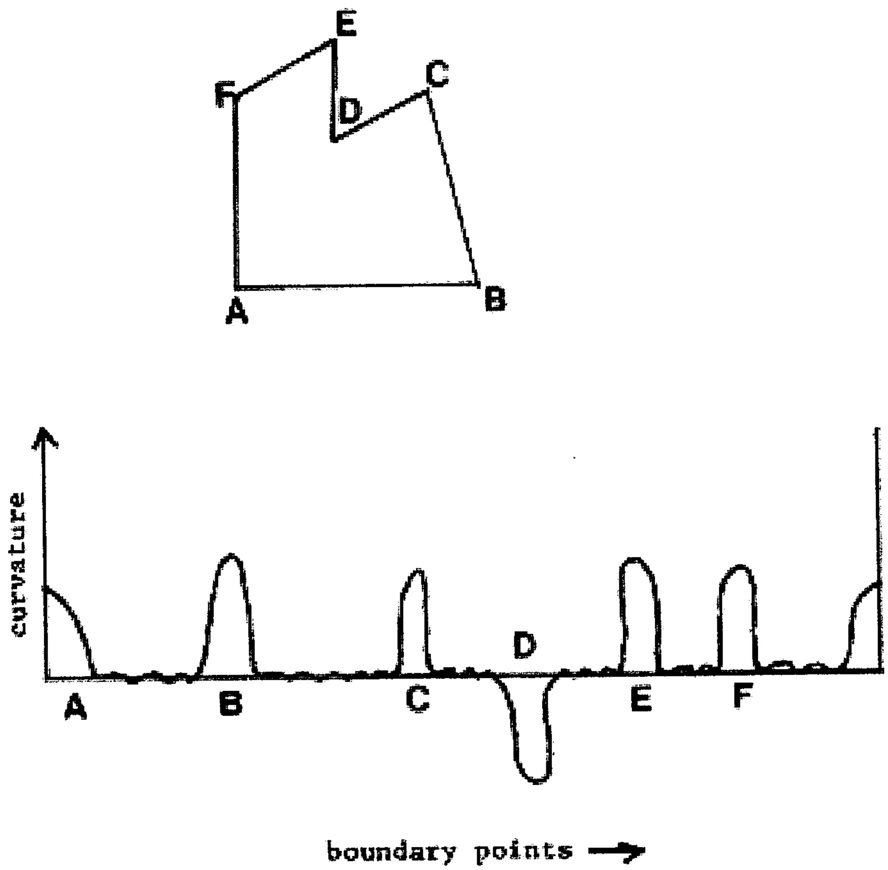
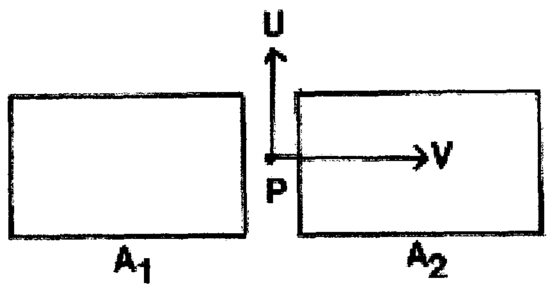
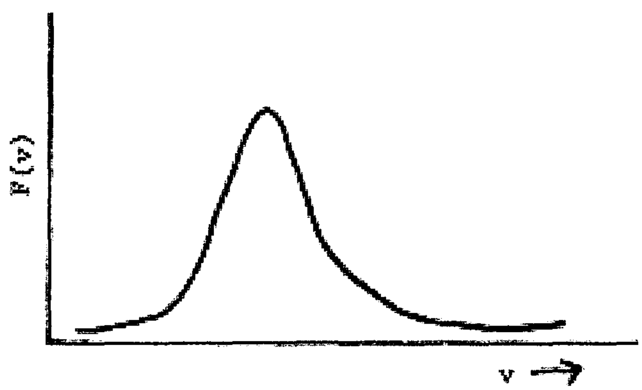


Fig. 12. Example of curvature.

(a) light intensity level profile



(b) Areas to compute E_u at P



(c) $F_u(v)$ for (a)

Fig. 13. Example of $F_u(v)$.

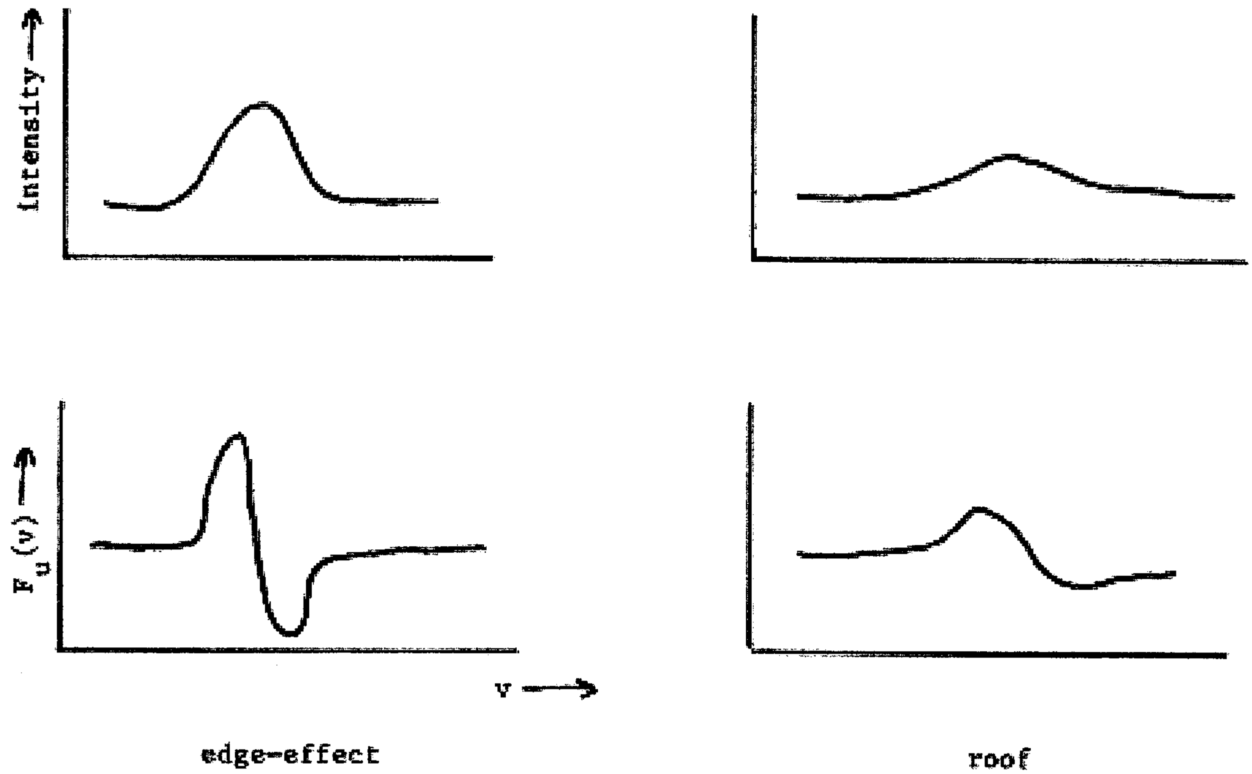


Fig. 14. Typical profile of $F_u(v)$.

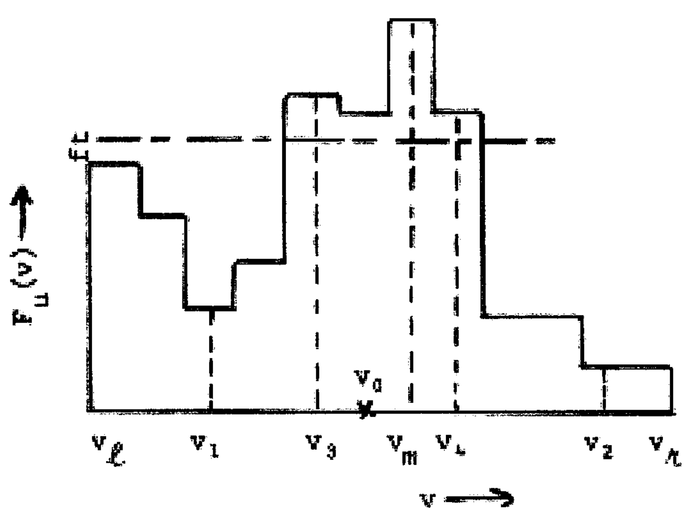


Fig. 15. Peak detection.

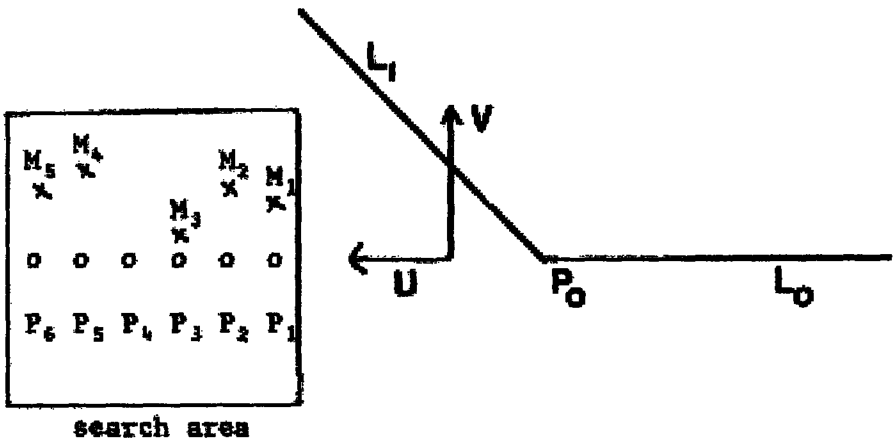


Fig. 16. Detection of feature points.

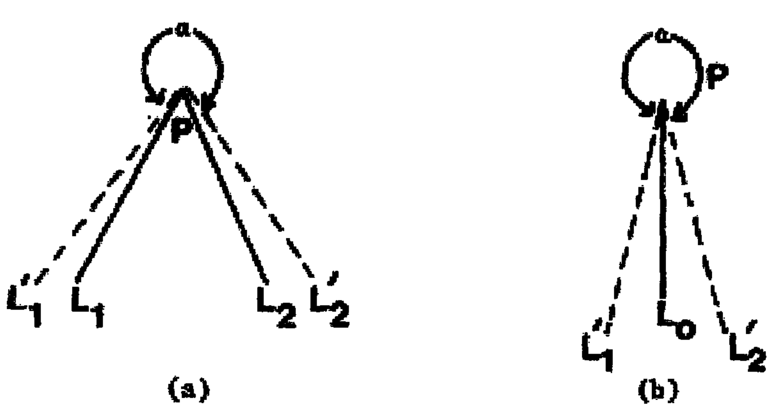


Fig. 17. Range of circular search.

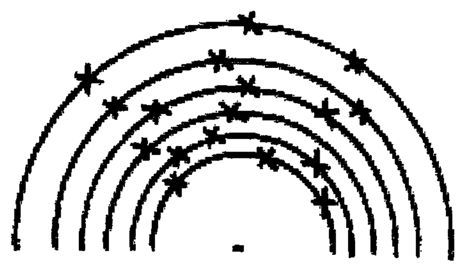


Fig. 18. Illustrates difficulty in classification of feature points.

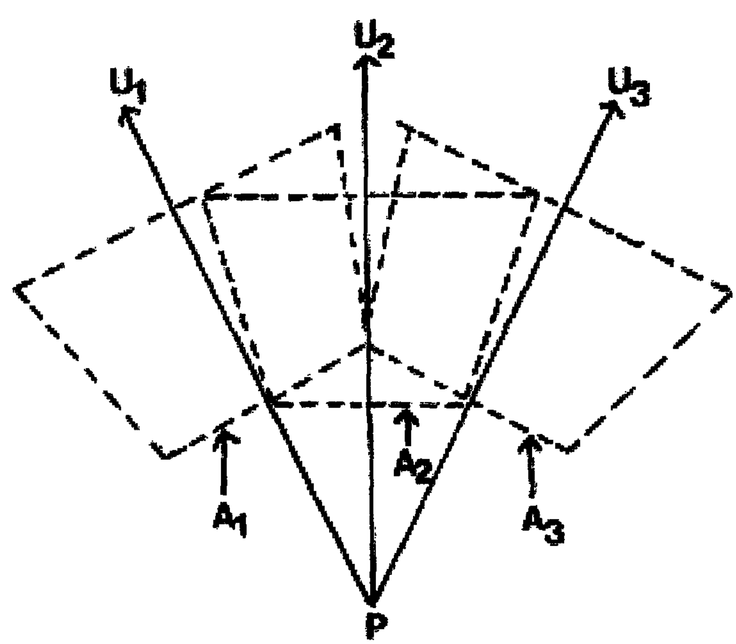


Fig. 19. Successive line detection in circular search.

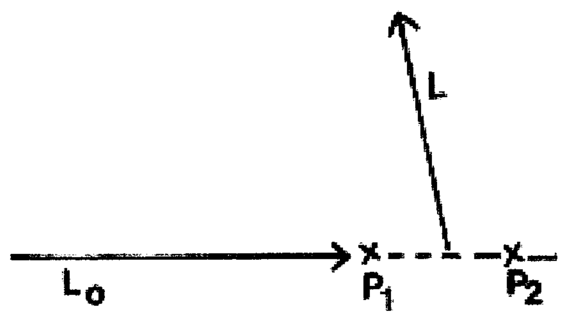


Fig. 20. Example of successive circular search.

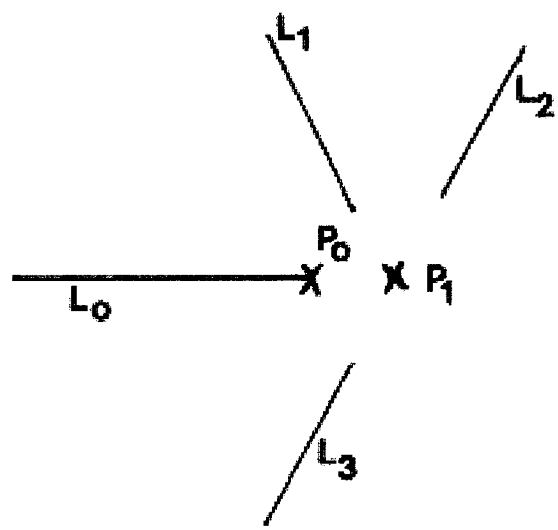


Fig. 21. Repeated circular search.

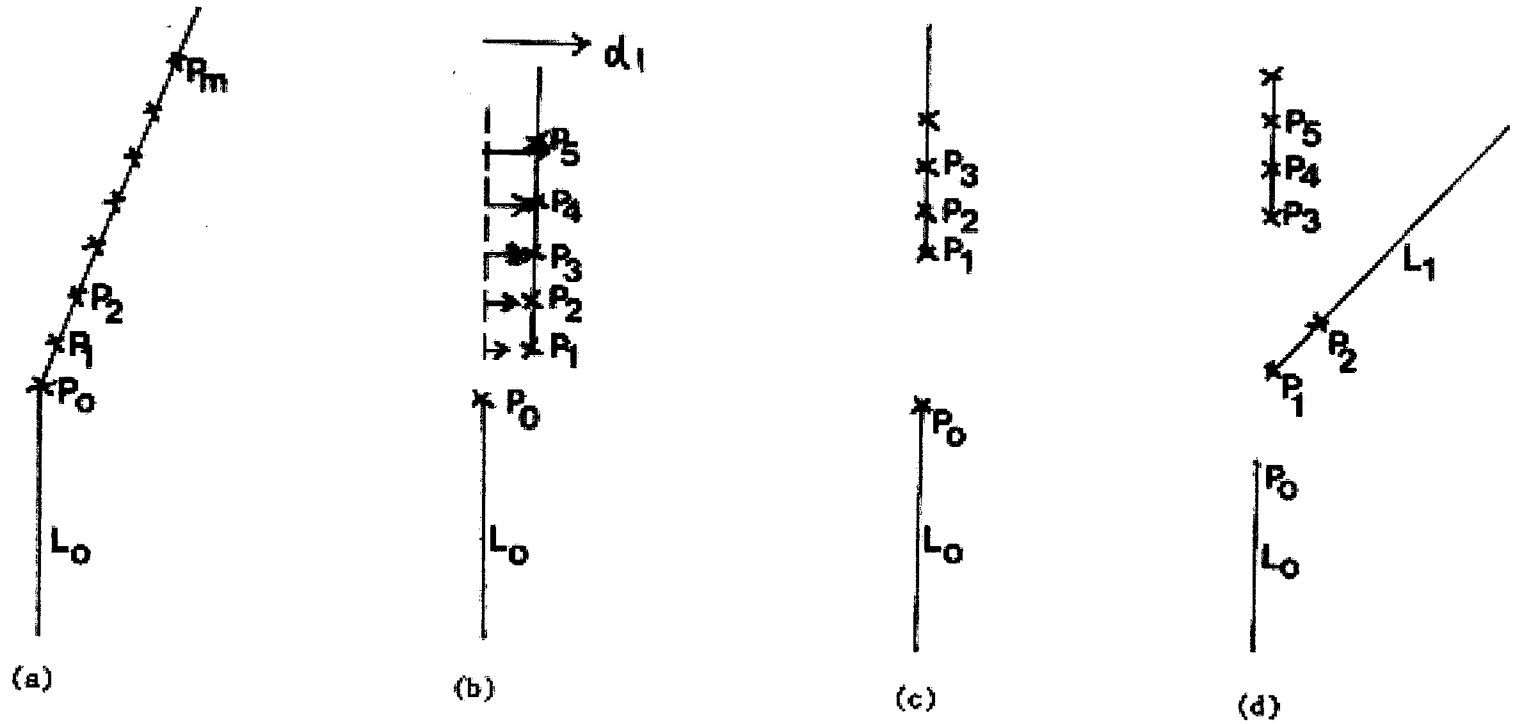
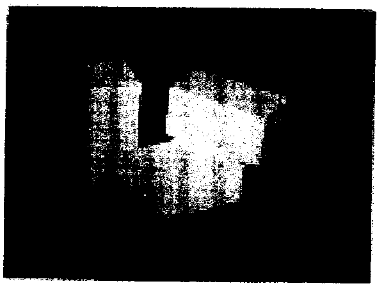
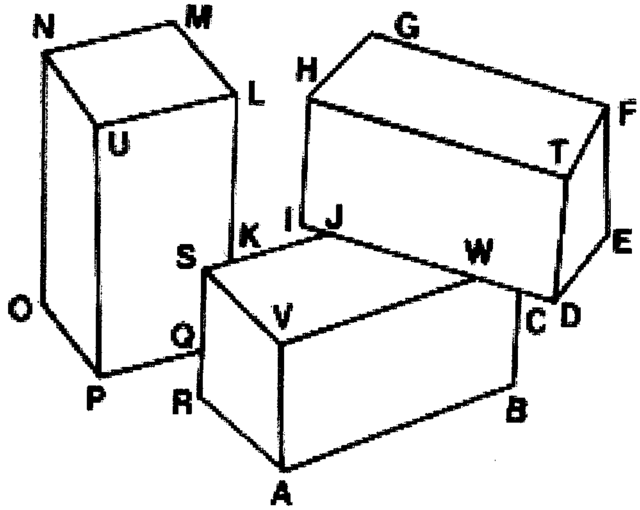


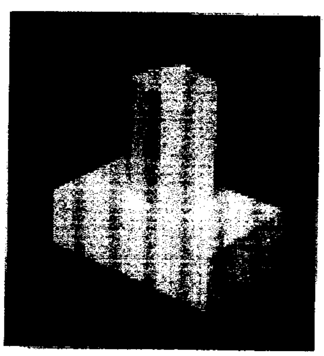
Fig. 22. Examples of tracking.



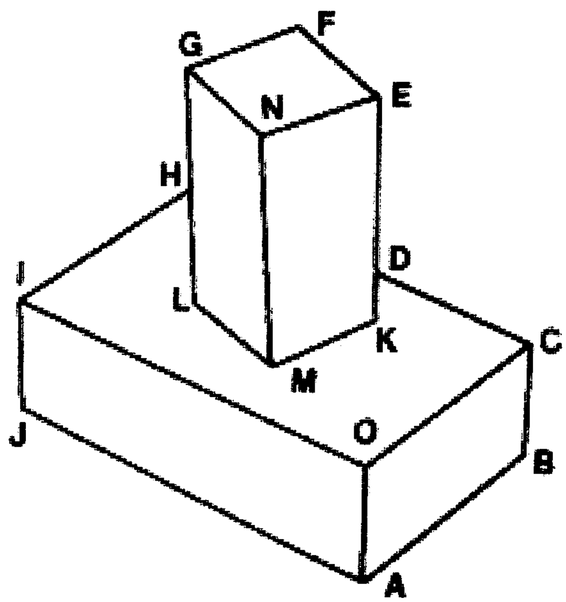
(a_1)



(c_1)



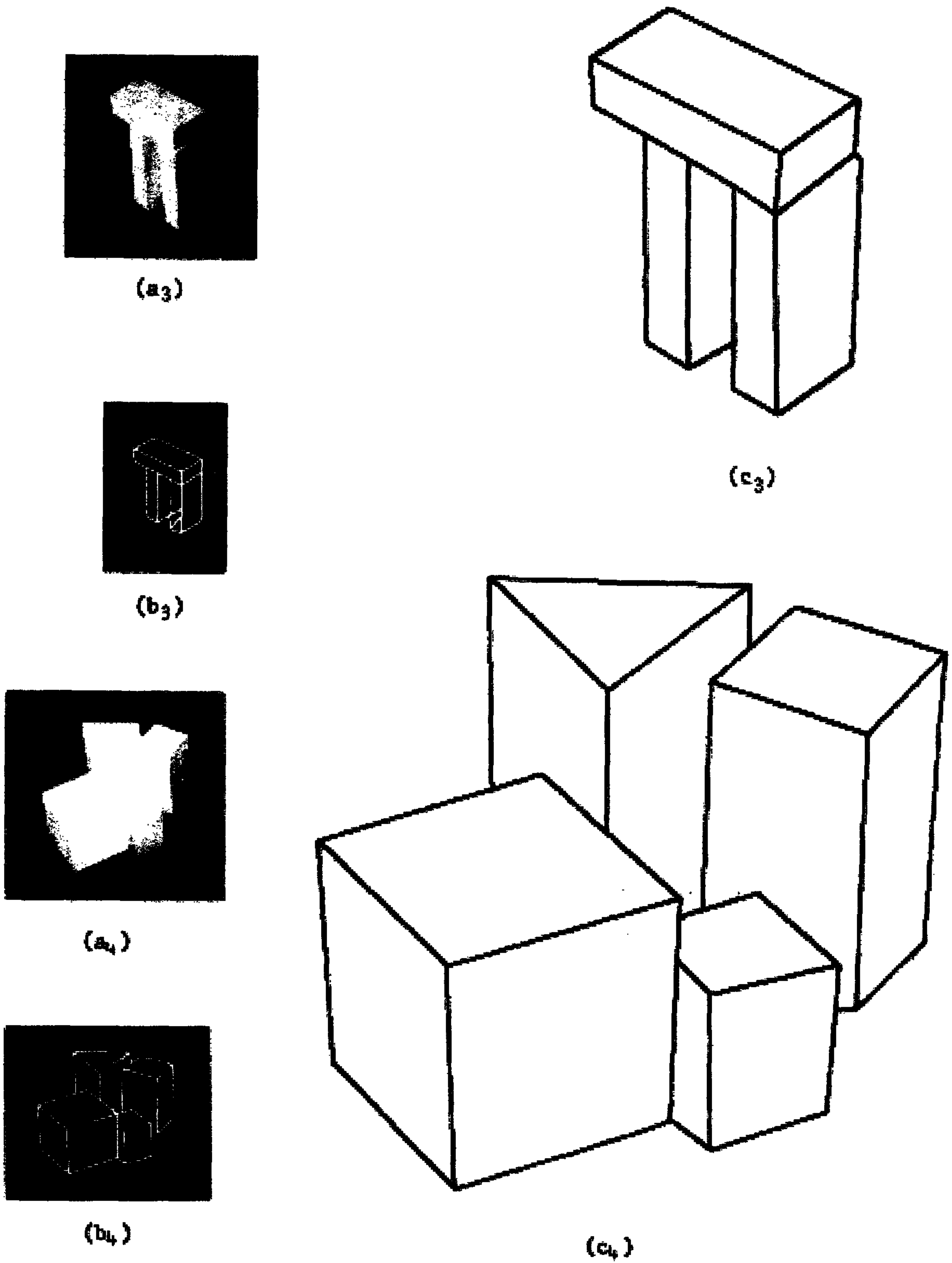
(a_2)



(c_2)

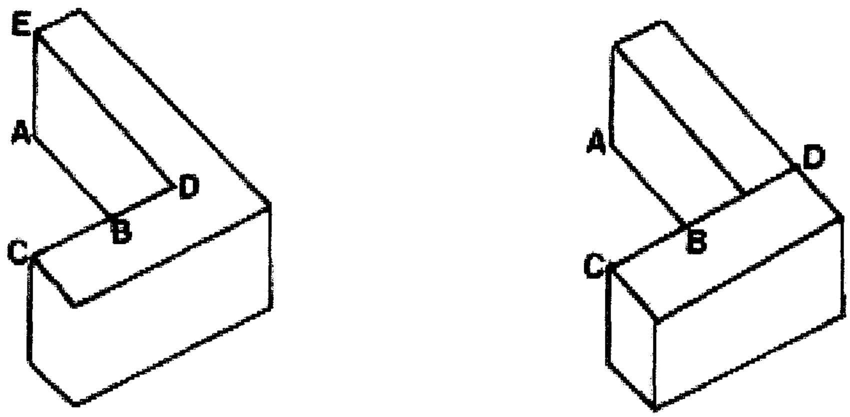
(a_i): Objects, (c_i): Result

Fig. 23. Examples of Experiment.



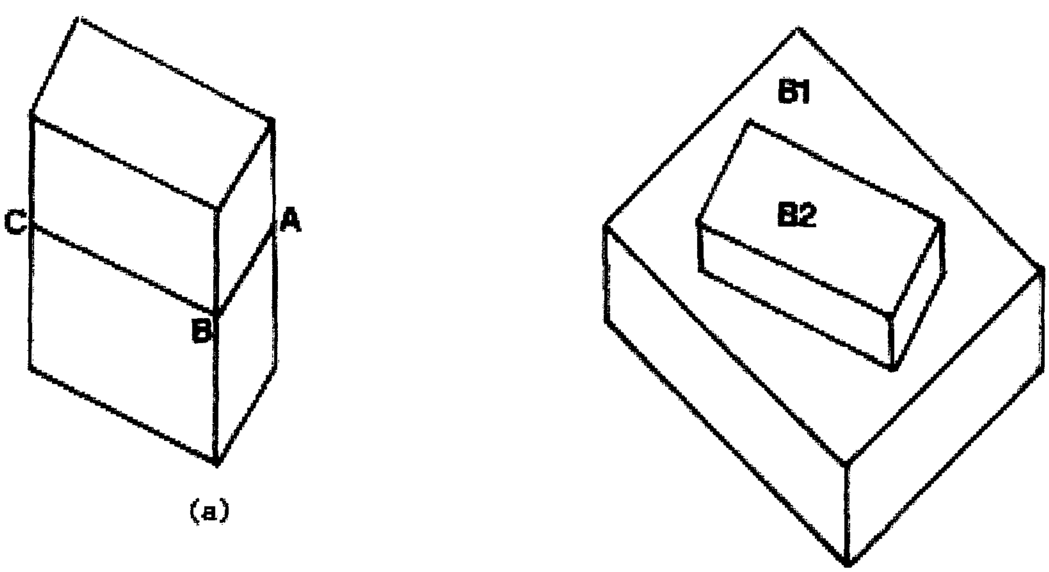
(a_i): Objects, (b_i): Result of Hierarchical Program
(c_i): Result of this program

Fig. 24. Examples of Comparison Between Hierarchical and Heterarchical Program.



(a) (b)

Fig. 26. Illustrates difficulty for concave body.



(a) (b)

Fig. 25. Illustrates lack of cues.