RECONSTRUCTION OF CURVED-SURFACE BODIES FROM A SET OF IMPERFECT PROJECTIONS*

Ruth Shapira
23 Sweden Street, Dania, Haifa, Israel

Herbert Freeman
Rensselaer Polytechnic Institute, Troy, N. Y. 12181

Abstract

A procedure is described for obtaining the description of solid bodies from a set of pictures taken from different vantage points. The bodies are assumed to be bounded by faces which are planar or quadric, and to have vertices formed by exactly three faces. The line and junction information provided to the program by the preprocessor is assumed to contain defects such as missing lines or wrongly classified junction types. The procedure is able to build a description of the bodies in spite of a moderate number of such imperfections. Use is made of a set of new grammar rules for line-drawing projections of curved and planar bodies.

1. Introduction

In computer scene analysis a picture of a three-dimensional scene is converted into an array of pixels, with each assigned a number corresponding to the average grey value in that local area of the picture. A common scheme for preprocessing the array for scene interpretation is to extract lines that are believed to correspond to the bodies' edges. To interpret a perfect line description of a scene, a picture grammar must be supplied that is based on the common general properties of the scene's bodies. When the line description contains errors (as a result of preprocessor limitations), its parsing and interpretation becomes difficult if not impossible. In such a case, understanding of the scene can be gained by using a set of line structures (each possibly defective), extracted from pictures describing the scene from different vantage points, a scheme which is also useful for obtaining three-dimensional information about the bodies.

The scene'interpretation given in terms of edges (grouped first according to faces and then according to bodies), is most suitable for polyhedra, but can also be applied to curved bodies. This paper is restricted to the analysis of bodies with planar or quadric faces. The technique, however, could easily be modified to handle also bodies whose curved faces are close to quadric surfaces, and this includes a large subset of man-made bodies.

Many papers were published in recent years dealing with polyhedral scene analysis, of which the papers by Roberts [7], Guzman [5], Falk [3], Huffman [6], Clowes [2], Waltz [11] and Ganapathy [4] are representative of the line of research that has been pursued. Some researchers contributed to the understanding of the process of perception by analyzing perfect (but unfeasible) line drawings. Others used real data (slightly edited) with its imperfections, but restricted themselves to a set of known bodies. Nearly all limited themselves to a single picture of the scene. (A restriction that led to many unnecessary difficulties). The work by Underwood and Coates [1.0] does utilize multiple pictures in the analysis of scenes of a single convex polyhedron - not to overcome data and analysis problems but merely to "have a look" at all sides of the body.

Attempting to extend the work from polyhedra to even a restricted family of curved bodies increases the difficulties to an extent which may not be apparent at first glance. Such an effort has been described by Chien and Chang [1]. They analyzed perfect line drawings representing bodies with planar, conic, or cylindric faces, where no intersection of two curved faces was allowed.

In this paper we shall show how a program can be made to "understand" a set of bodies with planar or quadric faces, utilizing defective data extracted from a set of multiple photographs. No pre-knowledge of the bodies is assumed, but certain general properties of the scene - such as the existence of precisely three faces in every vertex - are stipulated.

2. Basic Assumptions

The terminology used has been adapted from that employed by Woon and Freeman [12] and Clowes [2].

We impose the following restrictions: 1) Every vertex in the scene is formed by exactly three surfaces and belongs to exactly three edges; 2) Smooth transition between two different faces is not allowed (i.e., the derivative must be discontinuous across the edge); 3) The camera position is assumed to be "general". (For example, the projections of different vertices may not coincide in any picture); 4) No limb passes through a vertex.

In accordance with the foregoing, junctions can have no more than three lines. They can be classified into the types W, Y, V, T, A, and S. An example of each type is given in Fig. 1(a). Types Y and W must be the projections of vertices.

* This work was supported by the Directorate of Mathematical and Information Sciences, Air Force Office of Scientific Research, under grant AFOSR 76-2937.

Type T results from the covering of part of an edge or a limb by a face. Types S and A are the projections of virtual vertices (intersections of edges and limbs). In a perfect line description V is also a projection of a vertex.

The scene data is assumed to be extracted visually from a set of real photographs. The data includes line information (as a set of points) and junction information consisting of coordinates and types. The data is expected to be (in addition to the geometric inaccuracies) defective in the sense that (1) visible lines or parts of visible lines may be missing, (2) visible junctions may be missing, (3) junctions may be reported incorrectly (V instead of Y, W, T or S junctions).

We call a junction valid when it is a projection of a vertex. Thus all Y and W junctions are valid, all A, S, and T junctions are not valid, and the nature of the V junctions cannot be decided until more evidence is collected (as will be explained later). In those cases in which a preprocessor has difficulty in distinguishing between a T junction and a flat Y or W junction, they can be included in the group of the undecided junctions.

3. Cyclic Order Property

As already stated, *every* vertex belongs to exactly three edges. We define for these edges a cyclic order in the vertex [8,9], as the order induced by "walking around" the vertex in a clockwise manner, and numbering the edges in the order 1<2<3<1. If we trace out the edges of a boundary, we consistently change edges at the vertices, either always in a decreasing cyclic order or always in an increasing cyclic order. Let us always choose to trace edges in an increasing order. Then we shall traverse *every* edge in two different directions as we walk around the two faces that share it, as illustrated in Fig. 1(b). When a vertex's projection has three lines, their cyclic order is determined by the arrangement of the lines in the picture. (The clockwise sense of the cyclic order is preserved in a projection when the three edges are visible). However, when one line is invisible or missing, the cyclic order is not known. Let us define the notation AB<AC to mean that in junction A the line AC follows the line AB immediately in the cyclic order. Then if A is a two-line valid junction, we have either the relation AB<AC or the relation AC<AB.

Knowledge of the cyclic order at a junction is crucial to the assembly of the lines belonging to a single region. Therefore it is important to have means for determining the cyclic order in a junction when the order is not given naturally.

We define a line assembly (LA) as the directed path followed in tracing out the lines corresponding to a single boundary - or any continuous part of it - in increasing cyclic order. We denote an LA in picture i by the ordered set of junctions $A_{i,1}, \ldots A_{j,n}$ visited in the course of the trace. Now it follows from the definition that for every $1<k<n$ we have $A_{i,k}A_{k,(k-1)} < A_{i,k}A_{i,(k+1)}$. Two LA's are said to be distinct if they trace out lines corresponding to two distinct boundaries. Two LA's, $A_{i,1} \ldots A_{i,n}$ and $A_{j,1} \ldots A_{j,m}$, are distinct if

1) $A_{i,1} \equiv A_{i,n}$ and there is at least one A_j different from every A_i, or

2) There are two successive junctions in the two LA's such that $A_{i,k} \equiv A_{j,h}$ and $A_{i,(k-1)} \equiv A_{j,(h+1)}$, or

3) $A_{i,n} \equiv A_{j,1}$ and the relation $A_{j,1}A_{j,2} < A_{i,(n-1)}A_{i,n}$ exists.

Two rules forcing a cyclic order in a junction can be stated now:

Rule 1: If we have an LA $A_{i,1} \ldots A_{i,n}$ in which $A_{i,1} \equiv A_{i,n}$, then we must have in $A_{i,1}$ the relation $A_{i,1}A_{i,(n-1)} < A_{i,1}A_{i,2}$.

For example in Fig. 2(a) we have the LA B,C,D,A,B. (The LA is shown as an arrowed line). The lines in B are thus forced to have the relation

BA<BC

Rule 2: If we have two distinct LA's, $A_{i,1} \ldots A_{i,n}$ and $A_{j,1} \ldots A_{j,m}$ such that $A_{j,m} \equiv A_{i,1}$ and $A_{j,(m-1)} \neq A_{i,2}$ then we must have in $A_{i,1}$ the relation $A_{i,1}A_{i,2} < A_{j,m}A_{j,(m-1)}$.

For example in Fig. 2(b) we have the two distinct LA's: E,F,C,B and B,A,G,E. They are distinct because of the cyclic order already established in junction B by Rule 1. Now by Rule 2 the relation EF<EG must hold in E. (This in turn establishes the LA M,F,E,G,N,0).

A more complicated strategy may be adopted to force the correct order at a junction when these two rules cannot be applied directly. One assumes a cyclic order in the junction and tries to reach a contradiction by a sequence of inductive steps. If a contradiction is reached, the cyclic order opposite to the one selected is forced on the junction. For example, in Fig. 2(c) we assume an order in junction R such that RQ<RN. Then we can establish the LA's P,Q,R,N,G,A,D and D,C,F,M,P. They are distinct because of the existing natural order in D, and hence by Rule 2 force in P the relation PQ<PM. At the same time we have the two LA's R,Q,0 and 0,N,R which are distinct due to the assumed order in R. By Rule 2 the relation 0N<0Q is forced in 0. This in turn establishes the LA M,F,E,G,N,0, Q,P,M. Now by Rule 1 we must have in M the

relation MP<MF, which contradicts the natural order in M. This sequence of deductions was based on the assumption that the relation RQ<RN exists in R. The contradiction reached lets us conclude that the opposite relation, namely RN<RQ, must exist in R. This establishes the LA 0,N,R,Q,0 and, by Rule 1, the relation 0Q<0N in 0. We have thus been able to find the cyclic order in 4 of the original 5 unordered junctions. Junction P remains unordered. The final ordering of the junctions is given in Fig. 2(d).

A by-product of the foregoing is a new class of impossible objects, in addition to those described by Huffman [6]. The objects can be identified as being impossible if their structure is such as to lead to contradictory cyclic order at their junctions [8,9].

4. Junction Matching

To utilize the data from all pictures, we must know the parameters that relate each picture's coordinates to those of the other pictures and also to the universal coordinate system in which the final description should be given. These parameters are difficult to measure. Some of them are measured indirectly and some are calculated by minimizing the sum-of-squares differences between the measured and calculated projections of a set of predetermined points. A detailed description of the parameter determination is given in [8].

In order to arrive at a true description of the 3D scene in spite of the data imperfections, we compare the line structures from the given pictures (three in our case) against each other and use the information found in one picture to verify the information found in another.

Let us start with junctions. We have two goals. One is to determine those V junctions for which there is enough evidence to assume that they are valid junctions. The second is to group, from the different pictures, junctions that are the projections of the same vertex. From here on in this section, whenever we use the term "junction", we shall mean a V, Y, or W junction.

If two junctions in two different pictures are projections of the same vertex, they must obey certain geometric rules. Let P., Cj, and A. , be the picture plane, the center of projection, and the projection of vertex J, respectively, for picture i. Then for two pictures, i and j, C_j, $^{C}j'$ j, $^{A}i, J'$ and $^{A}j,J$ are copTanar, and the lines C C J, C^A^j, and C.A. j intersect any plane P in three colinear points: C. , J. and J. , respectively (Fig. 3). We shall call the line formed by these points a match line. Thus if we re-project into the plane P_q junvyion A. j from C, junction A^j from C., and C_1 from C, we can check whether this condition is satisfied. Two junctions from different pictures that satisfy this condition are said to be matchable.

Obviously, if a vertex I is in the plane of C.,C, and J, then A. $_T$ and A. , are matchable, and so are A. , and A. $_T$. To resolve such ambiguities, we refer to the third picture, picture k. We require that C_i, C_j, C_k be not colinear. Then if $A_{i.j,}$ A. ,, and A, , are the three projections of vertex J, the three different match lines on plane P_q form a triangle whose vertices are J. , J , , and J. (Fig. 4). We call the set of three junctions, from different pictures, that *are* matchable in pairs, a triple. By forming triples we can eliminate many pairs that are matchable but are not projections of the same vertex. However, even in the pure geometric case we may have three projections of three different vertices that form a triple [8].

We are dealing here with data extracted from actual photographs; thus we must accept A. , and A. , as matchable even if J. is distant from the match line C_{ijq} J_{jq}. by an amount less than some threshold. Relaxing the colinearity requirement on the match lines increases the ambiguity of matching. The larger the threshold distance, the greater the number of ambiguous cases.

Two matchable junctions that are the projections of the same vertex are said to match each other. A triple in which the junctions match each other is called a match triple. To establish matches, we find for every junction all matchable junctions in the two other pictures. Then we form all the triples. To find the match triples in the set of all triples, we must use some picture context. The kind of context we shall use is line connections between junctions. If we have three lines - each in a different picture - whose ends form two triples, the two triples are each considered to be a match triple. (The chance of naming a wrong triple as a match triple still exists but is now greatly reduced).

Since it is possible that a vertex's projection will be missing in one or more pictures, it is desirable to establish matches between matchable pairs. Naturally, more context must be used to establish a match on the basis of only a two-picture comparison. The pair match is done in two passes and only after the triple match phase is completed. (It reduces the number of free candidates and, therefore, the chance of error). We select a set of three junctions for which one of them - call it the middle - is connected by lines to the other two that are matchable to a set of three junctions connected (in the same way) in another picture. In the first pass we require that the middle junction be of type Y or W and that the two lines have the same cyclic precedence in the two junctions. The three junctions are then assumed to match the three junctions in the other picture. In the second pass

we allow one or both of the middle junctions to be of type V and do not check the cyclic precedence (it may be unknown for the V junction). The second pass is a little more hazardous since we are unable to use cyclic precedence, but since it is carried out after most junctions are already matched, there is only a small chance of making an error. If one junction in a triple is matched to the other two by two applications of a pairwise match, we consider the triple to be a match triple; that is, we consider the remaining two as matching also.

Every V junction that is matched to a junction in another picture is marked as valid. The match supplies additional evidence that the junction is a projection of a vertex. The validated V junctions are assigned a cyclic order, when possible, by applying the cyclic order rules.

5. Line Matching

The matching of lines in different pictures poses difficulties because of data imperfections and because the number of lines between two junctions can vary from zero to three. Matching is done for two pictures at a time for the lines terminating at matched junctions. There are three cases to be considered: (1) The two matched junctions are both 3-line junctions, (2) one is a 3-line and the other a 2-line junction, and (3) both junctions are 2-line junctions. For a detailed discussion of this the reader is referred to reference [8].

When two junctions around which a match of lines is attempted *are* both cyclically ordered, a match of *one* pair of lines yields in turn the corresponding match of the successors in the cyclic order. Also two lines in two pictures found to match to the same line in a third picture are declared to match each other.

When two lines of a cyclically ordered junction are matched to two lines of an unordered junction, the second junction can be ordered because its cyclic order must agree with that of the junction to which it is matched.

The line matching procedure makes it possible to detect missing connections between junctions. The lines around the junctions are assigned indices that agree with the cyclic order in ordered junctions, or re-assigned to agree if the cyclic order is determined at a later stage.

6. Object Formation

The collection of lines from all pictures that correspond to the boundaries of a single face will be referred to as a face group. Clearly, *every* line that is not a limb belongs to two face groups. We build a face group by selecting a line, in any picture, that does not yet belong to two face groups. We then follow the LA in a direction not previously followed, adding to the group *every* line traced and its matches in the other pictures. If the trace returns us back to the line from which we started, the trace is terminated - and this face-group generation is complete, assuming that the face has only one boundary. Otherwise when we reach a line for which no next line in the LA is available, we check whether an LA trace can be continued from the line's match in another picture. The face-group generation may thus proceed by "jumping" from one picture to another. If the process is blocked in all pictures, we return to the starting point and try to follow the LA in the opposite direction, again "jumping" from one picture to another, if necessary, until the process is blocked again in all pictures. All S and A junctions encountered in the trace are remembered. When the process terminates, the limbs at these junctions are checked and if a limb's other end junction belongs to a line not yet in the face-group (implying that another boundary of the same face has been detected), the trace resumes there. The limb lines are also added to the face group.

When all lines have been traced twice, further data recovery action is called for. This second-level data recovery can be carried out for the following three configurations, illustrated in Fig. 5.

1. Two lines *l* and k in the same picture, with a valid junction in only one *end* of each, have been assembled into two face-groups M *and* N, k, *l e* M *and* k, *l z* N

   Action: Combine *l and* k into *one line.*

2. A line *l with one valid end junction has been assembled into two face-groups M and* N, *and there is in the same picture a* 2-line valid junction J with lines m and n, m *e* M *and n* e N.

   Action: Extend the line l to junction J.

3. There are two 2-line valid junctions I and J in the same picture with lines k, *l* and h, g, respectively, where k and h have been assembled into face-group M, and *l* and g have been assembled into face-group N.

   Action: Create an "empty" line between the two junctions I and J. (An "empty" line is a line whose only known points are its end junctions and whose form has not yet been determined.)

The data recovery actions utilize the property that two components assembled simultaneously into two face-groups must correspond to the intersection of the associated faces. There is, of course, the possibility that the intersection is sectioned into several separate edges and that the two lines correspond to two different sections; but at this stage of the recognition phase, with no evidence to the contrary, we assume that they belong to the same section.

Each face group corresponds to a face of one of the bodies in the scene. After the face-group assembly and second-level data recovery is completed, we are ready for the description of the objects. Each object is a set of face-groups, with no common elements between two sets. *Eyery* face group must belong to some set. A new object is formed by taking a face group that is as yet unassigned and recursively adding to it *eyery* unassigned face group that has a line in common with any of the face groups already in the set.

We also wish to determine those face groups that correspond to curved faces. Clearly, every curved face must manifest itself somehow, or we have no way for recogniziing it as curved. We know that whenever two face groups share, in at least one picture, a curved line, at least one of them must correspond to a curved face. It is also obvious that *eyery* face group that contains a limb must correspond to a curved face. Thus to determine the curved face, we build, by means of a tree search, a minimal set of faces containing initially all those faces whose face groups have a limb, such that at least one partner (actually its corresponding face), from each pair sharing a curved line, is present in it. A procedure for extracting information about the nature of the faces and for getting the faces' equations is described in [8].

7. Experimental Results

A scene that is typical of the kind considered for analysis is shown in Fig. 6. Three views are shown, each from a different vantage point. Schematic descriptions of the three pictures are given in Fig. 7. In the input data, the two non-limb lines of an S junction, which are actually parts of the same line, have distinct labels; however, later on one of these can be deleted.

The procedure was able to validate 28 of the 34 V junctions in the junction-matching phase. After the line-matching process, all the junctions shown circled in Fig. 7 had been cyclically ordered. Data recovery action caused the "empty" lines (shown dashed in Fig. 7) to be generated, as well as the line extensions marked by arrows.

The final scene description reported by the program was given in the form of face groups (consisting of sets of triples of matched lines) for each of the 5 bodies in the scene. Detailed results, as well as other scenes processed, are described in [8].

8. Conclusion

The objective of the research described here was to investigate techniques for the computer understanding of pictures of scenes. Three principles guided the research: (a) No complete preknowledge of the scene's bodies should be assumed since that would limit the repertoire of bodies, (b) The number of bodies should be unrestricted and merely some properties of the bodies should be prespecified which are sufficiently general to permit inclusion of all bodies likely to be encountered, (c) Since the input is to be gotten from photographs, the presence of a limited amount of data imperfections such as missing lines, wrong junctions, and geometric inaccuracies should be permitted.

To achieve the objective, a set of multiple pictures of the scene was assumed to serve as the input data. New grammar rules needed for the analysis of these pictures were formalized. A procedure for establishing matches between features in the different pictures and for verifying doubtful features were devised, as well as procedures for eliminating wrong data and for recovering missing data. Finally a procedure for assembling the analyzed data into sets, each describing a single body of the scene, was constructed, including the determination of the bodies' faces and their nature. A computer program based on this approach was written. The program was able, given real input data, to "understand" the photographed scene, and to yield a plausible description.

References

1. R. T. Chien, Y. H. Chang, "Recognition of Curved Objects and Objects Assembly", Proc. of the Second International Joint Conference on Pattern Recognition, IEEE Publication No. 74 CH0885-4C, Copenhagen, pp. 465-510, August 1974.

2. M. B. Clowes, "On Seeing Things", Artificial Intelligence, 2, (1), 1970, pp. 79-116:

3. G. Falk, "Computer Interpretation of Imperfect Line Data as a Three-Dimensional Scene", doctoral dissertation, Computer Science Department, Stanford University, Stanford, CA, August 1970. (AD 715 665).

4. S. Ganapathy, "Reconstruction of Scenes Containing Polyhedra for Stereo Pair of Views", AI Memo 272, Stanford University, Dec. 1975.

5. A. Guzman, "Computer Recognition of Three-Dimensional Objects in a Visual Scene", doctoral dissertation, M.I.T., Cambridge, Mass., December 1968. (AD 692 200).

6. D. A. Huffman, "Impossible Objects as Nonsense Sentences", Machine Intelligence, 6, ed. B. Meltzer and B. Michie, American Elsevier, 1971, pp. 295-323.

7. L. G. Roberts, "Machine Perception of Three-Dimensional Solids", Optical and Electro-optical Information Processing, ed. J. T. Tippet et. al., M.I.T. Press, Cambridge, Mass., pp. 159-197, 1965.

8. R. Shapira, "Computer Reconstruction of Bodies Bounded by Quadric Surfaces from a Set of Imperfect Projections", doctoral dissertation, New York University, Sept.1976. (AD A035 563).

9. R. Shapira and H. Freeman, "A Cyclic-Order Property of Bodies with Three-Face Vertices", accepted for publication in IEEE Transactions on Computers.

10. S. A. Underwood and C. L. Coates Jr., "Visual Learning from Multiple Views", IEEE Transactions on Computers, C-24, (6), June 1975.

11. D. A. Waltz, "Generating Semantic Description from Drawings of Scenes with Shadows", doctoral dissertation, Artificial Intelligence Lab., M.I.T., Cambridge, Mass., November 1972. (AD 754 080).

12. P. Wood and H. Freeman, "A Procedure for Generating Visible-Line Projections of Solids Bounded by Quadric Surfaces", Information Processing 71, ed. bu C. Freiman, North-Holland Publ. Co., Amsterdam, 1972, pp. 1120-1125.

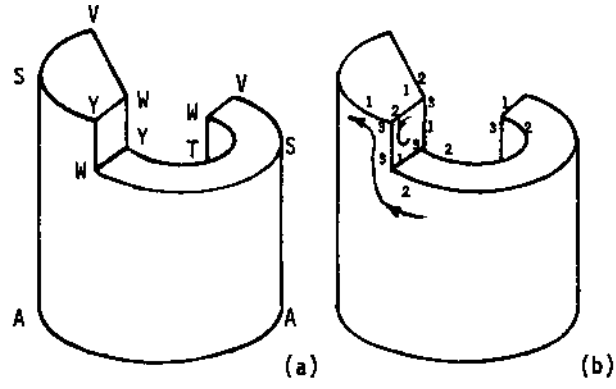


Fig. 1. Junction types and cyclic order.

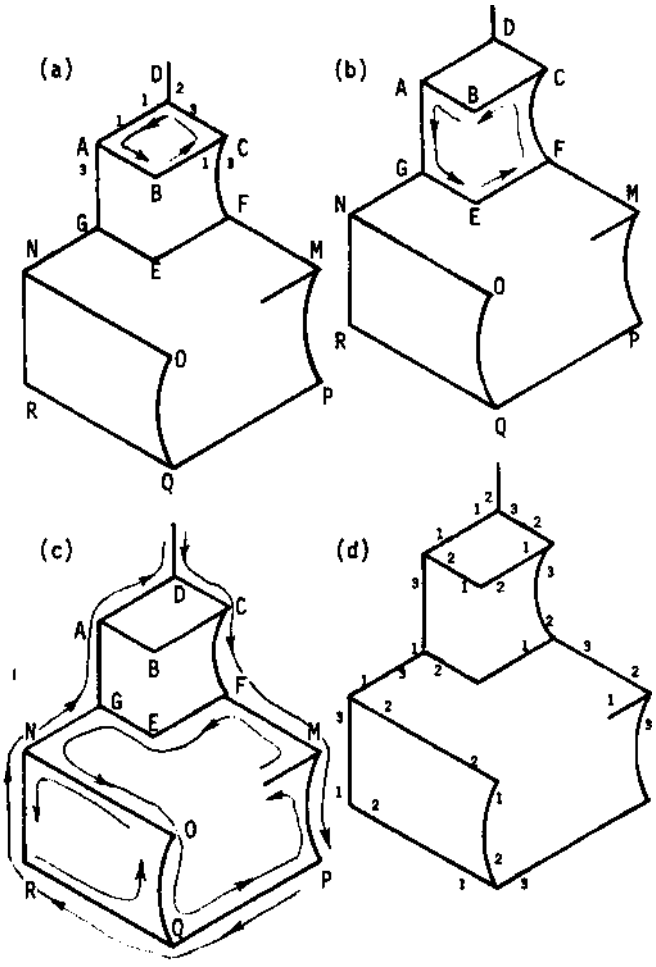


Fig. 2. Forced cyclic order.

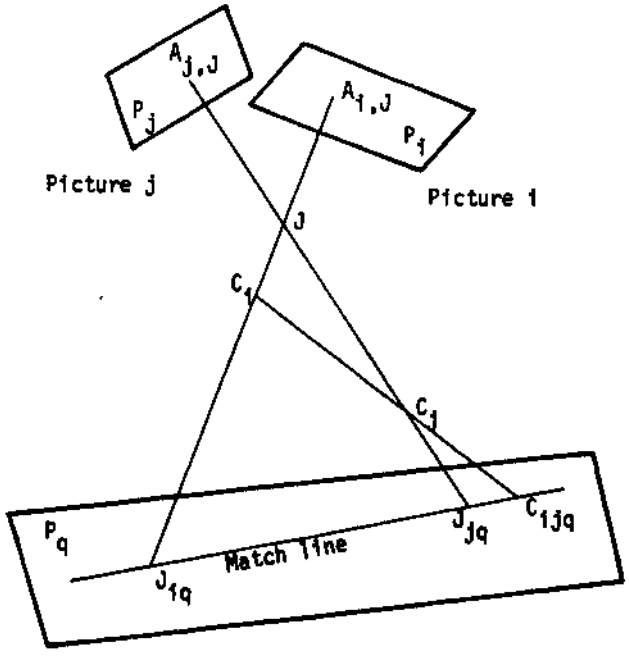


Fig. 3. The match line.

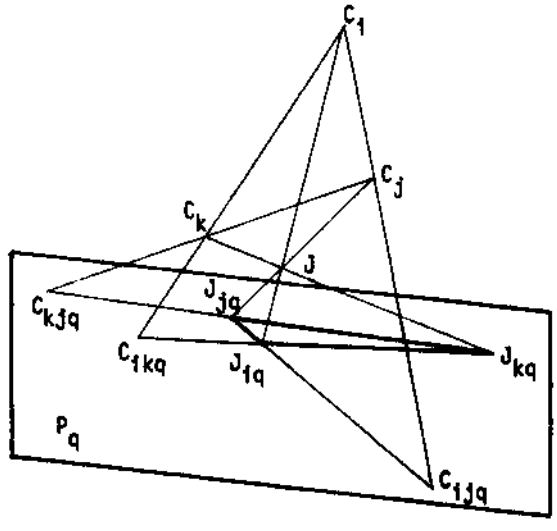


Fig. 4. The match-line triangle for vertex J. (Match lines shown bold.)

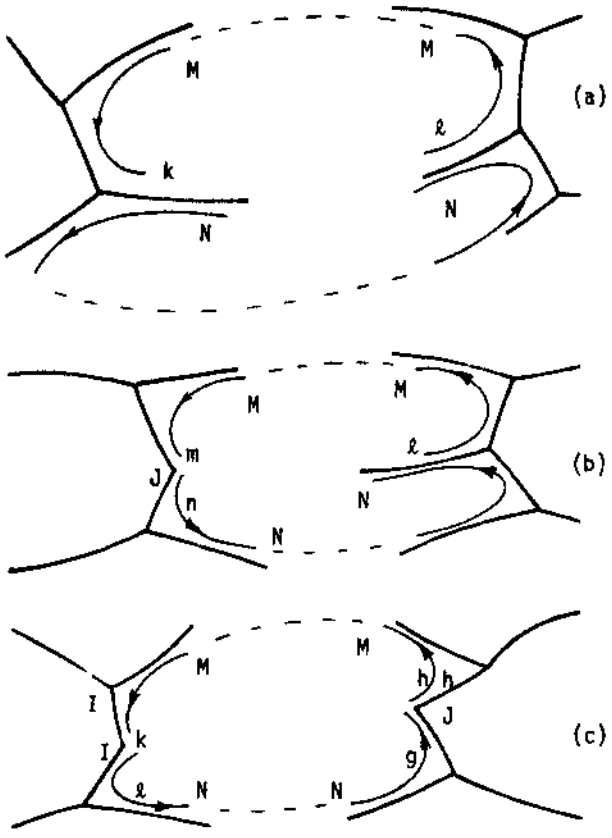
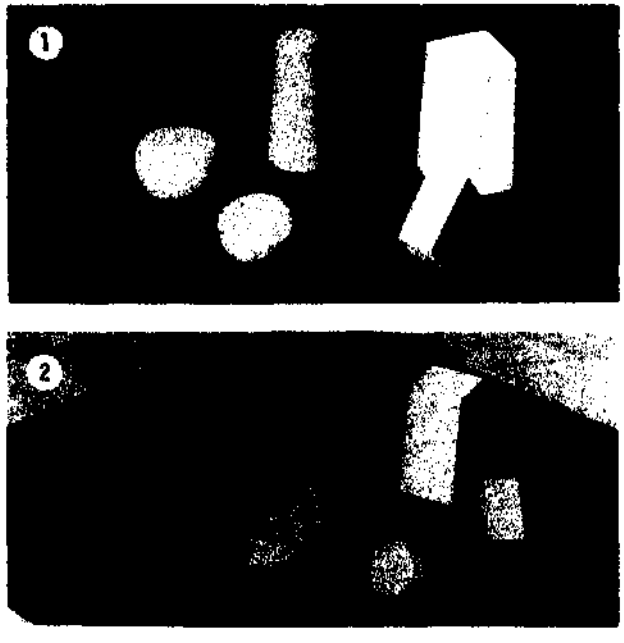


Fig. 5. Three configurations for second-level data recovery.



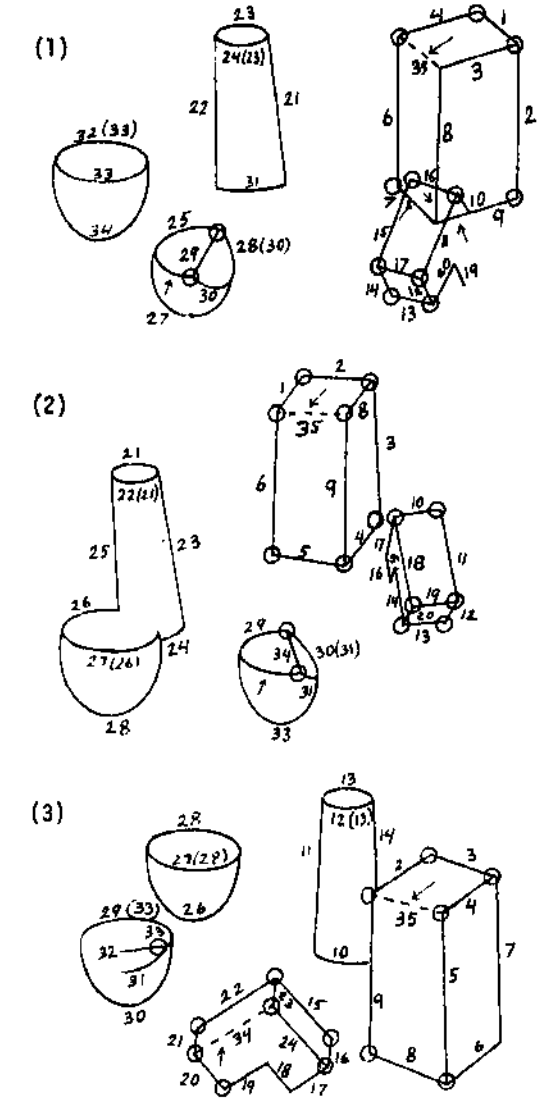


Fig. 7. Schematic descriptions of three pictures of sample scene. Results marked with circles, arrows, and dashed lines.

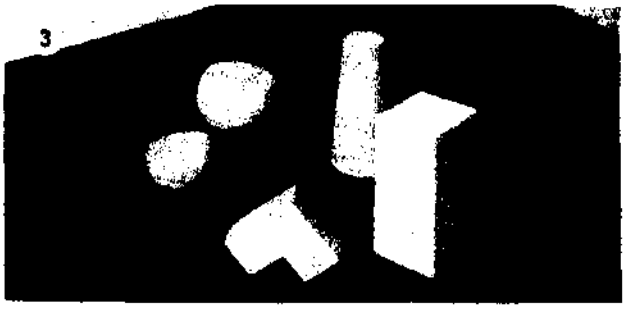


Fig. 6. Three pictures of sample scene.