Correoportdonce in Line Drawings of Multiple Views of Objects

Charles Thorpe and Steven Shefer

Carnegie-Mellon University

Abstract

As an object moves relative to a viewpoint, its appearance changes. In this paper we analyze the topological constraints on the changing appearance of line drawings of objects as the objects or the camera move. We start with a Huffman-Clowes junction dictionary. We show a way of deriving vertex types from junction types by inference rather than by table look-up. We then derive three constraints on the change in appearance of an object: conservation of vertices, conservation of vertex type, and conservation of adjacencies. Using these constraints, we develop a matching algorithm that traces vertices from one image to the next. Examples are given showing correct matching tor simple objects, including partially visible objects and multiple objects in the same scene.

1. Introduction

In this paper we examine the appearance of trihedral vertices from different viewpoints. This analysis is used to derive the effects cf changing viewing position on the appearance of line drawings of trihedral objects, That in turn helps solve the problem of identifying the same vertices in different pictures. The objects we consider are trihedral blocks; that is three planar sides meet at each vertex. The drawings are initially presumed to be perfect. The change in viewing position can come either from moving the camera, as in stereo vision, or from moving objects.

The first part of this paper examines the change in appearance of a vertex as the viewing angle changes. We start out with Huffman-Clowes enumeration of junctions, the two-dimensional images of three dimensional vertices. We show a way of deriving vertex types from junction types by inference rather than by enumeration and table look-up. We then develop a "transition table", showing how the image of a vertex can change from one type of junction to another as the viewpoint changes.

In the second part we show a matching process that relies on topology to identify the same vertices in two line drawings of the same scene. The only information used is line labels and connectivity. These give us three constraints: conservation of vertices, conservation of adjacencies, and conservation of vertex types. This process works for scenes with more than one object or with only part of an object visible. We discuss extensions and show several simple examples.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

2. Trihedral Vertices

2.1 Introduction: Huffman-Clowes Junction Dictionaries

Huffman [5] and Clowes [3] independently developed "junction dictionaries", lists of possible appearances in line drawings of all configurations of trihedral vertices. They first note that there are only two types of edges: concave and convex. The two planes that form the edge divide space into four sections, with the edge appearing topologically different from each section. In the case of a concave edge, the edge will be obscured from three of the sections and will appear simply as a concave edge from the fourth, labeled "-" by Huffman. For a convex edge, the edge will be visible from three sections. From the front, it will appear as a convex edge, designated with a "+ ". From either side, it will appear as an occluding (obscuring) edge, with the nearer surface blocking the farther surface from view. Huffman labels occluding edges with an arrow along the edge such that the surface on the right of the arrow occludes the surface on the left.

Trihedral vertices are formed by the intersection of three surfaces, which is also the intersection of three edges. There are two ways of enumerating all possible trihedral vertices. The first deals with the types of edges that meet at the vertex. Each of the three edges can be either convex or concave. This gives four different vertex types: all three edges convex, two convex and one concave, one convex and two concave, or all three concave.

The second method deals with the space filled by an object, and gives more geometric intuition. The three planes that form a trihedral vertex divide space into octants. Some of the octants can be filled and the rest left empty. The combinations of filled and empty octants that give single, connected objects with three surfaces meeting at the center have one, three, five, or seven filled octants. These vertex types are usually named by Roman numerals according to the number of filled octants. Each corresponds to one of the types enumerated by their edges: type I is all three convex, type III is two convex and one concave, type V is one convex and two concave, and type VII is all three concave.

Huffman and Clowes point out that each type of vertex can be viewed from each of the empty octants. The image of a vertex is called a junction. By viewing all vertex types from all empty octants they construct a "junction dictionary" showing all topologically different types of junctions that can occur in the trihedral world. Table 1 shows all junctions, each listed by type of vertex of which it is an image.

2.2 Reasoning From Junction Type to Vertex Type

Notice in table 1 that each junction type can result from only one kind of vertex. For instance, the arrow-shaped junction with one + and two > labels can only be the image of a convex-convex-convex (type I) vertex. So if the labels of all lines on a junction are known, it

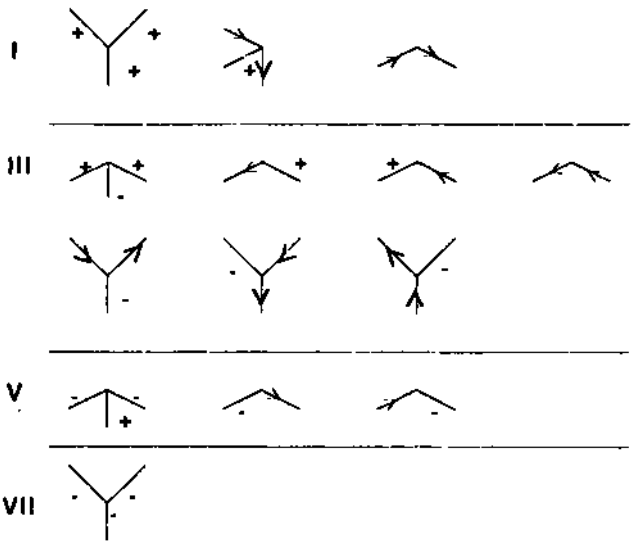


Table 1: Summary Of Junction Types

is possible to tell what three dimensional configuration must be pictured.

This can be explained by edge types Lines with + or arror labels must be inages of concave edges Lines with labels are images of concave edges. So every junction with three lines visible and labeled is easily traceable to the type of vertex that formed it. For instance, the Y junction labeled with two arrows and one minus must come from the type III configuration, since two of its edges are convex and one concave.

This reasoning from appearance to vertex type takes more analysis for L junctions. The key is to find where the hidden third edge must be. This gives the location of the hidden surface or surfaces, and thus gives the edge types.

The first case is when the edge is within the angle ABC (figure 1), and ABC is the only visible surface. Then AB and BC both appear as occluding edges, which means that they are both convex. The object must lie below the arc ABC, and the two hidden surfaces must meet in a convex edge. So this is the all-convex vertex, type I.

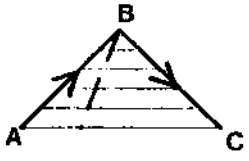


Figure 1:

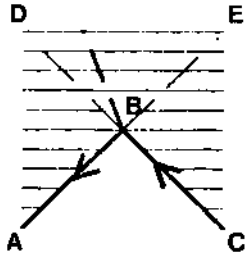


Figure 2:

If the hidden edge is within the arc DBE, the opposite situation exists. The one visible surface is the area outside of arc ABC (figure 2). Edges AB and BC are still occluding, and hence convex, but the direction of their labeling arrows is reversed. The object lies entirely under the visible surface, and the invisible surfaces must meet in a concave edge. This is the two convex and one concave vertex type.

As the hidden edge swings around into arc ABD (or, in the mirror image, arc CBE), one of the hidden surfaces now becomes partly visible (figure 3). The newly visible surface 1 joins the "top" surface 2 along edge BC; that edge is then labeled with a plus rather than an arrow. The one remaining hidden surface meets surface 2 along edge AB and must extend back underneath it. Its intersection with surface 1 along (he hidden edge will then be concave These must also be formed by vertices with two convex and one concave edge.

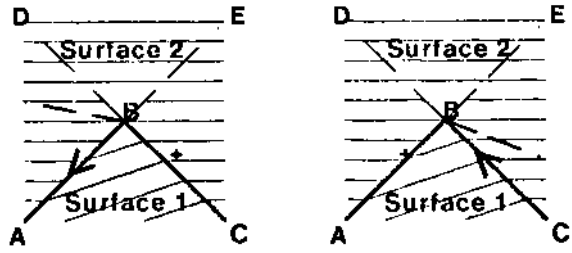


Figure 3: Two Surfaces Visible. Hidden Edge In ABD Or CBE

The final case is two surlaces visible and the third edge hidden by surface 1 The hidden edge is formed by the third surface and surface 2 Since this edge is under surface 1. surface 1 must be in front of surface 2 at that point This means that the intersection of surfaces 1 and 2 must be concave, since surface 1 sticks out over surface 2. The other edge of surface 1 must then be occluding. Then the hidden surface must run from the occluding edge to the hidden edge, and must form a concave edge with surface 2. This is the one convex two concave case. The two symmetric cases of this are shown in figure 4.

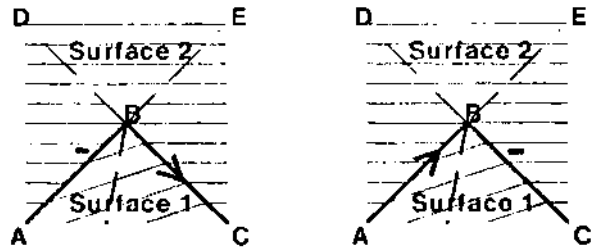


Figure 4: Two Surfaces Visible, Hidden Edge In Arc ABC

Note that there are a series of constraints that may be interchanged. It is not necessary to know line labels, surface visibilities, and hidden edge location in order to derive vertex type. For instance, in figure 1, it would be sufficient to know that there is a surface within arc ABC and none outside. In that case, all the rest of the information would be constrained. More information than just surface visibility would be needed to distinguish between cases like figures 3 and 4, but even there line label information could be traded for knowledge of hidden line position. This information could come from other sources, such as intensity or orientation. For instance, in urban scenes there is very often a vertical edge at each vertex. If the camera geometry is known, it is possible to predict the direction in the image of the vertical edge. If the vertical edge is the missing one in an L junction, it would be possible to determine the type of the vertex.

2.3 Effect of Motion on Appearance of Trihedral Vertices

As the viewpoint moves relative to the vertex, the appearance of the vertex may change. Ac long as the viewpoint stays in one octant, the image of that vertex remains the same type of junction. But when the viewpoint crosses into another octant, the vertex will appear as a different junction type. There are two strong constraints on the change in junction type. The first Is conservation of vertex type. Since the underlying vertex type remains the same, the junctions must all be images of that type of vertex. So, for instance, a junction in row I of table 1 can become, due to viewpoint change, any other junction in row I, but cannot become a junction from rows III, V, or VII. The second constraint is that each octant is adjacent to three other octants. As the viewpoint changes, it can only go from an octant to one of the three adjacent octants. So the junction type can only change to the junction type derived from an adjacent octant. These two constraints give a transition graph, showing all possible junction type transitions, as in figure 5. The conservation of type constraint splits the graph into four disjoint parts, one for each type of vertex. Each part of the graph has eight nodes, one for each octant, with 1, 3, 5, or 7 of the nodes marked "invisible" because they correspond to viewing positions behind the object.

Another constraint comes from the direction of the change. Changing octants means crossing one of the three planes that meet at the vertex. If the motion of the camera is known to be parallel to one of the planes, or is known not to be large enough to cross a plane, the possible junction type transitions are constrained. For instance, in interpretation of aerial photgraphs of urban scenes it is often easy to tell whether the camera has crossed the plane of a wall Furthermore, the motion of the camera will be nearly parallel to the plane of flat roots. These constraints can greatly reduce the possible appearance of vertices.

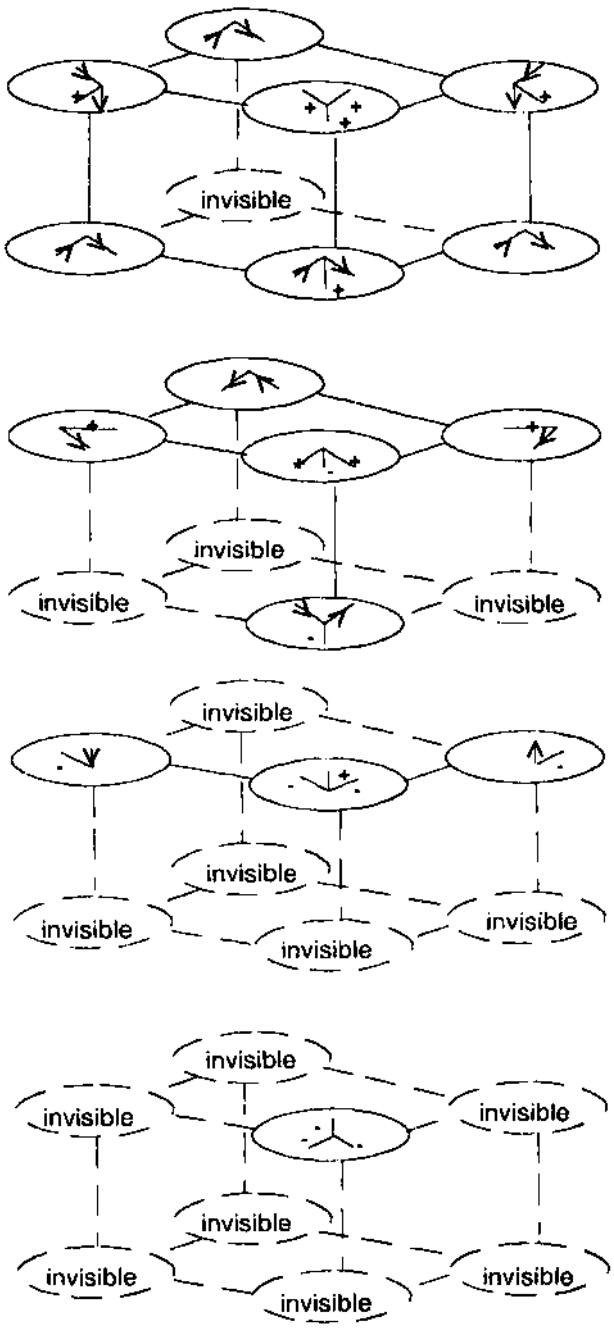


Figure 5: Transition Table

3. Correspondence

In this part of the paper, we show how to find correspondences consistent with the topological and line label constraints. Studying topological constraints in isolation will make it easier to incorporate them into a full system which makes use of quantitative as well as topological information.

Finding matching points is an essential step in tracking object motion and in calculating depth from stereo images. We discuss tracking the vertices of trihedral blocks from one line drawing to another. The two views can come from either stereo or from object motion. We make no assumptions about camera geometry (such as known epipolar lines), fixed relations among objects, or magnitude of the change between images. We assume that some other process has done Huffman-Clowes labeling on each image. This carries with it implicit assumptions of completeness of the scene and of a "general viewpoint", that is, no coincidental alignments in the image.

3.1 Constraints

The object of correspondence is find a match in the second image for each point in the first image. There are three constraints that a complete, consistent set of matches has to satisfy: conservation of vertices, conservation of type, and conservation of adjacencies.

Conservation of vertices means that since the same objects are in each image, the same vertices must also be present. They may not be visible in each image, however. Under perspective projection, it would even be possible to have all vertices that are visible in one image invisible in the other.

Conservation of type means that a vertex must always keep the same shape. Although its appearance may change, that change is constrained (see section 2.3). Since we assume that all lines are labeled, every junction's type is known. A junction in one image can therefore only match a junction in the recond image that is the image of the same type of vertex.

Conservation of adjacency is really conservation of edges. If two junctions in one image are directly connected by an edge, the junctions they match in the second image must have a line connecting them or the possibility of an invisible edge between them.

These constraints allow some types of noise in the scene. Missing lines can be handled by the "invisible edge" criterion. Extra lines cannot be handled Changing angles can be tolerated, since quantitative geometry is not important, but they must not change from concave to convex or vice versa, since vertex types, and thus edge types, must remain fixed. Changing line length is permitted, which may be especially useful in real scenes. But polygonal approximations to curved surfaces will probably not work, since it is difficult to guarantee the same number of vertices in two different polygon fits.

3.2 Matching Algorithms

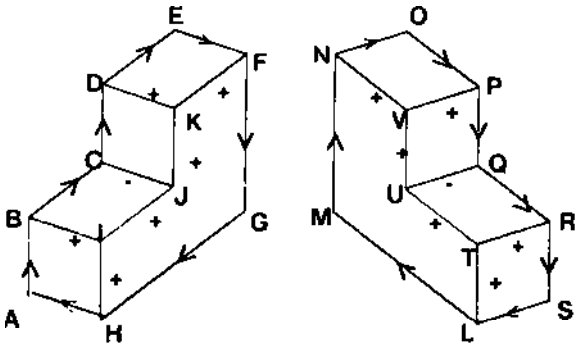


Figure 7: Two Labeled Pictures of an L Block

3.2.1 Correspondence Graph

Our matching algorithms are based on a central data structure, the correspondence graph. Each node in the correspondence graph is composed of a junction from one image and a junction from the second image, where the two junctions may be images of the same actual vertex. Node formation uses the "conservation of type" constraint In figure 7, junctions B and T are both images of a type I vertex, so BT would be a node. Junction U is an image of a type III vertex Since B and U come from different types of vertices, they cannot match, so BU would not be a node.

Invisible vertices must also be allowed for in the graph. Unless we know otherwise from outside sources of information, we must assume that there can be any number of invisible vertices and that they can be of any type. We create the special junction (invisible) that can match any or all of the junctions. Figure 8 shows all the nodes formed from the L blocks in figure 7.

Links in the graph connect nodes that are "consistent". Consistent means that both nodes could be part of the same overall match. Inconsistent means that if one node is correct, the other is incorrect. Consistency is defined using the "conservation of adjacency" constraint and its logical implications. We define the *distance between two vertices* to be the least number of edges that must be traversed to get from one vertex to the other. If all vertices and all edges were visible in an image as junctions and lines, it would be straightforward to determine the distance between two junctions. Then, given two nodes XX' and YY.' they would be consistent only if the distance from X to Y in one image was exactly the same as the distance from X' to Y" in the other image. For instance, node AL is consistent with node IR since the distance from A to I (two edges) is the same as the distance from L to R. The difficulty with this is that all vertices might not be visible, so it might not be possible to toil what the shortest distance is between two junctions. The way we handle this is to calculate the upper and lower hounds of the distance. The upper bound is the shortest distance using only visible lines. The lower bound uses invisible lines wherever they could possibly run Then the criterion for compatibility is that the range of distances must overlap. That is, the lower bound of the distance from X to Y must not exceed the upper bound of the distance from X' to Y' and vice versa.

AL, AM, AN, AO, AP, AR, AS, AT, AV, A(invisible),
BL, BM, BN, BO, BP, BR, BS, BT, BV, B(invisible),
CQ, CU, C(invisible),
DL, DM, DN, DO, DP, DR, DS, DT, DV, D(invisible),
EL, EM, EN, EO, EP, ER, ES, ET, EV, E(invisible),
FL, FM, FN, FO, FP, FR, FS, FT, FV, F(invisible),
GL, GM, GN, GO, GP, GR, GS, GT, GV, G(invisible),
HL, HM, HN, HO, HP, HR, HS, HT, HV, H(invisible),
IL, IM, IN, IO, IP, IR, IS, IT, IV, I(invisible),
JQ, JU, J(invisible),
KL, KM, KN, KO, KP, KR, KS, KT, KV, K(invisible),
(invisible)L, (invisible)M, (invisible)N, (invisible)O, (invisible)P,
(invisible)Q, (invisible)R, (invisible)S, (invisible)T, (invisible)U,
(invisible)V, (invisible)(invisible)

Figure 8: Nodes Formed From Figure 7

In figure 7, the upper bound on the distance from A to E is 4. In order to calculate the lower bound, we have to decide if there could be an edge directly connecting A and E. This takes some reasoning about the planes that form the surfaces of an object. If the lines are labeled, it is easy to tell which lines and points lie in the same plane. A line labeled + or lies in the planes visible on either side of it. An occluding line lies in the hidden plane and the occluding plane but not the occluded plane. Then in the example, lines FK and KJ lie in the same plane. So do lines JI, IH, and HG. So points F, G, H, I, J, and K all lie in a plane, called plane 1. Sim.larly, if there is an edge from E to A. then A. E. F, G. and H all lie in a plane, plane 2. Since F, G. and H all lie in both plane 1 and plane 2 the planes must be identical. ADIH is also a plane, and since A. I, and H are in plane 1, B must also be. So must C and D. But then the whole figure lies in plane 1, and is not a trihedral figure at all. This argument can be formalized and extended to show that in a line drawing of a trihedral object there can never be just one junction with only two lines accounted for. If there were a line from A to E, G would be the only vertex with two known edges. Since this is impossible, we infer that there must be at least one hidden vertex, and no direct connection from A to E. So the lower bound on the distance from A to E is two.

The lower bound on the distance from (invisible) to a junction is the distance from that junction to the nearest junction with less than three lines, plus one. The upper bound is infinite, since the connections between invisible junctions are unknown. Figure 9 shows all the links for node AL.

3.2.2 Searching the Correspondence Graph

A complete match consists of a subgraph of the correspondence graph such that the nodes contain every junction in each scene and are all linked to each other. A completely connected subgraph is called a clique. Finding cliques of a given size is NP-complete, which suggests that the best algorithms will be exponential. The algorithm we use for search makes no claims to optimality.

BM, BT, BS,
CU, C(invisible),
DO, DV, D(invisible),
EN, EO, EP, ER, EV, E(invisible),
FO, FV, F(invisible),
GN, GR, G(invisible),
HM, HT, HS,
IN, IR, I(invisible),
JQ, J(invisible),
KP, K(invisible).
(invisible)M, (invisible)N, (invisible)O, (invisible)P, (invisible)Q
(invisible)R, (invisible)S, (invisible)T, (invisible)U, (invisible)V,
(invisible)(invisible)

Figure 9: Links of Node AL

Wo define two sets of nodes, the set I of instantiated nodes (the clique) and the set P of possible nodes (to extend I). Initially I is empty and P is the correspondence graph. At each stop some node n from P is considered for being moved to I. There are two cases: either n is included in I or it is excluded. If it is excluded, it is removed from P. The new I is the old I and the new P is the old P minus n. If it is included, the now I is the old I plus n. To generate the new P, we take the old P minus n intersected with the links of p. This guarantees that all members of P are always linked to all members of I, and all members of I are linked to each other. Each case (n included and excluded) is generated and passed recursively to the search procedure. If at any point the union of I and P does not contain nodes that contain each junction, the conservation of vertices constraint is violated and that branch of the search terminates. Figure 10 summarizes the algorithm, and figure 11 traces its execution for a few steps, following the "inclusion" branches.

3.2.3 Invisible Vertices

In either of these cases invisible vertices must be handled as a special case. Normally, a vertex is allowed to have only one match. For the special symbol (invisible), however, more than one match must be allowed because there may be more than one invisible vertex. On the other hand, a visible vertex only has a limited number of invisible neighbors. There is a special check that, if X matches X' and X has some number i of invisible neighbors, only i of the neighbors of X' may match (invisible). Without this check we would generate matches such as A matches L and everything else matches (invisible). Yet we know that, given A matching L, one of L's neighbors must match B_r one must match H, and only one of L's neighbors can match (invisible).

3.3 Comments and Extensions

There may be more than one possible complete, consistent match for a pair of objects One cube, for example, can match another in 24 legitimate ways One vertex can match any of eight others. Having fixed that. the cube can be rotated to 3 different positions. These matches are all topologicaly valid.

```
Procedure include-node (P, I, J)
  n = first node in P
  P = (P - n) ∩ links of n
  I = I + n
  search (P, I, J)

Procedure exclude-node (P, I, J)
  n = first node in P
  P = P - n
  search (P, I, J)

Procedure search (P, I, J)
  for each junction j in J
    if j not found in P ∪ I
      return (fail)
  if P empty then
    print I
    return (success)
  include-node (P, I, J)
  exclude-node (P, I, J)

Procedure start-search
  P = correspondence graph
  I = empty
  J = junctions from image 1 ∪ junctions from image 2
  search (P, I, J)

In each procedure, the declarations are
  set-of-nodes P
  set-of-nodes I
  set-of-junctions J
  node n
  junction j
```

Figure 10: Summary of Search Algorithm

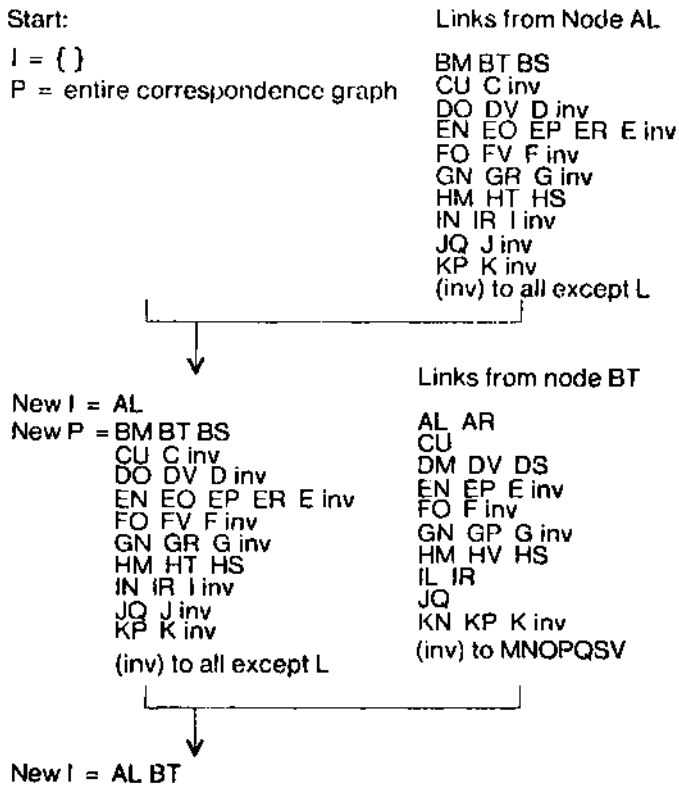


Figure 11: First Steps of Search

The algorithm will also produce 24 incorrect matches for the cube, each the mirror image of a correct match. Since the only information used is connectivity, and not sidedness, any object with a plane of topological symmetry will have spurious mirror image matches. These could be removed by using quantitative information. We currently use an interactive program.

The constraints used make no assumptions about all parts of an object being visible. So without modification the routines can be used to match part of an object with an object template or with another partial view of the object. There are two caveats. First, incomplete images are usually harder to label with Huffman Clowes line labels. This may cause ambiguities in the matching (see below). Second, incomplete images have more invisible vertices and more invisible edges. This allows lots of matches where only a few vertices are matched and all the rest match (invisible). These are all legitimate possibilities, and examples can Deconstructed in which they are the only correct matches. Put if it is known in advance that most of the vertices in the incomplete image are visible in the other image, either the search can return matches with the fewest "invisibles" matches or an upper bound can be set on the number of invisibles.

It is not always possible to assign unique Huffman'Clowes labels to an image. If more than one possible labeling exists, the correspondence process can be run separately with each possible labeling. Then incorrect labelings may have no possible match, or only the match all(invisible). So correspondence *can* be used to refire line labels if multiple views arc available.

Trihedrality is at the basis of the assumptions about invisible lines. Extending the algorithm to deal with non-trihedral objects would be possible, but the number of complete, correct matches possible for a pair of images would increase unmanageably unless other constraints were added. Additional information could include, for example, knowing which vertices were non-trihedral, or knowing how many non-trihedral vertices there were.

If quantitative data is available, it can be used in several ways. Conceptually, the matching process could be run just using topological data, and the quantitative information used in a post-processing step to find the most likely match or matches. It would be equivalent, but probably more efficient, to include all the data from the beginning. One scenario is that vertex positions are known to within some distance, or to within so many pixels of an epipolar line. Then the possible matches are constrained, and many fev/er nodes would have to be created and searched. If positions are known precisely, they can be used to check the validity of the match. Ullman [7], Aggarwal and his group [1,8], Ganapathy [4], and Lawton 16], have all worked with various numbers of points visible in two or more images, and the available constraints. Asada, Yachida, and Tsuji present a partial transition table use it with quantitative data in [2].

3.4 Examples

We have implemented and tested the correspondence algorithm. Here we show the results for a few simple figures. In each case, the images were labeled (with Huffman-Clowes line labels) by hand. Each of these simple images has a plane of symmetry, so the mirror-image program was run to remove reflections from the output. All the resulting matches are topologically correct.

For the L blocks of figure 7, the program found five possible matches (see table 2). The first match is the obvious one, which can be visualized as rotating the block approximately thirty degrees about a vertical axis. The second match is the same as the first,

1. AL BT CU DV EN FO G(invisible) HS IR JQ KP (invisible)M

2. AL BT CU DV EN FO G(invisible) HS IR JQ KP (invisible)M (invisible)(invisible)

3. AO BP CQ DR ES FL GM HN IV JU KT (invisible)(invisible)

4. A(invisible) B(invisible) C(invisible) D(invisible) E(invisible) F(invisible) G(invisible) H(invisible) I(invisible) J(invisible) K(invisible) (invisible)L (invisible)M (invisible)N (invisible)O (invisible)P (invisible)Q (invisible)R (invisible)S (invisible)T (invisible)U (invisible)V

5. A(invisible) B(invisible) C(invisible) D(invisible) E(invisible) F(invisible) G(invisible) H(invisible) I(invisible) J(invisible) K(invisible) (invisible)L (invisible)M (invisible)N (invisible)O (invisible)P (invisible)Q (invisible)R (invisible)S (invisible)T (invisible)U (invisible)V (invisible)(invisible)

Table 2: Matches for Figure 7

except that it presumes that there are additional invisible vertices seen in neither view. The third match can be thought of as tipping the block onto its back, so that the horizontal rjart of the L becomes the vertical part and vice versa, then rotating it some 60 degrees. The last two matches have no vertices visible in both images. The difference between these two is that the last one allows for vertices that are not visible in either image.

Figure 12 shows the same L-shaped blocks as figure 7, but with only part of the left-hand block visible. The additional constraint was given that there was a total of 6 invisible vertices. This eliminated matches in which one or two vertices from the partial block matched visible junctions in the right hand image and all other vertices matched (invisible). The first two matches correspond to match 1 in table 2, and the second two to match 3. The difference between matches 1 and 2 is in the match for H Without quantitative information, it is impossible to tell whether H matches S (as in match 2) or T (as in match 1). The same holds for matches 3 and 4.

Figure 13 shows a block with a corner cut out. It was matched against itself, with the extra constraint given to the program that there were no invisible type VII vertices. This is a strong constraint, since it forces M to match itself. The results are shown in table 4.

Finally, in figure 14 each image contains two objects, the L block and the block with the corner cut out. The program was told that there were no more than seven invisible vertices. It generated 18 matches: the three for the cutout block, times three for the L blocks (the last two L-block matches were eliminated because of the limit on invisible vertices), times a factor of two for different interpretations of the hidden part of the cutout block in the left image. There were three incorrect mirror images for each correct match, since the L block and the cutout block each have their own plane of symmetry.

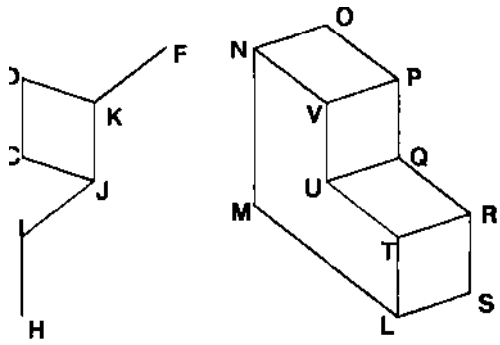


Figure 12: Part of an L block

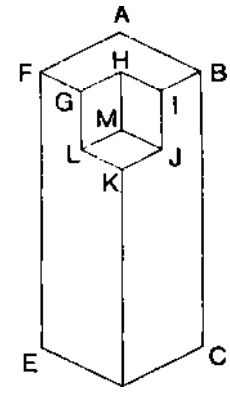
1.L(inv) M(inv) N(inv) OF PK OJ RI S(inv) TH UC VD (inv)(inv)

2. L(inv) M(inv) N(inv) OF PK OJ RI SH T(inv) UC VD (inv)(inv)

3 LF M(inv) N(inv) O(inv) PH QC RD S(inv) TK UJ VI (inv)(inv)

4. LF M(inv) NH O(inv) P(inv) OC RD S(inv) TK UJ VI (inv)(inv)

Table 3: Matches for Figure 12



Figu re 1 3: Block With One Corner Cut Out

1Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk LI Mm (invisible)(invisible)

2. Ac Bd Ce Df Ea Fb Gi Hj Ik Jl Kg Lh Mm (invisible)(invisible)

3. Ae Bf Ca Cb Ec Fd Gk Hl Ig Jh Ki Lj Mm (invisible)(invisible)

Table 4: Matches for Figure 13

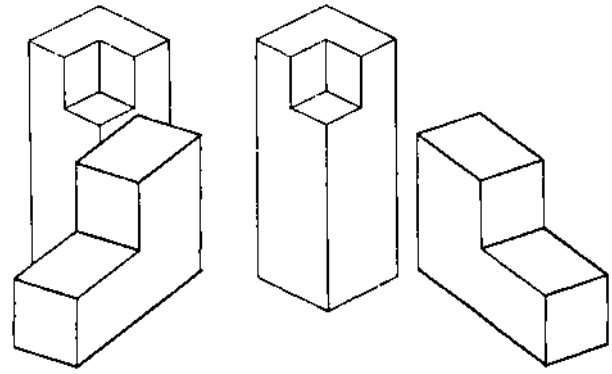


Figure 14: Two Objects

References

1. J. K. Aggarwal and W. N. Martin. Analyzing Dynamic Scenes Containing Multiple Moving Objects. In *image Sequence Analysis,* T. S. Huang, Ed.SpringerVerlag, 1981.

2. M. Asada, M. Yachida, and S. Tsuji. Three Dimensional Motion Interpretation for the Sequence of Line Drawings. Fifth International Joint Conference on Pattern Recognition, IEEE, 1980.

3. M. B. Clowes. "On Seeing Things." *Artificial Intelligence 2* (1971).

4. Sundaram Ganapathy. *Reconstruction of Scenes Containing Polyhedra From Stereo Pair of Views*. Ph.D. Th., Stanford University, January 1976.

5. D.A.Huffman. Impossible Objects as Nonsense Sentences. In *Machine Intelligence* 6, B. Meltzer and D. Michie, Ed..American Elsevier, 1971.

6. Daryl T. Lawton. Constraint-Based Inference From Image Motion. Proceedings of the AAAI, AAAI, 1980.

7. Shimon Ullman. *The Interpretation of Visual Motion.* The MIT Press, Cambridge. Massachusetts, and London, England, 1979.

8. Jon A. Webb and J. K. Aggarwal. Structure from Motion of Rigid and Jointed Objects. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 1981.

Acknowledgements

Many people, especially members of the CMU Image Understanding group, provided helpful comments and discussions. Thanks to Marty Herman, Rich Korf, Leslie Thorpe, and Jon Webb. Special thanks to Takeo Kanade, the IUS group leader, for encouragement and advice.