WHAT CORNERS LOOK LIKE

VISION FLASH 13

by

Mark Dowson

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Vision Group

June 1971

ABSTRACT

An algorithm is presented which provides a way of telling what a given trihedral corner will look like if viewed from a particular angle. The resulting picture is a junction of two or more lines each labelled according to Huffman's convention. Possible extensions of the algorithm are discussed.

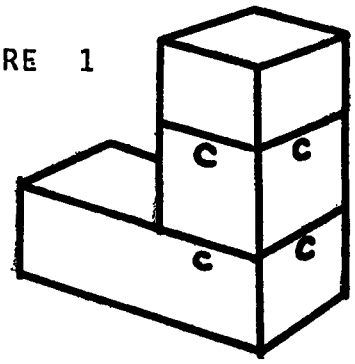
Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported by the Advanced Research Projects Agency of the Department of Defense, and was monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0002.

What Corners Look Like

Clowes' picture parsing algorithm (Clowes 1971) and Huffman's decision procedure both contain a table which exhibits a set of possible interpretations, as scene corners, of the various kinds of two and three line picture junction (see fig.2). The set of interpretations given is exhaustive under the following constraints: the corners are single trihedral vertices of opaque polyhedra viewed from a 'generalised position' (see Huffman, 1970) and there are no 'cracks' (see fig. 1)

FIGURE 1

'Cracks'



How are these interpretations arrived at? Essentially by looking at all the possible varieties of trihedral corner from each posssible position. A trihedral corner has three edges which associate its three faces; each of these edges may be convex (VX) or concave (CV) thus there are four distinct kinds of trihedral corner with various combinations of edge type. See table 1.

Huffman has observed that, as three intersecting planes, whether mutually orthogonal or not, divide space into eight octants, the four types of vertex can be further characterised by how many octants of space around the vertex are occupied by solid material.

A vertex can be viewed from any one of the octants not so occupied and all views of a particular vertex from any point within a given unoccupied octant will be essentially similar. Table 1 shows the number of views of each vertex.

TABLE 1

Vertex Type	Edge types at vertex	No. of occupied octants	No. of viewpoints
I	VX VX VX	1	7
II	VX VX CV	3	5
III	VX CV CV	5	3
IV	CV CV CV	7	1

The total number of different views is thus sixteen.

Some of these sixteen views are the same and, eliminating duplications, we are left with the twelve views shown in fig. 2. The algorithm used to obtain the junction labellings of fig. 2. - look at all the corner types from all possible positions and draw them - is unsatisfactory for two reasons. First, it is difficult to implement as a program and second, it provides little insight into how the various features of solid vertices interact to yield pictures of them, insight which is sorely needed if we are to extend picture parsing programs to handle less restricted situations. The next section sketches an algorithm for generating junction labellings which does not suffer from these defects.

We can now determine which faces of a vertex are visible from a particular unoccupied octant the 'viewpoint':-

Consider a solid octant , called SOLID, and an empty octant, called VIEWPOINT. The X face of SOLID cannot be seen from VIEWPOINT if SOLID and VIEWPOINT are on the same side of the X plane. Thus a neccessary condition (a) for the X face of SOLID to be visible is that the X components of the vectors defining SOLID and VIEWPOINT be different.

The other condition (b) which, with (a) is neccessary and sufficient, is that the octant sharing the X face of SOLID be empty. Corresponding rules hold for the Y and Z faces.

The analagous rules for edge visibility are:

The X edge of SOLID can only be obscured by SOLID itself if VIEWPOINT is the octant which shares its X face with SOLID.

Thus VISCONDITION:

The X edge of SOLID is unobscured by SOLID unless the vectors defining SOLID and VIEWPOINT differ only in their X components. The neccessary and sufficient condition (c) for the X edge of SOLID to be visible from VIEWPOINT is that VISCONDITION be satisfied for all occupied octants which share the X edge in question. Corresponding rules hold for the Y and Z edges.

Conditions (b) and (c) require the determination of 'equivalent faces' and 'equivalent edges' (that is: edges and faces which belong to more than one solid octant) These are easy to compute from the pairwise 'differences' between vectors identifying solid octants.

Edges satisfying VISCONDITION are -

X,Y and Z edges of A

X,Y and Z edges of B

Y and Z edges of C

The X edge of C is the only hidden edge but it is also the X edge of octants A and B so ,by (c) they are hidden too. This leaves,as visible edges -

Y edge of A Z edge of A

Y edge of B Z edge of B

Y edge of C Z edge of C

Note that as the Y edge of B is the Y edge of C and the Z edge of A is the Z edge of B there are actually only four different edges visible.

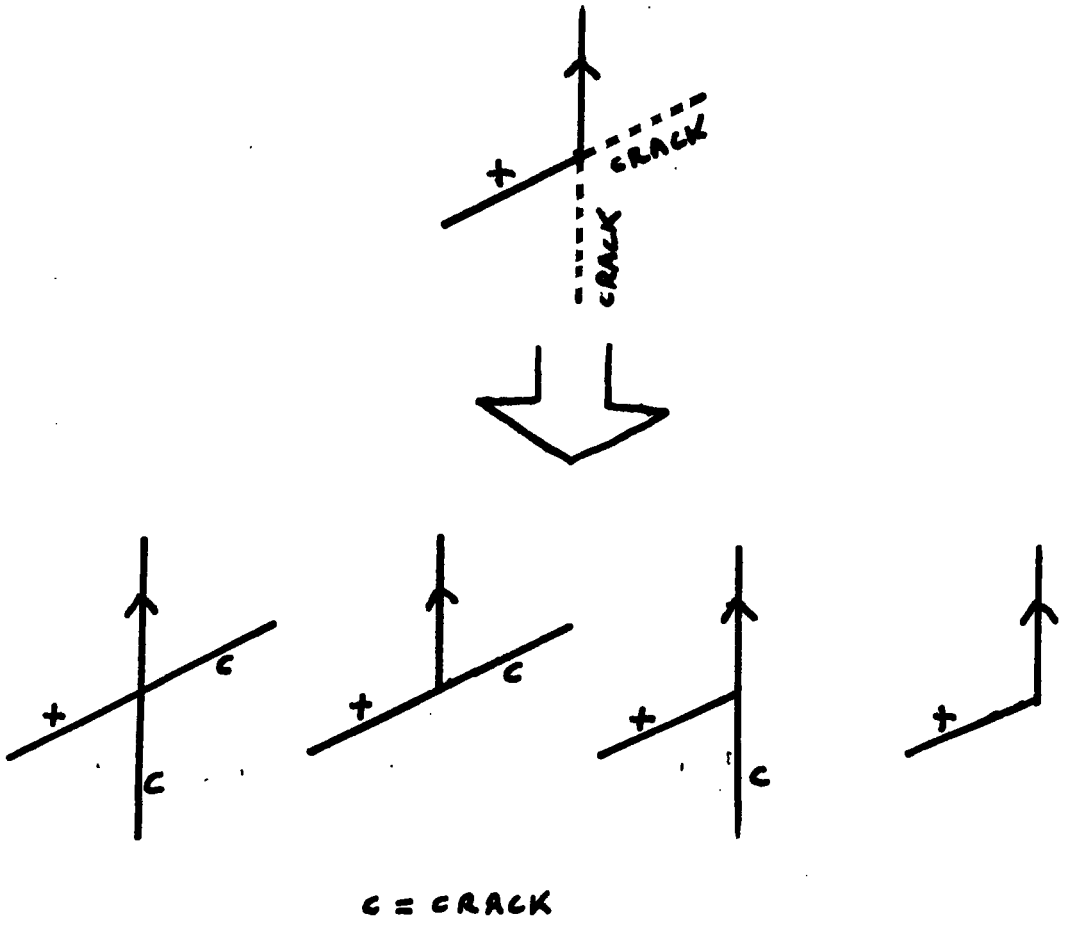
We would now like to know which Huffman labels to assign to the visible edges. The rules are as follows:

If a visible edge belongs to only one octant (as the Y edge of A and the Z edge of C above) it takes either a '+' or an 'arrow' label ;the '+' label is chosen if both surfaces adjoining the edge are visible. (a simple procedure gives the direction of the arrow)

If the visible edge belongs to two solid occupied octants it is marked as a 'crack' ; if it belongs to three solid octants it takes a '-'. Fig. 4 shows what this procedure yields applied to the fig. 3 example.

The labeling arrived at so far includes lines labelled as cracks in plane surfaces. Since both the Huffman and Clowes algorithms assume that all plane surfaces are continuous and featureless we may, to get a labelling given in Figure 2, 'merge' the octants separated by cracks and thus erase the lines at the junction marked by 'crack'. If we wish to extend the Figure 2 set of labellings to include 'crack' lines instead of simply erasing cracks we add to the algorithm above an 'optional' crack erasing rule. Our example now yields the set of junction labellings shown in Figure 6.

FIGURE 6



C = CRACK

A picture parsing algorithm is useful only insofar as it yields a useful description of the scene the picture represents. Winston, in discussing the picture of fig. 8 , has pointed out that "...the cube seems left of the arch's entrance even though all its vertices are clearly right of all the arch's vertices." Now this statement needs to be sharpened to include a 'frame of reference' and so we might say that "From a viewpoint on the road, the cube is to the left of the arch." What we want to do is to transform the viewpoint of the scene from that of the picture to another, hypothetical but well specified, position.

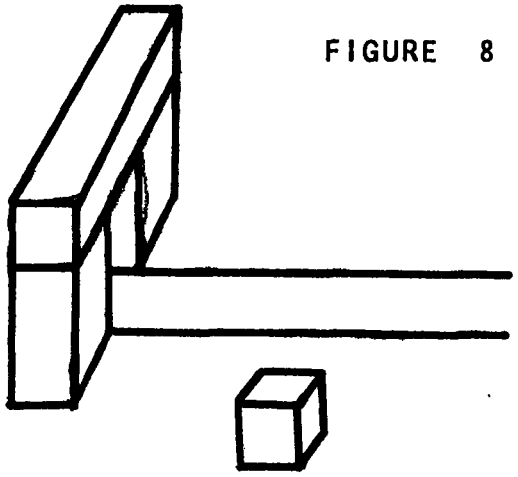


FIGURE 8

We now begin to see how to do this transformation of viewpoint. If we know, by some previous analysis, what kind of a vertex a particular picture junction depicts, we can use an adaption of the algorithm described above to determine which octant it is viewed from. Conversely, we can discover what it would look like if viewed from a different octant.

REFERENCES

Clowes M.B. On Seeing Things
A.I. Journal, Spring 1971.

Huffman D. A. Impossible objects as Nonsense Sentences
Machine Intelligence 6 , 1970

Winston P. H. Learning Structural Descriptions from
Examples. MAC TR-76 1970.