Detection of Outlines and their Use in Hidden-Line Elimination

Kenji Kira*

Introduction

Lately we have developed an experimental computer animation system IMAGES[3](Interactive Montage and Animation Generating System), which generates a series of animated pictures in three dimensional space and displays on a CRT. Each object in the system is a closed polyhedron surrounded with several polygons. A polyhedron may be concave and may have some holes. In this system we have to calculate not only hidden-line-eliminated perspective views of animated polyhedra but also all of their outlines, with which several perspective views can be superimposed onto a background view if wanted.

In this paper, the author desclibes an effective algorithm for outline detection and hidden line elimination. In this algorithm, the detection of outlines of three dimensional objects is regarded as an integral step in the hidden line elimination procedure and the obtained outline data are made full use of in the following steps of hidden line elimination. A new concept of "vertex condition in visibility" is also proposed, which is usefull for checking up the visibility of the edge segments connecting to a vertex quickly.

Input Data

The following three kinds of data are sufficient for outline detection as well as hidden line elimination. Their contents are as follows.

Vertex Data;	three dimensional coordinates Vi(Xi,Yi,Zi).
Face Data ;	a string of vertex numbers travelling clockwise around the face.
Edge Data ;	a couple of vertex numbers, a couple of face numbers intersecting from both sides, and a flag that indicates the edge to be convex or concave.

But the edge data can be made automaticaly from the other two kinds of data, so the vertex data and the face data are enough as actual input data.

Program Flow

The program flow is shown in Fig.1 and the summaries of these steps are as follows.

Initial Step;	The edge data are developed from the vertex data and the face data.
Step 1 ;	All the vertex coordinates in 3D space are calculated according to the specified motions, and then all the face vectors are calculated to classify the faces into front faces and back faces.
Step 2 ;	All the edges are classified into four types(H1,H2,H3,H4) as P.P.Loutrel[2]

This paper first appeared in Japanese in Joho — Shori (Journal of the Information Processing Society of Japan), Vol. 14, No. 3 (1973), pp. 188 ~ 195.

* NHK Technical Research Laboratories

suggested, and potentially visible edges (H3,H4) are sellected.

Step 3 ; H3 edges' mutual intersections are examined in the perspective view.

Step 4 ; The outlines are detected from the results of Step 3.

Step 5 ; The intersections between H3 edges and H4 edges are examined.

Step 6 ; All the segments which are not evident yet to be visible or invisible, are finally examined, and the hidden line elimination is completed.

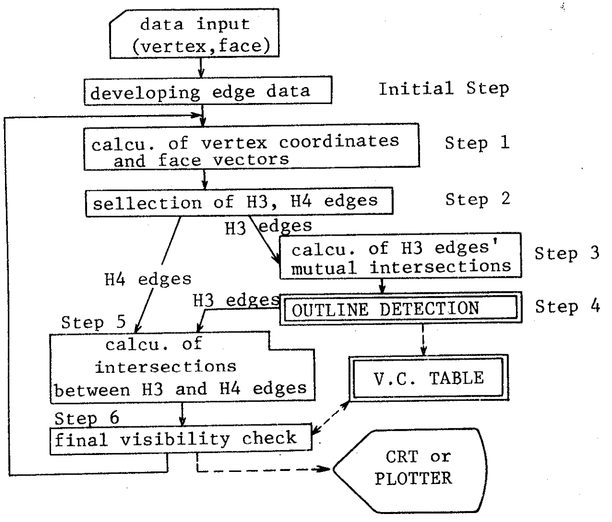

Fig.1 Program Flow

In the program flow, the most important parts are the routine of outline detection and the table of "vertex condition in visibility" (V.C.). With these availabilities the processing of Step 5 and Step 6 (especially the latter) has become much faster.

The points are as follows.

(1) As the outlines are composed of H3 edge segments only, the outlines are detected after the calculation of H3 edges' mutual intersections. (Step 4)

(2) As the outline-composing segments cannot be boundaries where other edges are partially hidden, the calculation of intersections between H3 edges, which compose outlines along whole length, and all H4 edges can be omitted. (in Step 5)

(3) As the outline-composing segments are naturally visible, it is no longer necessary to examine their visibility. (in Step 6)

(4) After the outline detection, V.C. of all the vertices through which the detected outlines pass are set visible. And in Step 6, the visibility checking of the segments connecting to a vertex whose V.C. is already set can be accomplished only by reffering V.C. table. Only in the case V.C. is not set yet, the check is made by calculation, and the result (visible or invisible) is set in the table for the following visibility checking of the other segments.

Vertex Condition in Visibility (V.C.)

Generally all the segments of H3 and H4 edges connecting to the same vertex are in the same condition in visibility. Therefore, if one of them is known to be visible (or invisible), the remainders can also be judged to be visible (or invisible). In this paper we name the visible or invisible condition of each vertex "vertex condition in visibility" (V.C.). But there exist two exceptional types of vertices whose V.C. must not be used, and we call them "peculiar vertices". They require good attentions and rather complicated processing in the outline detection procedure.

The contents of V.C. table are as follows.

1. unknown yet (remaining in initial condition)
2. visible (every segment connecting to the vertex can be judged to be visible.)
3. invisible (every segment connecting to it can be judged to be invisible.)
4. inavailable (peculiar vertex)

Characteristics of Outlines

Some characteristics of outlines, which are usefull for the outline detection procedure, are derived from the example in Fig.2. The lines ⓐ and ⓑ are the outlines of the three objects. In Fig.2 (B), only H3 edges are indicated as vectors. The direction of each vector is defined as the vector looks the front face in right side.

(1) All the segments composing outlines are on H3 edges only.

(2) The tips of these segments are on the vertices or H3 edges' mutual intersections. It is important that the intersections, hiding other edges, are unessential for hidden line elimination but essential for outline detection.

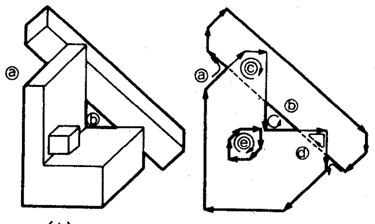

Fig.2 Detection of Outlines

(3) If there is no face surrounding a segment on the perspective view plane, that is an outline segment. (necessary and sufficient condition to be outline segment)

(4) The closed circuit obtained by the rule, starting from one of the outline segments, then travelling in the vector direction and changing to the new vector at the vertices or intersections, is certainly outline.

Preparations for Outline Detection

It is necessary to make some preparations for outline detection.

(1) Table of H3 edges' sequence

This table is made by examining the sequence of H3 edges sellected in Step 3. In the outline detection procedure, the sellection of the following segment via vertex is simply accomplished by reffering this table.

(2) Table of H3 edges' mutual intersecions

The table such as list 1 is made in the process of the calculation of H3 edges' mutual intersections (Step 4). This example is related to the illustration in Fig.3. The contents of the table are as follows.

1,2; coordinates (X,Y) of the intersection on the perspective view plane.

3* ; the partner edge number of the intersection.

4* ; the intersection number with which the intersection on the partner edge is found out.

5 ; the visibility condition in both sides. "0" indicates evidently inbisible.

- $1:1^*$; the partner edge is partially hidden.
- 1:0 ; end point side of the own edge is partially hidden.
- 0:1 ; start point side of the own edge is partially hidden.

Among these contents, the number marked with * indicates that they are necessary only for outline detection but not for hidden line elimination.

edge i

1	----	Xp	Xq	-----
2	----	Yp	Yq	----
3	----	j	k	----
4	----	Np	Nq	-----
5	----	1:1	1:0	----

edge j

---	Xp	---
---	Yp	---
---	i	---
---	Np	---
---	1:0	---

edge k

---	Xq	---
---	Yq	---
---	i	---
---	Nq	---
---	1:1	---

List 1 Intersection Table

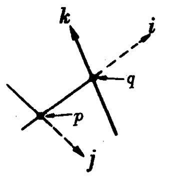

Fig.3 Intersections

In the table of H3 edges' mutual intersections, the segments, into which H3 edges are devided at the intersections, are the "unit segments" which have possibilities to compose the outlines.

Algorithm for Outline Detection

The outline segments can be picked up by examining whether each segment satisfies the necessary and sufficient condition to be outline segment. But this way is not so efficient in processing time, so in my algorithm all the closed circuits having possibilities to be outlines, are picked up first in the following manner.

(1) An optional segment remaining unsellected is sellected as a starting segment.

(2) The next segment sellection is made in the manner that if the former ends on a vertex, the segment starting from the vertex is sellected by reffering the table of H3 edges' sequence, and if the former ends on a intersection, the segment starting there is sellected by reffering the table of H3 edges' mutual intersections.

(3) If the starting segment is re-sellected, it is known that a closed circuit has been obtained. But if the circuit contains even partially the evidently invisible segments, it is omitted as it can not be outline. (such as ⓒ and ⓓ in Fig.2(B))

Like this way, some closed circuits, we call them potential outlines, can be detected. (such as ⓐ , ⓑ and ⓔ in Fig.2(B)) The potential outlines can not always be outlines.

The outlines are finally picked up from the potential outlines in the following way. An optional segment is sellected from each potential outline, and it is examined whether the segment satisfies the necessary and sufficient condition to be outline segment. If satisfies, the potential outline with the segment can be judged to be outline. In Fig.2, only ⓐ and ⓑ are judged to be outlines.

By the way the outline (such as ⓑ in Fig.2(B)) travelling arround in counter-clockwise direction is an inner outline (hole).

Peculiar Vertices

Outlines formed with "rather simple" objects such as shown in Fig.2, are rather simply detected by the predesclibed algorithm. The term "rather simple" means the number of H3 edge vectors connecting to a vertex in the perspective view is at most two, one coming in and the other going out. But there are some "rather complex" objects which have some peculiar vertices. At the peculiar vertices more than two H3 edge vectors start or end together. The outline detection of these objects requires exceptional treatments. The peculiar vertices can be classified into two types.

(i) A type peculiar vertex; vertex where more than two H3 edge vectors end.

Vertex p in Fig.4(b) is a simple example of this type. In the detection of the potential outlines, if the segment d (or e) is sellected as the starting segment, the circuit goes d , e, a, b, c and can not return to the starting segment, so the detection of the potential outlines ends in fail.

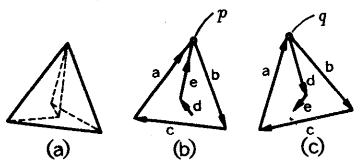

Fig.4 Peculiar Vertices

An adequate treatment of A type vertex is as follows.

If the segment which ends on this type vertex is sellected, the check whether it really composes outline is done and if satisfied, the next segment is sellected. If not satisfied, the circuit in process can not be outline, so from the new segment remaining unsellected, the new circuit detection is started again.

(ii) B type peculiar vertex; vertex where more than two H3 edges start.

Fig.4(c) is the reverse of Fig.4(b), and the vertex q is an example. In this figure, although two segments b and d follow the segment a, it is necessary to sellect the segment b as the next of the segment a.

An adequate treatment of B type vertex is as follows.

At this typed vertex the most counterclockwise-directed segment against the former segment is sellected.

These peculiar vertices can simply be found out in the establishing routine of the table of H3 edges' sequence.

By the way, it is evident in Fig.4 that the all the H3 and H4 edge segments connecting to a peculiar vertex are not always in the same condition in visibility. So V.C. of the peculiar vertex can not be used for checking up the visibilities of the segments connecting to the vertex.

Availabilities of Outlines and Vertex Condition in Visibility

In Fig.5 the broken lines indicate the segments which can be judged to be invisible from the result of calculations of H3 edges' mutual intersections, and the thick lines indicate the detected outlines and they are evidently visible.

(1) All the V.C. of the vertices marked with ◎ can be set visible, through which the outlines pass. (In this example no peculiar vertices exist.)

(2) Among sixteen H3 edges, nine edges which compose outlines along the whole length, can not intersect with every H4 edges, so their calculations in Step 5 can be omitted.

(3) On the H3 edges (a, b, and c), tip points of the outline-composing parts(thick lines) are on the intersections where other H3 edges are partially hidden, so they can be judged to be visible along whole length. Consequently V.C.of the vertices marked with ○ can be set visible.

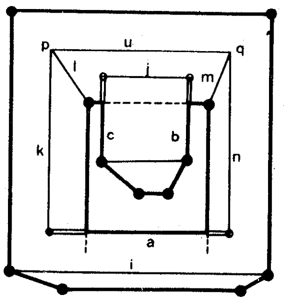

Fig.5 Availability of Outline and V.C.

(4) The segments ($i \sim n$) can be judged to be visible only by reffering V.C. table. Therefore only one segment u in this example requires the calculations for visibility checking. But if at least one segment among the four ($k \sim n$) has been examined before the segment u, V.C. of vertex (p or q) can be used for the segment u.

In this example, although this is a convenient one, the visibility checking in Step 6 are almost accomplished only by setting and reffering the V.C. table.

Reduction of Computation Time

List 2 shows the improvement on computation time. Each time in this list is an average computation time, rotating the sixteen faced objects shown in Fig.6 twelve times by thirty degrees. In this example the coputation time of Step 6 (final visibility check of the segments) has reduced beyond expectation to less than 10%, and as the result the total time including outline detection has reduced to about one third compared with that by the ordinary algorithm only for hidden line elimination.

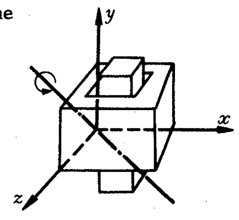

Fig.6 Example of polyhedra

	Total Time	Step 6
Ordinary Algorithm	3 sec.	2 sec.
My Algorithm	0.96sec. (0.2sec.)	0.1∼0.2 sec.

( )...time for outline detection

List 2 Reduction of computation time

Time Estimation and Examples

The graph in Fig.7 shows the relations between the number of the faces surrounding the objects and the computation time of hidden line elimination including outline detection by my algorithm. In the graph, ○ shows the results by FORTRAN and ● by ASSEMBLER using IBM 360/40. Several examples are shown in Fig.8.

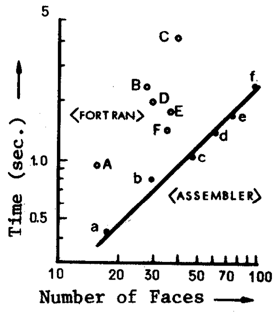

Fig.7 Computation Time

Conclusion

Generally in the procedure of hidden line elimination by a digital computer, the visibility check of the segments, devided from the edges at the intersections on the perspective view plane, takes considerable time. This check is accomplished by examining whether each segment is surrounded by any faces on the perspective view plane, and if so, which is nearer to the view point in three dimensional space. This calculation is rather complex and, what is more, the number of the combinations between the segments and the faces is considerably great. So how to reduce the number of the combinations directly leads to speeding up the hidden line elimination. In order to reduce them several efforts as follows have been tried.

(1) The number of faces to be taken into account for the check has been reduced by examining the normal vector of each face and excepting back faces.

(2) The number of edges to be examined on their mutual intersections has been reduced by classifying edges into four types and excepting evidently invisible edges.

(3) The number of segments to be checked in the final visibility check has been reduced by examining which side of the intersection is hidden in the calculation of intersections and excepting the evidently invisible segments.

In spite of these efforts, the final visibility check of the segments takes considerable time yet. (See the first line of List 2.) In the development of a new algorithm, I have considered how to speed up the hidden line elimination, especially the final visibility check, by using the outlines of which detection was necessary for the system. And I have been struck with the new concept "vertex condition in visibility", which makes the detected outlines more effective for the hidden line elimination. And the results seem to prove that my approach was not mistaken.

Although the necessity for outline detection is not so common, the algorithm can be applied in the problem of shadowing. On the other hand the concept "vertex condition in visibility" can be applied more broadly for speeding up the hidden line elimination.

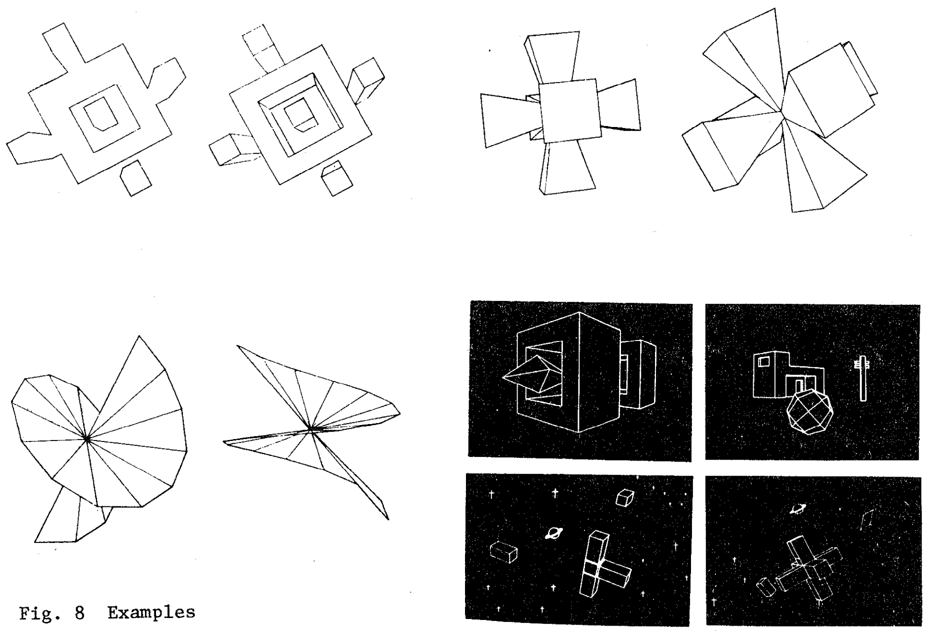

Fig. 8 Examples

References

1) I.E. Sutherland: Computer Graphics - Ten Unsolved Problems, Datamation. Vol.12, No.5, pp. 22 ～ 27 (May 1966)

2) P.P. Loutrel : A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra, IEEE Trans. Computers, Vol. C-19, pp. 205 ～ 213 (March 1970)

3) J. Kustuzawa, H. Yásuda, K. Kira and T. Omata : IMAGES - Interactive Montage and Animation Generating System, NHK Laboratories Note, No. 168 (August 1973)