An algorithm for the removal of hidden lines in 3D scenes

A. Ricci

CNEN Centro di Calcolo, Via Mazzini 2, Bologna, Italy

In the present paper a new algorithm is illustrated which removes hidden lines from a 3D scene comprising polyhedra, polygon-bounded planar surfaces and straight line segments.

(Received December 1970)

Introduction

In the field of computer graphics most efforts have been devoted, in the recent years, to develop and improve graphics systems. Nevertheless, a few problems of computational geometry have been thoroughly studied, due to their practical importance.

A problem of outstanding interest is that of the removal of hidden lines in projective representation of a 3D scene and many algorithms for its solution have been developed (see references). More properly, said algorithms generally relate to the removal of hidden surfaces, segments of straight lines not being admitted in the 3D scene unless they are edges of plane surfaces. This is a particularly serious limitation as far as engineering and architectural drawing is concerned, because it prevents easy representation of structural elements, for instance grids, pylons, etc., which are better represented by means of straight line segments.

In the present paper a new algorithm will be illustrated which removes hidden lines from a 3D scene comprising polyhedra, polygon-bounded planar surfaces and straight line segments. An example of application of the algorithm is shown in **Fig. 1.**

The basic idea of the algorithm is that in a line drawing the elements on which the removal procedure must be based should be lines and not surfaces. That means objects should be considered sets of edges, their ordering being immaterial. It should be possible to represent a real edge of an object by means of adjacent lines in order to permit a different definition of attributes for different parts of the real edge. The set of lines representing an object should be permitted to have as its elements not only lines belonging to real edges, but also lines lying on the surface of the object. In the following, lines belonging to the sets will be generally called edges.

1. The projection system and data structure

In the following the algorithm will be described supposing data for the 3D scene referred to a system $O(x, y, z)$ and the projection plane parallel to the plane $x = 0$, intersecting the positive part of the x-axis, with its own reference system $O'(x', y')$ such that for a point $P(P_x, P_y, P_z)$ its projected image $P'(P_{x'}, P_{y'})$ is given by $P_{x'} = P_y$ and $P_{y'} = P_z$. This gives a parallel projection with no lack of generality because this situation can always be obtained by a suitable sequence of rotations and translations of the 3D reference frame $O(x,y, z)$. The case of perspective projection will be later considered.

The data on which the algorithm acts comprise a list of straight line segments (here simply called segments), a list of edges and a list of objects. With the only exception illustrated in the following, all edges also appear in the segment list.

A segment or edge $\overline{P_0P_1}$ can be defined by

$$P(s) = P_0(1 - s) + P_1 s \qquad (1)$$

with $0 \leqslant s \leqslant 1$. For $-\infty \leqslant s \leqslant +\infty$, $P(s)$ is a point of the straight line on which the segment or edge lies, in the following respectively called segment line or edge line.

An object is a set of edges forming in the 3D space a convex polyhedron or a planar surface with a convex polygonal boundary. Nonconvex polyhedra and nonconvex bounded surfaces can be represented by means of adjacent convex objects, with the common edges required to be invisible not appearing in the segment list. The ordering of edges in the definition of an object is immaterial and an object is always considered opaque and not intersected by any other object or segment.

In parallel projection, the x-coordinate P_x of the point P in the system $O(x, y, z)$ is used to test if the point P is nearer than

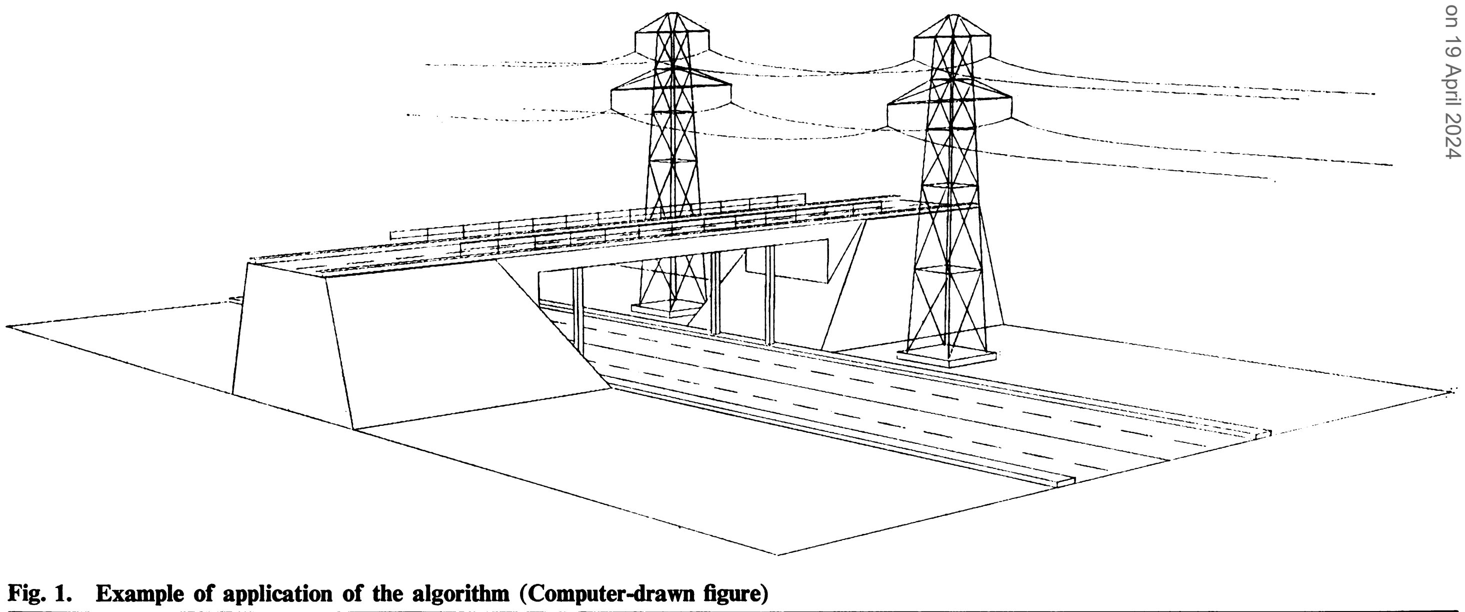

Fig. 1. Example of application of the algorithm (Computer-drawn figure)

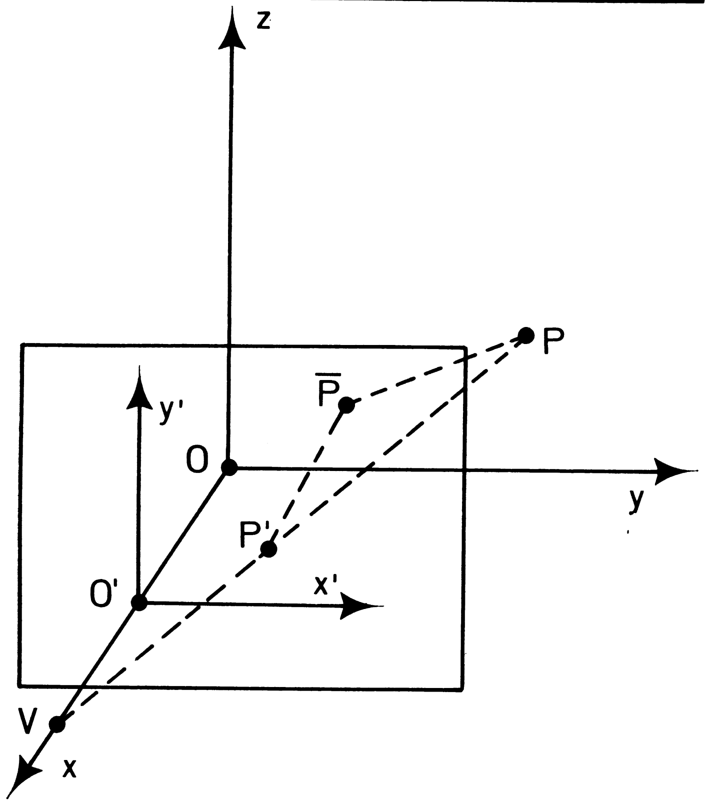

Fig. 2. 3D transformation for perspective view

other points to the projection plane. To apply the algorithm to perspective projection it is necessary to transform the coordinates in 3D space, from the system O(*x*, *y*, *z*) to a new system $\bar{O}(\bar{x}, \bar{y}, \bar{z})$, in such a way that a segment maps into a new segment. This may be done by means of the transformation

$$\bar{x} = a.x, \quad \bar{y} = a.y, \quad \bar{z} = a.z$$

with

$$a = (x_o - x_p)/(x_o - x) \tag{2}$$

where x_o is the *x*-coordinate of the vantage point *V*, always supposed to lie on the *x*-axis, and $x_p = 0$ is the equation of the projection plane in the O(*x*, *y*, *z*) system. The $\bar{x}$-coordinate keeps the decision capability to test the nearness to the projection plane the *x*-coordinate had before. In other words, the algorithm always operates a parallel projection, a perspective view being obtained by parallel projecting a previously transformed 3D scene (**Fig. 2**).

2. The removal algorithm

In the removal algorithm every object is compared with all segments in the scene. When a segment results totally or partially hidden by the object under consideration, the hidden part is removed by removing from the segment list the entire segment, if totally hidden, or substituting for it one or two new segments corresponding to the part or parts left visible. Since the objects are convex not more than two separate parts can be left visible.

After all objects have been compared with all segments in the scene, the procedure is completed and plotting or displaying the segment list will give a view with hidden lines removed. It is worth to be noted that the segment list is continuously updated during the procedure, with the result of avoiding any further comparison with segments already found hidden. In the computation an edge can disappear from, or be substituted for in, the segment list, but it will be integrally kept in the edge list to be used when the object it belongs to is compared with segments in the scene.

In the comparison between an object and a segment, the intersection between the image of every edge of the object and the image of the segment is computed.

Indicating, according to (1), with *P*(*s*), $-\infty \leqslant s \leqslant +\infty$, the segment line and with $Q_i(e)$, $0 \leqslant e \leqslant 1$, the *i*th edge of the object, in the adopted projection system the segment line image $P'(s)$ and the *i*th edge image $Q_i'(e)$ are obtained simply disregarding the *x*-coordinate. The intersection between the latter 2D lines can be individuated by the couple s_i, e_i satisfying the system

$$P'(s_i) = Q_i'(e_i) \tag{3}$$

If the 2 × 2 matrix of the system (3) is singular that means either the segment line and the edge are parallel or one, or both, of them is, or are, projected into a point. If $e_i < 0$, or $e_i > 1$, no intersection between the segment line image and the edge image exists. In all these cases the comparison with the *i*th edge is not continued, because either the information on the segment visibility can be completely obtained from the comparison with the other edges of the object or, when the segment line is projected into a point, the segment is considered invisible.

Since the algorithm always considers a parallel projection (as above said, for perspective projection the 3D scene must have been previously transformed), the quantity

$$d_i = P_x(s_i) - Q_{ix}(e_i) \tag{4}$$

where *x* denotes the *x*-coordinate, gives information on being the segment line, in correspondence of the intersection of its image, nearer to the projection plane than the edge or not.

Once the quantities s_i, e_i, d_i have been computed for all edge images of the object, since the latter is convex the potentially hidden part of the segment line is completely determined by the couple s_j, s_k, being s_j the minimum and s_k the maximum of the values s_i (**Fig. 3**).

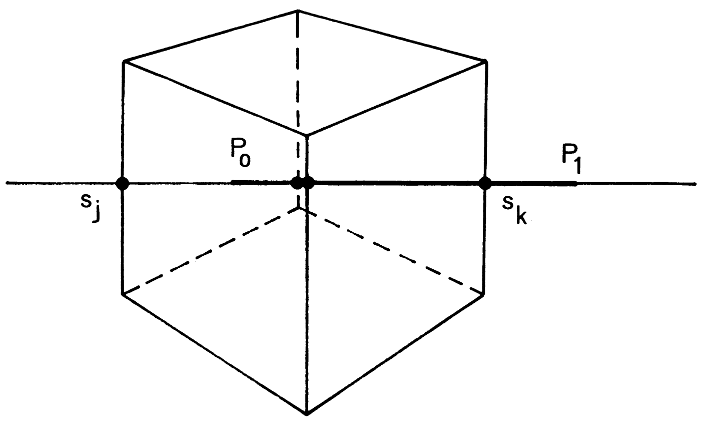

Fig. 3. Intersections on the projection plane

To find the potentially hidden part of the segment the values s_j and s_k are examined: if $s_k \leqslant 0$ or $s_j \geqslant 1$ the segment is entirely visible since its image does not intersect the object image, otherwise a potentially hidden part exists. This part is individuated by $\bar{s}_j$, $\bar{s}_k$ with

$$\bar{s}_j = \begin{cases} s_j \text{ if } 0 \leqslant s_j < 1 \\ 0 \text{ if } s_j < 0 \end{cases}$$
$$\bar{s}_k = \begin{cases} s_k \text{ if } 0 < s_k \leqslant 1 \\ 1 \text{ if } s_k > 1 \end{cases} \tag{5}$$

If $\bar{s}_j = \bar{s}_k$ the segment is obviously entirely visible; this can happen, for example, when the object is a planar surface and its image reduces to a line.

Considering now the line

$$L(v) = Q_j(e_j)(1 - v) + Q_k(e_k)v \tag{6}$$

it is possible to find a linear function *v*(*s*) such that the points *P*(*s*) and *L*(*v*(*s*)) lie on a line parallel to the *x*-axis. This permits the definition of a linear function *d*(*s*) such that

$$d(s_j) = d_j \qquad (7)$$
$$d(s_k) = d_k$$

from which new values

$$\bar{d}_j = d(\bar{s}_j) \qquad (8)$$
$$\bar{d}_k = d(\bar{s}_k)$$

can be obtained.

For $0 \leqslant v \leqslant 1$ the line (6) is inside the convex object or on its surface; since the segment cannot intersect the object, it also cannot intersect the line $L(v)$. Thus, if $\bar{d}_j$ and $\bar{d}_k$ are not negative and not both zero the segment is entirely visible, if $\bar{d}_j$ and $\bar{d}_k$ both are equal to zero, that means the line $L(v)$ is on the surface of the object, the segment line coincides with $L(v)$ and the segment is entirely visible only if none of the computed values d_i is negative. In all other cases the segment potentially hidden part is really hidden, the segment $\overline{P_0P_1}$ is removed from the segment list and substituted for by

$$\begin{array}{ll} \text{nothing} & \text{if } \bar{s}_j = 0 \text{ and } \bar{s}_k = 1, \\ \overline{P(\bar{s}_k)P_1} & \text{if } \bar{s}_j = 0, \\ \overline{P_0P(\bar{s}_j)} & \text{if } \bar{s}_k = 1, \\ \overline{P_0P(\bar{s}_j)} \text{ and } \overline{P(\bar{s}_k)P_1} & \text{if } 0 < \bar{s}_j < \bar{s}_k < 1. \end{array} \qquad (9)$$

In the comparison of an object with a segment, if the latter corresponds to an edge of the object the comparison procedure can be simplified because if at least one of the values d_i is negative the segment is totally hidden, otherwise it is entirely visible.

3. On the implementation of the algorithm

The data structure used to describe the 3D scene, namely the segment, edge, object lists, suggests the use of a set of functions for handling information items in the lists. The functions can be easily implemented in the form of a package to be used in a high level language program or through an interactive computer graphics system.

By the use of such a package the description of even very complicated scenes becomes relatively simple since subroutines can be written which describe objects or sets of objects, more generally whatever sets of edges or segments, to facilitate the description of more frequently used scene elements.

Functions to duplicate, rotate and translate elements should be provided in the package, together with functions for handling attentions when the package is used in conjunction with an interactive display unit.

The example of application of the algorithm shown in Fig. 1 has been realised using a prototype version of such a package, HLR for use in a FORTRAN environment.

With particular reference to the use of the above illustrated data structure in an application program, only few basic functions need to be provided for the description of the 3D scene. They must permit the definition of a current position in the 3D space and the addition of new items to the segment, edge and object lists. For instance, a possible set of FORTRAN subroutines could be

```
POS(A)
SEG(B)
OBJECT(NAME)
EDGE(B,L,NAME)
```

where A and B are point coordinates or pointers to the areas where they are actually stored.

POS(A) defines the point A as the current position, here indicated by C. SEG(B) adds the segment $\overline{CB}$ to the segment list. OBJECT(NAME) generates a new object in the object list and the variable NAME becomes a pointer to it. EDGE(B,L,NAME) adds $\overline{CB}$ to the edge list and relates the new edge to the object NAME, the value of L indicating if the edge is visible or not, namely if the edge has to be added to the segment list or not.

Conveniently, the subroutines POS,SEG and EDGE could have duplicates accepting incremental definition of points in the 3D space.

As far as computing time is concerned, in the previous algorithms it was generally proportional to n^2, where n is the number of elements in the 3D scene, with the exception of a few algorithms (Warnock, 1968, and Bouknight, 1970), particularly suitable for half-tone representation, in which the computing time is proportional to n.

In the present algorithm, due to the fact that the segment list is continuously updated during execution of the removal procedure, the computing time is in a certain sense unpredictable but normally substantially less than proportional to n^2.

References

Bouknight, W. J. (1970). A procedure for the generation of three dimensional half-toned computer graphics presentations, *Comm. of the ACM*, Vol. 13, pp. 527-536.

Encarnacao, J. L. (1970). A survey of and new solutions for the hidden-line problem, Symposium Interactive Computer Graphics, 26-28 October, 1970, Delft.

Galimberti, R., and Montanari, U. (1969). An algorithm for hidden line elimination, *Comm. of the ACM*, Vol. 12, pp. 206-211.

Kubert, B., Szabo, J., and Giulieri, S. (1968). The perspective representation of functions of two variables, *J. Assoc. Comp. Mach.*, Vol. 15, pp. 193-204.

Loutrel, P. (1970). A solution to the hidden-line problem for computer-drawn polyhedra, *IEEE Trans. on Comp.*, Vol. c-19, pp. 205-213.

Warnock, J. (1968). A hidden line algorithm for half tone picture representation, Tech. Rep. 4-5, Un. of Utah, Salt Lake City, May 1968.

Book review

Case Exercises in Operations Research, by M. J. C. Martin and R. A. Denison, 1971; 209 pages. (*John Wiley & Co.* £3·25)

It is important that students of Operational Research should be exposed to many different types of real life case studies. While most teachers can draw on their own experience to a certain extent, the number of suitable case studies within the experience of a teacher is usually limited. Teaching practical studies at second hand can be extremely difficult, because one normally has available only a final report on a project: in such a report many of the early problems will have been ignored. It is also difficult to separate the problem from the solution unless one has been intimately associated with the exercise under consideration. While one may invent situations for discussion in class, inventions usually lack somewhat in reality, and a need is felt for a library of case studies in which the problems are presented in such a way that the student can pause after the presentation to consider and evolve a method of solution.

The present book will be a valuable addition to the personal library of the operational research teacher or student. It describes fifteen situations which were tackled and solved in real life by operational research groups, although the solutions are not given with the body of this work; these are available to bona fide teachers on application. The teacher may therefore use the situations presented in the book as exercises to be set to a class of students, and to be discussed after the students have worked through them and formulated solution methods. The situations are clearly presented, and make challenging projects for either students or operational research workers desiring to broaden their experience. The separation of the solutions will force the student to make a serious personal attempt to solve the problems, although the industrious student may find some assistance from published papers describing some of the exercises, which have generally been disguised to a certain extent.

Anthony Wren (Leeds)