A Screen Subdivision Method for Half-Tone Representation of 3-D Objects Using Mini-Computers

TOMOYUKI NISHITA* and EIHACHIRO NAKAMAE**

A half-tone representation of three-dimensional objects is a useful tool for architectural, mechanical, and lighting designs. In many cases, however, a mini-computer is not available for such complex scenes consisting of many objects and/or light sources.

This paper describes a method suitable for a mini-computer on half-tone representation of complex scenes; a display method by means of subdividing a screen into several parts and processing each part by using only the minimum required data for it, is proposed. The proposed method is very useful not only for saving memory, but also for shortening computation time.

This paper discusses the methods for subdividing a screen and the procedures which extract objects existing in the region of a subdivided screen and objects casting shadows from the outside of that region onto the subdivided screen.

1. Introduction

The main criteria of evaluation of the half-tone representation of 3-D objects might be: realistic for rendering, saving memory, shortening computation time, and simplifying input data.

It is difficult to satisfy these criteria simultaneously because of conflicting interests among the first three criteria. The highly realistic shade images require handling many complex objects and displaying cast shadows[1]–[3]. Then, the necessary data and memory requirements become large and the computation time becomes long. These facts mean that for a mini-computer it is difficult to generate practical images.

This paper describes how to overcome these difficulties. A display method by means of subdividing a screen into several parts and of processing each part by using only the minimum required data for it is proposed. Concerning the subdivision of the field of view, the basic idea of the method is similar to that of the multi-screen of flight simulators[4], but the following points differ from it because of different use.

Flight simulators use a multi-screen in order to obtain an image corresponding with each window in a cockpit, and to obtain a wide field of view. The proposed method uses one screen instead of a multi-screen and this screen is divided into several parts depending on the complexity of the scene. Only the minimum required data for each subdivided screen have to be extracted from a disk, and then the hidden surface removal and the shadow detection are executed. For this purpose, the volumes formed by the field of view and light sources are defined, and they are used for extracting not only the polyhedra in the subdivided screen but also the polyhedra casting shadows onto them.

The advantages of the proposal are as follows:

(1) For a mini-computer with relatively small memory, extracting any needless objects must be avoided. Therefore, a method is proposed which efficiently extracts only indispensable data for each subdivided screen by using contour lines[3] of convex polyhedra.

(2) Complex objects and multi-light sources can be handled by a mini-computer because the screen is divided into several parts depending on the complexity of the scene. The division gives an additional advantage in some cases, that is, shortening the computing time of hidden surface removal and shadow detection, because this time is usually proportional to a square of the number of polyhedra(see Ref. 5). Furthermore, the modification of some partial areas in a scene including shadows becomes easy because of the ability of processing arbitrary parts as a subdivided screen. This process is especially useful for an animation dealing with moving objects in the scene.

(3) A high quality image without any distortion for a wide field of view can be realized by using a quasicylindrical projection.

In this paper, objects, details such as windows and doors on the objects depicting buildings, and light sources are treated with sets of convex polyhedra, convex polygons, and point and/or parallel light sources, respectively. The improved method of Ref. [3] is used for half-tone representation in this paper.

2. Method for Subdividing a Screen

The perspective surfaces (i.e. a projection plane) used here and the conception of the method of subdividing a screen are described.

*Faculty of Engineering, Fukuyama University.
**Faculty of Engineering, Hiroshima University.

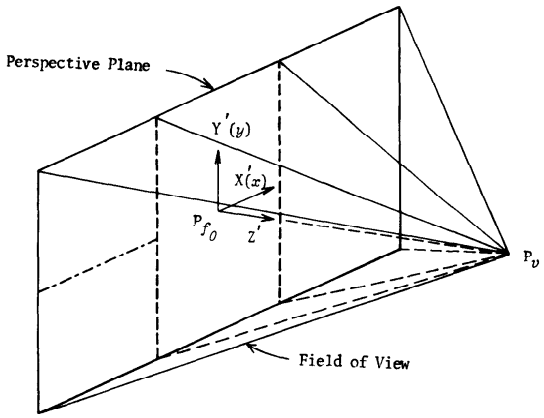

Fig. 1 Division of a projection plane.

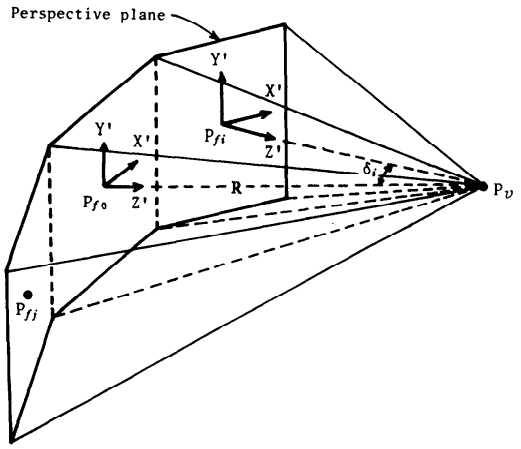

Fig. 2 Division of a cylindrical projection plane.

A perspective surface is classified into the following two types; for the standard horizontal view angle not exceeding fifty-five degrees, a plane is used which is vertically divided into several similar size rectangles (see Fig. 1), while for the wide field of view, a cylinder approximated by several rectangles (see Fig. 2) is applied*. When a subdivided screen still includes too many polyhedra, it is redivided into two parts with a horizontal line (see the chain line in Fig. 1).

As mentioned before, indispensable polyhedra for processing the subdivided screen are polyhedra in the subdivided view field and polyhedra casting shadows onto them. In order to put the polyhedra from a disk into the main memory efficiently, polyhedra are divided into groups and a Bounding Box is defined for each group.

[Def. 1] Group: A set of convex polyhedra. A user can make a set with some convex polyhedra which exist closely with each other or work as parts of specific articles such as desks and houses. This set is called a group and can be given a name.

*The coordinate systems in Fig. 1 and Fig. 2 are described in the Appendix.

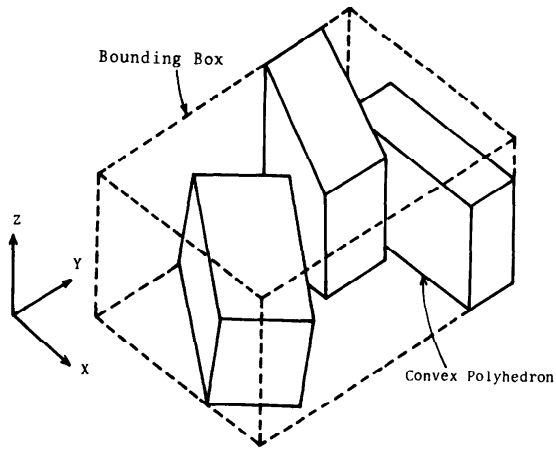

Fig. 3 Bounding Box for a group.

[Def. 2] Bounding Box (BB): If the maximum and minimum values concerning every axis of the coordinate system (X, Y, Z) are extracted from all vertices which form convex polyhedra belonging to a group, and two triples consisting of the set of the maximum or minimum values are made, a bounding box BB is defined as a rectangular parallelpiped surrounded by six planes which are parallel to the X-Y, Y-Z and Z-X planes (see Fig. 3).

The outline for subdividing a screen is shown in Fig. 4, where the object data and the bounding boxes are stored on a disk.

The additional explanation of Fig. 4 is as follows:

1) In step *1, the bounding boxes in the field of view are extracted from the disk, and the number of divisions is determined by the following method:
   As shown in Fig. 5, all bounding boxes are projected onto an X-Y plane. The bounding boxes overlapping or intersecting with the field of view (the shaded region in Fig. 5) are searched for, and the total number of polyhedra in the field of view is roughly calculated from those bounding boxes. The initial number of divisions in the total number of polyhedra divided by the allowable number of polyhedra in one subdivided screen.
2) The view field pyramid for each subdivided screen is formed, and polyhedra in the view field are extracted from the disk by using the relationship between the bounding boxes and the view field pyramid (see *2 in Fig. 4). When there are too many for main storage, the subdivided screen is redivided into two parts with a horizontal line.
3) For each subdivided screen, the volume formed by a light source and the space, including the polyhedra in the view field of the subdivided screen, is determined. The polyhedra casting

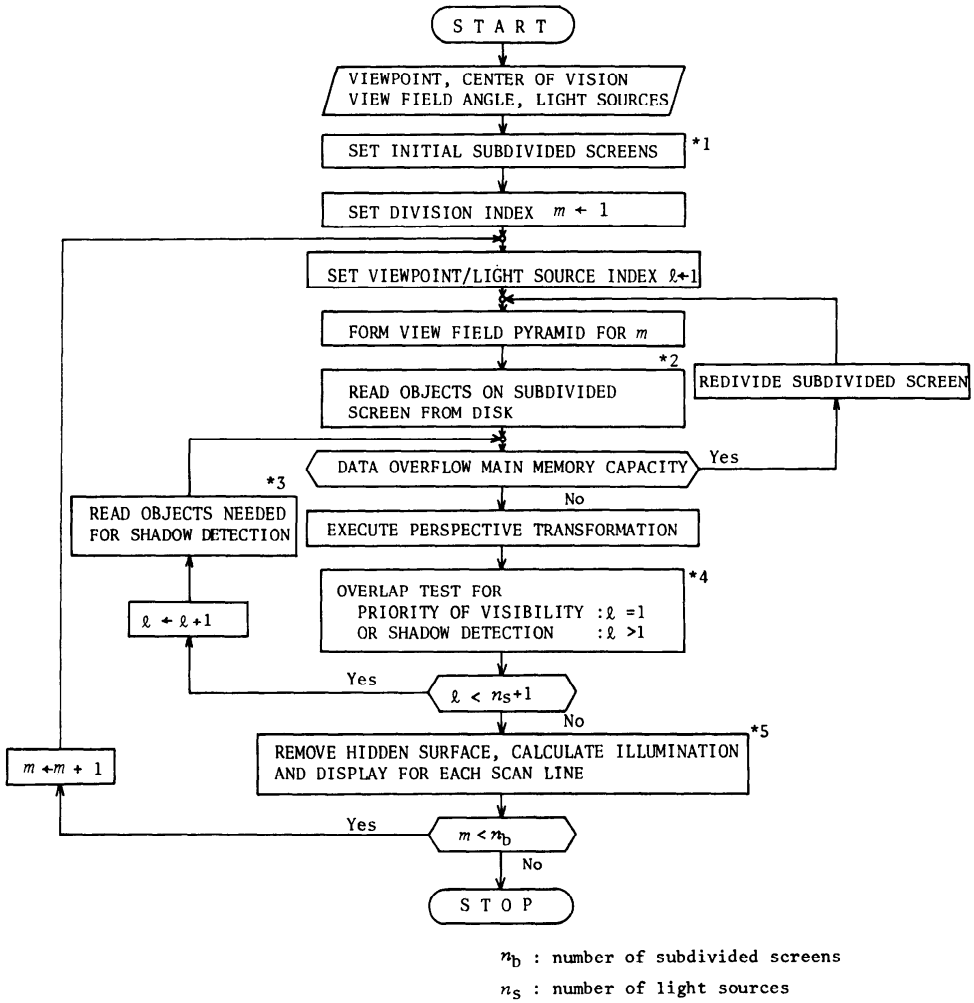

Fig. 4 Flow chart of subdividing a screen.

shadows on the above polyhedra are extracted from the disk (see *3 in Fig. 4). This process is repeated as many times as the number of light sources. When the data overflow main storage, the subdivided screen is horizontally redivided into two parts.

The procedures, 2) and 3), are repeated as many times as the number of divisions given by 1). In spite of no description in the flow chart, horizontal division has a certain limitation determined by the ratio of the vertical size to the horizontal size of a subdivided screen (e.g. 1.5). When the ratio becomes smaller than the limitation, the procedure stops. In this case, the number of vertical divisions can be increased and restarted from *1 by the user's demand, because there are some cases requiring more divisions depending on the arrangement of polyhedra or their shapes. In a few cases (e.g. too many overlapped polyhedra exist on a part of the screen), the above process is useless.

Refer to Ref. [3] about the priority of visibility and shadow detection in *4. The overlap tests for both of them are the same procedure, once for the viewpoint and once for each light source. Their tests are performed on the perspective plane (see Appendix for perspective transformation). For the polyhedra which can not be projected on it (e.g., vertices of the polyhedron exist behind the viewpoint), the procedure is executed in the three dimensional space.

The details of 2) and 3) are described in the following sections.

3. Extracting Data for a Subdivided Screen

The capacity of main memory of a mini-computer is small compared with that used for a flight simulator. Therefore, the extraction methods used in the latter are

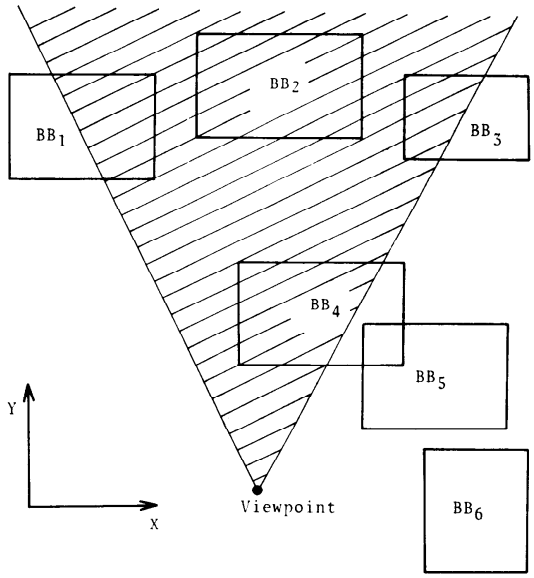

Fig. 5 Decision method for the number of divisions of a screen.

not always available for the former.

In almost any flight simulator, spheres (or rectangular boxes) enclosing each polyhedra are used for extracting the required polyhedra for each screen. If the spheres (or rectangular boxes) are within the view pyramid, they are extracted [4], [6]. This method is outstanding because the inclusion test is simple. On the other hand, a drawback of the method is the possibility of extracting unnecessary polyhedra which actually exist outside the view pyramid, especially when many long and slender polyhedra, such as pillars, exist. The possibility of needless extraction becomes large. Extracting these unnecessary polyhedra should be avoided for a minicomputer. For this reason, the pyramids formed by the viewpoint (or the light sources) and the contour lines of convex polyhedra are used instead of the spheres. The method using the pyramid takes longer for extracting polyhedra than the one using the sphere, but the computation time of the former is shorter than that of the latter for the processing of hidden surface removal and shadow detection. Furthermore, the pyramids (or the contour lines) can be used again for the overlap test, which is necessary for hidden surface removal and shadow detection.

For shortening the extracting time, the extraction of the minimum required polyhedra for each subdivided screen is performed by the following two steps. The first step is for each group (see Def. 1) constructed with several polyhedra, and the second step is for each polyhedra. In this process, an access between a disk and CPU is made for each group.

Before discussing the procedures for extracting the data for subdivided screens, some definitions are given.

[Def. 3] View Field Pyramid (VF): The domain of a quadrangular cone containing a viewpoint P_v and four

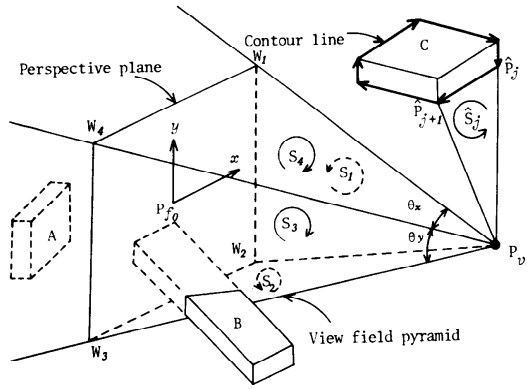

Fig. 6 The relation between a view field pyramid and convex polyhedra.

vertices $W_l(l=1, 2, 3, 4)$ forming a subdivided rectangle. The domain is infinite from the viewpoint P_v to the center of vision P_{f0}. The four planes $S_l[P_v, W_l, W_{l+1}](l=1, 2, 3, 4)$ surrounding the domain are described clockwise for an observer, where $W_5=W_1$ (see Fig. 6).

[Def. 4] Contour Plane of Convex Polyhedron $\hat{S}_j$: Assume that the contour lines of a convex polyhedron V_i consisting of vertices $P_k(k=1, 2, \cdots, n)$ for the viewpoint P_v are expressed by $[\hat{P}_1, \cdots, \hat{P}_j, \hat{P}_{j+1}, \cdots, \hat{P}_m]$ $(m \leqq n)$. The contour plane of convex polyhedron $\hat{S}_j$ is a plane which contains a triangle having the three vertices $[P_v, \hat{P}_j, \hat{P}_{j+1}]$ (where $\hat{P}_{m+1}=\hat{P}_1$) as shown in Fig. 6, where the symbol $\hat{}$ shows the vertices making contour lines, and the planes forming the convex polyhedron V_i are described in a clockwise direction for an observer. Therefore, the contour lines are also clockwise from the viewpoint, and its shape is a convex polygon.

[Def. 5] Classification of Convex Polyhedron V_i(or Bounding Box BB_i):

$$\begin{aligned} &\text{Type A: } V_i \subset VF \\ &\text{Type B: } (V_i \cap VF \neq \phi) \cap (V_i \not\subset VF) \\ &\text{Type C: } V_i \cap VF = \phi \end{aligned} \tag{1}$$

In order to extract the data for a subdivided screen, the bounding boxes $BB_i(i=1, 2, \cdots, n_e)$ are classified into three types, and the bounding boxes Type A and B are extracted. Then, the convex polyhedra $V_j(j=1, 2, \cdots, n)$ belonging to the bounding boxes Type B are classified into three types and extracted. The procedure is described as follows:

step 1. Set $i \leftarrow 1$; calculate the coefficients a_l, b_l, c_l, and d_l (refer to eq. 2) of the plane S_l forming a view field pyramid.

step 2. Classify BB_i.

step 3. Extract the data for a subdivided screen:

a) If BB_i is Type C, go to step 4.

b) If BB_i is Type A, set all V_j belonging to BB_i Type A; call all V_j and the details belonging to the front faces of each V_j to the main memory; go to step 4.

c) If BB_i is Type B, call all V_j belonging to BB_i to the main memory; classify each V_j; call the details on the front faces of V_j of which the Type is A or B; remove V_j of Type C; remove the details belonging to V_j of Type B existing outside the domain of VF; go to step 4.

step 4. If $i<n$, set $i \leftarrow i+1$ and return to step 2; otherwise stop.

The classifications of polyhedra (or bounding boxes) are obtained by using the features of convex polyhedra and the following equation; i.e., the relationships between a plane S_i containing the three vertices P_1, P_2, and P_3 and an arbitrary point P(X, Y, Z) are decided by the sign of $F_{S_i}(P)$:

$$F_{S_i}(P)=(P_2-P_1)\times(P_3-P_1)\cdot(P-P_1)$$
$$=a_iX+b_iY+c_iZ+d_i \qquad (2)$$

if $F_{S_i}(P)>0$; point P exists in a space from where the triangle $P_1P_2P_3$ is observed in a counter-clockwise direction,

if $F_{S_i}(P)<0$; point P exists in a space from where the triangle $P_1P_2P_3$ is observed in a clockwise direction,

and if $F_{S_i}(P)=0$; point P exists on plane S_i.

For the view field pyramid VF formed by faces S_i and vertices W_i, the convex polyhedron V_i constructed by vertices P_k are classified by the following procedure: i) If any S_i satisfies $F_{S_i}(P_k)\leqq 0$ for all vertices P_k of V_i (all P_k are on the opposite side of S_i to VF), V_i is Type C. ii) If $F_{S_i}(P_k)\geqq 0$ for every P_k and S_i, V_i is Type A. iii) In the other case it is required to obtain the contour line of V_i when viewed from the viewpoint. If any contour plane $\hat{S}_j$ satisfies $F_{\hat{S}_j}(W_i)\leqq 0$ for every W_i, V_i is Type C, if not so Type B.

4. Extracting Convex Polyhedra for Processing of Shadow Detection

As mentioned in Sec. 2, the polyhedra in the view field pyramid are not enough for processing a subdivided screen. The convex polyhedra existing outside a view field pyramid VF may cast their shadows on some convex polyhedra existing inside the VF. Therefore, these polyhedra must be called into main memory.

In order to extract these indispensable polyhedra, the following definitions are given.

[Def. 6] Visual Field Polyhedron (VP): A visual field polyhedron VP is a quadrangular truncated pyramid formed by a visual field pyramid VF and two planes; each plane is normal to the visual direction P_vP_{f0} and contains two vertices whose depths from P_v are the maximum or minimum of all vertices belonging to all convex polyhedra in the VF(see Fig. 7).

The planes of a VP are defined clockwise for an observer, and the depths of these vertices are obtained by using the denominator $(R-Z_k')$ in eq. (3) (see Appendix).

[Def. 7] Light Polyhedron (LP): A light polyhedron LP is a convex polyhedron formed by the following vertices; a light source P_l, and the vertices belonging to the back face of the visual field polyhedron VP when P_l is assumed as viewpoint.

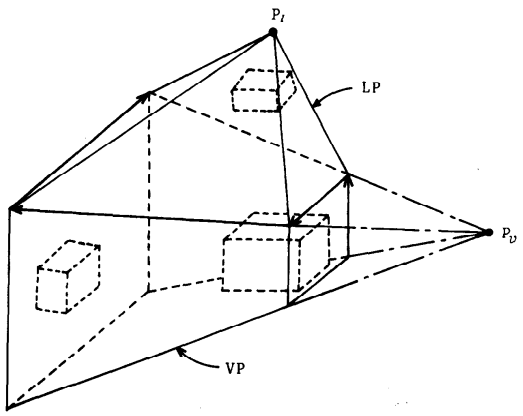

Fig. 7 Visual field polyhedron VP and light polyhedron LP.

Then, $LP \supset VP$ where if $P_l \in VP$, it is assumed that LP=VP (see Fig. 7). The planes of the LP are defined clockwise for an observer.

The procedure for extracting data of the processing of shadows made by each light source is as follows: If $P_l \in VP$, there is no convex polyhedron to add ($V_i \cap VP \neq \phi$ has been obtained in Sec. 3). If not so, the following processes are required. The LP is constructed and all convex polyhedra which satisfy $(V_i \cap VP=\phi) \cap (V_i \cap LP \neq \phi)$ are extracted. Then the contour lines of them are stored*. Where extracting polyhedra in LP is followed by extracting groups in LP by using the classification of bounding boxes. Thus, the classification of V_i for LP can be determined by means of a similar technique used for classification of the V_i for VP mentioned before.

Furthermore, to distinguish between the inner convex polyhedra of VP obtained at Sec. 3 and the outer convex polyhedra of VP obtained by the above procedure, gives the following additional benefit.

Assume that the numbers of the inner polyhedra of VP and the outer ones are n_p and n_l, respectively. When there is no distinction, the number of decisions needed for shadow processing (*4 in Fig. 4) is $(n_p+n_l)(n_p+n_l-1)/2$, whether the contour lines of convex polyhedra for a light source intersect or not. On the other hand, the times decrease to $n_p(n_p-1)/2+n_p\cdot n_l$ by means of the distinction. Furthermore, when there is no distinction, the decision concerning which of the convex polyhedron intersecting each other is in front or behind must be executed for all of them. However in the illustrated case it is needed only for the inner convex polyhedra of VP.

*The shadows occur when intersections exist between the contour lines of convex polyhedra for the light source. Therefore, no other information is needed to decide whether the V_i makes a shadow on the other convex polyhedra in the VP.

5. Examples

Examples are shown in Fig. 8. Picture (a), a three-phase transformer, is depicted as one of the examples of a complex object with many vertices. Each cylinder expressing a coil is composed of the set operation of two polygonal prisms, that is, the cylinder is formed by calculating the difference between a large prism and a small one, then the cylinders consist of many polyhedra. The transformer consists of six groups; three groups, each one consists of one core and four coils, two groups, the upper and lower yokes, and the parts of binders. Picture (b) is an example of lighting simulation where the number of light sources is four. The picture consists of four groups; a building, the ground and two street lamps. Each lamp is treated as one group because of designing the suitable position of it. In order to show the effects of cylindrical projection for a wide field of view, pictures (c) and (d) are depicted. The former is projected onto a plane and the latter onto the planes approximating a cylinder. The groups are constructed from each block of buildings, mountaines, trees and etc. The data are shown in Table 1.

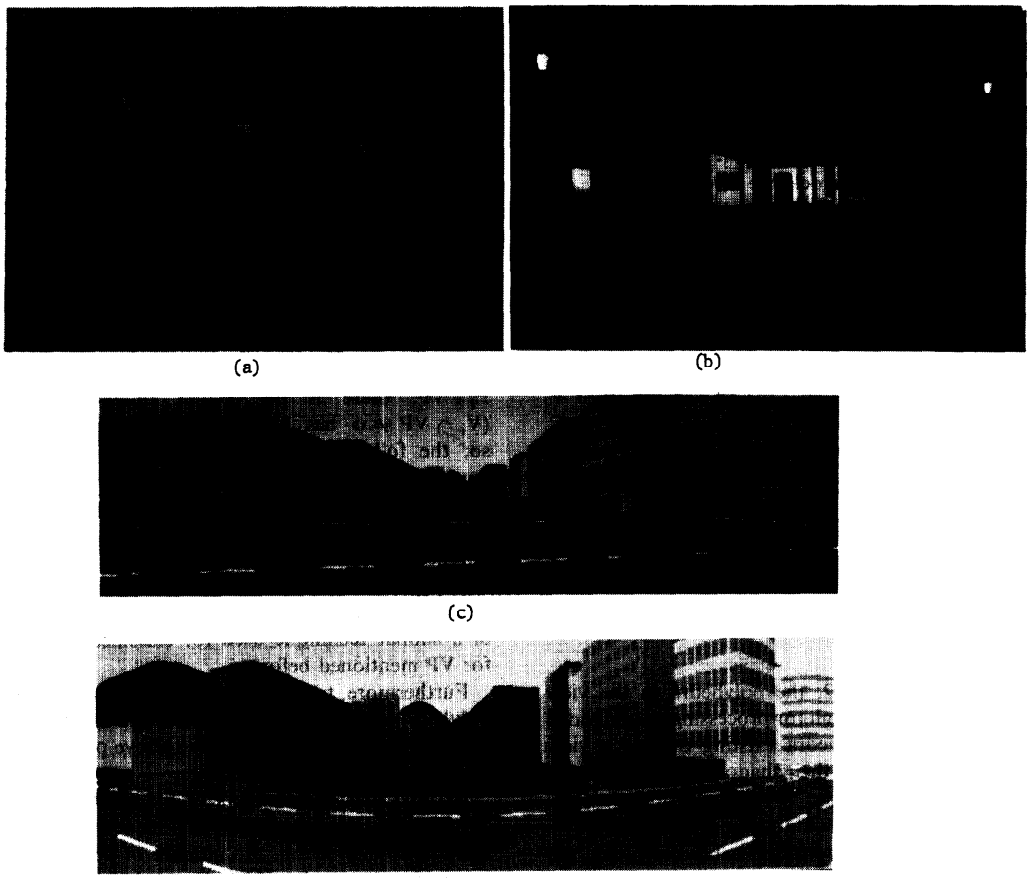

Fig. 8 Examples.

Table 1 Data and execution time of examples.

example	number of polyhedra	number of vertices	number of vertices of details	number of groups	number of divisions	horizontal visual angle (degree)	size of picture	execution time (sec)
a	165	1486	0	6	8	9	512×344	1067
b	42	392	229	4	3	55	512×338	1296
c	168	1380	1268	20	10	140	512×139	467
d					10	140	512×167	491

Fig. 9 shows the relationship between the number of subdivided screens n_b and the execution time of picture (b) which is the only possible case to calculate with the mini-computer used here. It also shows the relationship between n_b and the maximum required memory capacity for data of the picture (b) and (c) (the memory of (c) is calculated by using a large scale computer).

Concerning the picture (b), the maximum required memory capacity drastically decreases until $n_b=3$, but after that it doesn't change much. The execution time shows the minimum value at $n_b=3$, but the time has a tendency to increase for a larger number of n_b. These facts suggest that $n_b=3$ is the optimal number in this case for the execution time.

The reasons for the existence of the optimal number of subdivided screens are as follows: In this system, most execution time is spent for the overlap tests of hidden surface removal and shadow detections (see *4 in Fig. 4). The times for the overlap tests are in proportion to the square of the number of polyhedra n_0. If no polyhedra cast shadows and/or overlap with the screen boundaries, the execution time decreases in proportion to $1/n_b$, because the number of polyhedra of each subdivided screen is n_0/n_b. However, as it is clear from the required memory capacity (proportionate to the number of required polyhedra) in Fig. 9, the numbers of polyhedra overlapping the screen boundaries and of polyhedra needed for shadow detection increase as n_b increases. In picture (b), these number of latter can not be ignored because of multi-light sources. It means that these polyhedra don't decrease in proportion to n_b. On the other hand, the above discussion suggests that subdividing the screen is inefficient for the cases overlapping too many polyhedra on screen boundaries.

Concerning the picture (c), the required memory capacity for data drastically decreases until $n_b=3$ (one-third), but the mini-computer used here is still unavailable for the data. As n_b increases, the required memory gradually decreases and finally it reaches to one-sixth at $n_b=10$ (from 79 kW to 13 kW), then the mini-computer is able to handle the picture.

The computer and color display equipment used in this paper are OKITAC System 50/40 (main memory for user's area is 48 kW) and GRAPHICA M508 (512 × 512, 256 levels), respectively. The upper limit of polyhedra in one subdivided screen and light sources are 46 and 4, respectively.

6. Conclusions

This paper brought out the following facts: Concerning saving memory, both main memory and disk memory are effectively used. As shown in Fig. 9, it realized the complex pictures by using a relatively small computer. Concerning execution time, a proper division of a screen results in shortening the execution time as shown in Fig. 9, and grouping convex polyhedra is useful for extracting minimum required data for subdivided screens.

Concerning the quality of images, the example (d) in Fig. 8 depicts the effect of the projection on quasi-cylindrical planes.

Appendix

Assumed that the coordinate system of a set of the convex polyhedra $V_i(i=1, 2, \cdots, n)$ (these are given as input data) is (X, Y, Z) the perspective transformation of the vertices of polyhedra is performed as follows: We assume that the coordinates of a center of vision P_{f0} are (X_{f0}, Y_{f0}, Z_{f0}) and viewpoint P_v is given as the polar coordinates (R, θ, ϕ) having the origin P_{f0}; i.e., if the coordinate system (X″, Y″, Z″) having the origin P_{f0} is parallel to the coordinate system (X, Y, Z), R is the $P_{f0}P_v$, θ is an angle between the visual direction $P_{f0}P_v$ and the plane X″−Z″, and ϕ is an angle between $P_{f0}P_v$ and the plane X″−Y″ (see Fig. 10).

Assumed that the origin of orthogonal coordinate

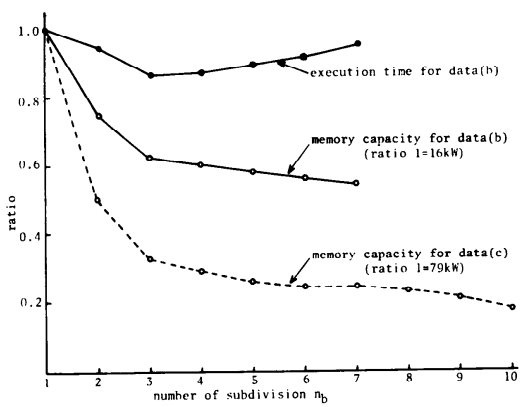

Fig. 9 The execution time and maximum required memory capacity with respect to the number of divisions.

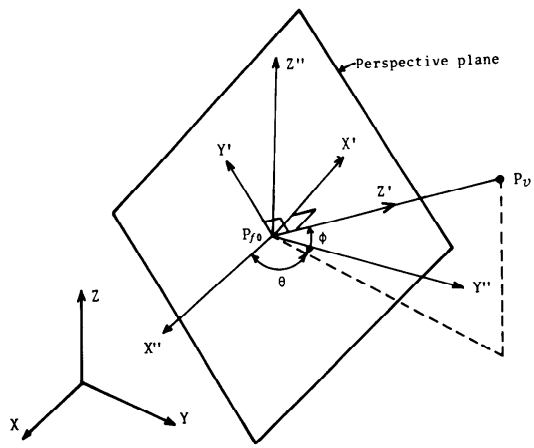

Fig. 10 Coordinates system.

system (X', Y', Z') is P_{f0}, the Z' axis is the visual direction $P_{f0}P_v$, and the X' axis is the horizontal line, the X'−Y' plane constructs the perspective plane (x, y). If an arbitrary point $P_0(X_0, Y_0, Z_0)$ in the coordinate system (X, Y, Z) is transformed to the coordinate system (X', Y', Z') and its coordinates are expressed by $P_0(X'_0, Y'_0, Z'_0)$, the point P_0 transformed to the perspective plane having the origin P_{f0} is expressed by the following equation (see Fig. 1):

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{R}{R - Z'_0} \begin{bmatrix} X'_0 \\ Y'_0 \end{bmatrix} \tag{3}$$

in the case where $R - Z'_0$ is negative, the point exists behind the viewpoint.

The perspective transformations of the other subdivided perspective planes are obtained by rotating the center of vision; as shown in Fig. 2, a quasi-center of vision $P_{fi}(X_{fi}, Y_{fi}, Z_{fi})$ of the ith subdivided perspective plane is obtained by rotating P_{f0} with the origin P_v as much as δ_i in the plane X'−Y', where the directions of Y of all subdivided perspective planes are the same due to the features of a cylinder. The coordinates of P_{fi} are expressed by

$$\begin{bmatrix} X_{fi} \\ Y_{fi} \\ Z_{fi} \end{bmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta \cdot \cos\phi \\ \cos\theta & \sin\theta \cdot \cos\phi \\ 0 & \sin\phi \end{bmatrix} \cdot \begin{bmatrix} R \sin\delta_i \\ R(1-\cos\delta_i) \end{bmatrix} + \begin{bmatrix} X_{f0} \\ Y_{f0} \\ Z_{f0} \end{bmatrix} \tag{4}$$

Assumed that the unit vector of P_vP_{f0} is U(u, v, w), the transformation of the point $P_0(X_0, Y_0, Z_0)$ to the coordinate system (X', Y', Z') is obtained by the following equations:

$$\begin{bmatrix} X'_0 \\ Y'_0 \\ Z'_0 \end{bmatrix} = \begin{bmatrix} -w \cdot \cos\theta \cdot \sin\phi - v\cos\phi & u\cos\phi + w\cos\theta\sin\phi & u\sin\theta\sin\phi - v\cos\theta\sin\phi \\ -\cos\theta\sin\phi & -\sin\theta\sin\phi & \cos\phi \\ u & v & w \end{bmatrix} \begin{bmatrix} X_0 - X_{fi} \\ Y_0 - Y_{fi} \\ Z_0 - Z_{fi} \end{bmatrix} \tag{5}$$

Thus, by using eq. (3), $P_0(x_0, y_0)$ is obtained.

References

1. CROW, F. C. Shaded Computer Graphics in the Entertainment Industry, *Computer Graphics* **11**, 2 (1977), 242.
2. ATHERTON, P., WEILER, K. and GREENBERG, D. Polygon Shadow Generation, *Computer Graphics* **12**, 3(1978), 275–281.
3. NISHITA, T. and NAKAMAE, E. An Algorithm for Half-Toned Representation of Three Dimensional Objects, *Information Processing in Japan*, **14** (1974), 93–99.
4. SCHACHTER, B. J. Computer Image Generation for Flight Simulation, *IEEE Computer Graphics & Applications* **1**, 4 (1981), 29–68.
5. SUTHERLAND, I., SPROULL, R. and SCHUMACKER, R. A Characterization of Ten Hidden Surface Elimination Algorithms, *ACM Computing Surveys* **6**, 1 (1974), 1–55.
6. CLARK, J. H. Hierarchical Geometric Models for Visible Surface Algorithms, *Comm. ACM* **19**, 10 (1976), 547–554.
7. NAKAMAE, E. and NISHITA, T. An Algorithm for Hidden Line Elimination of Polyhedra, *Information Processing in Japan*, **12** (1972), 134–141.

(Received May 10, 1982; revised March 1, 1984)