Systems and Computers in Japan, Vol. 38, No. 8, 2007
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J88-D-II, No. 7, July 2005, pp. 1205–1215

Hierarchical Face Cluster Partitioning of Polygonal Surfaces and High-Speed Rendering

Tokuo Tsuji,[1] Hongbin Zha,[2] Tsutomu Hasegawa,[1] and Ryo Kurazume[1]

[1]Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, 812-8581 Japan

[2]Center for Information Science, Peking University, Beijing, 100871, China

SUMMARY

A computer's rendering capacity cannot handle the increased amount of data when rendering high-quality, three-dimensional computer graphic images. This results in problems such as rendering not being possible depending on the subject image, or the rendering speed dropping too far. To resolve these problems, we propose a method for generating a new hierarchical three-dimensional data structure based on recursive face cluster partitioning, and a high-speed rendering data extraction algorithm using that data structure. In the proposed algorithm, local resolution control using the approximating polygon of a face cluster, and precise invisible surface culling using the bounding volume of the face cluster can be executed in parallel. High-speed rendering can be performed with little deterioration in quality, even for a large amount of data. The effectiveness of this algorithm was confirmed with a local resolution control experiment and a large-scale three-dimensional model rendering experiment. © 2007 Wiley Periodicals, Inc. Syst Comp Jpn, 38(8): 32–43, 2007; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.20581

Contract grant sponsors: 21st Century COE Program "Forming Fundamental Social Systems in Systems and Information Science" and support was received from the Strategic Information and Communications R&D Promotion Programme, Ministry of Internal Affairs and Communications.

Key words: polygon model; hierarchical modeling; LOD control; occlusion culling.

1. Introduction

With improvements in the capacities of three-dimensional measuring devices such as range finders, stereo cameras, and medical imaging devices, three-dimensional forms of real objects can now be measured at high precision and high density, and the measurements easily read into computers. Also, improvements in the capacities of 3D modelers and computers have made it easy to generate complex and detailed virtual objects. High-quality, three-dimensional computer graphic (CG) images can be generated using these geometrical models.

Higher precision and higher density three-dimensional data have resulted in increasing amounts of data to be processed. For example, in Stanford University's Digital Michelangelo Project [1], 2 billion polygons were collected from the statue of David. Thus, scans of art objects or buildings will amount to several gigabytes of data. A computer's rendering capacity cannot handle this increase in data, resulting in problems such as rendering becoming impossible, depending on the image, or excessive drops in the rendering speed.

However, the amount of data necessary for rendering a three-dimensional model placed in a given scene changes depending on the relative positions of the viewpoint and the model. For example, when the viewpoint is changed and the three-dimensional model is rendered as shown in Fig. 1, the surface can apparently be expressed with a small number of polygons and the image can be rendered without a loss in quality in the case of (c) where the viewpoint is far from the model. Also, when the viewpoint is close to the model (a), polygons outside the field of view need not be rendered. A three-dimensional model comprising a large amount of data can be rendered at a high speed when only the minimum amount of data necessary depending on the viewpoint is rendered.

Level of detail (LOD) control and invisible surface culling are proposed as methods for selecting the rendering data according to the viewpoint and rendering large amount of data efficiently. LOD control accelerates the process by controlling the LOD of the image and simplifying and rendering the object [2–10]. On the other hand, invisible surface culling improves the rendering speed by removing surfaces not to be rendered [11–17]. However, using either LOD control or invisible surface culling alone will sometimes not result in a sufficient and stable rendering speed in an environment with a large number of models comprising large amounts of data. Consequently, a process which combines LOD control and invisible surface culling is necessary. In fact, controlling the level of detail according to the resolution of the image and deleting invisible surfaces will reduce dependence on the complexity of the rendered image and rendering can be performed at high speeds even for large amounts of data. Ideally, the maximum amount of processing will depend only on the number of pixels.

Before now, there have been proposals for algorithms combining LOD control and invisible surface culling, using a hierarchical data structure expressing three-dimensional models at different resolutions and describing the relationships thereof with a tree structure [18–24]. However, these methods have problems such as perceptible errors (artifacts) appearing in the images and reduced rendering speed when the viewpoint changes greatly. These are a result of the conventional methods not consistently performing processing at each level of the hierarchy for error-free invisible surface culling, and processing for control of levels of detail in arbitrary sections.

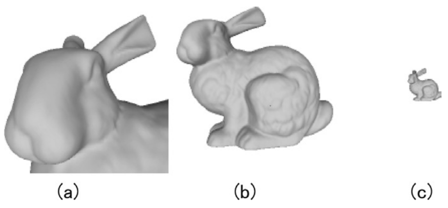


Fig. 1. A model displayed in different viewpoints.

A tree structure with vertices as nodes is used for control of levels of detail in arbitrary sections. The vertex nodes are selected according to the desired level of detail and the surface to be rendered is reconstituted from those vertices. Meanwhile, in error-free invisible surface culling, the model is recursively partitioned and the hierarchy of bounding volumes, enclosing progressively smaller regions as one moves toward the lower levels, is generated in advance. In rendering, processing is accelerated by examining bounding volumes in order from the upper nodes of the hierarchy and removing faces included in bounding volumes determined to be invisible. However, when this type of invisible surface removal is performed using a tree structure of vertices, the bounding volumes of each node are highly overlapped because faces traverse the tree structure, and the efficiency drops substantially.

In this study, we propose an algorithm for generating a new hierarchical three-dimensional data structure based on recursive face cluster partitioning, and a high-speed rendering data extraction algorithm using that data structure. In preprocessing, the operation of recursively partitioning face clusters, the set of connected faces, from the original model is repeated and a binary tree with face clusters as nodes is generated. At each node in the hierarchy, the approximating polygon of the face cluster and the bounding volume are computed. During rendering, the approximating polygon is selected according to the desired LOD, adjoining polygons are connected, and a connected mesh is generated. Even when connecting approximating polygons, the determination of invisibility can be made without error at all the nodes in the tree structure because those approximating polygons are included in the bounding volume of a cluster. With this algorithm, both error-free invisible surface culling and LOD control of arbitrary portions of a model, which are not implemented with conventional methods, can be performed at all levels of the hierarchy, and rendering without artifacts and without delay, even in the case of large shifts of viewpoint, is possible.

This paper is organized as follows. In Section 2, relevant algorithms for LOD control and invisible surface culling are introduced and issues therewith are clarified. An overall picture of the proposed algorithm is discussed in Section 3. Next, the proposed hierarchical data structure based on recursive face cluster partitioning is described in detail in Section 4 and an effective method for selecting rendering data using the proposed hierarchical data structure is discussed in Section 5. In Section 6 are disclosed the results of the local LOD experiment and the large-scale three-dimensional model rendering experiment. Conclusions are discussed in Section 7.

2. Relevant Algorithms

2.1. LOD control

LOD control is performed at a high speed through controlling the LOD of the mesh and simplifying and rendering objects which are distant from the viewpoint. Two types of LOD control are discrete LOD control [10] and continuous LOD control [2–9]. In discrete LOD control, several preparatory steps are taken in advance for objects with different levels of detail for the entire model and the display switches among the models during rendering. In continuous LOD control, a tree structure of vertices is prepared and the vertices are dynamically selected during rendering.

Discrete LOD control is easy to set up, but has the problem that images change greatly when the level of detail is switched and artifacts will occur. Also, rendering efficiency will sometimes drop because it is not possible to establish LOD with fine differences within the model.

The progressive mesh proposed by Hoppe [2] is a data structure for implementing continuous LOD control. Therein, a single edge within the model is selected according to appropriate metrics, the endpoints of that edge are brought to a single point, and the edge is removed; this operation is repeated until a predetermined simplification metric is satisfied. At the same time, a tree structure is prepared wherein child nodes are the endpoints of removed edges and parent nodes are the vertices newly generated upon edge culling. The LOD of the model can be easily changed by controlling the hierarchy of data accessed during rendering.

2.2. Invisible surface culling

Invisible surface culling accelerates rendering by removing surfaces outside the field of view which are not rendered. Invisible surface culling encompasses an approximate algorithm for reducing the resolution of regions which are presumed to be unseen, and a conservative algorithm for removing regions which are certain to be unseen.

A problem with the approximate algorithm is that artifacts occur because it is not an exact calculation.

Meanwhile, the conservative invisible surface culling algorithm does not have artifacts. The calculations for conservative invisible surface culling require that a hierarchy of bounding volumes be prepared in advance, from a volume containing the entire model down to progressively smaller regions. At this time, the bounding volumes are found as simple forms with convexities such as spheres or cubes. Expressing the bounding volumes with smaller elements can result in faster calculations. Moreover, the calculation efficiency improves as the bounding volumes are brought into close contact with the model and superfluous space is not included.

2.3. Integrated algorithm for LOD control and invisible surface culling

An integrated algorithm using discrete LOD control [19–21] and an integrated algorithm using approximate invisible surface culling [22, 23] have problems with artifacts. Continuous LOD control and the conservative invisible surface culling algorithms must be integrated in order to suppress the generation of artifacts.

Yoon and colleagues proposed an algorithm which integrates continuous LOD control and conservative invisible surface culling [24]. However, the determination of invisibility and LOD control are performed in two stages, reducing efficiency. In this algorithm, preprocessing is performed so that a hierarchy with partial trees as nodes is prepared by partitioning the progressive mesh in space and bounding volumes are found for each partial tree. During rendering, the partial tree nodes to be rendered are selected according to the determination of invisibility using the bounding volumes. Thereafter, LOD control is performed within the partial trees. A consequent problem is that the determination of invisibility is not performed within a partial tree. Furthermore, a vertex comprising one face is sometimes associated with a different partial tree. As such surfaces are not included in the bounding volume, the determination of invisibility must be performed for these surfaces individually. In order to accelerate rendering, similarities between frames of rendering data are used. Data rendered in the previous frame are used in the selection of rendering data for a given frame. Consequently, big shifts in the viewpoint result in a drop in rendering speed because of reduced similarity between frames.

3. Overview of Proposed Algorithm

In this paper, data expressed with a connected mesh is treated as the input data. An operation is performed wherein all faces of the input data are treated as one cluster which is repeatedly and recursively divided in half. The hierarchical data structure necessary to accelerate rendering is generated in the process which repeats the cluster partitioning operation.

The following steps are performed in the partition operation.

(1) Using multivariate analysis, data distribution within the cluster is analyzed and the center of gravity and principal component vector are found.

(2) The cluster is partitioned at the plane passing through the cluster center of gravity and perpendicular to

the principal component vector calculated with multivariate analysis, and two new clusters are generated.

(3) A tree structure is generated in which the parent is the partition cluster and the children are the clusters generated by partitioning.

(4) The clusters generated by partitioning are approximated with polygons.

With the repetition of this operation, a binary tree structure is generated wherein the root node is the cluster including all the faces, and the leaf nodes are clusters including one original face. Each node in the generated binary tree hierarchy corresponds to an approximating polygon. In fact, the resulting data structure is also a hierarchy of approximating polygons.

During rendering, a connected approximating mesh is generated by selecting approximating polygons satisfying the given level of detail from the upper level nodes in the tree structure, and connecting adjacent approximating polygons together.

In the hierarchy of approximating polygons, the conservative determination of invisibility can be performed at all nodes. Even when deformed by being connected with adjacent polygons, an approximating polygon is constituted with vertices on the cluster boundary of its node. Consequently, polygons are not generated by combining vertices within a cluster associated with different clusters. Also, each polygon is constituted from vertices of each face cluster and is therefore included in the bounding volume of the face cluster. Furthermore, the face cluster of each node is a partial aggregation of the face cluster of the node above it. Each polygon is therefore enclosed within the bounding volume of the cluster of the node above. Consequently, conservative invisible surface culling, using the bounding volume of the face cluster of the node above, can be performed in parallel with continuous LOD control.

The proposed algorithm has the following three characteristics.

(1) In face clustering, forms closer to the surface than in the conventional algorithm generate a simple face cluster. Conventional recursive partitioning of data points [10] does not necessarily result in the partitioned data point distribution becoming flat. Also, with the face cluster merging method [25], cluster forms in the node above easily become complex.

(2) Approximating meshes with arbitrary levels of detail are generated in real time from the face cluster hierarchy. The face cluster hierarchy structure itself has already been proposed with the object being to accelerate radiosity calculations [25], but has never been used as a method for generating an approximating mesh. Conventional methods [10, 26] for generating an approximating mesh from a face cluster or vertex cluster do not generate tree structures and do not implement continuous LOD control in real time.

(3) A polygon enclosing a cluster in an image is generated by expanding the approximating polygon. The bounding polygon can be attached to and enclose a cluster with a smaller number of elements than a conventional oriented bounding box (OBB) or the like.

4. Hierarchical Data Structure Based on Recursive Face Cluster Partitioning

4.1. Multivariate analysis

Using multivariate analysis [27], the face distribution and normal distribution constituting a face cluster are analyzed and used in cluster partition orientation and model error calculation. Before now, principal component analysis was used to generate hierarchical models. For example, a method was proposed for finding the main axis from an aggregation of data points with principal component analysis, and then partitioning the aggregation at a plane passing through the center of the data points and perpendicular to the main axis [10]. In face cluster merging, the quadric error (least squares error) derived with principal component analysis is used for selecting the merged cluster [25]. Also, methods for generating the progressive mesh include a method for using the quadric error to select the edges to be collapsed [7, 8].

In this research, the face distribution was analyzed with principal component analysis and used for cluster partitioning and modeling error evaluation. In the modeling error calculation, an error evaluation independent of data point density is implemented by expanding the quadric error [25] calculated for conventional discrete point data to calculations for continuous surfaces. Partitioning using principal component analysis has been used before, but the normal distribution of the faces was not considered and the partitioning results sometimes did not become flat. Thus, this is a proposal for a new method for analyzing the distribution of faces in normal space using principal correlation analysis. This method is used in cluster partitioning along with principal component analysis.

Principal component analysis (PCA)

When data points $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ are present, the variance-covariance matrix of the points is expressed as follows:

$$\mathbf{Z} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{v}_i - \bar{\mathbf{v}})(\mathbf{v}_i - \bar{\mathbf{v}})^T \tag{1}$$

$$= \frac{1}{N-1}\left(\sum_{i=1}^{N}\mathbf{v}_i\mathbf{v}_i^T - k\bar{\mathbf{v}}\bar{\mathbf{v}}^T\right) \tag{2}$$

Here, $\bar{\mathbf{v}} = (\Sigma_{i=1}^{k} \mathbf{v}_i)/k$.

This formula is expanded to points on a surface. The variance-covariance matrix $\mathbf{Z}_{rr}$ from a point $\mathbf{r}$ on a face cluster region R is expressed as follows:

$$\mathbf{Z}_{rr} = \frac{1}{A}\int_R (\mathbf{r}-\bar{\mathbf{r}})(\mathbf{r}-\bar{\mathbf{r}})^T d\mathbf{r} \tag{3}$$

$$= \frac{1}{A}\left(\int_R \mathbf{r}\mathbf{r}^T d\mathbf{r} - A\bar{\mathbf{r}}\bar{\mathbf{r}}^T\right) \tag{4}$$

Here, A is the total surface area of the mesh, $d\mathbf{r}$ is the surface element, and $\bar{\mathbf{r}} = 1/A \int_R \mathbf{r}d\mathbf{r}$.

Because all polygon surfaces can be partitioned into triangular patches, $\mathbf{Z}_{rr}$ can be found with the base unit being triangular patches. When the triangular patch f_i has three vertices $\{\mathbf{v}_{i1}, \mathbf{v}_{i2}, \mathbf{v}_{i3}\}$, the points on each triangular patch can be expressed as follows:

$$\mathbf{r}_i = s(\mathbf{v}_{i2} - \mathbf{v}_{i1}) + t(\mathbf{v}_{i3} - \mathbf{v}_{i1}) + \mathbf{v}_{i1} \tag{5}$$

Here, s and t are in the range of $D = \{(s, t) : 0 < s, t, s + t < 1\}$. Also, $\mathbf{r}_i\mathbf{r}_i^T$ is the integral over the triangular patch and is expressed as follows:

$$\int\int_D \mathbf{r}_i\mathbf{r}_i^T \left|\frac{\partial \mathbf{r}_i}{\partial s} \times \frac{\partial \mathbf{r}_i}{\partial t}\right| dsdt$$
$$= A_i\left\{\frac{1}{6}\left(\mathbf{v}_{i1}\mathbf{v}_{i1}^T + \mathbf{v}_{i2}\mathbf{v}_{i2}^T + \mathbf{v}_{i3}\mathbf{v}_{i3}^T\right)\right.$$
$$+ \frac{1}{12}\left(\mathbf{v}_{i1}\mathbf{v}_{i2}^T + \mathbf{v}_{i2}\mathbf{v}_{i3}^T + \mathbf{v}_{i3}\mathbf{v}_{i1}^T\right.$$
$$\left.\left. + \mathbf{v}_{i2}\mathbf{v}_{i1}^T + \mathbf{v}_{i3}\mathbf{v}_{i2}^T + \mathbf{v}_{i1}\mathbf{v}_{i3}^T\right)\right\} \tag{6}$$

Here, A_i is the area of the triangular patch. When $\mathbf{r}_i$ is integrated over the triangular patch, the following results:

$$\bar{\mathbf{r}}_i = \frac{1}{A_i}\int\int_D \mathbf{r}_i \left|\frac{\partial \mathbf{r}_i}{\partial s} \times \frac{\partial \mathbf{r}_i}{\partial t}\right| dsdt$$
$$= \frac{1}{3}(\mathbf{v}_{i1} + \mathbf{v}_{i2} + \mathbf{v}_{i3}) \tag{7}$$

Consequently, the variance-covariance matrix $\mathbf{Z}_{rr}$ of points on the triangular patch $\{f_1, \ldots, f_M\}$ is expressed as

$$\mathbf{Z}_{rr} = \frac{1}{A}\sum_{i=1}^{M}\left(A_i\sum_{j=1}^{3}\sum_{k=1}^{3} m_{jk}\mathbf{v}_{ij}\mathbf{v}_{ik}^T\right)$$
$$- \frac{(\sum_{i=1}^{M} A_i\bar{\mathbf{r}}_i)(\sum_{i=1}^{M} A_i\bar{\mathbf{r}}_i)^T}{A^2} \tag{8}$$

$$m_{jk} = \begin{cases} 1/6\ (j = k) \\ 1/12\ (j \neq k) \end{cases}, \quad A = \sum_{i=1}^{M} A_i$$

The eigenvector corresponding to the maximum eigenvalue of $\mathbf{Z}_{rr}$ becomes the principal component vector.

The modeling error is calculated for each cluster with multivariate analysis. The modeling error ε is defined as follows:

$$\epsilon = \frac{1}{A}\int_R \{(\mathbf{r}-\bar{\mathbf{r}})\cdot \mathbf{p}_{min}\}^2 d\mathbf{r}$$
$$= \mathbf{p}_{min}^T \mathbf{Z}_{rr}\mathbf{p}_{min}$$

Here, $\mathbf{p}_{min}$ is the eigenvector corresponding to the smallest eigenvalue of $\mathbf{Z}_{rr}$. The modeling error is the mean square of the distance from the approximating surface of a point on the cluster. The formula is an expansion of the quadric error from the discrete data points, which has already been proposed, to a continuous surface. For smaller values of the modeling error ε, the cluster is closer to a flat surface; and the error is smaller when approximated with a polygon. The modeling error is calculated when the hierarchical model is prepared and is preserved in the tree structure in advance.

Canonical correlation analysis (CCA)

Let $\mathbf{n}$ be a normal vector on the point $\mathbf{r}$, and $\mathbf{n}_i$ be a normal vector of a surface f_i, then the normal variance-covariance matrix $\mathbf{Z}_{nn}$ is found as follows:

$$\mathbf{Z}_{nn} = \frac{1}{A}\int_R (\mathbf{n}-\bar{\mathbf{n}})(\mathbf{n}-\bar{\mathbf{n}})^T d\mathbf{r} \tag{9}$$

$$= \frac{1}{A}\left(\int_R \mathbf{n}\mathbf{n}^T d\mathbf{r} - A\bar{\mathbf{n}}\bar{\mathbf{n}}^T\right) \tag{10}$$

$$= \frac{1}{A}\left(\sum_{i=1}^{M} A_i\mathbf{n}_i\mathbf{n}_i^T - A\bar{\mathbf{n}}\bar{\mathbf{n}}^T\right) \tag{11}$$

Here, $\bar{\mathbf{n}} = (\Sigma A_{i=1}^{M} A_i\bar{\mathbf{n}}_i)/A$. Also, the correlation matrix normal to the position $\mathbf{Z}_{rn}$ is calculated as follows:

$$\mathbf{Z}_{rn} = \frac{1}{A}\int_R (\mathbf{r}-\bar{\mathbf{r}})(\mathbf{n}-\bar{\mathbf{n}})^T d\mathbf{r} \tag{12}$$

$$= \frac{1}{A}\left(\int_R \mathbf{r}\mathbf{n}^T d\mathbf{r} - A\bar{\mathbf{r}}\bar{\mathbf{n}}^T\right) \tag{13}$$

$$= \frac{1}{A}\left(\sum_{i=1}^{M} A_i\bar{\mathbf{r}}_i\mathbf{n}_i^T - A\bar{\mathbf{r}}\bar{\mathbf{n}}^T\right) \tag{14}$$

The canonical correlation matrix $\mathbf{B}_{rn}$ facing the position is expressed as

$$\mathbf{B}_{rn} = \mathbf{Z}_{rr}^{-1}\mathbf{Z}_{rn}\mathbf{Z}_{nn}^{-1}\mathbf{Z}_{rn}^T \tag{15}$$

The principal component vector of $\mathbf{B}_{rn}$ becomes the direction with the highest normal dispersion on the three-dimensional space.

4.2. Cluster partitioning

Cluster partitioning is performed by cutting with a surface perpendicular to the principal component vector and passing through the center of gravity of the cluster. However, the cluster growth method [26] is used so that the cluster contains a continuous region. Two seed surfaces farthest from the cutting surface are selected and then clusters are grown from the seed surfaces to the cutting surface. At this time, a region which is not included in the two newly generated clusters exists depending on the form of the cluster. A continuous cluster is generated by merging this region with the contacting clusters.

Partitioning using the principal component vector calculated with PCA or CCA results in the following characteristics. Partitioning in the principal component direction calculated with PCA results in the cluster being partitioned in the direction with the broadest distribution, and therefore the distribution is contained within a more narrow range. This is effective for invisible surface culling. Furthermore, the form of the cluster becomes nearly circular rather than long and thin. However, flat clusters are not necessarily generated. On the other hand, partitioning in the principal component direction calculated with CCA results in flatter clusters than is the case with PCA, because the cluster is partitioned into an aggregation of surfaces having similar orientations. However, upon repetition of cluster partitioning, sometimes long and thin, smoothly curving clusters are generated rather than clusters which are nearly flat surfaces. Combining the use of PCA and CCA results in both types of characteristics and implements the generation of clusters which are more like flat surfaces.

A detailed partitioning procedure is discussed below.

Selecting two seed surfaces

Two seed surfaces are selected on a cluster. The two seed surfaces are the surfaces with the maximum separation along the principal component vector. The value R_i for selecting seed surfaces is defined as follows:

$$R_i = \mathbf{p}_{max} \cdot (\bar{\mathbf{r}}_i - \bar{\mathbf{r}}) \tag{16}$$

Here, $\mathbf{p}_{max}$ is the principal component vector of $\mathbf{Z}_{rr}$ or $\mathbf{B}_{rn}$. The surface with the maximum value of R_i and the surface with the minimum value of R_i are selected as the two seed surfaces. Here, f_{s1} and f_{s2} are the seed surfaces and R_{s1} = max(R_i) and R_{s2} = min(R_i).

Cluster growth

Clusters are grown from seed surfaces to form two clusters (Fig. 2). In the initial condition, a cluster includes only one seed surface. C_{s1} and C_{s2} are clusters corresponding to f_{s1} and f_{s2} and in the initial state, $f_{s1} \in C_{s1}$ and $f_{s2} \in C_{s2}$.

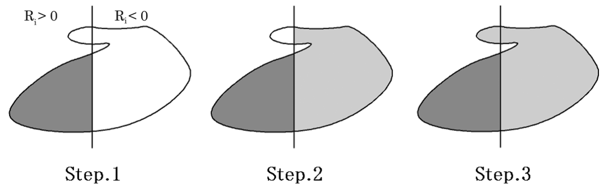


Fig. 2. Face cluster partitioning.

Step 1: Among the surfaces adjacent to the surface including C_{s1}, the surfaces where $R_i > 0$ are added to C_{s1}. The step ends when all adjacent surfaces have been checked.

Step 2: Among the surfaces adjacent to the surface including C_{s2}, the surfaces where $R_i < 0$ are added to C_{s2}. The step ends when all adjacent surfaces have been checked.

Step 3: The cluster is completely partitioned by combining regions not included in the two clusters with adjacent clusters.

Selecting generated clusters

Each cluster is partitioned using the principal component vector calculated with PCA and the sum of the modeling error ε of the two resulting clusters is found. Next, partitioning is performed in the same way using the principal component vector calculated with CCA. The sum of the modeling error ε of the two resulting clusters is found. The results from the partitioning process yielding the least sum of modeling errors ε are employed.

4.3. Polygon generation

Concurrent with the cluster partitioning operation, polygons approximating each cluster are generated (Fig. 3). The detailed procedure is shown below (Fig. 4).

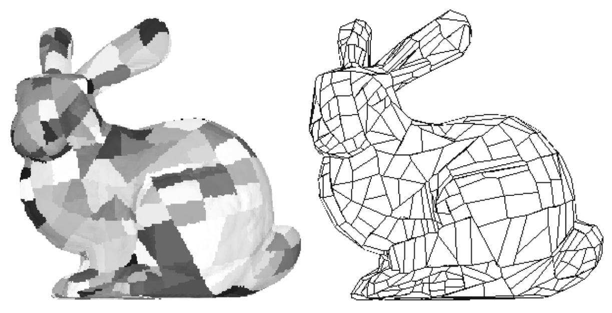


Fig. 3. Clusters and polygons.

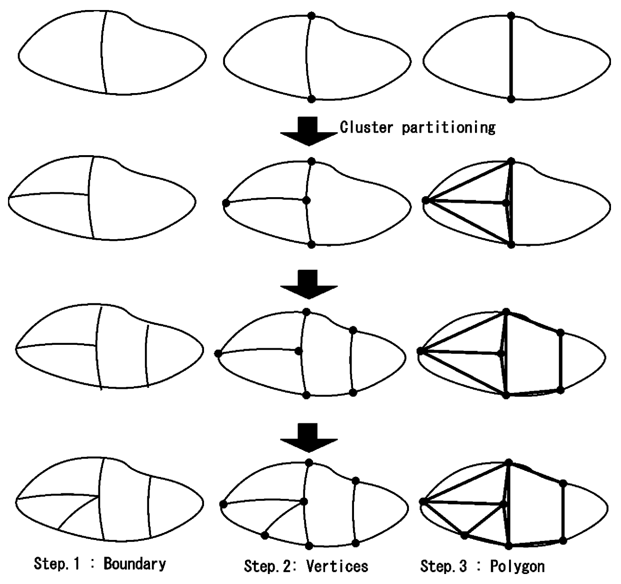


Fig. 4. Polygon generation.

Step 1: Extract the cluster partitioning boundary by extracting the boundary between two newly generated clusters.

Step 2: Register the points at both ends of the boundary in Step 1 as vertices of the polygon. When the boundary of Step 1 is closed, do not register vertices and do not execute Step 3.

Step 3: Generate a polygon corresponding to the cluster by linking the registered vertices along the cluster boundary.

5. High-Speed Rendering Using the Hierarchical Model

5.1. LOD control

The method for performing LOD control using the proposed hierarchical model is discussed. The nodes of the tree structure are examined by traversing each child node from the root node of the tree structure until a child node satisfying the following is reached:

$$\epsilon \leq \epsilon' \tag{17}$$

Here, ε' is the tolerance established according to the object. ε is the modeling error calculated when the tree structure was generated.

For example, the tolerance may be found with the following formula in order to dynamically change the level of detail according to the distance from the viewpoint:

$$\epsilon' = k \cdot d^2 \tag{18}$$

Here, d is the distance along the line of sight from the viewpoint to the center of the target cluster, and k is a constant determined according to the resolution of the screen, or the like.

When nodes are selected with the procedures discussed above and the child node of only one of two adjacent clusters is selected, a gap is formed between two polygons [Fig. 5(a)]. The gap is filled by connecting the vertices of the adjacent polygons while maintaining the alignment of the vertices of the boundary. A connected mesh is generated with this procedure [Fig. 5(b)].

5.2. Back surface culling and invisible volume culling

Because a cluster is partitioned so as to approach a flat surface, the normal vectors of surfaces of a cluster come to have nearly the same orientation upon partitioning. High-speed calculations for back surface culling become possible by including the distribution range of the normal vectors of a cluster in each node of the tree structure.

For each node of the tree structure, the inner product of the mean normal vector in a cluster and the normal vector of each surface is calculated and the minimum value thereof found. The normal of a surface exists inside the normal cone (Fig. 6) calculated from the normal showing this minimum. The back surface of this cluster can therefore be determined at once by comparing this normal cone and the line of sight during rendering. If it is determined that a node in the hierarchy is a back surface, all child nodes thereof need not be rendered and the rendering process can be accelerated.

The determination of whether a surface is within the field of view can be made quickly and with the same process. Specifically, for each node in the tree structure, a sphere enclosing that cluster is generated in advance and

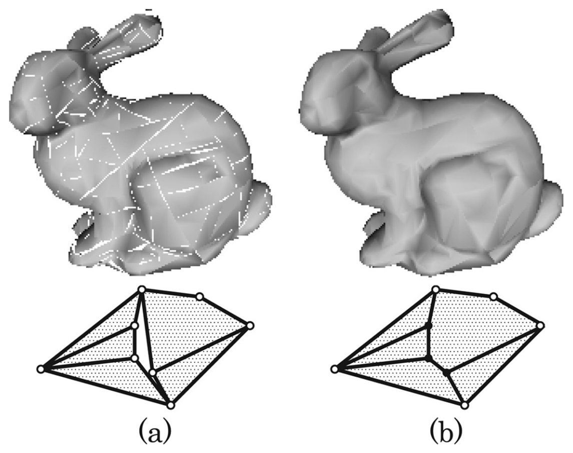


Fig. 5. Polygons connecting.

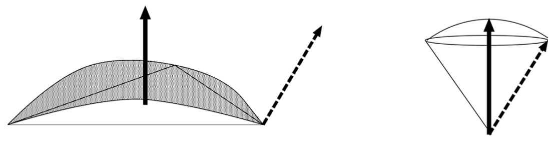


Fig. 6. Normal cone.

stored in the tree structure. During rendering, this bounding volume and the visible volume are compared. If the bounding volume is outside the visible volume, the rendering process is not performed for child and descendant nodes. If within the visible volume, that node and the child and descendant nodes are determined to be within the visible volume. Also if the bounding volume is applied to the boundary of the visible volume, the decision regarding the interior of the visible volume is made for each child node until it is determined whether the bounding volume is entirely within or without the physical volume.

In this way, high-speed rendering can be implemented by efficiently performing back surface culling and invisible volume culling using the hierarchical data structure.

5.3. Occlusion culling

When models overlap within the image, there is a surface which is behind the surface to the front and cannot be seen. The rendering process can be accelerated by removing these hidden surfaces.

Occlusion culling is performed using a visibility test (occlusion query) which is provided in commercially available graphics processors. Using a rough model approximated with a small number of surfaces (Fig. 7), the depth value thereof is written to a z buffer. Afterwards, for each node in the tree structure, a visibility test is performed for the polygons (Fig. 8) enclosing clusters in the image by establishing and expanding the depth value in front of the cluster. If it is determined that the polygon enclosing the cluster is hidden, that node is removed. As the bounding

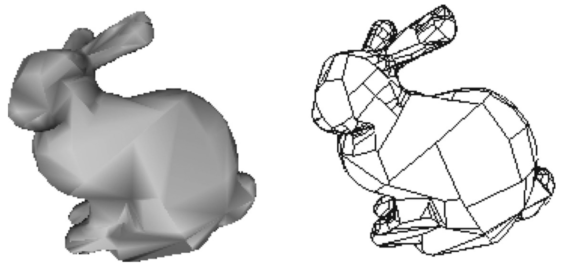


Fig. 7. Example of occluder.

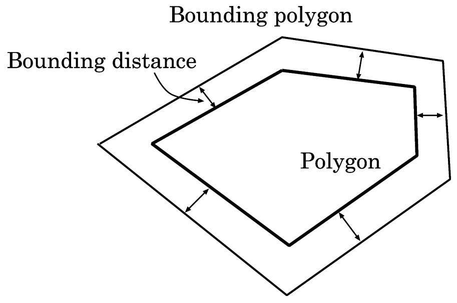


Fig. 8. Bounding polygon.

polygon is attached to the cluster, invisible surface culling can be executed with good efficiency.

6. Experimental Results

A personal computer (CPU: Pentium IV 2 GHz, memory: 4 GB) with a video card (ATI Radeon 9800 Pro) is used in this experiment. The shape data used in this experiment is the data "bunny" comprising 70,000 polygons, published by Stanford University.

6.1. Experiment to generate a hierarchical model

The results of generating hierarchical models using different principal component vectors were compared. Figure 9 shows a graph comparing the mean modeling error per unit area. The case in which both PCA and CCA were used had a lower error compared to both the case of using only PCA and the case of using only CCA. Also, partitioning using CCA generally had a lower error than PCA.

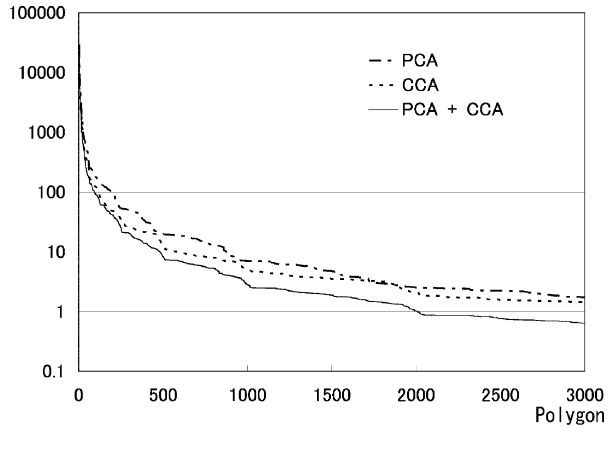


Fig. 9. Approximation error.

Figure 10 shows the results of generating an approximation model comprising 250 polygons. The mean modeling error per unit area was (a) 53.4, (b) 33.7, and (c) 24.4. Case (c) using both PCA and CCA generated polygons more similar to the original model (d) than case (a) using only PCA and case (b) using only CCA.

Figure 11 shows the results of generating approximation models of the bunny using different numbers of polygons. More detailed shapes are expressed as the number of polygons increases.

6.2. Local LOD control experiment

Rendering was performed using local LOD control. Figure 12 shows an image rendered with half the model at a high resolution and the remainder at a low resolution. When the hierarchical data structure generated with the proposed algorithm was used, the different resolutions within the model can be finely established and smooth connections can be made even between areas of different resolution.

6.3. Large-scale model rendering experiment

One hundred copies of the bunny were randomly established within a virtual space and a rendering experiment was performed while changing the viewpoint of a

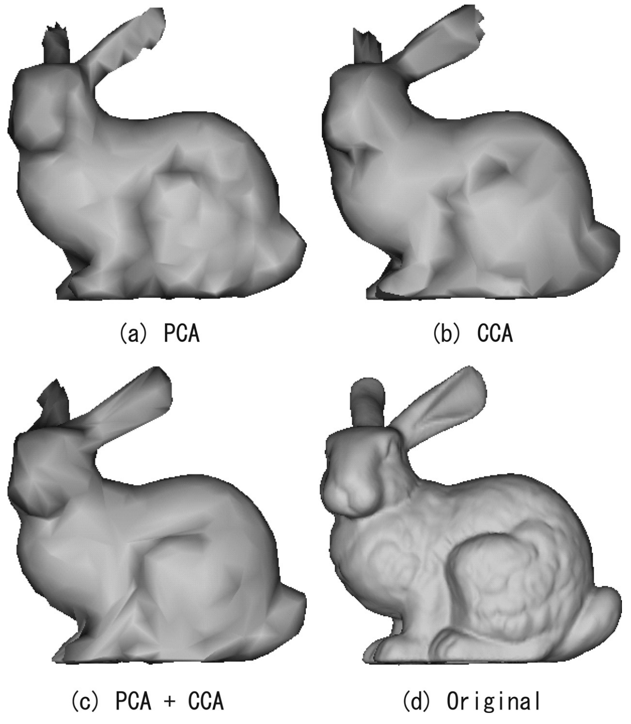


Fig. 10. Approximation model with 250 polygons.

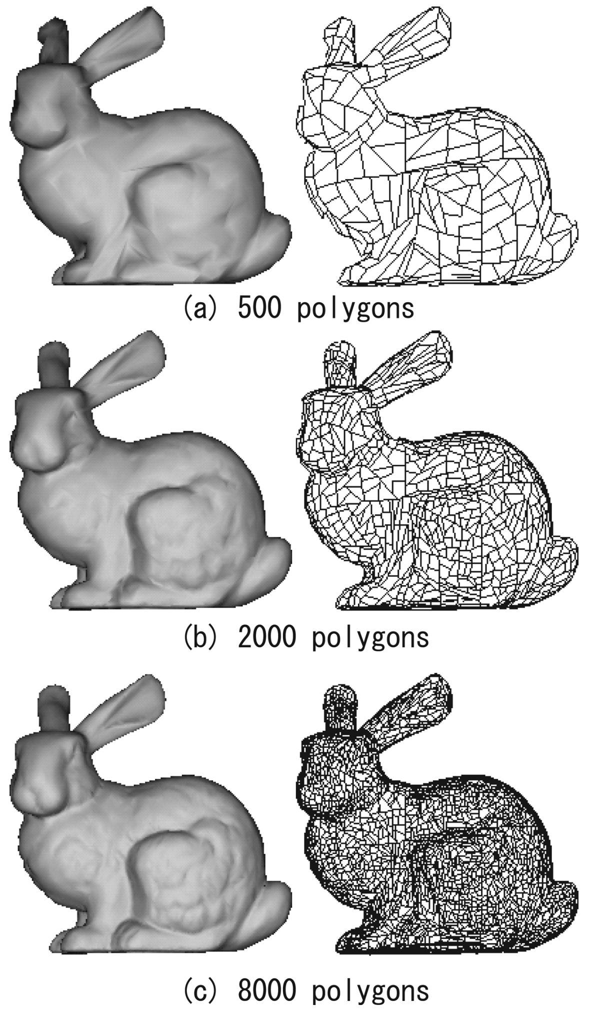


Fig. 11. Approximation models of bunny.

model comprising 7 million polygons. The depth values of a rough model approximated with a small number of surfaces were recorded as in Fig. 15(a). Using those results, occlusion culling was performed to generate the image shown in Fig. 15(b). The case of combining LOD control and invisible surface culling and the case of using only LOD

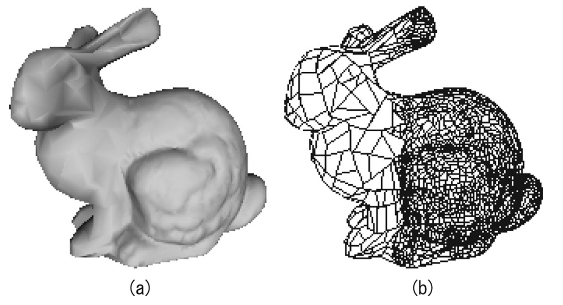


Fig. 12. One model with different resolution parts.

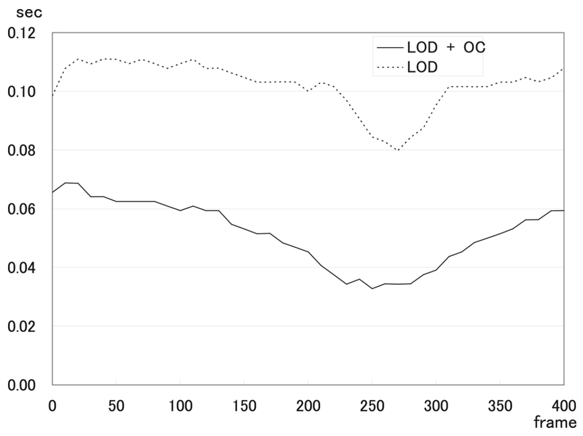


Fig. 13. Rendering time per frame.

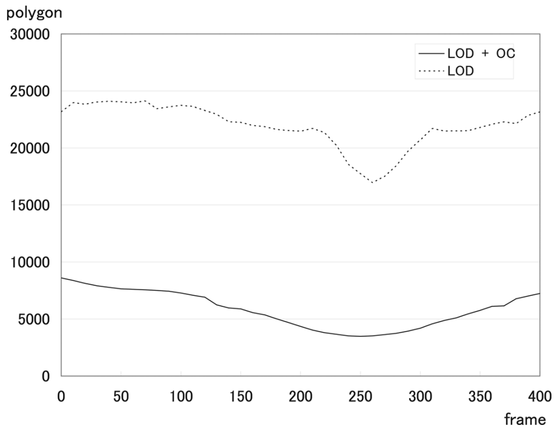


Fig. 14. The number of polygons per frame.

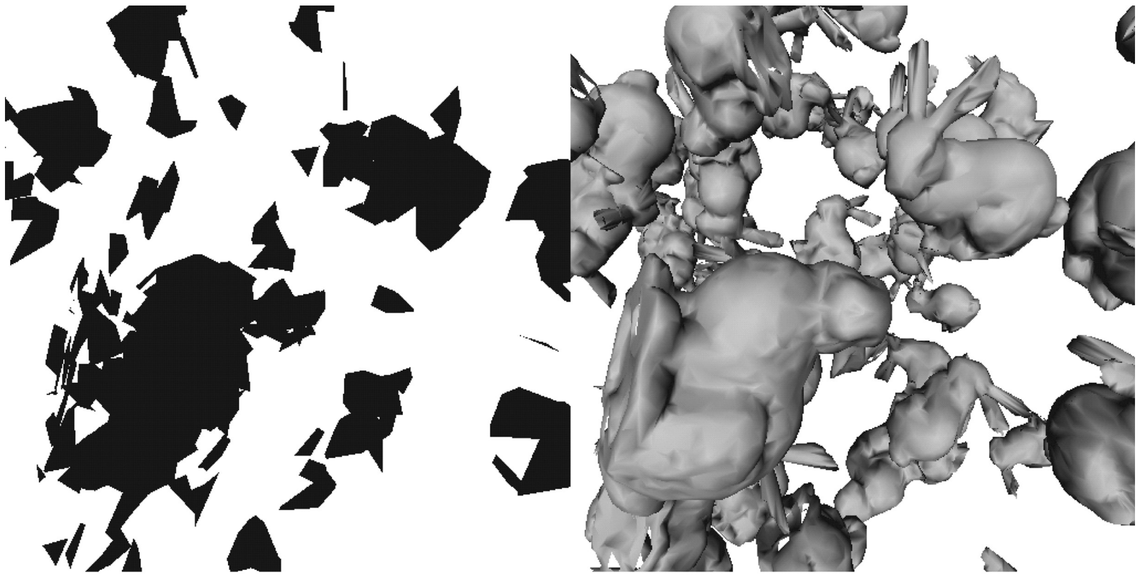
control are shown in Figs. 13 and 14, respectively as graphs of the rendering time per frame and the number of rendered polygons. Rendering at 15 to 30 frames/s is possible when LOD control and invisible surface culling are combined. A simple comparison is not possible because of dependence on the experimental environment, but the model comprising 7 million polygons in real time was displayed at a sufficiently high speed, even in comparison to the conventional algorithm. Meanwhile, in the case of LOD control alone, the frame rate dropped to below 10 frames/s. Also, in the case of only invisible surface culling, a maximum 10-second rendering time per frame was necessary.

When the speed of movement of the viewpoint was multiplied by 100 and the same experiment was performed, the rendering time depended only on the number of polygons rendered and rendering was possible at a speed of 15 frames per second or greater. Consequently, it could be confirmed that high-speed rendering was possible with the proposed algorithm, regardless of the amount of movement of the viewpoint.

Continuous LOD control and conservative invisible surface culling were performed with the proposed algorithm. As a result, artifacts did not occur as in the case where conventional discrete LOD control and approximate invisible surface culling were performed.

7. Conclusions

In this paper, a new hierarchical, three-dimensional data structure, based on recursive face cluster partitioning, was proposed and a high-speed rendering process for large-



(a) Occluder (b) Image

Fig. 15. Result image of rendering.

scale form data using this data structure was implemented. By combining principal component analysis and canonical correlation analysis, clustering was implemented with a smaller modeling error than with the conventional method. By using the proposed hierarchical data structure, the level of detail in any location within the model could be freely controlled. It was also possible to correctly detect and remove invisible surfaces which did not need to be rendered in parallel with the LOD control processing. Consequently, even in the case of a large movement in the viewpoint, an image with few artifacts could be efficiently rendered without a delay. The effectiveness of this algorithm was confirmed with experiments on local LOD control and a rendering experiment for a large-scale model.

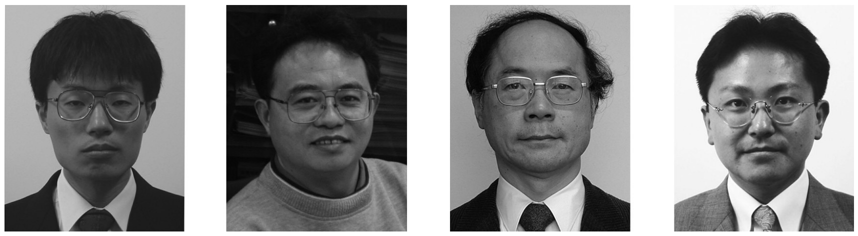
Acknowledgments. A portion of this research was performed as part of the 21st Century COE Program "Forming Fundamental Social Systems in Systems and Information Science" and support was received from the Strategic Information and Communications R&D Promotion Programme, Ministry of Internal Affairs and Communications.

REFERENCES

1. Levoy M, Pulli K, Curless B, Rusinkiewicz S, Koller D, Pereira L, Ginzton M, Anderson S, Davis J, Ginsberg J, Shade J, Fu D. The digital Michelangelo project: 3D scanning of large statues. Computer Graphics (SIGGRAPH 2000 Proc), p 131–144.
2. Hoppe H. Progressive meshes. Computer Graphics (SIGGRAPH '96 Proc), p 99–108.
3. Hoppe H. View-dependent refinement of progressive meshes. Computer Graphics (SIGGRAPH '97 Proc), p 189–198.
4. Luebke D, Erikson C. View-dependent simplification of arbitrary polygonal environments. Computer Graphics (SIGGRAPH '97 Proc), p 199–208.
5. Xia JC, Varshney A. Dynamic view-dependent simplification for polygonal models. Proc IEEE Visualization '96, p 327–334.
6. El-Sana J, Varshney A. Generalized view-dependent simplification. Computer Graphics Forum 1999;18:83–94.
7. Garland M. Quadric-based polygonal surface simplification. Ph.D. thesis. Carnegie Mellon University, CS Dept, Tech Rep CMU-CS-99-105, 1999.
8. Garland M, Heckbert PS. Surface simplification using quadric error metrics. Computer Graphics (SIGGRAPH '97 Proc), p 209–216.
9. Yoshida K, Kitajima K. Adaptive visual field-dependent display of a polygon model. Trans IEICE 2000;J83-D-II:1507–1515.
10. Heckel B, Uva AE, Hamann B. Clustering-based generation of hierarchical surface models. In: Wittenbrink CM, Varshney A (editors). IEEE Visualization '98 Late Breaking Hot Topics. p 41–44.
11. Airey J, Rohlf J, Brooks F. Towards image realism with interactive update rates in complex virtual building environments. Proc Symposium on Interactive 3D Graphics, p 41–50, 1990.
12. Coorg S, Teller S. Real-time occlusion culling for models with large occluders. Proc ACM Symposium on Interactive 3D Graphics, p 83–90, 1997.
13. Hudson T, Manocha D, Cohen J, Lin M, Hoff K, Zhang H. Accelerated occlusion culling using shadow frusta. Proc ACM Symposium on Computational Geometry, p 1–10, 1997.
14. Klosowski J, Silva C. The prioritized-layered projection algorithm for visible set estimation. IEEE Trans Vis Comput Graphics 2000;6:108–123.
15. Klosowski J, Silva C. Efficient conservative visibility culling using the prioritized-layered projection algorithm. IEEE Trans Vis Comput Graphics 2001;7:365–379.
16. Luebke D, Georges C. Portals and mirrors: Simple, fast evaluation of potentially visible sets. Proc ACM Interactive 3D Graphics Conference, p 105–106, 1995.
17. Teller SJ. Visibility computations in densely occluded polyhedral environments. Ph.D. thesis. CS Division, UC Berkeley, 1992.
18. Aliaga D, Cohen J, Wilson A, Zhang H, Erikson C, Hoff K, Hudson T, Stuerzlinger W, Baker E, Bastos R, Whitton M, Brooks F, Manocha D. Mmr: An integrated massive model rendering system using geometric and image-based acceleration. Proc ACM Symposium on Interactive 3D Graphics, p 199–206, 1999.
19. Funkhouser T, Khorramabadi D, Sequin C, Teller S. The ucb system for interactive visualization of large architectural models. Presence 1996;5:13–44.
20. Baxter B, Sud A, Govindaraju N, Manocha D. Gigawalk: Interactive walkthrough of complex 3d environments. Proc Eurographics Workshop on Rendering, 2002.
21. Govindaraju N, Sud A, Yoon S, Manocha D. Interactive visibility culling in complex environments with occlusion-switches. Proc ACM Symposium on Interactive 3D Graphics, p 103–112, 2003.
22. Andujar C, Saona-Vazquez C, Navazo I, Brunet P. Integrating occlusion culling and levels of detail through hardly-visible sets. Computer Graphics Forum (Proc Eurographics 2000) 1999;19:681–692.

23. El-Sana J, Sokolovsky N, Silva C. Integrating occlusion culling with view-dependent rendering. Proc IEEE Visualization 2001, p 371–575.
24. Yoon SE, Salomon B, Manocha D. Interactive view-dependent rendering with conservative occlusion culling in complex environments. Proc IEEE Visualization 2003, p 163–170.
25. Garland M, Willmott A, Heckbert P. Hierarchical face clustering on polygonal surfaces. Proc ACM Symposium on Interactive 3D Graphics, p 49–58, 2001.
26. Kalvin AD, Taylor RH. Superfaces: Polygonal mesh simplification with bounded error. IEEE Comput Graph Appl 1996;16:64–77.
27. Okuno T, Kume H, Iga T, Yoshizawa T. Multivariate analysis. JUSE; 1971.

AUTHORS (from left to right)



Tokuo Tsuji (active member) received a B.S. degree in electrical engineering and computer science from the Department of Engineering, Kyushu University, in 2000 and a Ph.D. degree in intelligent systems from the Graduate School of Information Science and Electrical Engineering in 2005. Since then, he has been a COE Researcher at the Graduate School of Engineering, Hiroshima University. His research interests include researching computer vision and computer graphics. He is a member of the Robotics Society of Japan and the Japan Society of Mechanical Engineers.

Hongbin Zha received a B.S. degree in electrical engineering from Hefei University of Technology (China) in 1983 and enrolled in the Kyushu University Graduate School of Engineering in 1985, receiving a Ph.D. degree in 1990. He then became an assistant in the Computer Science Department, an assistant professor in the Engineering Department in 1991, and an assistant professor of research in the Graduate School's Information Science and Electrical Engineering Department in 1996. Since 2000, he has taught at the Information Science Center of China's Beijing University. He holds a doctorate. He is engaged in research on robotics, robot vision, and the modeling and recognition of three-dimensional models. He is a member of the Information Processing Society of Japan and the Robotics Society of Japan.

Tsutomu Hasegawa (active member) received a B.S. degree in electron physics from Tokyo Institute of Technology in 1973 and became a researcher at the National Institute of Advanced Industrial Science and Technology. Since 1992 he has been a professor at the Graduate School of Information Science and Electrical Engineering, Kyushu University. His research interests include intelligent robots. He holds a doctorate, and is a member of the Society of Instrument and Control Engineers, the Institute of Electrical Engineers, and the Japan Society of Mechanical Engineers.

Ryo Kurazume received an M.S. degree in mechanical physics engineering from Tokyo Institute of Technology in 1991 and joined Fujitsu Laboratories. In 1995, he became an assistant in the Department of Mechano-Aerospace Engineering at Tokyo Institute of Technology. In 2000, he was a guest researcher at Stanford University and also engaged in doctoral research at the University of Tokyo Institute of Industrial Science. He has been an assistant professor at the Graduate School of Information Science and Electrical Engineering, Kyushu University, since 2002. His research interests include multirobot systems, walking mechanics, and laser measurement. He is a member of the Robotics Society of Japan and the Japan Society of Mechanical Engineers.