Volume visualization of 3D finite element method results

by K. Koyamada
T. Nishio

This paper describes a method for visualizing the output data set of a 3D finite element method result. A linear tetrahedral element is used as a primitive for the visualization processing, and a 3D finite element model is subdivided into a set of these primitives, which are generated at every solid element. With these primitives, isosurfaces are visualized semitransparently from scalar data at each node point. Two methods are developed for the visualization of isosurfaces with and without intermediate geometries. The methods are applied to output data sets from some simulation results of a semiconductor chip. These are visualized, and the effectiveness of the method is discussed.

Introduction

The 3D finite element method (FEM) has recently become very important in the manufacturing industry as a result of the increase in computing power and the progress of numerical computation technologies. This analysis produces very large volumetric data sets on unstructured grids, and the effective visualization of these data sets has become an important problem. However, the current postprocessors for the 3D FEM compel users to imagine the volumetric distribution only from cutting planes or exterior surfaces. It is necessary to migrate from "volume imagination" to "volume visualization," because the 3D FEM results are essentially volumetric, and volume visualization helps users to optimize a mechanical design from a 3D FEM result. In volume visualization, an isosurface, which is a set of points with the same scalar value, plays a very important role, because it can represent complex features that are difficult to display by using several cutting planes.

Recently, visualization technologies for isosurfaces have been studied intensively, especially in the medical imaging area [1–5]. Lorensen and Cline have developed a "marching cubes" method using a look-up table for surface classification [5], and Levoy has reported a "volume rendering" method incorporating such a classification into the projection process [1]. To be applied to 3D FEM result visualization, these methods must employ a gridding operation, because they assume that data sets are defined on a structured grid. However,

it is very difficult to arrange an adequate structured grid for the operation. To avoid this difficulty and maintain accuracy in the data, we use the original grid data on which a 3D FEM result is defined. Recently, Gallagher and Nagtegaal have applied an extension of the marching cubes method to 3D FEM results, using the original grids and visualized isosurfaces [6]. In the case of a linear element, this method yields an effective visual representation of slope-continuous surface segments. But in the case of a higher-order element, because a linear interpolation cannot be applied to the intersection search, two or more intersections between an isosurface and a finite element may be calculated on the edge line, which makes a look-up table very complex. To effectively fit a linear interpolation to an isosurface visualization, we introduce a concept of element subdivision, namely *tetrahedronation*, following the generation of new node points inside the element, and propose the volume visualization of 3D FEM output data sets, using a tetrahedral cell as a process primitive. In tetrahedronation, various kinds of elements such as a linear wedge element, a linear brick element, and a parabolic element are reconstructed into linear tetrahedral elements to generate primitives. In the following section, we discuss tetrahedronation for this primitive generation in order to generalize the visualization method on the basis of a tetrahedral primitive. In the field of chemical graphics, Koide and Doi have used this primitive to visualize isopotential surfaces from structured grid data [7]. Here, we use it for the visualization of unstructured grid data.

The tetrahedron has been used in many methods for generating grids from objects of arbitrary shape, because of its geometric flexibility [8–11]. Phai has reported a technique for connecting manually defined nodal points to a network structure consisting of tetrahedral grids which have an optimum form for numerical computation [8]. Yerry and Shephard have presented a method which follows from the basic concepts of the octree encoding technique to generate tetrahedral grids completely automatically [9]. Shenton and Cendes have proposed a method based on an extension of the Delaunay triangulation algorithm to 3D geometry [10]. The geometrical flexibility of tetrahedral grids is so attractive in terms of automatic grid generation that these grids are beginning to be used in 3D FEM analysis [12]. Consequently, using a tetrahedral cell as a process primitive is very effective. In terms of volume visualization, this primitive has the following features:

1. Linear data distribution on the edge line.
2. Linear data distribution along any line segment.

The first feature, which is included under the second, makes it possible to find intersections of edges of a primitive and an isosurface by using a very simple look-up table; this led us to develop a method for isosurface visualization based on triangular facets. The second feature allows a point on the isosurface along the viewing ray to be found easily by using the values on the entry point to the primitive and the exit point from it; this led us to develop a method for the direct imaging of isosurfaces.

We apply the latter method to the thermal stress analysis of a solder joint in a semiconductor chip, and visualize isothermal surfaces and isostress surfaces. The method can also be applied to simulation results from the 3D finite difference method (FDM) based on the boundary-fitting coordinate (BFC) system.

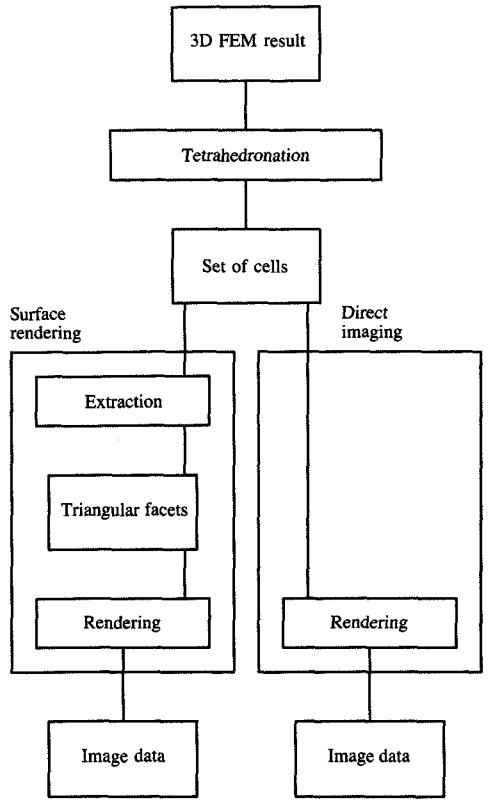


Figure 1
Overview of the method.

Overview of the method

An overview of our method is presented in **Figure 1**. We begin with a 3D FEM result defined at each node, as

described in the third section. When the result is defined at each element, we can convert it to the result at each node point by extrapolating and then taking the average of the values computed at parametric integration points within the elements, using the interpolation function of the element. The first step is tetrahedronation, described in the third section, which converts each solid finite element into several tetrahedral cells. The output of this step is a set of tetrahedral cells with scalar data at each node. Two independent steps, a surface-rendering step and a direct-imaging step, use this set as input. In the surface-rendering step, described in the fourth section, triangular facets are first extracted as intermediate geometries and then rendered. In the direct-imaging step, described in the same section, this set is directly rendered by a volume ray-tracing method. Each step produces image data as output.

Data of 3D FEM result

Because the data structure of a 3D FEM result is not intended for volume visualization, it does not allow efficient visualization unless modified. In this section, we first describe the characteristics of the data structure of 3D FEM results and investigate the data structure needed for efficient visualization. We then describe how FEM result data are interpolated, and explain the effectiveness of using tetrahedral cells in volume visualization.

- *Data structure*

To solve a partial difference equation such as a Navier–Stokes equation, we often use an approximate solution which converts the equation into an algebraic form. The weighted residual method, which is often used in numerical simulation, builds algebraic equations by integrating the weighted error over the whole region.

In FEM analysis, the above integration is performed at each subregion, called an element, and subalgebraic equations are formed. Finally, whole algebraic equations are constructed by superposing the subequations on a whole matrix. The superposition in FEM analysis allows the data structure to independently express the element topological data simply by using the element–node relation. Such an expression of an element does not offer enough information for the volume visualization of 3D FEM data.

Volume visualization includes processes on edges or faces of an element and the traversal of elements along a viewing ray. For example, if the generation of points on an edge or face is performed independently at each element, duplicate calculations may occur if the edge or face is shared by other elements. Without a list of adjacent elements, ray-tracing spends much CPU time searching for the element which the viewing ray encounters next.

One solution to the problem is to rebuild the data structure so that it involves an element–face relation, a face–edge relation, an edge–node relation based on a boundary representation (B-Rep), and a list of adjacent elements. Such a data structure is often introduced in solid-modeling systems, but it is not realistic for volume visualization because many more elements must be processed than solids. To identify an edge or a face which can be shared by elements, we adopt hashing [13]. As a preprocess for ray-tracing, we generate a list of adjacent elements called a link table.

- *Data interpolation*

For effective visualization, attention must be given to data interpolation at an arbitrary point in an element, as well as to the data structure. In FEM analysis, the data in an element are interpolated from nodal data by using some function, which is expressed not in a global but in an element–local coordinate system. Because an arbitrary point in the element is expressed in the global system, the point coordinates should be transformed into the local system. This transformation is performed by iterative calculation based on the Newton method, the convergence of whose calculation depends on the initial value. If the value is not adequate, the calculation diverges. Since it is very difficult to forecast the initial value of a given point, especially in a curved element, the data should be interpolated in global coordinates in order to avoid a transformation that may be accompanied by instability and low performance.

From this viewpoint, a linear tetrahedral element (a tetrahedral cell) is suitable, because the data in this cell are directly interpolated in a global system. Moreover, in a tetrahedral cell, the data distribution is linear in any direction, which makes ray-tracing very efficient. For this reason, we use the tetrahedral cell as a process primitive for volume visualization. Usually, various kinds of element are mixed in a 3D FEM result. In this case, we subdivide these elements into a number of tetrahedral cells in a preprocess that we call tetrahedronation.

In the visualization of 2D FEM results, elements are often subdivided into linear triangular elements for contour plotting [14, 15]. Our approach is a natural expansion of this subdivision to 3D results. Converting all elements into tetrahedral cells fixes the numbers of nodes composing an element and the adjacent elements at (4,4), which is very desirable in terms of the efficient use of memory resources.

- *Tetrahedronation*

Koide and Doi have developed a cell-based tetrahedronation method for efficient extraction of an isosurface as a set of triangular facets [7]. They subdivide each grid cell independently into five tetrahedral cells

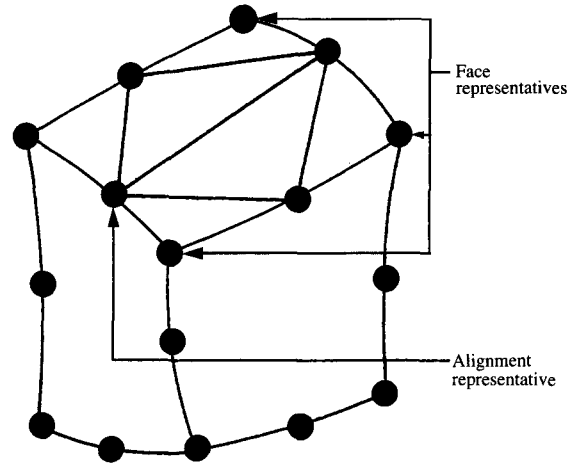
because, for a regularly ordered grid, there are some rules for subdivision which warrant alignment with the face of the adjacent cell. Since elements are not arranged regularly in 3D FEM data, independent tetrahedronation can cause misalignment with the face of an adjacent cell. For correct alignment, an element should be subdivided after the information has been obtained about the face of the adjacent element. As we stated previously, we use a hash table for this purpose.

Let us take a parabolic brick element as an example for tetrahedronation (see **Figure 2**).

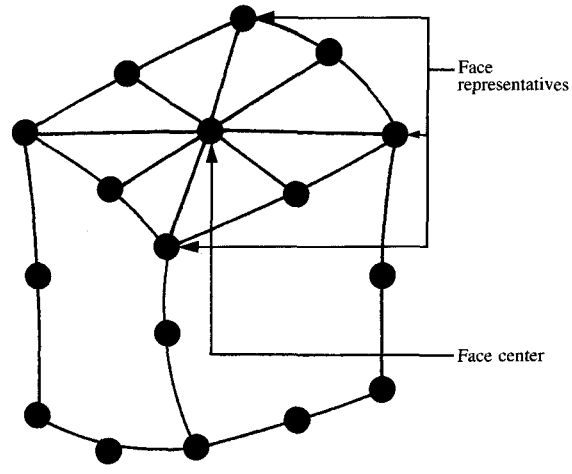
First, each face of this element is subdivided into six triangles. Because there are two alignments in this subdivision, we should register which alignment is generated on the face in the hash table. The identifiers of one of two nodes to which five edges are attached can be made to represent the alignment. As entries in the hash table, we also need node identifiers to represent a face itself. It is sufficient to select the identifiers of the first three out of four main nodes in descending order of value. Because we chose the sum of these identifiers as a hash value, we actually register their maximum and the minimum values in the hash table.

Besides six-triangle subdivision, we can consider an eight-triangle subdivision accompanying an additional node generation at the center of a face. This subdivision is axisymmetric about the face center at every ninety degrees, so no misalignment of the subdivision occurs. The identifier of the additionally generated node must be registered in order to avoid duplicate generation of a node.

Next, we independently generate an additional node at the center of a volume, which does not require hashing. Finally, we generate tetrahedral cells by connecting the generated node and the vertices of 36 or 48 triangles. For other types of elements, we also generate tetrahedral cells in a similar manner. In **Figure 3**, we show the face subdivision related to the tetrahedronation of various kinds of elements.



(a)



(b)

Figure 2

Tetrahedronation of a parabolic brick element: (a) six-triangle subdivision; (b) eight-triangle subdivision.

Isosurface visualization

An isosurface, which is a set of points with the same value in 3D space, is very useful for interpreting volume data such as 3D FEM results. Our goal is to visualize isosurfaces from a set of tetrahedral cells. There are two approaches to visualization. One is to render polygonal geometries which are locally extracted as isosurfaces at each cell. In this approach, we calculate the intersections of a cell with the surface at the edges of the cell, and compose polygons whose vertices are the intersections. The other is to use a method based on ray-tracing, in which we calculate intersections along a viewing ray and their pixel values directly without creating intermediate geometries such as polygons. We develop a volume ray-tracing method for tetrahedral cells and apply the method to the visualization of isosurfaces. First, we explain how to calculate normal vectors, which are used for the shading of isosurfaces. We then give two methods based on two approaches, a triangular-facet method and a direct-imaging method.

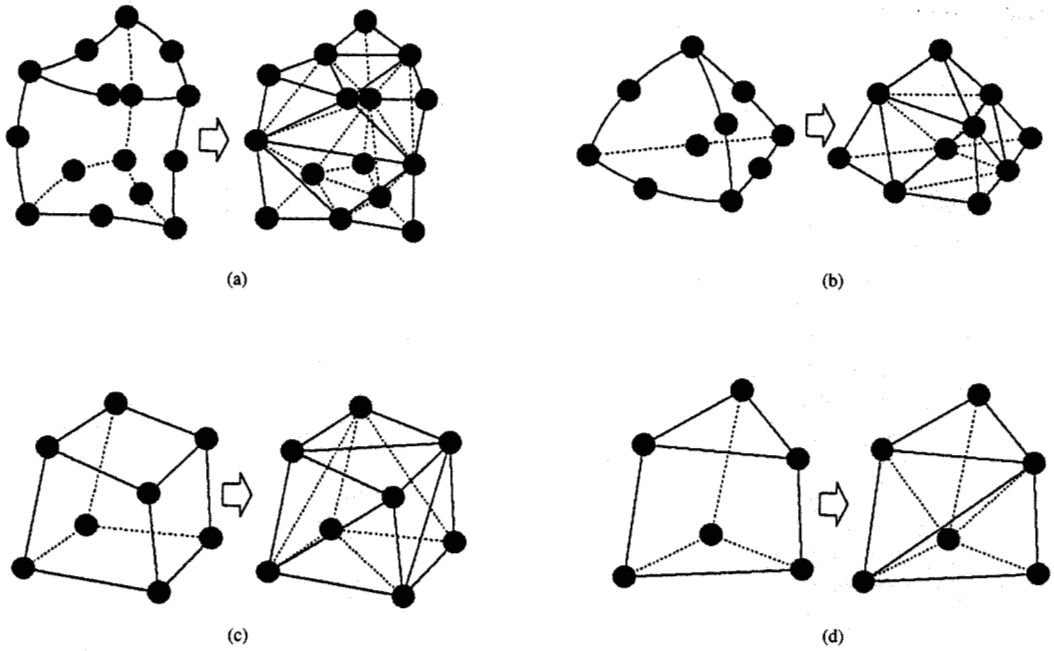


Figure 3

Face subdivision of various element types: (a) parabolic wedge; (b) parabolic tetrahedral; (c) linear brick; (d) linear wedge.

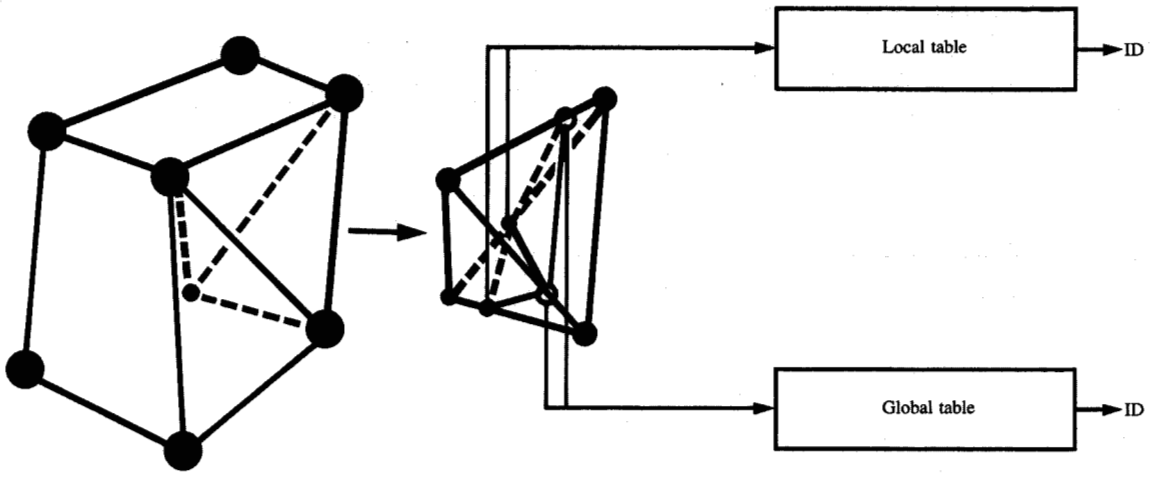


Figure 4

Hashing by a local table.

• *Normal vector calculation*

The shading of an isosurface is synthesized by using normal vectors and some illumination model. We can calculate these normals as normalized gradients of a given function, which is defined at each cell. In a tetrahedral cell, the function is

$$F(X, Y, Z) = a_0 + a_1X + a_2Y + a_3Z,$$

where the coefficients a_j ($j = 0, 3$) are determined by the nodal coordinates and data. The normal N can be calculated as follows:

$$N = \text{grad}\,[F(x, y, z)].$$

The normals in the cell are all the same, because partial derivatives of F with respect to x, y, and z are constant. This means that a local isosurface in this cell is a plane. The use of a normal at each cell leads to the isosurface having a constant shading image, which is not good for expressing the smoothness of the surface. To avoid this disadvantage, we convert the vector of a cell into the vector at each node. As a result, normals are stored at every node, and in a cell, a normal is interpolated through four normals at nodes. This conversion is also advantageous in terms of memory utilization. The number of cells is larger than the number of nodes, because the number of cells connecting a node is usually greater than four. Therefore, storing a normal at each node requires less memory space than doing so at each cell.

• *Triangular-facet method*

This method represents an isosurface as a set of triangular facets that are extracted from a cell. The process is as follows:

1. Search for edges which intersect the isosurface.
2. Calculate the intersections by linear interpolation.
3. Connect the intersections to form triangular facets.

In general, an edge is shared by several cells. To give a unique identifier to each intersection and avoid duplicate calculations on the edge, hashing is introduced. The method is basically inspired by that of Koide and Doi, in which a grid is subdivided with only the given grid points. Since in our method an element is subdivided into tetrahedral cells along with the generation of an additional node, it is possible to generate a cell with an edge whose vertex is this node. It is efficient to have a local table for hashing at each element, because hashing with respect to the intersection at the above edge can be localized. We prepare a table at each element. In the case of the element shown in **Figure 4**, the maximum number of rows in the table is eight, because there are eight edges which extend to nodes from the volume center. If we take one cell from the element which contains two triangles of an isosurface, we give unique identifiers to two vertices at these edges by using the local table and to two vertices at the other edges by using the global table.

• *Direct-imaging method*

The above representation of isosurfaces sometimes requires much memory space for polygonal geometries such as triangular facets; this is one of the motives for development of the volume-rendering method. We have developed a method for directly rendering not only polygonal geometries but also tetrahedral cells by volume ray-tracing. First, we propose a tetrahedral model for efficiently traversing an unstructured collection of cells. Next, we explain the volume ray-tracing method based on this model and its application to isosurface visualization.

Tetrahedral model

The tetrahedral model is composed of a node table, a cell topological table, a link table, and an exterior face table, which are intimately connected to one another. The node table is an array of coordinates, normal vectors, and scalar data. The cell topological table is an array of cells, each of which is represented by the identifiers of its four nodes (cell–node relation). The node identifiers are chosen according to their addresses in the node table. The link table is another array of the same cells, each of which is represented by the identifiers of the four cells connected to its four faces (cell–cell relation). The cell identifiers are chosen according to their addresses in the cell topological table. If there is no cell connected to a face, the exterior face identifier, the sign of which is changed, is registered instead. The exterior face table is an array of faces, each of which is represented by the identifiers of the cells connected to it and a face number. This number is an internal face identifier of a cell. When the node identifiers of a tetrahedral cell are n_1, n_2, n_3, and n_4, a face labeled n ($n = 1, 4$) is defined by the following face–node relations:

$$\text{face } 1 = (n_2, n_3, n_4),$$

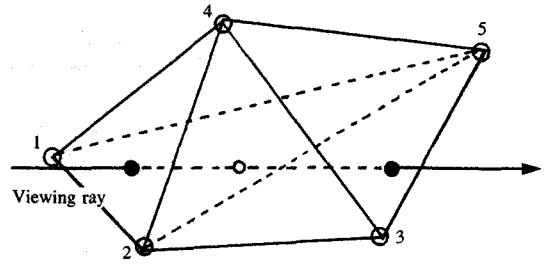
$$\text{face } 2 = (n_3, n_4, n_1),$$

$$\text{face } 3 = (n_4, n_1, n_2),$$

$$\text{face } 4 = (n_1, n_2, n_3),$$

where n denotes an internal face identifier. The exterior face table and the cell topological table realize two-way linkage between an exterior face and a tetrahedral cell.

Figure 5 shows a tetrahedral model which consists of two tetrahedral cells. A cell labeled 1 is composed of nodes labeled 1, 5, 4, and 2 and connected to a cell labeled 2 at face 1. Faces 2, 3, and 4 are exterior faces



(a)

Cell table

Cell topological table

Cell ID	Node 1	Node 2	Node 3	Node 4
1	1	5	4	2
2	2	4	5	3

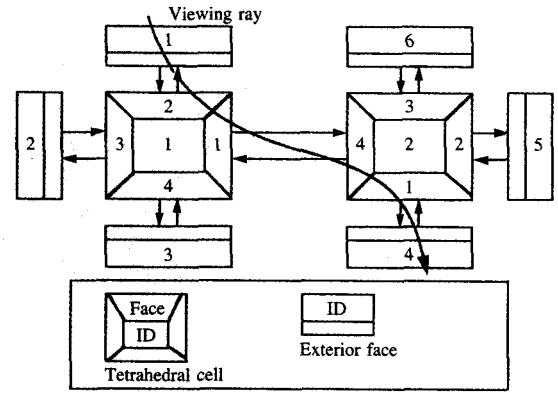
Cell link table

Face 1	Face 2	Face 3	Face 4
2	−1	−2	−3
−4	−5	−6	1

Exterior face table

Face ID	Cell ID	Face number
1	1	2
3	1	4
5	2	2
2	1	3
4	2	1
6	2	3

(b)



(c)

Figure 5

(a) Connected cells; (b) tetrahedral model; (c) schematic expression.

labeled 1, 2, and 3. A cell labeled 2 is composed of nodes labeled 2, 4, 5, and 3 and connected to a cell labeled 1 at face 4. Faces 1, 2, and 3 are exterior faces labeled 4, 5, and 6. The viewing ray first intersects the model at the exterior face labeled 1. By referring to the exterior face table, we find it is face 2 of the cell labeled 1. The viewing ray exits from face 1, which is found to be identical to face 4 of the cell labeled 2 by referring to the link table. Next, it exits from face 1 of the cell labeled 2, which is found to be identical to the exterior face labeled 4. We illustrate this traversal schematically in the lower part of the figure.

Volume ray-tracing

The volume ray-tracing method based on the tetrahedral model can be divided into two processes. One is searching for the cell which the viewing ray encounters first, and the other is calculating the brightness in the traversal of the model.

The first process, in other words, is searching for the nearest exterior face to the viewing point, which is an application of hidden-surface removal. Many algorithms for this have already been developed, and our purpose is not to develop more. We therefore adopt the Z-buffer algorithm to obtain the exterior face identifier, and determine the cell by referring to the exterior face table. Reference to the link table enables us to search for cells which intersect the viewing ray successively until the ray reaches an exterior face.

In the next process, we use the following brightness equation in each cell:

$$B = \sum_{i=1}^{n} [B_i \, a_i \prod_{j=1}^{i=1} (1 - a_j)] \, .$$

The term B_i denotes the brightness which is calculated from the interpolated normal at a sampling point and from Phong's model [11], and a denotes the opacity which is calculated from some transfer function of the interpolated scalar data and gradient. (B_1, a_1) and (B_n, a_n) are the brightness and opacity at the exterior faces (see **Figure 6**). At a sampling point in a cell which is D_i apart from the viewing point, we calculate (B_i, a_i) as follows:

1. Interpolate the normals and scalar data at the entry and exit points.
2. Interpolate the normals and scalar data at the sampling point. Note that the weights are $(D_{ext} - D_i)/(D_{ext} - D_{ent})$ and $(D_i - D_{ent})/(D_{ext} - D_{ent})$, where D_{ent} and D_{ext} are the distances of the entry and exit points from the viewing point, respectively.
3. Calculate (B_i, a_i) from the interpolated normal and scalar data.

By performing the processes at each viewing ray, we can create a volume-rendering image into which the exterior surface-rendering image has been integrated.

Application to isosurface visualization

To visualize isosurfaces by the volume ray-tracing method, we usually raise the local opacity around the visualizing data in general. However, there is a slight loss of surface sharpness because the sampling points are not always on the isosurface. There are many requirements

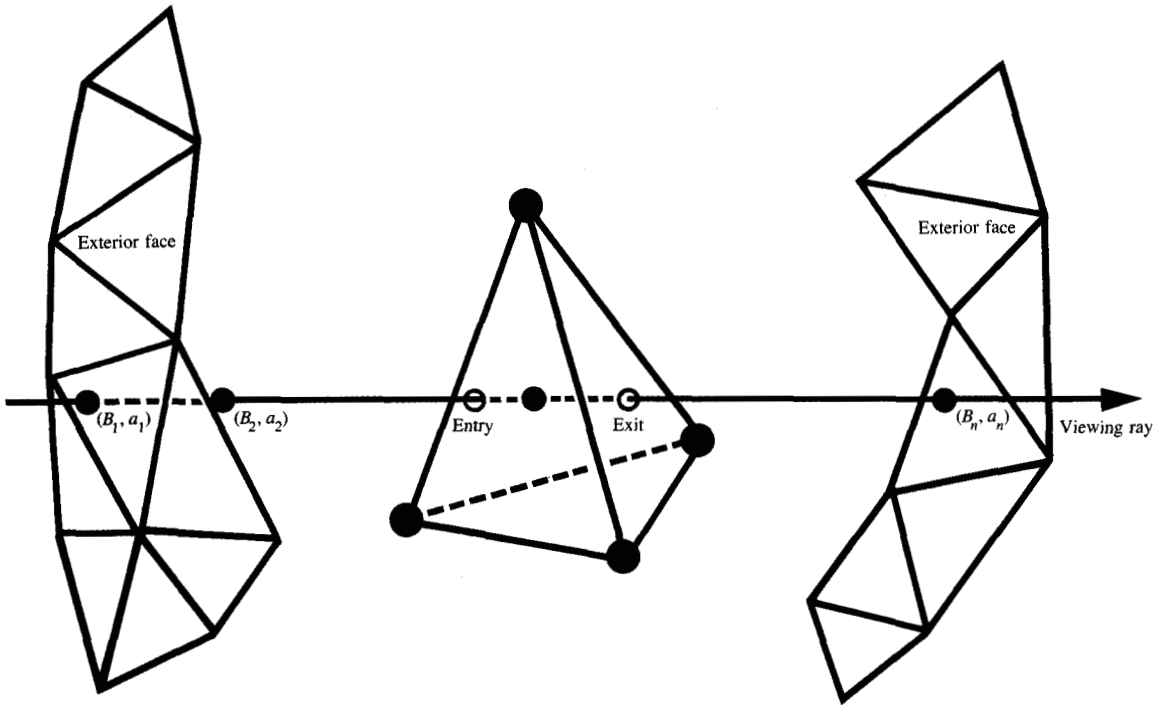


Figure 6

Sampling points on a viewing ray.

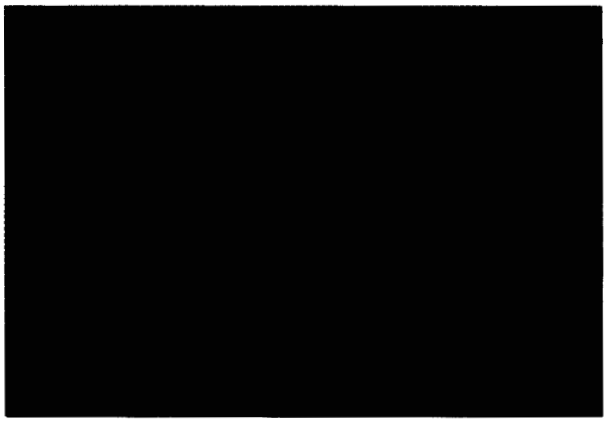
for achieving sharpness in visualizing the result of a numerical simulation such as a 3D FEM result; we achieve it by confining the sampling points to the intersections between the viewing ray and isosurfaces. This approach is efficient because the number of sampling points is significantly reduced. In **Figure 7**, we show two images which are created by the volume ray-tracing method from the same pressure data in a clean room. The upper one is based on uniform sampling with the opacity in proportion to the data, and the lower one is based on sampling only at intersections.

Application to chip design

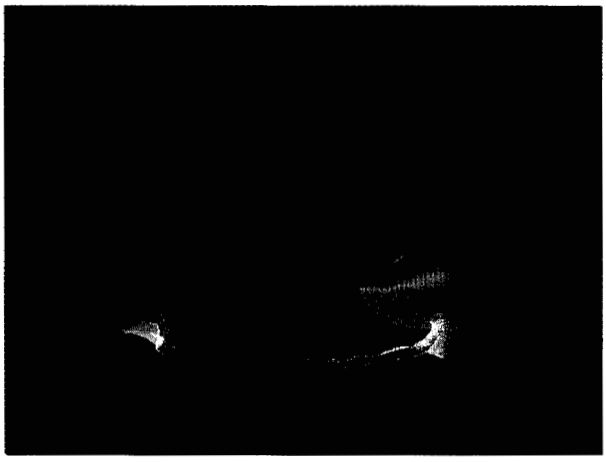
We have applied our visualization method to the packaging of a semiconductor chip which is based on controlled collapse chip connection (C4) technology. First, we present a general view of this technology. Next, we explain the model used for the FEM analysis, and finally, we discuss the effectiveness of our visualization method for interpreting 3D scalar data.

• *C4 technology*

C4 is a technology for attaching silicon circuit chips to a ceramic substrate through a tin–lead solder joint, as shown in **Figure 8** [16–18]. This technology achieves high packaging density; however, there is a problem with the reliability of C4, namely that fatigue can occur in a solder joint because of the thermal stress introduced by a difference in coefficient of thermal expansion (CTE) between the chip and the substrate. The process leading to fatigue in standard C4 technology can be explained as follows: When a chip is active, the difference in CTE between the chip and the substrate results in a shear strain between the top and the bottom of a solder joint as the temperature of the chip increases. Because the displacement is removed when the chip becomes inactive, the power-on–off cycle imposes a cumulative strain on the solder joint. When the equivalent strain exceeds a critical value, a fracture appears in the portion of the solder joint where the strain concentration occurs, and this finally results in fatigue. Therefore it is desirable



(a)



(b)

Figure 7

Pressure distribution.

that the CTE of the substrate, which is 2.7–3.0 ppm/°C, be close to that of the chip. But, for reasons of productivity and cost, alumina, whose CTE is 6.5–9.0 ppm/°C, is chosen to be used as the substrate in production. Because the CTE of the substrate is more than twice the CTE of the chip when the temperature of each material increases, the chip size and the thermal environment are restricted to avoid fracture. Furthermore, Goldmann has reported [18] that the fatigue life of the solder joint based on C4 technology depends on its diameter and height. If such restrictions can be removed, C4 technology will have potential applications in many areas.

One technique for enhancing the reliability of C4 technology is to encapsulate solder joints between the chip and the substrate with epoxy resin, as shown in **Figure 9**.

In the encapsulated solder joint, the stress distribution is caused not only by the shear strain but also by the compressive/tensile strain resulting from the expansion/shrinkage of the epoxy resin with the change in temperature. Because the strain distribution is too complex to analyze theoretically, FEM analysis is needed to investigate the strain distribution of the new structural design.

• *FEM model*

This new C4 technology is applied to the memory chip. Several chips are connected on a circuit card. For simplicity, only one chip is treated as a finite element model. The model can be described as a quarter of a chip, because the location of the solder joints is symmetrical with respect to the A–A surface and B–B surface, as shown in **Figure 10**. The model is defined as the area occupied by a chip. The defined FEM model is composed of 4155 nodes (including 455 for the shell), 2740 linear bricks, and 1048 linear wedges (see **Figure 11**).

Zero-displacement boundary conditions have been imposed along the center surfaces A–A and B–B, and heat sources have been placed under the surface of the chip. Convection surfaces have been imposed on top of the chip and the card, and under the card. The side surfaces are assumed to be adiabatic. The solder joint has material properties that are highly dependent on temperature and time. Therefore, in FEM analysis, the material property of the solder joint is assumed to be plastic and dependent on temperature. That of the other portion is assumed to be elastic and dependent on temperature.

• *Result*

Figure 12 shows the steady-state thermal contour map after the chip has begun to consume 0.625-W power. Temperature values at each node point are needed to set up a load condition for stress/strain analysis after the chip has been activated, and to evaluate a cooling effect for the chip that does not endure beyond 85°C. Although the thermal distribution on an exterior surface can be understood from this image, the volumetric distribution inside the chip cannot be determined. **Figure 13** shows the corresponding isothermal surfaces in the encapsulation and the joints. This visualization makes it easier to understand the internal thermal distribution. To understand such volumetric distribution in detail, we should effectively express the depth information of isosurfaces in a graphic display. For this purpose, several images should be displayed from two or more viewing points. We introduce the following two methods for this:

- Wire-frame display using local graphic functions such as zooming, rotation, and translation.

- Stereoscopic display using a double-buffering mechanism.

The first is used in order to understand the volumetric distribution at high speed in routine work. The second is used in order to examine the depth in detail, displaying two shaded images produced by changing the viewing position.

Figure 14 shows the Von Mises stress contour map which corresponds to the previous thermal distribution. Our concern is to understand the stress distribution in the encapsulation surrounding solder joints. In this case, the contour map on an exterior surface does not provide sufficient information. **Figure 15** shows the Von Mises stress surfaces obtained by our visualization method. Note particularly that there is an isolated region with a high stress value in the upper left corner. Naturally, it can be expected that there will be a high possibility of a fracture at the solder joint nearest to this region, which agrees qualitatively with experimental results. Furthermore, we can understand that the stress value is not simply proportional to the distance from the center of a chip, which is another reason why FEM analysis is needed for the development of encapsulated C4 technology.

Figure 16 shows the strain contour on the outer surface of the solder joint previously mentioned, and **Figure 17** shows the corresponding isostrain surfaces. The latter indicates that surfaces with high strain are attached to the edge of the solder joint, where a fracture may occur.

Through 3D FEM analysis and volumetric visualization, we have reached the first step in developing an advanced chip based on C4 technology, which may be summarized as follows:

- We found a solder joint which can be heavily damaged.
- We found the portion of the above joint in which a fracture may occur.

Discussion

We have developed two methods for visualizing isosurfaces from a 3D FEM result. In each method, all elements are subdivided into tetrahedral cells. If elements to be processed are confined to linear elements, the tetrahedronation may be omitted in the triangular-facet method, which is based on the linear distribution only along the edges of the cells. However, when higher-order elements are processed, tetrahedronation is very important, since it converts a complex data distribution in these elements into a linear distribution set. Moreover, it is only in a tetrahedral cell that data are linearly distributed along an arbitrary line segment. In this sense, tetrahedronation is indispensable for the direct-imaging method.

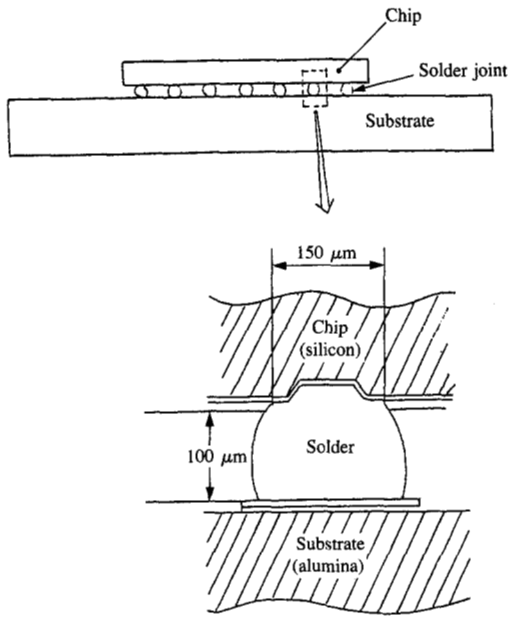


Figure 8

Standard C4 structure.

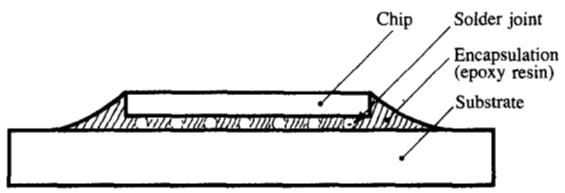


Figure 9

Encapsulated C4 structure.

The triangular-facet method allows the benefits of hardware technology to be enjoyed in polygon-based graphics, because a triangle is a basic primitive of graphics hardware. These benefits lead to interactive manipulation of isosurfaces, such as selecting one of the vertices of which an isosurface is composed, and inquiring about its coordinates. This advantage originates in the explicit extraction of triangular facets, which has the disadvantage that a very large area of memory or DASD space may be required for such intermediate

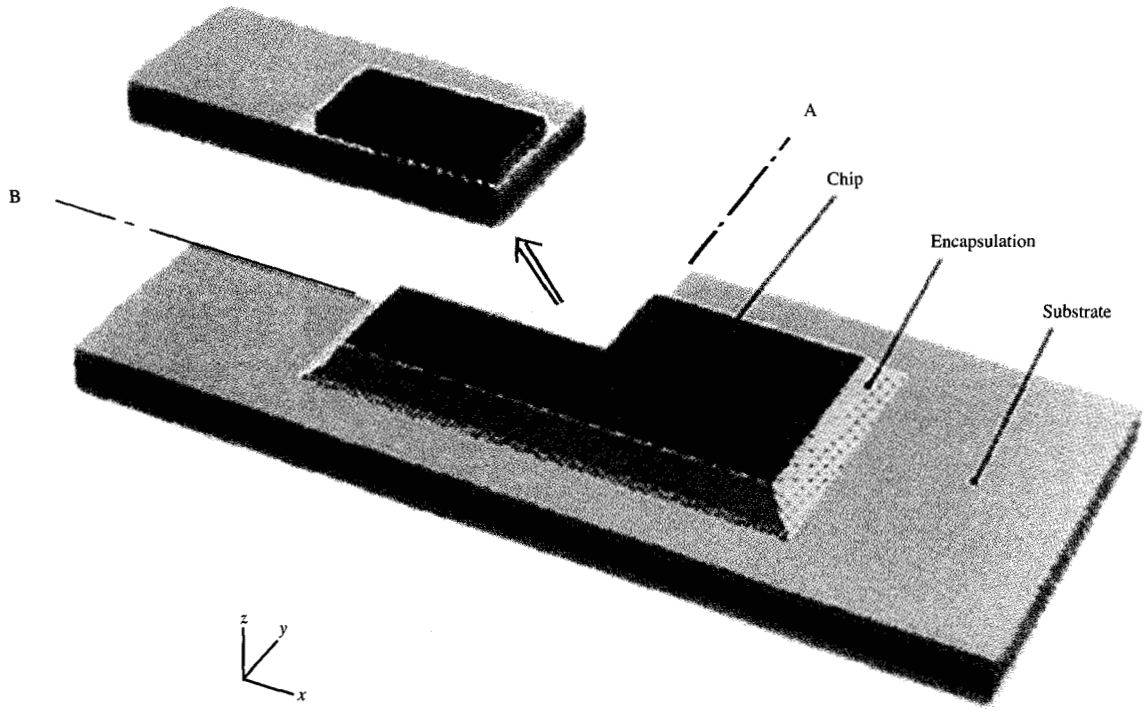


Figure 10

Cutout model.

geometries. The direct-imaging method overcomes this disadvantage by incorporating the extraction process with the shading.

The direct-imaging method makes the most of the fact that data are linearly distributed along an arbitrary line segment which is proper to the tetrahedral cell. To find an intersection along the viewing ray, it is sufficient to solve a linear equation, which makes it possible to check the existence of an intersection from two scalar data at the entry and exit points. Other primitives do not allow such simplicity in searching. For example, because the data are interpolated trilinearly in the cubic primitive, the data distribution along an arbitrary line segment is expressed as a cubic function. To search for an intersection on the segment, a cubic equation must be solved. The search cannot be based on the two values at the entry and exit points.

Here, we compare the two methods in terms of their computational costs. Ray-tracing is assumed to be used for rendering isosurfaces in the triangular-facet method in order to create uniform conditions. In each method, the processes can be categorized into linear interpolation along an edge and over a triangle. The first consists of calculating weights at the vertices of the edge (process 11) and interpolating scalar data by using the weights (process 12). The second consists of calculating weights at the vertices of the triangle from the intersection of a viewing ray and the triangle (process 21) and interpolating scalar data by using the weights (process 22). The triangular-facet method performs process 11 three and a half times on the average at each cell and process 12 six times at each edge to extract one or two triangles. It then performs process 21 once and process 22 three times at the intersection to interpolate a normal. Assuming that 1) the projection area of extracted triangles is half that of the cell, 2) the number of rays that intersect the triangles is K, and 3) an edge is shared by six cells, we can estimate the computation time T_1 at each ray and each cell as

$$T_1 = [3.5C_{11} + 3.5(6C_{12})]/6K + (C_{21} + 3C_{22})/2$$

$$= (0.583/K)C_{11} + (3.5/K)C_{12} + 0.5C_{21} + 1.5C_{22},$$

where C_n denotes the computation time for process n.

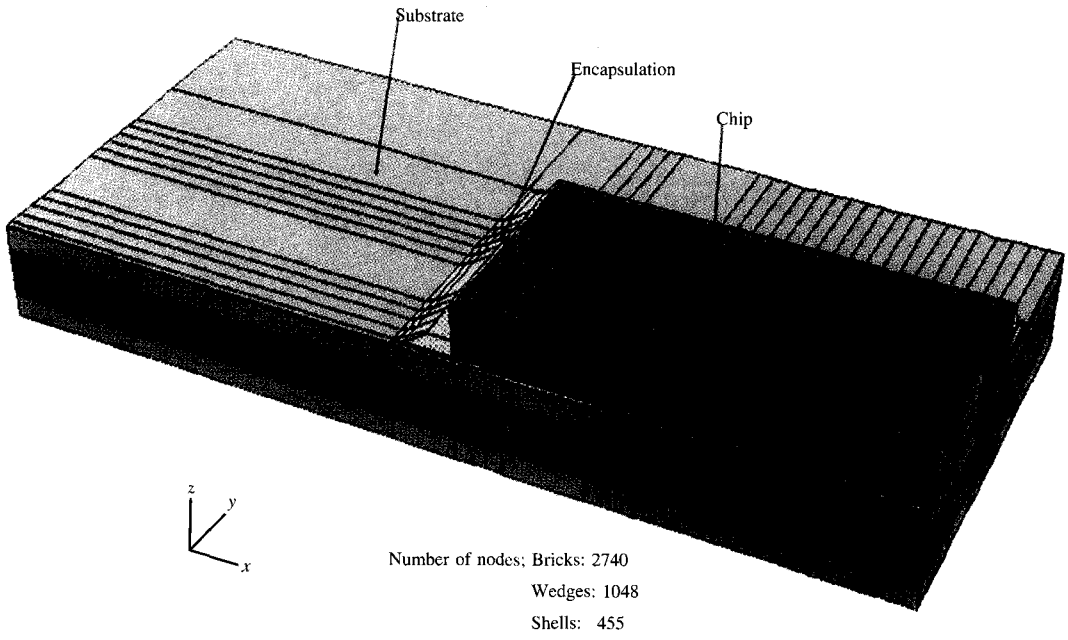


Figure 11

Defined FEM model.

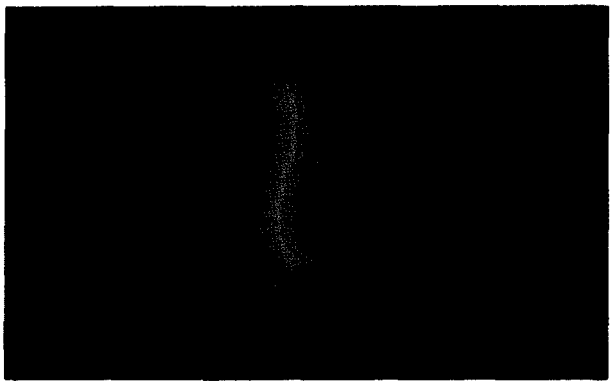


Figure 12

Steady-state thermal contour.

Figure 13

Isothermal surface.

The direct-imaging method performs process 21 once and process 22 four times to interpolate a normal and a scalar, since it may divert the result of the adjacent cell. It then performs process 11 once and process 12 three times to interpolate a normal on the isosurface. Consequently, the computation time T_2 can be estimated as

$$T_2 = (C_{11} + 3C_{12}) + (C_{21} + 4C_{22})$$

$$= C_{11} + 3C_{12} + C_{21} + 4C_{22}\,.$$

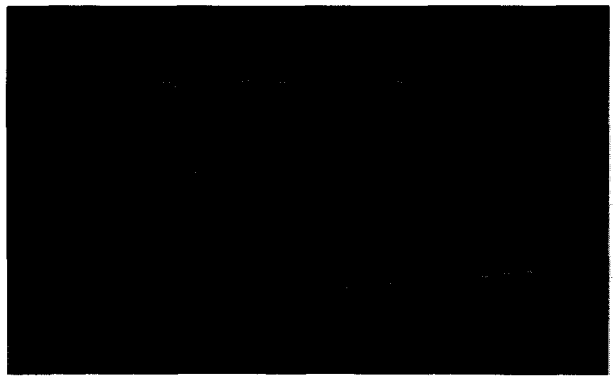


Figure 14

Stress contour map on an exterior surface.

Figure 15

Isostress surface.

Figure 16

Strain contour map of the outermost joint.

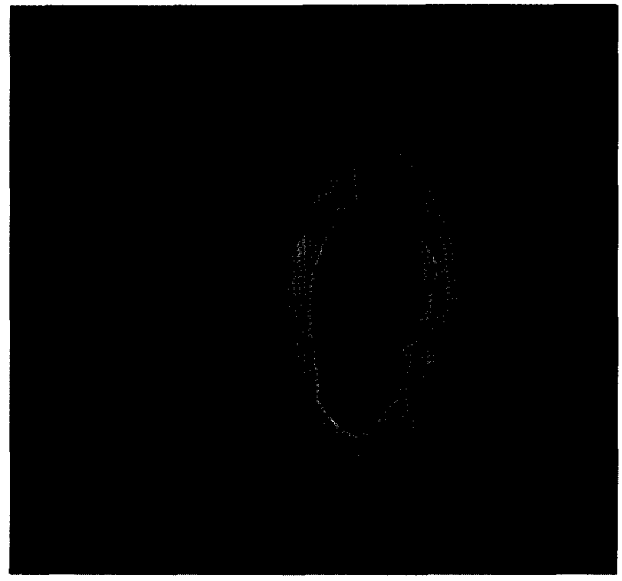


Figure 17

Corresponding isostrain surfaces.

Apparently, C_n can be arranged according to magnitude as follows:

$$C_{21} >> C_{11} > C_{22} > C_{12} .$$

That is, in terms of computation time, the triangular-facet method is superior to the direct-imaging method, which is confined to the case of one isosurface at each cell. However, when N isosurfaces are visualized in a cell, in T_1 all terms are increased by N times, but in T_2 only terms related to C_{11} and C_{12} are increased by N times. Accordingly, when N is greater than 2, the direct-imaging method is advantageous. When the application requires only isosurface visualization, the triangular-facet method should be selected. But when it requires a cloudlike image, the direct-imaging method is very attractive, because it can create both a surface-rendering image and a volume-rendering image.

Acknowledgments

We would like to thank K. Sugimoto, manager of applied graphics, and T. Kaneko, manager of the Tokyo Scientific Center, Tokyo Research Laboratory, IBM

Japan, for their encouragement in this work. We wish to thank R. Matoba, K. Yasaka, Y. Shibasaki, and M. Terashima for assisting with the FEM analysis, and Y. Tsukada and S. Tsuchida for their meaningful overall suggestions. The comments of H. Samukawa of the NIC Marketing Center, IBM Japan, on an early version of this paper were also helpful.

References

1. M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graph. & Appl.* **8,** No. 3, 29–37 (February 1988).
2. R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graph.* **22,** No. 4, 65–74 (August 1988).
3. C. Upson and M. Keeler, "V-BUFFER Visible Volume Rendering," *Computer Graph.* **22,** No. 4, 59–64 (August 1988).
4. P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graph.* **22,** No. 4, 51–58 (August 1988).
5. W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graph.* **21,** No. 4, 163–169 (July 1987).
6. R. Gallagher and J. Nagtegaal, "An Efficient 3-D Visualization Technique for Finite Element Models and Other Coarse Volumes," *Computer Graph.* **23,** No. 3, 185–194 (July 1989).
7. A. Koide and A. Doi, "A Novel Triangulation Method of Equi-Valued Surfaces Based on Tetrahedral Grids," *Research Report TR87-1017*, IBM Tokyo Research Laboratory, Tokyo, Japan, 1987.
8. N. V. Phai, "Automatic Mesh Generation with Tetrahedron Elements," *Int. J. Num. Meth. Eng.* **18,** 273–289 (1982).
9. M. Yerry and M. Shephard, "Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique," *Int. J. Num. Meth. Eng.* **20,** 1965–1990 (1984).
10. D. Shenton and Z. Cendes, "Three-Dimensional Finite Element Mesh Generation Using Delaunay Tesselation," *IEEE Trans. Magnetics* **MAG-21,** 2535–2538 (1985).
11. M. Sluiter and D. Hansen, "A General Purpose Automatic Mesh Generator for Shell and Solid Elements," *Computers in Engineering*, Vol. 3, Book No. G00217, American Society of Mechanical Engineers, 1982, pp. 29–34.
12. P. Parikh, R. Lohner, D. Gumbert, and S. Pirzadeh, "Numerical Solution on a PATH-FINDER and Other Configurations Using Unstructured Grids and Finite Element Solver," *Paper 89-0362*, American Institute of Aeronautics and Astronautics, 1989.
13. J. Ullman, *Fundamental Concepts of Programming Systems*, Addison-Wesley Publishing Co., Reading, MA, 1976, pp. 108–114.
14. J. F. Stelzer, "A Simple but Effective Method to Produce Color FEM Result Presentations," *Eng. Computation* **1,** 227–231 (September 1984).
15. Michael F. Yeo, "An Interactive Contour Plotting Program," *Eng. Computation* **1,** 273–279 (September 1984).
16. K. C. Norris and A. H. Landzberg, "Reliability of Controlled Collapse Interconnections," *IBM J. Res. Develop.* **13,** 266–271 (1969).
17. L. F. Miller, "Controlled Collapse Reflow Chip Joining," *IBM J. Res. Develop.* **13,** 239–250 (1969).
18. L. S. Goldmann, "Geometric Optimization of Controlled Collapse Interconnections," *IBM J. Res. Develop.* **13,** 251–265 (1969).

Received November 20, 1989; accepted for publication September 17, 1990

Koji Koyamada *Tokyo Scientific Center/Tokyo Research Laboratory, IBM Japan Ltd., 5-19 Sanban-cho, Chiyoda-ku, Tokyo 102, Japan 81-3-3288-8247.* Mr. Koyamada received his B.S. and M.S. degrees in electrical engineering from Kyoto University in 1983 and 1985, respectively. In 1985, he joined the marketing section of IBM Japan, where he supported customers in the area of numerical simulation, e.g. structural analysis, computational fluid dynamics, and magnetic field analysis. Mr. Koyamada is currently a researcher at the Tokyo Scientific Center in the Tokyo Research Laboratory, studying scientific simulation and visualization; he is interested in finite element analysis and volume visualization. He is a member of the Japanese Information Processing Society.

Toshihiko Nishio *Yasu Technology Application Laboratory, IBM Japan Ltd., 800 Ichimiyake, Yasu-cho, Yasu-gun, Shiga-ken 520-23, Japan 81-775-87-4737.* Mr. Nishio received his B.S. in industrial engineering and his M.S. in systems engineering from Hiroshima University in 1976 and 1978, respectively. Since joining IBM Japan in 1988, he has worked at the Yasu Technology Application Laboratory, where his current efforts involve the development of new packaging technology for the personal computer system. Mr. Nishio's work interests include estimating the reliability of the new packaging structure and evaluating its thermal performance using the finite element method.