Fast Isosurface Generation Using the Volume Thinning Algorithm

Takayuki Itoh, *Member*, *IEEE Computer Society*,
Yasushi Yamaguchi, *Member*, *IEEE Computer Society*, and
Koji Koyamada, *Member*, *IEEE Computer Society*

Abstract—One of the most effective techniques for developing efficient isosurfacing algorithms is the reduction of visits to nonisosurface cells. Recent algorithms have drastically reduced the unnecessary cost of visiting nonisosurface cells. The experimental results show almost optimal performance in their isosurfacing processes. However, most of them have a bottleneck in that they require more than $O(n)$ computation time for their preprocessing, where n denotes the total number of cells. In this paper, we propose an efficient isosurfacing technique, which can be applied to unstructured as well as structured volumes and which does not require more than $O(n)$ computation time for its preprocessing. A preprocessing step generates an extrema skeleton, which consists of cells and connects all extremum points, by the volume thinning algorithm. All disjoint parts of every isosurface intersect at least one cell in the extrema skeleton. Our implementation generates isosurfaces by searching for isosurface cells in the extrema skeleton and then recursively visiting their adjacent isosurface cells, while it skips most of the nonisosurface cells. The computation time of the preprocessing is estimated as $O(n)$. The computation time of the isosurfacing process is estimated as $O(n^{1/3}m + k)$, where k denotes the number of isosurface cells and m denotes the number of extremum points since the number of cells in an extrema skeleton is estimated as $O(n^{1/3}m)$.

Index Terms—Isosurface, extremum points, volume thinning, extrema skeleton, lattice classification.

1 Introduction

Visualizing isosurfaces is one of the most effective techniques for understanding scalar fields, such as the results of three-dimensional numerical simulations and the results of three-dimensional measurements in the medical field. An isosurface is usually approximated as a set of triangular facets [1], [2] and displayed as a set of edges of triangles or as a set of filled triangles.

In the numerical simulation field, visualization tools that support a function for the continuous generation of isosurfaces with changing scalar values are used to understand the distribution of scalar fields. When a volume is huge and contains an enormous number of cells, the cost of generating an isosurface may be high. It may even prevent the user from understanding the distribution of the scalar field because a long time is necessary to generate an isosurface from a huge volume of data. Fast isosurfacing methods are therefore needed to facilitate understanding scalar fields.

In a basic isosurfacing procedure, all cells are visited and those intersected by an isosurface, so-called *isosurface cells*, are extracted. Polygons inside the isosurface cells are then generated and, finally, the positions and normal vectors of the polygon-vertices are calculated. Some fast isosurfacing techniques focus on the acceleration of polygonization and the rendering processes, such as parallelization [3], graphics acceleration by generation of triangular strips [4], and geometric approximation [5]. However, it seems that the most effective technique for developing fast isosurfacing algorithms is the reduction of the number of visited nonisosurface cells. In our experience, the ratio of the number of nonisosurface cells to the number of isosurface cells is usually large.

Actually, many techniques have been proposed for reducing the number of nonisosurface cells that are visited. Algorithms that classify or sort cells according to their scalar values [6], [7], [8] have been previously proposed, but the number of cells visited in these algorithms is still estimated as $O(n)$, where n is the total number of cells. Algorithms that classify cells by using space-subdivision [9], [10] have also been proposed, but they are difficult to apply to unstructured volumes.

Recently, many efficient isosurfacing algorithms have been developed that can be applied to unstructured volumes and whose computation time for isosurfacing processes is much less than $O(n)$. They can be categorized into two approaches: range-based search methods and seed set generation methods.

The first approach uses an interval $[a, b]$ of a scalar range, where a is a cell's minimum value and b is its maximum value. The cell is intersected by an isosurface if the interval satisfies $a \leq C$ and $C \leq b$, where C is a constant value of the isosurface, the so-called *isovalue*. Such cells are efficiently

- T. Itoh is with IBM Research, Tokyo Research Laboratory, 1623-14 Shimotsuruma, Yamato-shi, Kanagawa, 242-8502, Japan. E-mail: itot@trl.ibm.co.jp.
- Y. Yamaguchi is with the Department of Graphics and Computer Sciences, University of Tokyo, 3-8-1, Komaba, Meguro-ku, Tokyo 153-8902, Japan. E-mail: yama@graco.c.u-tokyo.ac.jp.
- K. Koyamada is with the Department of Software and Information Science, Iwate Prefectural University, 152-52, Sugo, Takizawa-mura, Iwate-ken, 020-0193, Japan. E-mail: koyamada@soft.iwate-pu.ac.jp.

Manuscript received 30 Mar. 1998; revised 24 Mar. 2000; accepted 26 June 2000.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 109308.

extracted by using traditional searching algorithms [11], [12], [13]. Livnat et al. proposed an algorithm using a K-d tree method [11], whose isosurfacing cost is estimated as $O(n^{1/2} + k)$, where k is the number of isosurface cells. Shen et al. proposed an algorithm using a lattice classification [12], whose isosurfacing cost is estimated as $O(\log \frac{n}{L} + \frac{n^{1/2}}{L} + k)$, where L is a user-specified parameter. Cignoni et al. proposed an algorithm using an interval tree [13], whose isosurfacing cost is estimated as $O(\log n + k)$. These can be applied to unstructured volumes and are more efficient than the previously mentioned methods since the numbers of cells visited in the algorithms are much less than $O(n)$. However, these algorithms involve costs of over $O(n)$ for constructing substructures in preprocessing since they use sorting processes. The lattice classification can be implemented without the sorting process; however, it is desirable to include the sorting process in order to make the ranges of classifications vary in order to have a similar number of cells in each lattice.

The second approach generates small groups of cells, the so-called *seed set*, which includes at least one isosurface cell for every isosurface [14], [15], [16], [17]. This approach uses isosurface propagation algorithms, which recursively visit adjacent isosurface cells [18], [19], [20], starting from isosurface cells extracted from the seed set. Here, an adjacent cell means a cell that shares a face with the visited cell. When an isosurface consists of multiple disjoint parts, the approach extracts isosurface cells in all disjoint parts of the isosurface from the seed set. Two of the present authors, Itoh and Koyamada, reported a method that uses an extrema graph [14], [15]. An extrema graph connects extremum points in a volume by means of arcs and cells through which the arcs pass are registered. Bajaj et al. reported a method for generating smaller groups of cells, which sweeps cells in a grid space and removes many cells whose ranges of values are entirely shared by their adjacent cells [16]. The approaches described in these papers do not guarantee that cells intersected by all parts of an isosurface separated by through-holes of an unstructured volume will always be extracted from the seed sets. So as not to miss the intersected cells of all disjoint parts of an isosurface, the extrema graph method also traverses boundary cells. Kreveld et al. reported a method for generating a contour tree [17], which connects extremum and saddle points. The method solves the above-mentioned problem, but it requires over $O(n)$ computation time for generating the contour tree.

These two approaches, whose complexities are much smaller than $O(n)$ [11], [12], [13], [14], [15], [16], [17], have drastically reduced the unnecessary cost of visiting non-isosurface cells. For example, Table 6 in [14] shows that the cost of searching for isosurface cells in the extrema graph method is slight. These experimental results show that the differences in computation times of isosurfacing processes between such techniques seem relatively small. We should note another bottleneck of these approaches is that most of these fast isosurfacing methods require over $O(n)$ complexities for preprocessing.

In this paper, we propose an efficient isosurfacing technique which can be applied to unstructured volumes as well as structured volumes and does not require more than $O(n)$ computation time for its preprocessing. The technique generates an *extrema skeleton*, which is an extension of an extrema graph [14], as a small seed set. An extrema skeleton consists of cells and connects all extremum points like an extrema graph. The technique generates an extrema skeleton by the *volume thinning* method [15], which is an extension of the thinning method used for image recognition, in the preprocessing. It then extracts isosurface cells from the extrema skeleton and generates an isosurface by the propagation method [18], [19], [20].

This technique has the following characteristics:

- Though the complexity of the preprocessing in most of the above-mentioned fast isosurfacing methods is more than $O(n)$, in this technique, it is estimated as $O(n)$ since both the extremum point extraction process and the volume thinning process visit most cells constant times.
- The number of cells in an extrema skeleton is estimated as $O(n^{1/3}m)$, where m denotes the number of extremum points. The computation time of the isosurfacing process is therefore estimated as $O(n^{1/3}m + k)$, where k denotes the number of isosurface cells. The process can be accelerated by the combination with the range-based search approaches, such as the lattice classification method [12].
- The extrema skeleton can be used to understand the scalar topology [21] of volumes since it connects the critical points of the volume and preserves the topology of the volume.

The original volume thinning algorithm was reported in [15]; in this paper, we also describe the implementation for hexahedral cells and the face-connected to node-connected conversion process for the reduction of the number of seed cells. We also show more detailed experimental results.

2 Related Works

2.1 Isosurface Propagation

Isosurface propagation algorithms generate isosurfaces by recursively visiting adjacent isosurface cells, as shown in Fig. 1. In a typical algorithm [20], some isosurface cells are first inserted into a FIFO queue. Cells are then extracted from the FIFO one by one and polygons are generated inside the cell by the Marching Cubes method or some other polygonization method. At the same time, adjacent isosurface cells of the extracted cells are inserted into the FIFO. These cells are marked so that they are not inserted twice. By repeating the above processes, an isosurface is finally generated when the FIFO becomes empty.

The method needs additional computation time in the preprocessing step for constructing the adjacency of cells. It also needs additional memory for preserving the adjacency information. It seems that the method is memory-consuming; however, Kreveld et al. [17] describe that it is not worse than range-based search methods [11], [12], [13] since the methods may require more memory for their auxiliary substructures used while searching for isosurface cells.

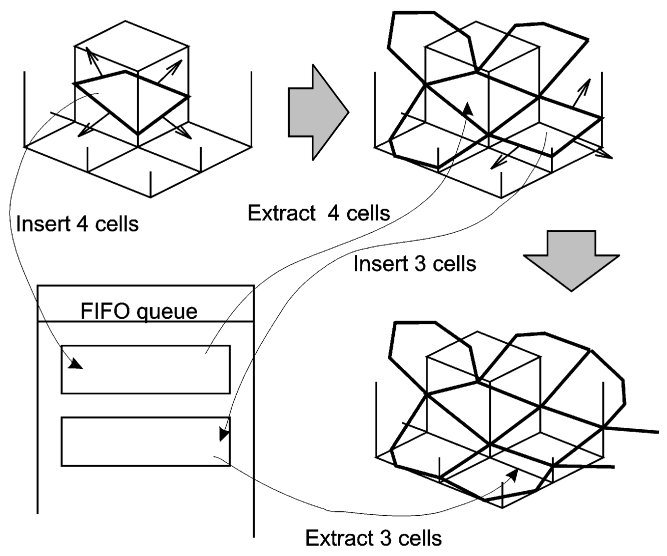


Fig. 1. Isosurface propagation.

2.2 Extrema Graph Method

The extrema graph method [14] is an algorithm for extracting isosurface cells in all disjoint parts of an isosurface. It uses the following rules governing the relationship between an isosurface and a volume:

Definition. *An extremum point is a node whose scalar value is higher (or lower) than those of all adjacent nodes that are connected to the node by cell-edges.*

Rules. *If there is a closed isosurface, then extremum points exist both inside and outside it. If there is an open isosurface, then it intersects the boundary of the volume.*

According to the above rules, cells intersected by a closed isosurface are found by traversing a graph that is generated by connecting extremum points. Cells intersected by an open isosurface are found on the boundary. In the preprocessing part of the method, extremum points in a volume are first extracted. A pair of close extremum points is then selected and adjacent cells are traversed in order, starting from one of the selected extremum points, continuing toward the other selected point. If the traversal is successfully completed, an arc of the extrema graph is allocated for the selected two extremum points and the visited cells are registered in a cell list for the arc. This process is repeated until all extremum points have been connected to form a graph. At the same time, boundary cells have been registered in a list and sorted according to the minimum and maximum values of their nodes. Fig. 2 shows an example of cells registered in the above cell lists.

Given an isovalue, cells in the extrema graph and in the sorted boundary cell lists are visited. The method extracts isosurface cells in all disjoint parts of an isosurface and the propagation algorithm generates all the parts of the isosurface.

The computation time of preprocessing of the extrema graph method is estimated as $O(n) + O(m^2 \log m + n^{1/3}m) + O(n^{2/3} \log n^{2/3})$, where m denotes the number of extremum points. In our experiments, the cost is shown to be linear in many cases. However, the

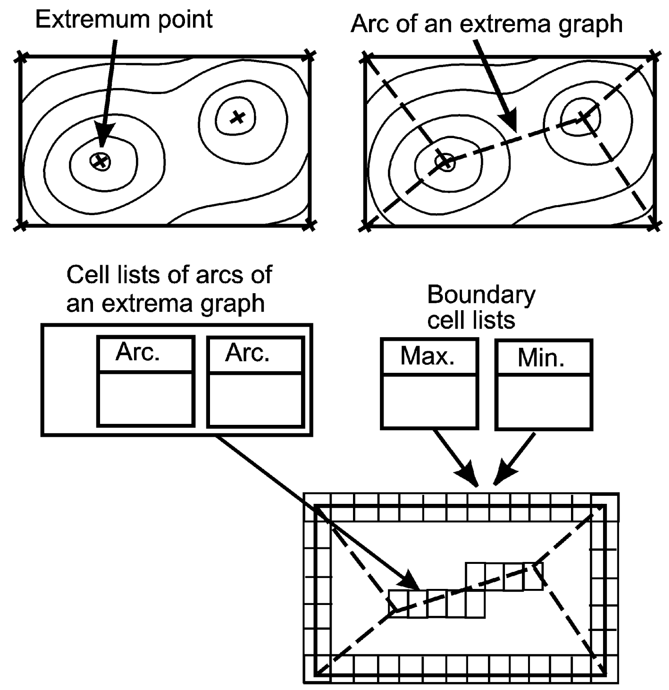


Fig. 2. Extrema graph and boundary cell lists.

process may be costly when a volume is very noisy and therefore has an enormous number of extremum points. The computation time for the isosurfacing process of the extrema graph method is estimated as $O(n^{2/3} + n^{1/3}m + k)$, where k is the number of isosurface cells. The details of these computation times are described in Appendix A.

Here, this paper estimates the complexity of algorithms with the following assumptions:

- The number of cells on the boundary of a volume is estimated as $O(n^{2/3})$,
- The maximum degree of any node is bounded by a constant, and
- The number of faces, edges, and nodes are all $O(n)$.

2.3 Contour Tree Method

Unstructured volumes may have *through-holes* or *voids* [22]. In this paper, a through-hole is defined as a topological feature that causes a genus of a boundary. A void is defined as an empty space enclosed by a disjoint part of the boundary of a volume.

An isosurface separates a volume into several subdomains. There are extremum points in all of the subdomains and an extrema graph therefore always connects all of the subdomains. An isosurface may also consist of several disjoint parts. If two adjacent subdomains share only one part of an isosurface, an arc of an extrema graph necessarily intersects the part. On the other hand, it is possible that the two subdomains share two or more disjoint parts of an isosurface if a volume has through-holes. In this case, arcs of the extrema graph do not always intersect all the parts of the isosurface. This means that an extrema graph may not find the intersection with several parts of an isosurface when a volume has through-holes. In other words, the extrema graph needs to preserve the topology of through-

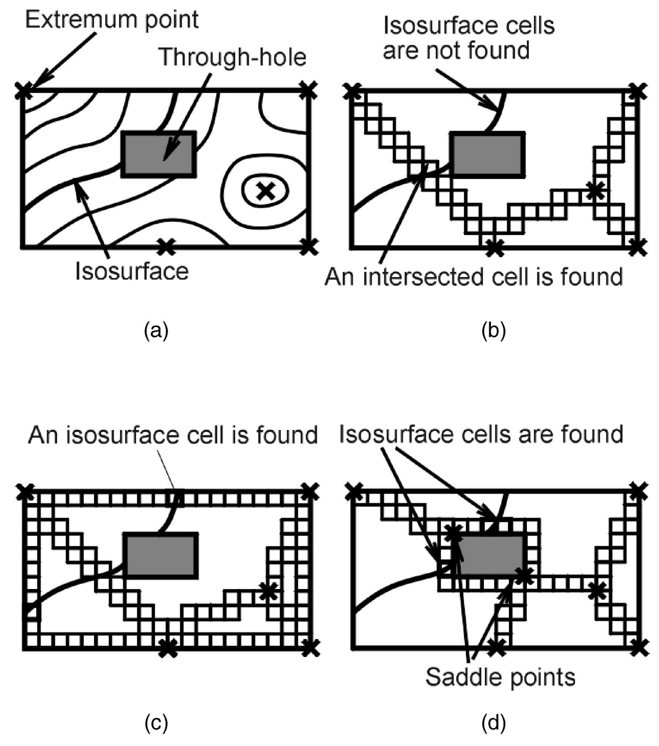

Extremum point
Through-hole
Isosurface
(a)
Isosurface cells are not found
An intersected cell is found
(b)
An isosurface cell is found
(c)
Isosurface cells are found
Saddle points
(d)

Fig. 3. Topology of an extrema graph and an isosurface. (a) A scalar field and extremum points. (b) An extrema graph may miss founding isosurface cells. (c) Isosurface cells are found on the boundary. (d) Boundary cells are not necessary if there is a cycle around a through-hole.

holes so that it intersects with all disjoint parts of every isosurfaces.

Fig. 3a shows an example of a scalar field of a volume that contains the through-hole. An isosurface is separated by a through-hole and a volume is separated into two subdomains. Both parts of the isosurface touch the two subdomains. In such cases, an extrema graph does not necessarily have intersections with all parts of the separated isosurface, as shown in Fig. 3b. In [14], cells on the boundary are visited so as not to miss isosurface cells, as shown in Fig. 3c.

Kreveld et al. proposed a contour tree method [17] which connects not only extremum points but also saddle points by using a tree. Here, a saddle point is defined as a node that the gradient vectors of the scalar values split or merge there. Kreveld et al. described what happens around saddle points in [17]. The method generates arcs of the tree by traversing cells. The traversal starts at local maximum points, merges or splits at saddle points, and finally terminates at local minimum points. The seed sets are then generated by traversing cells along the contour tree and selecting the minimum number of cells while the selected cells do not miss scalar ranges across the tree. The method requires over $O(n)$ computation time for the construction of the contour tree since it uses heap operations. However, a much smaller seed set than that of the extrema graph method can be obtained by using the contour tree since this preserves the topology of a volume and boundary cell lists are not therefore necessary, as shown in Fig. 3d.

```
void main(){

   /* preprocessing */ \\
   ExtremumPointExtraction();
   VolumeThinning();

   /* isosurfacing process */
   while (1){
     Specify an isovalue;
     Extract isosurface cells
        from the extrema skeleton;
     IsosurfacePropagation();
   }

}
```

Fig. 4. Pseudocode of the whole implementation described in this paper.

3 Algorithm Overview

3.1 Overview of Preprocessing and Isosurfacing Process

This paper proposes an efficient isosurfacing technique that drastically reduces visit of nonisosurface cells and does not require more than $O(n)$computation time for preprocessing. Fig. 4 shows the pseudocode of the technique.

During the preprocessing, our technique first extracts extremum points by using an algorithm shown in [14]. It then generates an *extrema skeleton* by the *volume thinning* method. An extrema skeleton is a set of cells that connects all extremum points while it intersects all disjoint parts of every isosurface. The preprocessing is performed only once and the extrema skeleton can be used until the visualization application is terminated.

Given an isovalue by a user or automatically, the technique extracts isosurface cells by traversing the extrema skeleton and then generates an isosurface by the propagation method. The computation time of the isosurfacing process is estimated as $O(n^{1/3}m + k)$, where k denotes the number of isosurface cells and m denotes the number of extremum points. The process can be accelerated by the combination with the range-based search approaches.

We have implemented the technique and confirmed that it normally generated isosurfaces in our experimental tests described in Section 5.

3.2 Overview of the Volume Thinning Method

We have already given an overview of the volume thinning method and its implementation for tetrahedral cells in [15]. In this paper, we also describe the implementation for hexahedral cells and a new implementation for reducing seed cells in a process for converting an extremum skeleton from face-connected to node-connected.

The volume thinning method is an extension of the previously reported image thinning method (see Appendix B). While the image thinning method generates a skeleton of a painted area of an image consists of pixels, the volume thinning method generates an extrema skeleton of a volume that consists of cells. The extrema

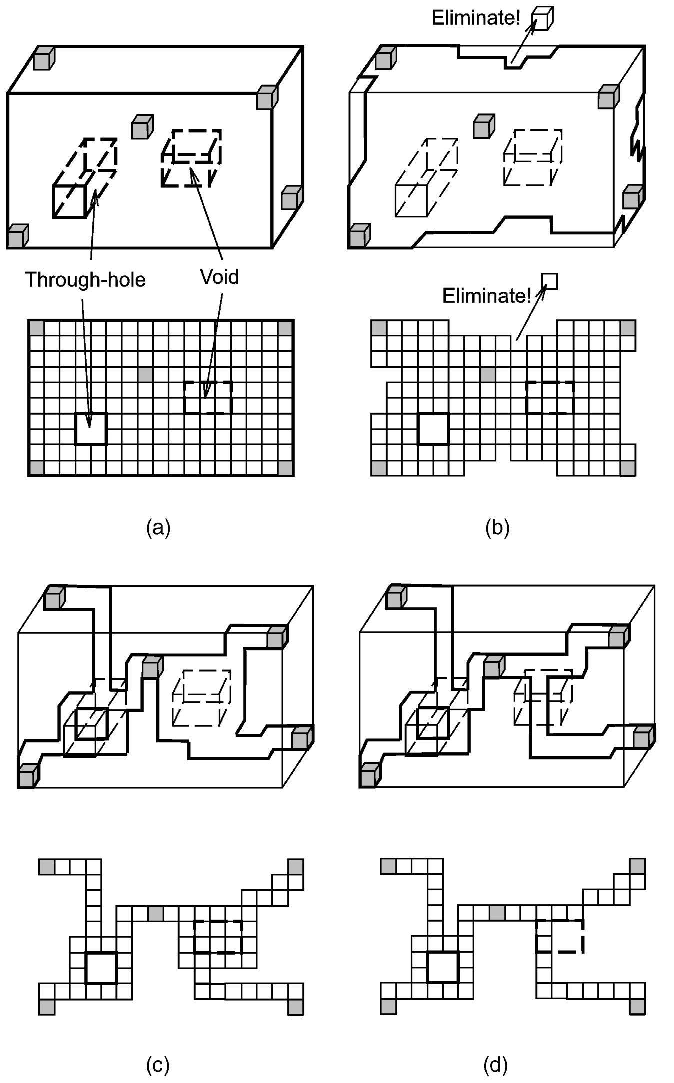


Fig. 5. Overview of the volume thinning method. (a) Cells which touch extremum points are marked. (b) Many cells are eliminated out of a seed set. (c) An extrema skeleton with a bubble-like layer around a void. (d) An extrema skeleton after removing the bubble-like layer.

skeleton connects extremum points like an extrema graph [14] while it preserves the topology of the volume like a contour tree [17]. Our technique does not need to traverse boundary cells to extract isosurface cells and seed sets generated by the volume thinning method are therefore smaller than those in the extrema graph method.

Our volume thinning method initially assumes that a seed set of a volume contains all cells in the volume. It then marks each cell which touches an extremum point, as shown in Fig. 5a. The marked cells will never be eliminated from the seed set during the volume thinning process. The method then visits unmarked cells on the boundary of the non-eliminated cells, and eliminates many of them from the seed set, as shown in Fig. 5b. Finally, the seed set forms a one-cell-wide skeleton while the cells in the skeleton preserve the connectivity of the marked cells. The visit procedure is similar to the image thinning algorithm, which visits pixels at the boundary of a painted area and eliminates many of them until a one-pixel-wide skeleton is generated.

The shape of the skeleton and the number of cells in it strongly depend on the order of visiting cells. However, we do not consider optimizing the order because the optimization causes a more complicated implementation and only slightly reduces of the number of seed cells.

The preprocessing part of our technique consists of the extremum point extraction process and the volume thinning process. The extremum point extraction process visits every cell once and traverses adjacent cells to compare the scalar values between adjacent nodes, as described in [14]. While the number of adjacent nodes of a node is constant according to an assumption described in Section 2.2, the computation time of the extrema point extraction process is estimated as $O(n)$. The computation time of the volume thinning process is also estimated as $O(n)$ since it visits each cell constant times. The computation time of the preprocessing is therefore estimated as $O(n)$.

4 Implementation of the Volume Thinning Method

4.1 Data Structures for the Volume Thinning Method

Our study assumes that all nodes are located at the vertices of cells and that a scalar value at an arbitrary position is calculated by linear interpolation. It also assumes that a volume has lists of the following data structures:

4.1.1 *Cell*

This paper describes the *classifications* of cells as C_n ($n = -1, 0, 1, 2, 3, 4$ for tetrahedral cells, $n = -1, 0, 1, 2, 3, 4, 5, 6$ for hexahedral cells). The method initially assumes that all cells are in a seed set and classifies them according to the number of adjacent cells. Cells that touch extremum points are classified as C_{-1} in the initialization stage and they are never updated. The method then eliminates many cells from the seed set. The classifications of the eliminated cells are changed to C_0. During the volume thinning process, the classifications of cells are dynamically changed according to the number of adjacent noneliminated ($=$ non-C_0) cells.We suppose that a tetrahedral cell e_i has the following variables:

- pointers to its four nodes, $(n_{i,1}, n_{i,2}, n_{i,3}, n_{i,4})$,
- pointers to its adjacent cells, $(e_{i,1}, e_{i,2}, e_{i,3}, e_{i,4})$, and
- a classification value, c_i, that represents the number of adjacent noneliminated cells.

In the case of hexahedral cells, the number of nodes is 8 and the number of adjacent cells is 6.

4.1.2 *Boundary Face*

In the initialization stage, a boundary face f_i is defined as a face of a cell that is not shared by another cell. During the volume thinning process, f_i is defined as a face of a cell that is not shared by another noneliminated cell. The process yields new boundary faces when it eliminates cells from a seed set one by one.

This paper describes the *boundary ID* as G_i ($i = -1, 0, 1, \ldots m$), where m denotes the number of voids. In the initialization stage, G_0 is assigned to the node and boundary faces on the outer boundary of a volume, and G_i

($i > 0$) is assigned to the node and boundary faces on the void boundary. G_{-1} is not assigned to boundary faces but to nodes which are not on the boundary of noneliminated cells.

We suppose that f_i has the following variables:

- a pointer to its cell, e_i and
- a boundary ID, g_i.

4.1.3 Node

We assume that a node n_i has the following variables:

- a position value, $\mathbf{p}_i$,
- a scalar value, s_i, and
- a boundary ID, g_i that n_i touches.

4.1.4 Extremum Point

An extremum point x_i is defined as a node whose scalar value is larger or smaller than those of any other adjacent nodes. It has the following variables:

- a pointer to the node, n_i and
- a pointer to one of the cells, e_i, that x_i touches.

Our implementation extracts all extremum points by an algorithm described in [14].

4.2 Detailed Algorithm of the Volume Thinning Method

Fig. 6 shows the algorithm of the volume thinning method for tetrahedral cells as a pseudocode. The algorithm for hexahedral cells is very similar, except the consideration of C_4 and C_5 cells.

The initialization stage of the volume thinning method first classifies cells according to the number of their adjacent cells. It then allocates FIFO queues for C_n cells that are on the boundary of the noneliminated cells ($n = 1, 2, 3$, for tetrahedral cells, and $n = 1, 2, 3, 4, 5$, for hexahedral cells) and inserts such cells into the FIFOs. At that time, C_4 tetrahedral cells or C_6 hexahedral cells are not inserted into FIFOs, but most of them will be inserted during the volume thinning process because their classification will be changed when their adjacent cells are eliminated from the seed set.

At the same time, the initialization process extracts a boundary face and assigns a boundary ID G_i. It then visits adjacent boundary faces recursively and assigns them the same boundary ID G_i. The process is repeated until it assigns boundary IDs to all boundary faces. It then visits all boundary faces again and assigns the boundary ID of the visited faces to nodes on the faces. Finally, it assigns G_{-1} to all other nodes which are not on boundary faces. At that time, many nodes are assigned G_{-1}, but most of them will be changed during the volume thinning process because the process eliminates many cells and faces adjacent to most of nodes then become boundary faces.

The main loop of the volume thinning process extracts cells from FIFOs and checks the connectivity between the extracted cells and their adjacent cells, according to the conditions described in Section 4.3 and 4.4. The cells are eliminated from the seed set if the connectivity of their adjacent cells can be preserved without them. At that time, their classifications are changed to C_0.

```
void VolumeThinning(){

  /* Initialization */
  Classify cells that
    touch extremum points as $C_{-1}$,
    and other cells as $C_1$, $C_2$, $C_3$, and $C_4$;
  Insert $C_1$, $C_2$, and $C_3$ cells into FIFOs;
  Assign boundary ID to boundary faces and nodes;

  /* Main loop */
  while (1)
    if(a $C_1$ cell $e_i$ is extracted){
      EliminateCell($e_i$);
    } else if(a $C_2$ cell $e_i$ is extracted) {
      if($e_i$ cannot be eliminated) continue;
      EliminateCell($e_i$);
    } else if(a $C_3$ cell $e_i$ is extracted) {
      if($e_i$ cannot be eliminated) continue;
      EliminateCell($e_i$);
    } else if(There are
       bubble-like layers around voids ) {
       Select three cells to prick a layer;
       for(each above selected cells $e_i$) {
         EliminateCell($e_i$);
       }
    } else break;
  } /* end of main loop */

  /* Post processing */
  Register non-$C_0$ cells to a cell list;
}

void EliminateCell($e_i$){
  Change the classification $c_i$ into $C_0$;
  Assign the same boundary ID
      to all faces and nodes of $e_i$;
  for(each adjacent cell $e_{i,j}$) {
    if($c_{i,j} == C_0 \| c_{i,j} == C_{-1}$) continue;
    Update the classification $c_{i,j}$;
    Insert $e_{i,j}$ to a FIFO;
  }
}
```

Fig. 6. Pseudocode of the volume thinning method for tetrahedral cells.

The image thinning method checks a pixel's connectivity by traversing its adjacent remaining pixels through their shared edges, as described in Appendix B. Similarly, the volume thinning method checks a cell's connectivity by traversing its adjacent noneliminated cells through their shared faces. If a cell e_i has two adjacent noneliminated cells, e_j and e_k, the method traverses the noneliminated cells starting from e_j to e_k, through the adjacency of the cells. If the traversal arrives at e_k, e_i can be eliminated since the connectivity between e_j and e_k can be preserved without e_i. More details about this part of the implementation are described in Sections 4.3 and 4.4.

Suppose that a cell e_i is eliminated. At that time, the process changes the classifications of its adjacent noneliminated cells from C_n to C_{n-1} if $n > 0$ and inserts them into the C_{n-1} FIFO. Here, some nodes of e_i have been assigned G_i $(i > -1)$ and others have been assigned G_{-1} because the eliminated cell e_i is the boundary cells of noneliminated cells. At that time, the process assigns G_i to all nodes of e_i and faces of e_i shared by its adjacent noneliminated cells. The shared faces become new boundary faces of noneliminated cells at that time.

In our implementation, all C_1 cells are first extracted from a FIFO. An extracted C_1 cell is unconditionally eliminated from the seed set since the connectivity of its adjacent noneliminated cell is not changed by the elimination. When the C_1 FIFO becomes empty, C_2 cells are extracted. When both C_1 and C_2 FIFOs become empty, C_3 cells are extracted. In the case of hexahedral cells, C_4 or C_5 cells are similarly extracted when all FIFOs up to C_3 or C_4, respectively, become empty.

When all FIFOs become empty, the noneliminated cells form a skeleton that connects all extremum points. The skeleton preserves the topological features of the volume, such as through-holes and voids [22]. Cycles of cells are generated around through-holes since the skeleton contains the cycle of the through-holes. Bubble-like layers of cells are generated around the voids since the skeleton retains any disjoint boundary faces around voids, as shown in Fig. 5c. The bubble-like layers occupy the large part of an extrema skeleton. However, we do not need to preserve the layers because we need only the topology of the through-holes, as discussed in Section 2.3. As shown in Fig. 5d, our implementation eliminates such bubble-like layers of cells to reduce the number of cells in the extrema skeleton. The details of this part of implementation are described in Section 4.5.

After eliminating bubble-like layers of cells around voids, all noneliminated cells are finally extracted as a seed set. Here, other fast isosurfacing methods, such as k-d tree [11], lattice classification [12], or interval tree [13], can be applied to skip nonisosurface cells in the seed set. We apply the lattice classification method; however, our implementation does not sort cells according to their minimum and maximum values, so as not to exceed $O(n)$ computation time in the preprocessing.

When an isovalue is given by a user or automatically, our method extracts isosurface cells from the extrema skeleton and generates an isosurface by the propagation algorithm. Fig. 7a shows an example of a scalar field in a volume and Fig. 7b shows an example of an extrema skeleton. Cells in the skeleton are classified in a span space, as shown in Fig. 7c, and isosurface cells are efficiently extracted from the span space when an isovalue is specified. The painted cells in Fig. 7d are an example of cells extracted from the span space and Fig. 7e shows all the cells that have been visited during the isosurfacing process.

4.3 Conditions for Tetrahedral Cells

In the case of volumes consisting of tetrahedral cells, C_2 and C_3 cells are processed according to the following conditions. Fig. 8 also shows the conditions.

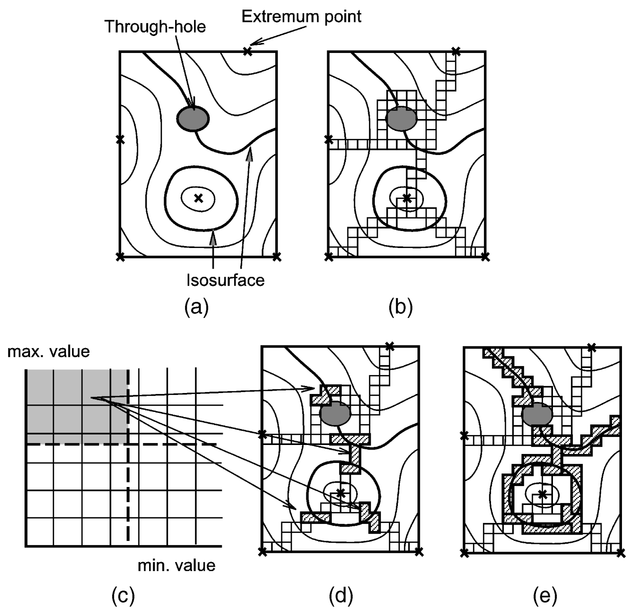


Fig. 7. Visited cells in the isosurfacing process. (a) Scalar field in a volume. (b) Extrema skeleton. (c) Lattice classification of cells in a skeleton. (d) Visited cells in a skeleton. (e) All visited cells until an isosurface is generated.

4.3.1 Condition for C_2 Cells

A C_2 cell, e_i, has an edge a shared by its two adjacent noneliminated cells, e_j and e_k. Our method traverses noneliminated cells sharing a through their shared faces, starting from e_j, without passing through e_k at once. If the traverse arrives at e_k, the classification of e_i, c_i, is changed to C_0 since the connectivity between e_j and e_k is preserved without e_i. This also means that a is not on the boundary of the noneliminated cells.

4.3.2 Condition for C_3 Cells

A C_3 cell, e_i, has a node n shared by its three adjacent noneliminated cells, e_j, e_k, and e_l. If the boundary ID of n is G_{-1}, n is not on the boundary of the noneliminated cells. In this case, an edge of e_i shared by e_j and e_k is not on the boundary of the noneliminated cells and, therefore, the

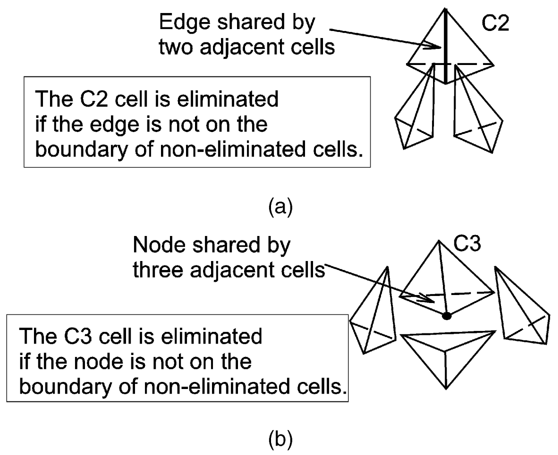
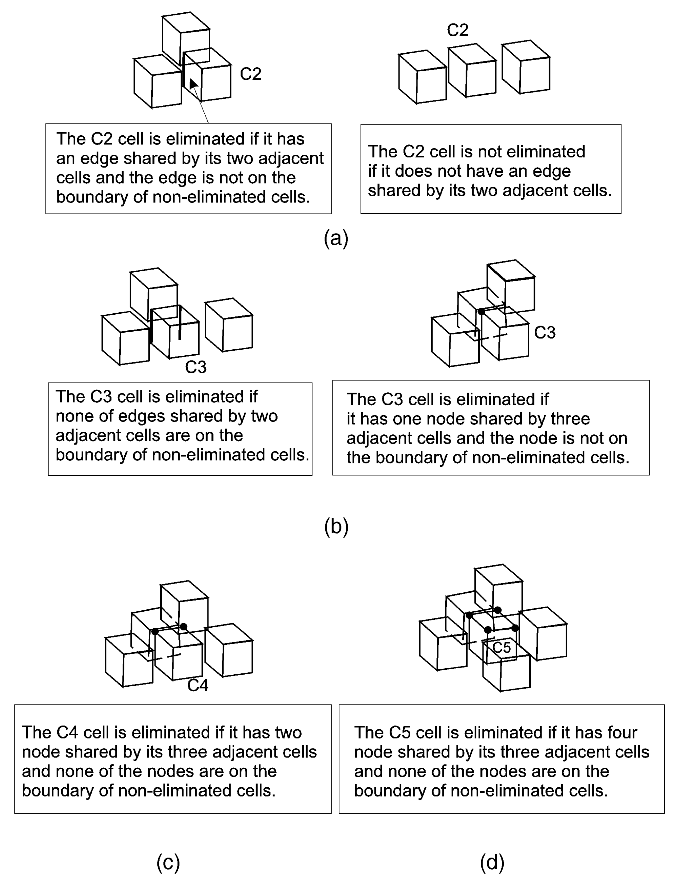

Edge shared by two adjacent cells
C2
The C2 cell is eliminated if the edge is not on the boundary of non-eliminated cells.
(a)
Node shared by three adjacent cells
C3
The C3 cell is eliminated if the node is not on the boundary of non-eliminated cells.
(b)

Fig. 8. Conditions for tetrahedral cells: (a) C2 cell, (b) C3 cell.



The C2 cell is eliminated if it has an edge shared by its two adjacent cells and the edge is not on the boundary of non-eliminated cells.

The C2 cell is not eliminated if it does not have an edge shared by its two adjacent cells.

The C3 cell is eliminated if none of edges shared by two adjacent cells are on the boundary of non-eliminated cells.

The C3 cell is eliminated if it has one node shared by three adjacent cells and the node is not on the boundary of non-eliminated cells.

The C4 cell is eliminated if it has two node shared by its three adjacent cells and none of the nodes are on the boundary of non-eliminated cells.

The C5 cell is eliminated if it has four node shared by its three adjacent cells and none of the nodes are on the boundary of non-eliminated cells.

Fig. 9. Conditions for hexahedral cells: (a) C2 cell, (b) C3 cell, (c) C4 cell, (d) C5 cell.

connectivity between e_j and e_k can be preserved without e_i, according to the condition for C_2 cells. Similarly, an edge of e_i shared by e_k and e_l, and an edge of e_i shared by e_l and e_j, are not on the boundary of the noneliminated cells and, therefore, the connectivity among e_j, e_k, and e_l can be preserved without e_i. In this case, the classification of e_i, c_i, is changed to C_0.

4.4 Conditions for Hexahedral Cells

The original volume thinning method [15] has been applied only to tetrahedral cells, but it can also be applied to hexahedral cells. In the case of volumes consisting of hexahedral cells, C_2, C_3, C_4, and C_5 cells are processed according to similar conditions. Fig. 9 also shows the conditions.

In this process, a cell is eliminated if none of its edges that are shared by two adjacent noneliminated cells are on the boundary of the noneliminated cells. The condition is similar to the condition of C_2 tetrahedral cells. It is also eliminated if none of its nodes that are shared by three adjacent noneliminated cells are on the boundary of the noneliminated cells. The condition is similar to the condition of C_3 tetrahedral cells.

4.4.1 Condition for C_2 Cells

If a C_2 cell, e_i, has an edge a shared by its two adjacent noneliminated cells, e_j and e_k, the process traverses adjacent cells of e_i, starting from e_j. If the traversal arrives at e_k, a is

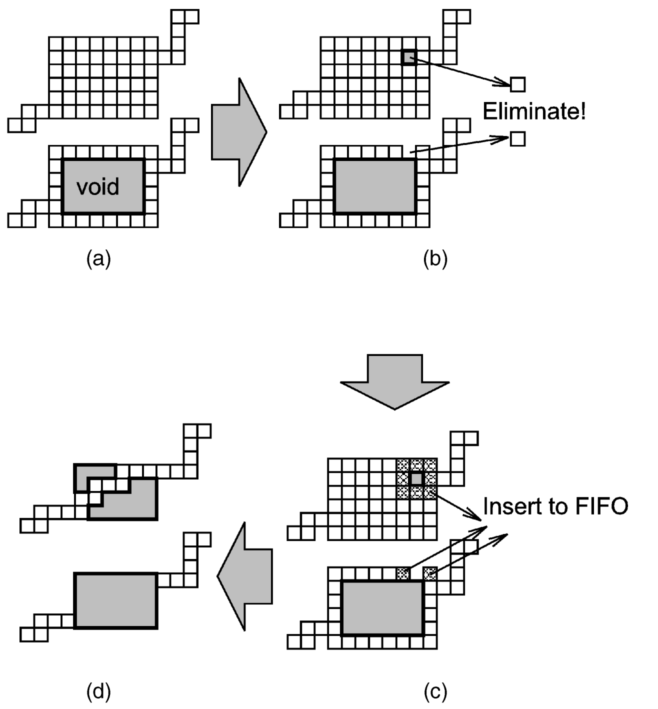


Fig. 10. Elimination of cells in bubble-like layers around voids. (a) A layer of cells around a void. (b) Eliminate cells so that disjoint boundary faces are connected. (c) Insert cells adjacent to the eliminated cells into FIFO. (d) Many cells in the layer are eliminated by the volume thinning.

not on the boundary of the noneliminated cells. In this case, the classification of e_i, c_i, is changed to C_0. If none of the edges of e_i are shared by e_j and e_k, c_i is not changed.

4.4.2 Condition for C_3 Cells

If a C_3 cell, e_i, has a node n that is shared by its three adjacent noneliminated cells, e_j, e_k, and e_l, the process checks the boundary ID of n. If it is G_{-1}, n is not on the boundary of noneliminated cells. In this case, c_i is changed to C_0.

Or, if e_i has two edges a_1 and a_2 shared by its two adjacent noneliminated cells, and either a_1 or a_2 is not on the boundary of the noneliminated cells, c_i is changed to C_0.

4.4.3 Condition for C_4 Cells

If a C_4 cell e_i has two nodes n_1 and n_2, shared by its three adjacent noneliminated cells, and either n_1 or n_2 is not on the boundary of the noneliminated cells, c_i is changed to C_0.

4.4.4 Condition for C_5 Cells

If a C_5 cell, e_i, has four nodes n_1, n_2, n_3, and n_4, shared by its three adjacent noneliminated cells, and none of them are on the boundary of the noneliminated cells, c_i is changed to C_0.

4.5 Elimination of Bubble-Like Layers of Cells

As mentioned in Section 4.2, bubble-like layers of cells are generated around voids when all the FIFOs become empty, as shown in Fig. 10a. Our implementation removes the bubble-like layers to reduce the number of cells in the final extrema skeletons.

Here, these layers have disjoint boundary faces, some faces assigned the boundary ID G_0 and others assigned G_n

($n > 0$). The topology of the layers is destroyed when the two disjoint groups of boundary faces are connected. Our method connects these groups of disjoint boundary faces by eliminating several cells, like pricking a hole through the layer, as shown in Fig. 10b. In the case of tetrahedral cells, the method first extracts a cell, e_i, in which two nodes are assigned G_0 and the other two nodes are assigned G_n. e_i should have two kinds of noneliminated adjacent cells: e_j that has at least one boundary face assigned G_0 and e_k that has at least one boundary face assigned G_n. The method then eliminates the three cells, e_i, e_j, and e_k, to connect the two disjoint boundaries, G_0 and G_n.

At that time, the classification of e_i, e_j, and e_k is changed to C_0. The classifications of their noneliminated adjacent cells are then changed from C_n to C_{n-1} and these cells are inserted into the C_{n-1} FIFO, as shown in Fig. 10c. The method restarts the thinning process by extracting cells from FIFOs and many cells in the bubble-like layer are then eliminated from the seed set, as shown in Fig. 10d. Finally, a smaller skeleton has been finally generated while cycles of cells around through-holes are preserved.

In the case of hexahedral cells, layers of cells around voids can also be eliminated by similarly pricking the layers.

4.6 Face-Connected to Node-Connected Conversion

When the volume thinning process terminates, the noneliminated cells, corresponds to non-C_0 cells, form a skeleton. Pairs of the seed cells share their faces; however, this is not necessary. The extrema skeleton contains at least one isosurface cell of every isosurface even if adjacent seed cells only share a node. The number of seed cells can therefore be reduced while the connectivity of extremum points is preserved through other nodes of the seed cells.

Consider a tetrahedral seed cell e_i and a cell e_j adjacent to e_i. If e_j has another adjacent seed cell e_k and e_k has nodes shared with e_i, then e_j can be eliminated. If the adjacent seed cell of e_k, e_l, also has nodes shared with e_i, then e_k can also be eliminated. Starting from seed cells that touch extremum points, this process eliminates many seed cells.

A seed set generated by the grid-plane sweeping method [16] looks like a node-connected extrema skeleton. Fig. 4 in [16] shows that their seed set includes cells that touch extremum points and many cells in the seed set are connected by their shared nodes.

5 Experimental Results

This section compares the experimental results of our isosurfacing method with those of other methods. In these experiments, the face-connected to node-connected conversion process had not been implemented in the volume thinning method. The experiments were carried out on an IBM PowerStation RS/6000 (Model 42T). Five datasets for unstructured volumes consisting of tetrahedral cells containing the results of numerical simulations, were used. Table 1 shows the sizes of these datasets. In Table 1:

- N_c denotes the number of tetrahedral cells.
- N_{gp} denotes the number of grid points.
- N_{ep} denotes the number of extremum points.

TABLE 1
Size of Datasets

No	1	2	3	4	5
N_c	20736	61680	346644	458664	557868
N_{gp}	4002	11624	62107	80468	97473
N_{ep}	21	46	135	5986	540

5.1 Images of Extrema Skeletons and Isosurfaces

The extrema skeleton for Dataset no. 2 is shown in Fig. 12, where the color of a cell represents its scalar value. Fig. 13 shows an example of generated isosurfaces. The volume represents a human as a through-hole and a box as a void. The through-hole starts from one foot, goes through both of the legs, and ends up in the other foot. In the skeleton, a line of cells passing between the legs preserves the cycle of the through-hole.

The extrema skeleton for Dataset no. 3 is shown in Fig. 14. Fig. 15 shows an example of isosurface generation.

The volume thinning process for generating the extrema skeleton of the volume for Dataset no. 2 is shown in Figs. 16, 17, 18, and 19.

5.2 Experimental Results of Preprocessings

We implemented the sweeping simplices method [8] and the lattice classification method [12] for the comparisons. We also used the lattice classification method to classify cells in each extrema skeleton. However, the ranges of classifications were fixed when we applied it to an extrema skeleton because we did not use a sorting process so as to avoid exceeding an $O(n)$ computation time in the preprocessing of the volume thinning method. The number of lattices was 100×100 in all experiments. Our results for the volume thinning method were also obtained without any lattice classification.

Table 2 shows the computation times in the preprocessing parts of the methods. In Table 2:

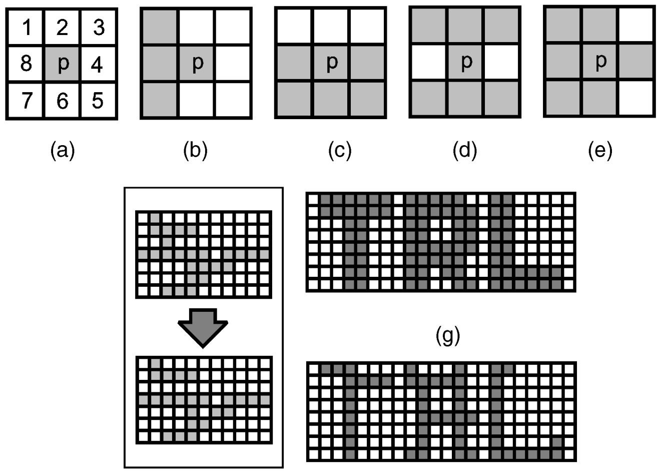


Fig. 11. Image thinning.

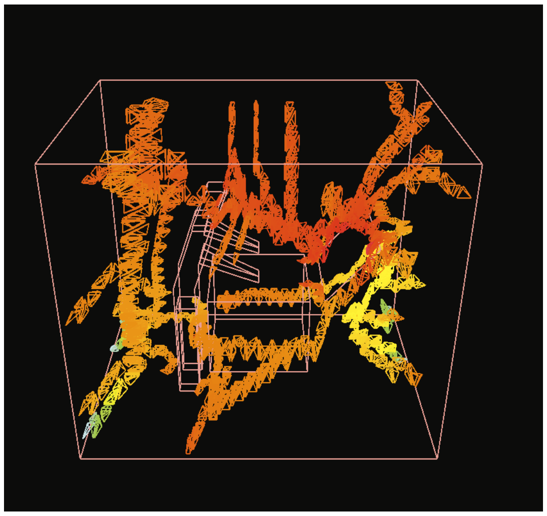


Fig. 12. Image of an extrema skeleton in Dataset no. 2.

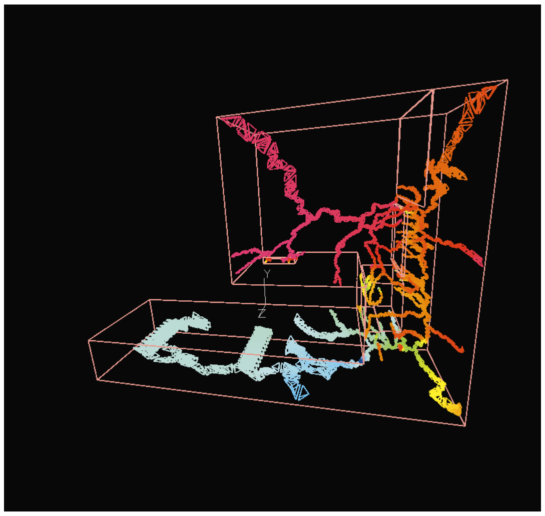


Fig. 14. Image of an extrema skeleton in Dataset no. 3.

- T_{ss} denotes the cost of preprocessing in the sweeping simplices method [8].
- T_{lc} denotes the cost of preprocessing in the lattice classification method [12].
- T_{eg} denotes the cost of preprocessing in the extrema graph method [14].
- T_{vt0} denotes the cost of preprocessing in the volume thinning method without lattice classification.
- T_{vt+lc} denotes the cost of preprocessing in the volume thinning method with lattice classification.

Fig. 20 shows the relationship between the number of total cells N_c and four of the above computation times, T_{ss}, T_{lc}, T_{eg}, and T_{vt0}. T_{vt+lc} is not shown in Fig. 20 because it is very similar to T_{vt0}.

Here, we note that the computation time of the volume thinning method is proportional to the number of cells and smaller than the computation times required by other methods, such as sweeping simplices or lattice classification methods. The results show that the differences in computation times of preprocessings are larger than the differences in those of isosurfacing processes and, therefore, the complexity of preprocessing should be taken into consideration.

We also note that the computation time of the extrema graph method is not directly proportional to the number of cells. The time required for Dataset no. 4 is very much larger than for the other datasets, although there are fewer cells than in Dataset no. 5, owing to the relatively much larger number of extremum points. This is problematic for users since it makes it impossible for them to know the number of extremum points without counting them and the cost of preprocessing is therefore not predictable. The cost of preprocessing in the volume thinning method looks more stable since it is almost proportional to the total number of cells.

We also compared the preprocessing of our volume thinning method and extrema graph method [14] in terms of the numbers of seed cells. Table 3 shows the numbers of seed cells in the two methods. In Table 3,

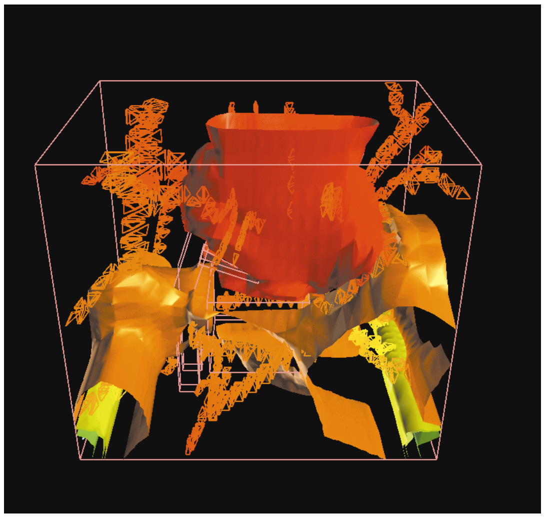


Fig. 13. Image of isosurfaces in Dataset 2.

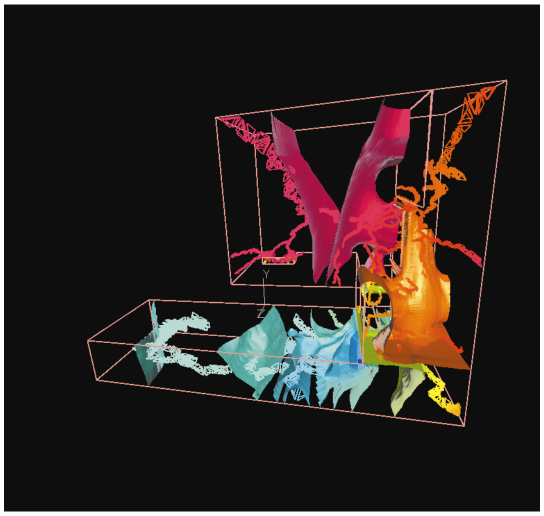


Fig. 15. Image of isosurfaces in Dataset 3.

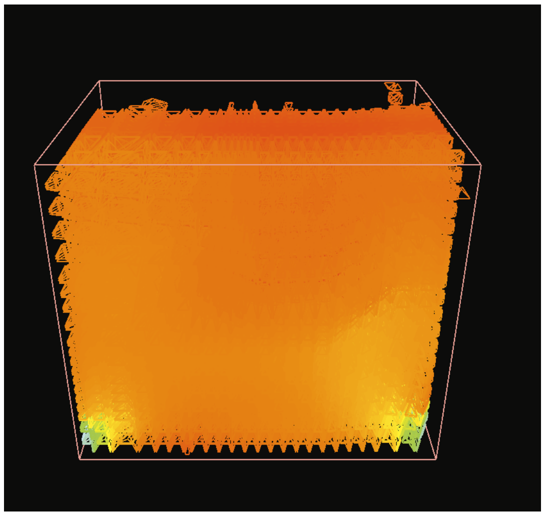


Fig. 16. Image of volume thinning process (1).

- N_{c1} is the number of seed cells in the extrema graph method.
- N_{c2} is the number of seed cells in the volume thinning method.

The results show that the number of seed cells obtained by volume thinning is much smaller than in the extrema graph method. Notice that all seed cells are not always visited to generate an isosurface. In the extrema graph method, arcs of an extrema graph preserves the maximum and minimum values of nodes of cells registered into cell lists of arcs. Given an isovalue, many arcs in an extrema graph are skipped in the isosurfacing process by comparing the isovalue with maximum and minimum values of arcs. Cells registered in the skipped arcs are therefore not visited. In the volume thinning method, only cells in specified lattices in a span space are visited in the isosurfacing process.

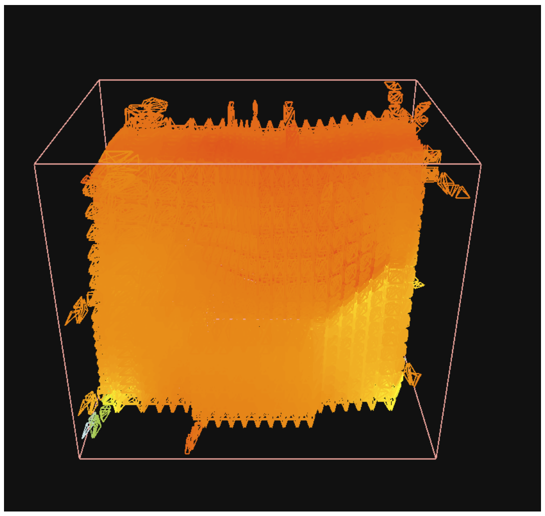


Fig. 17. Image of volume thinning process (2).

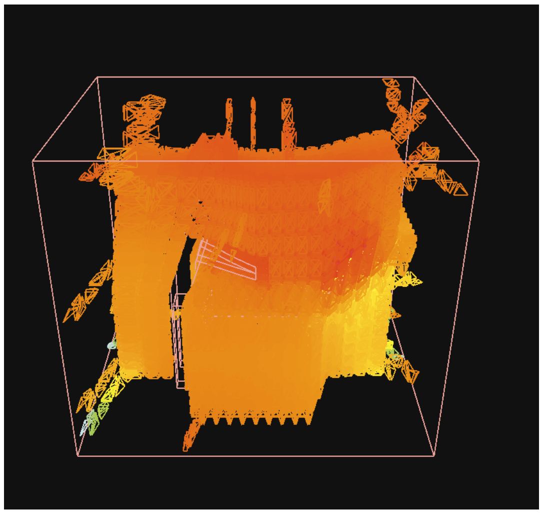


Fig. 18. Image of volume thinning process (3).

5.3 Experimental Results of Isosurfacing Processes

Next, we compare the efficiency of isosurfacing processes. In the experiments, a series of 20 isosurfaces were generated for each volume, with various scalar values. Table 4 shows the sizes of isosurfaces generated from the five datasets. In Table 4,

- N_i denotes the number of isosurface cells of the 20 isosurfaces.
- N_c denotes the number of triangular polygons in 20 isosurfaces.
- N_v denotes the number of vertices in 20 isosurfaces.

We confirmed that N_c, N_c, and N_v are the same in all five implementations and that our new method therefore generates isosurfaces properly.

Table 5 shows the computation times for the isosurfacing processes of the methods. In Table 5,

- T_{ss} denotes the total time for the isosurfacing process of the sweeping simplices method.

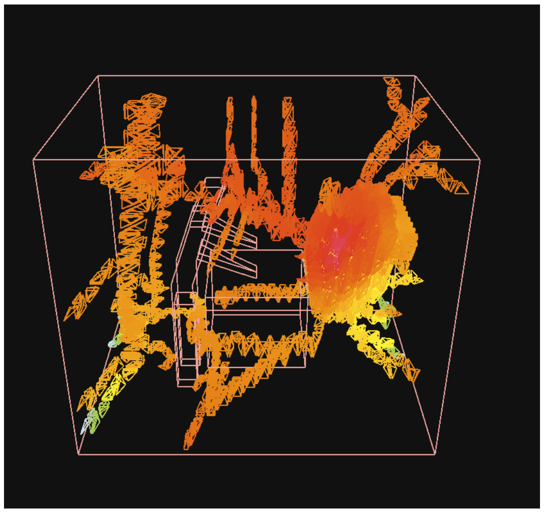


Fig. 19. Image of volume thinning process (4).

TABLE 2
Computation Times in the Preprocessing Stages

No	1	2	3	4	5
T_{ss}	0.52	2.08	21.70	29.11	27.23
T_{lc}	0.74	2.72	24.60	31.22	29.96
T_{eg}	0.25	0.77	3.53	39.85	6.95
T_{vt0}	0.47	1.35	8.00	9.04	11.21
T_{vt+lc}	0.47	1.37	8.03	9.12	11.34

- T_{lc} denotes the total time for the isosurfacing process of the lattice classification method.
- T_{eg} denotes the total time for the isosurfacing process of the extrema graph method.
- T_{vt0} denotes the total time for the isosurfacing process of the volume thinning method without lattice classification.
- T_{vt+lc} denotes the total time for the isosurfacing process of the volume thinning method with lattice classification.

This result shows that the computation times of our two methods were especially small. It also shows that we succeeded in reducing the computation times of the volume thinning method by the combination with the lattice classification method.

Table 6 shows the cost of checking if the cells in seed sets are isosurface cells or not. In Table 6,

- T_{ss} denotes the time spent searching for isosurface cells in the sorted cell lists in the sweeping simplices method.

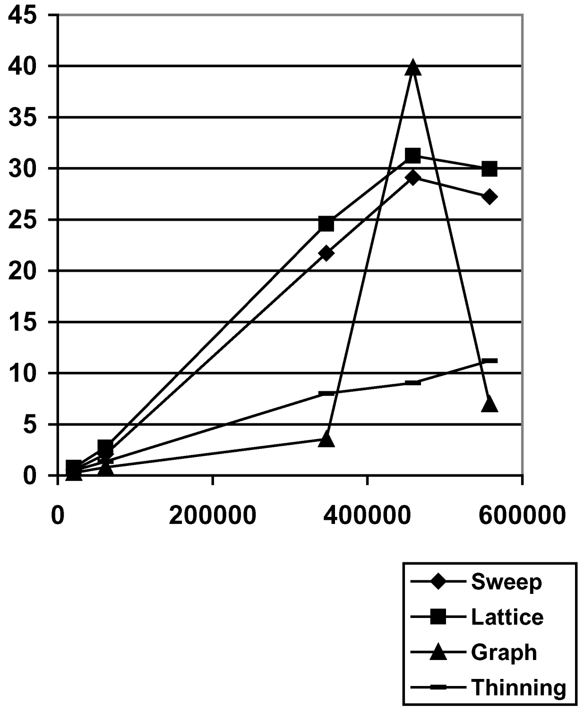


Fig. 20. Computation times in the preprocessing.

TABLE 3
Number of Registered Cells

No	1	2	3	4	5
N_{c1}	2516	6480	20158	121492	28086
N_{c2}	436	1365	3757	55536	10967

- T_{lc} denotes the time spent searching for isosurface cells in the classified cell lists in the lattice classification method.
- T_{eg} denotes the time spent searching for isosurface cells in an extrema graph and boundary cell lists and traversing adjacent intersected cells in the extrema graph method.
- T_{vt0} denotes the time spent searching for isosurface cells in a skeleton and traversing adjacent intersected cells in the volume thinning method without lattice classification.
- T_{vt+lc} denotes the time spent searching for isosurface cells in a skeleton and traversing adjacent intersected cells in the volume thinning method with lattice classification.

This result shows that the cost of searching for isosurface cells is small in all five implementations because the computation times in Table 6 are much smaller than those in Table 5.

Fig. 21 shows the relationship between the number of isosurface cells N_i and the above computation times, T_{ss}, T_{lc}, T_{eg}, T_{vt0}, and T_{vt+lc}. This result shows that the computation times for searching for isosurface cells are very slight in our two methods, the extrema graph and the volume thinning methods. It also shows that we archived to reduce the computation time in the volume thinning method by the combination with the lattice classification method.

Finally, we compare our volume thinning method and extrema graph method in terms of the numbers of visited cells. Table 7 shows the numbers of visited cells. In Table 7,

- $20N_c$ denotes 20 times the value of N_c, as shown in Table 4. It corresponds to the number of visited cells without any fast isosurfacing algorithms.
- N_i denotes the number of isosurface cells of the 20 isosurfaces, as shown in Table 4.
- N_{eg} denotes the number of cells visited in the extrema graph method.
- N_{vt0} denotes the number of cells visited in the volume thinning method without lattice classification.

TABLE 4
Size of Isosurfaces

No	1	2	3	4	5
N_i	51448	63509	102675	384176	881801
N_t	67875	80995	135358	494480	1164616
N_v	34921	43158	71358	251506	588796

TABLE 5
Computation Times in Isosurfacing Processes

No	1	2	3	4	5
T_{ss}	1.95	2.78	7.49	18.27	39.87
T_{lc}	1.87	2.32	4.40	14.63	33.71
T_{eg}	1.54	1.79	3.09	13.98	26.54
T_{vt0}	1.51	1.85	3.59	14.73	28.38
T_{vt+lc}	1.42	1.61	2.95	10.19	23.89

- N_{vt+lc} denotes the number of cells visited in the volume thinning method with lattice classification.

These results show that our two methods skip most of nonisosurface cells because the number of visited cells in these methods are very close to the number of isosurface cells.

Here, the results seem to indicate that the extrema graph method is sometimes more efficient than the volume thinning method; however, the comparison of N_{eg} and N_{vt0} is not always appropriate. In the extrema graph method, many arcs of a graph are skipped since each arc has the minimum and maximum values of the nodes of registered cells and they are compared with a given isovalue. N_{eg} does not denote the sum of the number of seed cells and isosurface cells because many cells registered in skipped arcs are not visited. On the other hand, N_{vt0} does denote the sum of the number of seed cells and isosurface cells. We do not think that range-based search methods, such as the lattice classification method, make the extrema graph method more efficient since many cells are already skipped without using these methods. We think that the combination of the volume thinning method and the range-based search method is a better solution for developing fast isosurface algorithms.

6 Conclusion

This paper proposed a fast isosurfacing technique that skips most of the nonisosurface cells and does not require more than $O(n)$ computation time for preprocessing. The technique generates an extrema skeleton by the volume thinning method during the preprocessing. The extrema skeleton consists of cells that connect all extremum points, and preserves the topology of the through-holes of a volume.

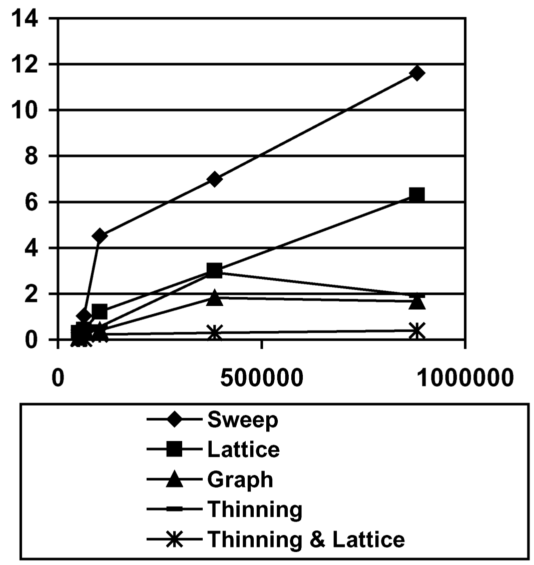


Fig. 21. Computation times in the isosurface processes.

The technique extracts isosurface cells from the extrema skeleton and then generates an isosurface by the propagation method, while it skips most of the nonisosurface cells. The experimental results given in this paper showed the efficiency of our technique.

As a topic for future work, we are thinking of using the extrema skeleton not only for isosurface generation but also for other purposes, such as the analysis of scalar topology.

Additional examples of the animation of the volume thinning process can be found at http://ciel.me.cmu.edu/itot/mytopics/isosurf.html.

Appendix

A. Computation Time of the Extrema Graph Method

The preprocessing of the extrema graph method consists of three parts: sorting cells on the boundary, extracting extremum points, and connecting extremum points. The cost of sorting cells on the boundary is estimated as $O(n^{2/3} \log n^{2/3})$ since the number of cells on a boundary of a volume is proportional to the area of the boundary and is therefore estimated as $O(n^{2/3})$ in typical cases. The cost of extracting extremum points is estimated as $O(n)$ since all cells are visited once and the number of adjacent nodes for each node is constant, according to an assumption described in Section 2.2. The cost of connecting extremum

TABLE 6
Computation Times of Searching for Intersected Cells

No	1	2	3	4	5
T_{ss}	0.42	1.04	4.51	6.98	11.61
T_{lc}	0.29	0.44	1.21	3.01	6.29
T_{eg}	0.09	0.44	0.41	1.82	1.66
T_{vt0}	0.05	0.14	0.56	2.92	1.92
T_{vt+lc}	0.02	0.03	0.23	0.30	0.39

TABLE 7
Numbers of Visited Cells

No	1	2	3	4	5
$20N_c$	414720	1233600	6932880	9173280	11157360
N_i	51448	63509	102675	384176	881801
N_{eg}	54420	66618	115639	550605	907319
N_{vt0}	59033	89007	175399	1452699	1082789
N_{vt+lc}	52956	64438	107981	442350	889064

points is about m times the cost of generating an arc, where m denotes the number of extremum points and the number of arcs is roughly estimated as m. To generate arcs that connect pairs of extremum points, the closest pairs of extremum points are selected and the cells between each pair are then traversed in order. The cost of sorting extremum points according to the distance from a selected extremum point is $O(m \log m)$. This process is used for coupling close extremum points to connect them by arcs of a graph. The cost of traversing cells between a pair of extremum points is proportional to the number of the traversed cells and is therefore estimated as $O(n^{1/3})$ in typical cases since the number of the traversed cells is proportional to the length of a segment connecting the two extremum points. The two processes are repeated about m times and, thus, the total cost of generating a graph is estimated as $O(m^2 \log m + n^{1/3}m)$. In many cases, this is not expensive since m is much smaller than n. The total cost of the preprocessing is therefore estimated as $O(n^{2/3} \log n^{2/3}) + O(n) + O(m^2 \log m + n^{1/3}m)$.

The isosurfacing process of the extrema graph method consists of three parts: searching for isosurface cells in the extrema graph, searching for isosurface cells on the boundary, and propagating an isosurface. The number of cells in an arc is estimated as $O(n^{1/3})$ and the total number of cells in a set of arcs is therefore regarded as $O(n^{1/3}m)$. The number of boundary cells is estimated as $O(n^{2/3})$ in typical cases since the number of cells on a boundary of a volume is proportional to the area of the boundary. The cost of the isosurfacing process of the extrema graph method is therefore estimated as $O(n^{2/3} + n^{1/3}m + k)$, where k is the number of isosurface cells.

B. Image Thinning Method

Image thinning [23] is a technique for generating skeletal features of an image. Essentially, the thinning is used for analyzing and recognizing the features of figures in the image-processing field. It visits pixels that touch the boundary of a painted area and eliminates many of them if features of the painted area, such as holes or projections, can be contained without the visited pixels. This process is repeated until no pixels are eliminated and, finally, a one-pixel-wide painted area, called a skeleton, is generated. An example of the conditions for determining a visited pixel's elimination is as follows:

1. If all the adjacent painted pixels (at most eight pixels) of the visited pixel p that share a vertex or an edge with p cannot be visited by traversing the adjacent pixels through their shared edges in order, then p should not be eliminated.
2. If the visited pixel p has only one adjacent painted pixel that shares an edge, p cannot be eliminated because it is on a projection of the painted area.

As shown in Fig. 11a, p is the current pixel and its adjacent pixels are numbered from 1 to 8. The black pixels in Fig. 11b, Fig. 11c, Fig. 11d, Fig. 11e are the remaining painted pixels. In Fig. 11b and Fig. 11c, p can be eliminated since all the adjacent painted pixels can be traversed through their shared edges in order (7, 8, 1 in Fig. 11b, and 4, 5, 6, 7, 8 in Fig. 11c) and p has more than one adjacent pixel that shares an edge. On the other hand, p cannot be eliminated in Fig. 11d and Fig. 11e. Fig. 11f shows that the painted area may be disjoint when visited cells in Fig. 11d or Fig. 11e are eliminated.

Fig. 11h shows an example of a skeleton generated by the thinning method, using the image shown in Fig. 11g. The skeleton contains topological features such as the genus of the painted area, and the thinning method is therefore used to understand the geometry of the images.

References

[1] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics,* vol. 21, no. 4, pp. 163-169, 1987.

[2] A. Doi and A. Koide, "An Efficient Method of Triangulating Equi-Valued Surfaces by Using Tetrahedral Cells," *IEICE Trans.,* vol. E74, no. 1, pp. 214-224, 1991.

[3] C.D. Hansen and P. Hinker, "Massively Parallel Isosurface Extraction," *Proc. IEEE Visualization '92,* pp. 77-83, 1992.

[4] C.T. Howie and E.H. Blake, "The Mesh Propagation Algorithm for Isosurface Construction," *Computer Graphics Forum (Eurographics),* vol. 13, no. 3, pp. C-65-74, 1994.

[5] J.W. Durkin and J.F. Hughes, "Nonpolygonal Isosurface Rendering for Large Volume," *Proc. IEEE Visualization '94,* pp. 293-300, 1994.

[6] R.S. Gallagher, "Span Filtering: An Optimization Scheme for Volume Visualization of Large Finite Element Models," *Proc. IEEE Visualization '91,* pp. 68-74, 1991.

[7] M. Giles and R. Haimes, "Advanced Interactive Visualization for CFD," *Computer Systems in Eng.,* vol. 1, no. 1, pp. 51-62, 1990.

[8] H. Shen and C.R. Johnson, "Sweeping Simplices: A Fast Iso-Surface Extraction Algorithm for Unstructured Grids," *Proc. IEEE Visualization '95,* pp. 143-150, 1995.

[9] J. Wilhelms and A. Van Gelder, "Octrees for Fast Isosurface Generation," *ACM Trans. Graphics,* vol. 11, no. 3, pp. 201-227, 1992.

[10] D. Silver and N.J. Zabusky, "Quantifying Visualization for Reduced Modeling in Nonlinear Science: Extracting Structures from Data Sets," *J. Visual Comm. and Image Representation,* vol. 4, no. 1, pp. 46-61, 1993.

[11] Y. Livnat, H. Shen, and C.R. Johnson, "A Near Optimal Isosurface Extraction Algorithm Using the Span Space," *IEEE Trans. Visualization and Computer Graphics,* vol. 2, no. 1, pp. 73-84, Mar. 1996.

[12] H. Shen, C.D. Hansen, Y. Livnat, and C.R. Johnson, "Isosurfacing in Span Space with Utmost Efficiency (ISSUE)," *Proc. IEEE Visualization '96,* pp. 287-294, 1996.

[13] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno, "Speeding Up Isosurface Extraction Using Interval Trees," *IEEE Trans. Visualization and Computer Graphics,* vol. 3, no. 2, pp. 158-170, Apr.-June 1997.

[14] T. Itoh and K. Koyamada, "Automatic Isosurface Propagation by Using an Extrema Graph and Sorted Boundary Cell Lists," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 4, pp. 319-327, Dec. 1995.

[15] T. Itoh, Y. Yamaguchi, and K. Koyamada, "Volume Thinning for Automatic Isosurface Propagation," *Proc. IEEE Visualization '96,* pp. 303-310, 1996.

[16] C.L. Bajaj, V. Pascucci, and D.R. Schikore, "Fast Isocontouring for Improved Interactivity," *Proc. ACM Symp. Volume Visualization '96,* 1996.

[17] M. Kreveld, R. Oostrum, C.L. Bajaj, V. Pascucci, and D.R. Schikore, "Contour Trees and Small Seed Sets for Isosurface Traversal," *Proc. 13th ACM Symp. Computational Geometry,* pp. 212-219, 1997.

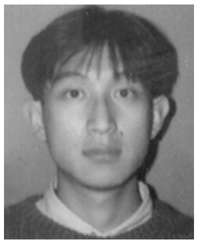
[18] G. Wyvill, C. McPheeters, and B. Wyvill, "Data Structure for Soft Objects," *The Visual Computer,* vol. 2, no. 4, pp. 227-234, 1986.

[19] J. Bloomenthal, "Polygonization of Implicit Surfaces," *Computer Aided Geometric Design,* vol. 5, no. 4, pp. 341-355, 1988.

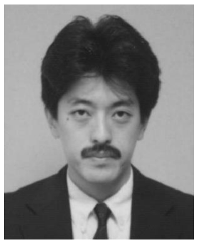
[20] D. Speray and S. Kennon, "Volume Probe: Interactive Data Exploration on Arbitrary Grids," *Computer Graphics,* vol. 24, no. 5, pp. 5-12, 1990.

[21] C.L. Bajaj, V. Pascucci, and D.R. Schikore, "Visualization of Scalar Topology for Structural Enhancement," *Proc. IEEE Visualization '98,* pp. 51-58, 1998.

[22] J.R. Munkres, *Topology: A First Course.* Prentice Hall, 1975.
[23] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Computer Science Press, 1982.



Takayuki Itoh received his BS, MS, and PhD degrees from the Department of Electronics and Communications at Waseda University in 1990, 1992, and 1997, respectively. He has been a researcher at IBM Research, Tokyo Research Laboratory (TRL), since 1992. His research mainly focuses on mesh generation, surface reconstruction, photo-realistic rendering, scientific visualization, and information visualization.He is a member of the IEEE Computer Society, ACM, and IPSJ (the Information Processing Society of Japan).



Yasushi Yamaguchi received a BE degree in precision machinery engineering and a DrEng degree in information engineering from the University of Tokyo in 1983 and 1988, respectively. He is an associate professor in the Department of Graphics and Computer Sciences of the University of Tokyo. His research interests lie in computer aided-design and geometric modeling, including parametric modeling, topology models for B-reps, surface surface intersection, and surface interrogation. He is a member of the ACM SIGGRAPH, IEEE Computer Society, and SIAM. He was an assistant professor of Tokyo Denki University from 1989 to 1993.

Koji Koyamada received BS, MS, and PhD degrees in electrical engineering from Kyoto University in 1983, 1985, and 1994. He is currently an associate professor at Iwate Prefectural University. His research interests include scientific visualization and optimization technologies. He is a member of the IEEE Computer Society and the Information Processing Society of Japan.