Volume Thinning for Automatic Isosurface Propagation

Takayuki ITOH* Yasushi YAMAGUCHI** Koji KOYAMADA*

Tokyo Research Laboratory, IBM Japan*
Graduate School of Arts and Sciences, The University of Tokyo**
IBM Japan, 1623-14 Shimotsuruma, Yamato, Kanagawa 242 JAPAN
itot@trl.ibm.co.jp* yama@graco.c.u-tokyo.ac.jp**

Abstract

An isosurface can be efficiently generated by visiting adjacent intersected cells in order, as if the isosurface were propagating itself. We previously proposed an extrema graph method, which generates a graph connecting extremum points. The isosurface propagation starts from some of the intersected cells that are found both by visiting the cells through which arcs of the graph pass and by visiting the cells on the boundary of a volume.

In this paper, we propose an efficient method of searching for cells intersected by an isosurface. This method generates a volumetric skeleton consisting of cells, like an extrema graph, by applying a thinning algorithm used in the image recognition area. Since it preserves the topological features of the volume and the connectivity of the extremum points, it necessarily intersects every isosurface. The method is more efficient than the extrema graph method, since it does not require that cells on the boundary be visited.

1 Introduction

In the area of numerical simulation, visualization tools that support a function for the continuous display of isosurfaces with changing scalar values are used to understand the distribution of scalar fields.

Elimination of non-intersected cells outside the process is one of the most effective approaches to developing fast algorithms for generating isosurfaces, since the number of cells intersected by an isosurface is regarded as $O(n^{2/3})$. Efficient algorithms that classify or sort cells according to their scalar values [1, 2], or space-subdivide cells [3], have been proposed. They do not visit many non-intersected cells, since the cells to be visited are grouped in a pre-process. However, the number of cells visited in these algorithms is still estimated as $O(n)$.

Livnat, Shen, and Johnson have reported a span-space-decomposition algorithm [4], in which cells are categorized by using a Kd-tree. This algorithm is more efficient than the abovementioned methods, since the number of cells visited in the algorithm is estimated as $O(n^{1/2}+k)$. In the worst case, however, it requires $O(n^2)$ computational time for the construction of the Kd-tree.

Polygonization algorithms of implicit functions [5, 6] visit adjacent cells intersected by an implicit surface in order, and a surface is generated as if it were propagating itself. Here, an adjacent cell means a cell that shares a face with the visited cell. These propagation algorithms are used to generate an isosurface in a volume dataset [7, 8]. The algorithm is efficient, sice it visits only intersected cells, however, it requires that the starting cells for propagation are specified. When an isosurface consists of multiple disjoint parts, starting cells in all the parts must be specified. Silver and Zabusky have reported a space-subdivision algorithm [9] in which extremum points are first extracted and cells around them are then divided into subgroups by using a spatial structure such as an octree. All starting cells are efficiently found by visiting cells in such the structure. The cost of the searching process is regarded as $O(n)$, and may be higher than that of the propagation process.

Two of the present authors, Itoh and Koyamada, reported a more efficient algorithm for detecting starting cells by reducing the number of visited cells in a pre-process[10]. In the pre-process, extremum points are first extracted and then connected as a graph. Cells through which arcs of the graph pass, as well as cells touching the boundary of the volume, are registered in a list. The cells in the list are visited to find the starting cells for propagation. Here, the number of cells on the arcs of the graph is regarded as $O(n^{1/3})$, and the number of cells on the boundary is regarded as $O(n^{2/3})$.

In this paper we propose a method for generating a volumetric skeleton consisting of cells, like an extrema graph. The thinning method for image recognition is used to generate a skeleton. A skeleton is generated in $O(n)$ computational time, because the thinning method visits most cells once. Since the skeleton preserves the topological features of the volume and the connectivity of the extremum points, it necessarily intersects every isosurface. The method is more efficient than the extrema graph method, since it does not visit cells on the boundary of a volume, and the cost of searching for intersected cells is therefore regarded as $O(n^{1/3})$.

2 Related work

2.1 Isosurface propagation

In an isosurface propagation algorithm, the IDs of adjacent intersected cells are put into a FIFO queue when an intersected cell is visited. Enqueued cells are marked so that they are not enqueued twice. The cells are visited in that order by dequeuing from the FIFO queue. A set of patches is efficiently generated by repeating this process until the FIFO queue becomes empty. The propagation algorithm has a great advantage in efficiency, since it does not visit any non-intersected cells. However, it has the problem that the starting cells must be specified in advance.

2.2 Extrema graph method

The extrema graph method [10] can detect all starting cells for the isosurface propagation. It uses the following rules governing the relationship between an isosurface and a volume:

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

According to the above rules, cells intersected by a closed isosurface are found around an inner extremum point, and cells intersected by an open isosurface are found on the boundary.

In the pre-process, extremum points are first extracted. Extremum points are defined as nodes whose scalar values are higher or lower than the values of all adjacent nodes. In our implementation, the scalar values of all nodes for each cell are compared. All nodes except the maximum-valued ones are marked as "not maximum." Similarly, all nodes except the minimum-valued ones are marked as "not minimum." After the values in all cells have been compared, only nodes that have either a "not maximum" or "not minimum" mark are extracted as extremum points.

The closest unselected pair of extremum points is then selected, and adjacent cells are traversed in order, starting from one of the selected extremum points. The traverse continues until it arrives at the other selected extremum point, and visited cells are registered in a list. This process is repeated until all extremum points are connected to form a graph. At the same time, boundary cells are registered in a list and sorted according to the minimum and maximum values of their nodes (see Figure 1).

When a scalar value is specified, cells in the extrema graph and the sorted boundary cell list are visited. At least one intersected cell is necessarily found, and every isosurface having the specified scalar value is generated by the propagation algorithm. This method necessarily extracts intersected cells in all disjoint parts of an isosurface.

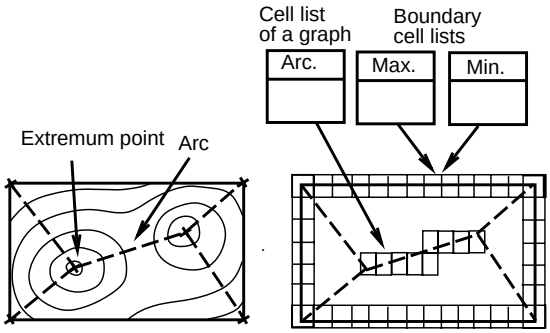

Figure 1: Extrema graph and boundary cell lists.

2.3 Cost of the extrema graph method

The pre-process of the extrema graph method consists of the following three parts: sorting cells on the boundary, extracting extremum points, and connecting extremum points. The cost of sorting cells on the boundary is regarded as $O(n^{2/3} \log n^{2/3})$, since the surfaces of boundaries are nearly planar and the number of cells on a boundary is therefore regarded as $O(n^{2/3})$. The cost of extracting extremum points is regarded as $O(n)$, since all cells are visited once. In the part of the process in which extremum points are connected, closest pairs of extremum points are selected and cells between the two extremum points are then traversed in order. The cost of traversing cells is regarded as $O(n^{1/3})$. The cost of selecting a close extremum point is regarded as $O(m \log m)$, where m denotes the number of extremum points. This process is repeated m times, and therefore the total cost is regarded as $O(m^2 \log m)$. In many cases this is not expensive, since m is much smaller than n. Table 4 in Itoh and Koyamada [10] shows that this part is not costly in any of the four datasets. However, it may be very expensive if m is large, especially in unconverged or noisy volumes.

The main process of the extrema graph method consists of three parts: searching for intersected cells in the extrema graph, searching for intersected cells on the boundary, and propagating an isosurface. The number of cells in a set of arcs is regarded as $O(n^{1/3})$ if all of the arcs are nearly straight. The number of boundary cells is regarded as $O(n^{2/3})$ if the entire boundary is nearly planar. The number of visited cells in isosurface propagation is regarded as $O(n^{2/3})$. The extrema graph method is especially efficient for large volumes, since the cost of generating an isosurface is smaller than $O(n)$. Table 5 in Itoh and Koyamada [10] shows that the extrema graph method is faster than other methods, especially for large volumes. This re-

sult is also shown in Table 3 of the present paper.

2.4 Topology of an extrema graph and an isosurface

In the extrema graph method, the cost of searching for intersected cells is governed by the number of cells on the boundary, which is regarded as $O(n^{2/3})$. Here we consider of the necessity of sorted boundary cell lists.

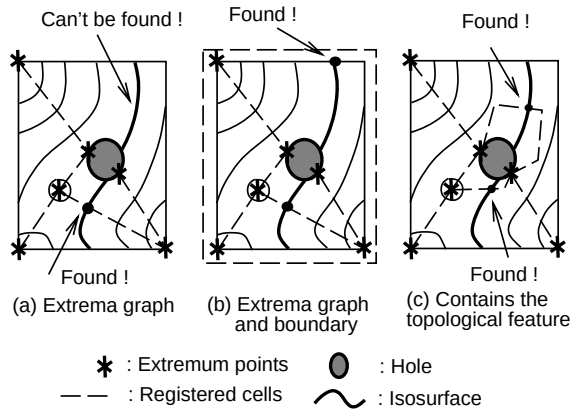

Figure 2: Topology of an extrema graph and an isosurface.

Unstructured volumes may have through-holes or voids. 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 discontinuous part of the boundary of a volume. Though an isosurface is not separated by a void, it may be separated by a through-hole.

Figure 2 shows an example of a volume which has a through-hole. An isosurface may be separated into multiple discontinuous parts when a volume has through-holes. In such volumes, an extrema graph does not necessarily have intersections with all parts of the separated isosurface. Therefore, cells on the boundary are visited, and consequently cells intersected by an open isosurface are necessarily found (see Figure 2 (b)).

If an extrema graph contains such topological features, i.e., cycles corresponding to through-holes of a volume, it necessarily intersects all parts of an isosurface (see Figure 2 (c)). Sorted boundary cell lists will not be necessary, and the cost of searching for intersected cells by using such an extrema graph will be $O(n^{1/3})$.

We applied the thinning method, which is used in the image recognition area, to generate a skeleton of a volume that consists of cells containing topological features of volumes. In the next section, we describe how we apply the thinning method to unstructured volumes.

3 Volume Thinning for Isosurface Generation

3.1 Thinning Method

Thinning [11] is a technique for generating a skeleton features of an image. Essentially, it is a technique for analyzing and recognizing the features of figures in the image-processing area. It generates a skeleton of a one-pixel-wide painted area by eliminating pixels that touch the boundary of the area if they are determined to be unnecessary. This process is repeated until all unnecessary pixels have been eliminated and all the remaining pixels touch the boundary of the area. Various conditions for determining a visited pixel's necessity have been proposed, for example,

- If all the adjacent painted pixels (8 pixels at maximum) that share a vertex or an edge with the visited pixel cannot be visited by traversing adjacent pixels in order, the visited pixel should not be eliminated, because doing so many create a blank spare in the printed area.
- If the visited pixel has only one adjacent painted pixel that shares an edge, the visited pixel cannot be eliminated, because the visited pixel is the edge of the line of the painted area.

As shown in Figure 3 (a), p is the current pixel and its adjacent pixels are numbered from 1 to 8. The black pixels in Figure 3(b)-(e) are the remaining painted pixels. In cases (b) and (c), the visited pixels can be eliminated, since all the painted pixels can be traversed in order (7, 8, 1 in Figure 3(b), and 4, 5, 6, 7, 8 in Figure 3(c)). On the other hand, the visited pixels cannot be eliminated in cases (d) and (e).

Figure 3 (g) shows an example of a skeleton generated by the thinning method, using the image shown in Figure 3 (f). The skeleton contains topological features such as genus of the painted area, and the thinning method is therefore used to understand the geometry of the images.

3.2 Volume Thinning

In this section we describe how to use the thinning method to generate a skeleton that connects all extremum points and contains the topological features of a volume. Cells that touch the boundary are visited, and unnecessary cells are eliminated outside the process in our method, in the same way that unnecessary boundary pixels were eliminated in the existing thinning method. The process is repeated until all cells in a volume have been visited at least once and a one-cell-wide skeleton has been generated.

In our method, each cell that touches an extremum point is marked "KEEP", and will never be eliminated during the thinning process. Though many un-

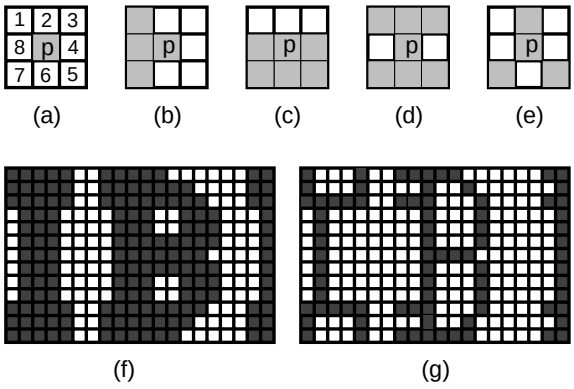

Figure 3: Thinning of an image

marked cells are then eliminated, the connectivity of the marked cells is retained. When all unmarked cells have been visited by the thinning method, a skeleton consisting of cells is generated. The skeleton contains the topological features of the volume, such as through-holes or voids [12]. Cycles of cells are generated around through-holes, since the skeleton contains the cycle of through-holes. Layers of cells like bubbles are generated around voids, since the skeleton retains any discontinuities of boundary faces around voids. See Figure 4 (b). The number of bubble-like layers of cells is regarded as $O(n^{2/3})$. In our method, these layers of cells around voids are eliminated, because isosurfaces cannot be separated by voids, while cycles of cells around through-holes are preserved. A layer of cells has two discontinuous groups of boundary faces, namely, inner boundary faces originating to the void and outer boundary faces. These faces are connected by eliminating some cells in the layer. This process looks like pricking a hole in a layer of cells. The thinning process is then restarted by visiting cells adjacent to the eliminated cells, and many cells in the layer are finally eliminated. By pricking all the layers of cells around voids, a smaller skeleton is generated in which the number of cells is regarded as $O(n^{1/3})$. See Figure 4 (c).

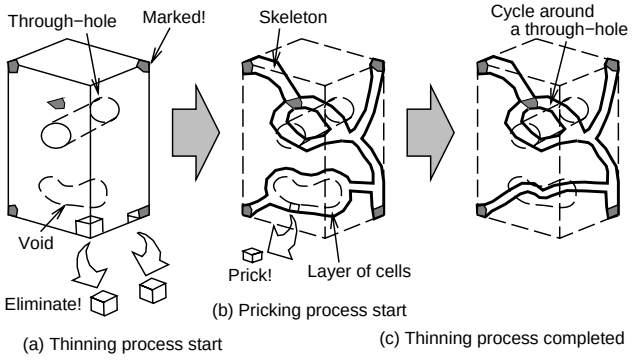

Figure 4: Volume thinning.

Though our method can be applied to both structured and unstructured volumes, in this paper we describe its application only to volumes consisting of unstructured tetrahedral or hexahedral cells.

The conditions for determining the necessity of a visited cell are defined similarly to those of existing thinning methods, described in Section 3.1. A cell is determined to be necessary if not all of its adjacent cells can be traversed along shared edges or nodes. However, this condition is not sufficient for minimizing the skeleton. Figure 5 shows an example of the accidental generation of a loop in the skeleton. A cell C has a node (or an edge), that is shared by its adjacent cells and is on the boundary of the skeleton. If C is eliminated, a cycle is generated and may remain in the skeleton after the thinning process is completed. C should therefore not be eliminated, since the generation of a cycle increases the number of cells in the skeleton. On the other hand, if the shared edge or node is not on the boundary but inside the cells, all adjacent cells of C can be traversed. In our method, the shared edge or node of the visited cell is checked if it is on the boundary, to determine the necessity of the cell.

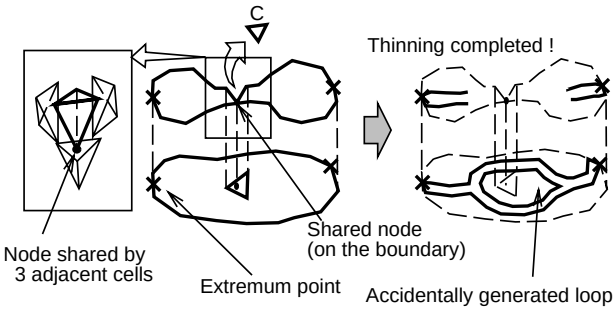

Figure 5: An Accidentally generated loop in a skeleton.

The number of cells in a skeleton obtained by the thinning method is regarded as $O(n^{1/3})$, and it is usually smaller than the number of cells on the boundary. This method is therefore more efficient than the extrema graph method, since the number of visited cells for detecting intersected cells is smaller.

This method has other advantages. The cost of generating a skeleton is regarded as $O(n)$ and does not strongly depend on the number of extremum points. When a volume has many extremum points, the thinning method is therefore more efficient and stable than the cost of the extrema graph method. It is also easy to implement, since it does not include geometric operations.

4 Implementation

In this section, we discuss an implementation of our method. The pseudo-code of the volume thinning method for tetrahedral cells is also shown in Figure 6.

The method consists of three processes: setup, thinning and pricking, and registration.

```
void VolumeThinning(){
  /* Setup process */
  Classify and enqueue cells;
  Number boundary faces and nodes;
  /* Thinning and pricking process */
  while (1){
    if( C1 FIFO is not empty ) Process a C1 cell;
    else if( C2 FIFO is not empty ) Process a C2 cell;
    else if( C3 FIFO is not empty ) Process a C3 cell;
    else if( C4 FIFO is not empty ) Process a C4 cell;
    else if( There are layers around voids )
      Prick one of the layer;
    else break;
  }
  /* Registration process */
  Extract non-C0 cells;
  Classify the extracted cells by scalar values;
}
```

Figure 6: Algorithm of the volume thinning method for tetrahedral cells.

4.1 Setup of the thinning method

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

Cell: A cell has pointers to its nodes, pointers to its adjacent cells, and a flag for its classification.

Node: A node has a position value, a scalar value, and a flag showing which void (or the boundary) it touches.

Boundary face: A boundary face is defined as a face of a cell that is not shared by another cell. It has a pointer to its cell, and a flag showing which void (or the boundary) it touches.

Extremum point: An extremum point has a pointer to its node, and a pointer to one of cells that touch the extremum point. All extremum points are extracted by an algorithm described in Itoh and Koyamada [10].

First, cells are classified by the number of their adjacent cells, and the classifications are described as C_n ($n = 0,1,2,3,4$, for tetrahedral cells, $n = 0,1,2,3,4,5,6$, for hexahedral cells). At the same time, FIFO queues for C_n ($n = 1,2,3,4$, for tetrahedral cells, and $n = 1,2,3,4,5,6$, for hexahedral cells) are allocated. Cells that touch extremum points are first marked "KEEP". Other cells on the boundary are enqueued in the FIFO queue in order. In the main loop of volume thinning, a cell is visited by dequeuing it from a FIFO queue, and its classification is altered to C_0 if it is determined to be unnecessary. The classifications of all adjacent cells of the visited cell are also altered from C_n to C_{n-1}, and they are enqueued into the C_{n-1} FIFO queue, unless they are marked "KEEP". The classifications denote the number of adjacent non-C_0 cells during and after the main loop process. The C_0 cells will be eliminated so that it does not belong to the skeleton when the thinning is finished.

At the same time, all boundaries, i.e., voids and an outer boundary, are numbered, and boundary faces that form a boundary are then marked with the number of the boundary. All nodes that are adjacent to the boundary face are also marked with the number of that boundary. All the other nodes are marked with another number indicating that they lie inside the volume. When a visited cell is altered to C_0, all its nodes numbered as inside are re-marked with the number of the boundary that the other nodes have touched.

4.2 Main loop of the thinning process

After the setup process, cells are dequeued from the FIFO queues. Many of them are determined to be unnecessary, and altered to C_0, i.e., eliminated from the volume. In our implementation, all C_1 cells are dequeued first, since the topology of the skeleton is not changed by the elimination of C_1 cells. C_2 cells are visited when the C_1 FIFO queue is empty. If any C_2 cell is altered to a C_1 cell and put into the C_1 FIFO queue caused by the elimination of an adjacent C_2 cell, it is dequeued before the remaining C_2 cells. C_3 cells are visited when both the C_1 and C_2 FIFO queues are empty. If any C_3 cell is altered to a C_2 cell and put into the C_2 FIFO queue caused by the elimination of an adjacent C_3 cell, it is dequeued before the remaining C_3 cells. C_4 and C_5 cells are similarly dequeued when all of the FIFO queues up to C_3 and C_4, respectively, are empty, in the case of hexahedral cells.

All dequeued C_1 cells are altered to C_0 cells, since the topology of the skeleton is not changed by their elimination. In the case of C_2, C_3, C_4, C_5, and C_6 cells, only cells that satisfy the following conditions are altered to C_0 cells. Some cells turn out to be necessary and remain after the thinning process is completed. These cells form cycles around through-holes, or layers around voids, preserving the topological features of the original volume.

In the case of volumes consisting of tetrahedral cells, cells are processed according to the following conditions:

Condition for C_1 cells: All dequeued C_1 cells are altered to C_0.

Condition for C_2 cells: A C_2 cell has an edge shared by its two adjacent cells, C_a and C_b. Adja-

cent cells sharing the edge are traversed in order, starting from C_a. If the edge is inside the volume, the traverse arrives at C_b and the dequeued C_2 cell will altered to a C_0 cell. See Figure 7 (a).

Condition for C_3 cells: A C_3 cell has a node shared by its three adjacent cells, C_a, C_b, and C_c. If the node is numbered as inside of the volume, the dequeued C_3 cell is altered to a C_0 cell. See Figure 7 (b).

Condition for C_4 cells: All faces of a C_4 cell are adjacent to other cells. If a graph consisting of the edges and the nodes that are outside of the volume forms a simply connected tree, the dequeued C_4 cell is altered to a C_0 cell. See Figure 7 (c).

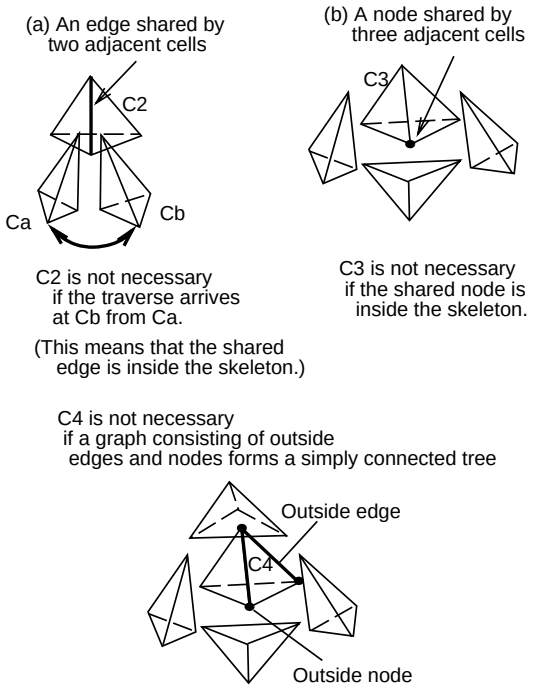

Figure 7: Conditions for tetrahedral cells.

In the case of volumes consisting of hexahedral cells, C_1, C_2, C_3, C_4, and C_5 cells are processed according to similar conditions.

4.3 Elimination of cells in layers around voids

Layers of cells around voids have discontinuous boundary faces, some facing outside the volume and others facing the voids. Both parts of faces can be connected by eliminating some cells, like pricking a hole through the layer.

In the case of tetrahedral cells, a cell in which two nodes are marked with the outer boundary's number and others are marked with some void's number is extracted first. Since the cells form a thin layer, the cell should have two kinds of adjacent cells. One has a face on the outer boundary, the other has a face on the void boundary. Our algorithm extracts a set of those three cells and alters to C_0 cells. According to this elimination, the classifications of their adjacent cells are also altered, and they are enqueued into the FIFO queues.

The above process is executed when all FIFO queues are empty, and is repeated until all layers have been pricked.

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

4.4 Registration of cells

When the thinning and pricking processes are completed, non-C_0 cells are extracted to form a skeleton. In our implementation, cells are registered in several arrays, each of which has a range of scalar values. A cell is registered in an array if at least one node of the cell has a scalar value that lies within the range of the array. When a scalar value is specified by a user or automatically, cells registered in one of the arrays whose specified value is within the range are visited and intersected cells are found.

5 Benchmark tests

This section gives the result of some benchmark tests of our thinning method in comparison with the results of other methods [2, 1, 10]. The tests were carried out on an IBM PowerStation RS/6000 (Model 560). Five datasets for unstructured volumes consisting of tetrahedral cells, which contain the result of numerical simulations, were used for the tests.

The skeleton of a volume, Dataset no. 1, is shown in Figure 8. The color of a cell is given by the number of its adjacent cells registered in the skeleton. A yellow cell has only one adjacent registered cell, and is located at the end of a line. It coincides to an extremum point. A blue cell has two adjacent registered cells and a red cell has more than three adjacent registered cells. Figure 10 shows an example of isosurface generation. Starting cells in all discontinuous isosurfaces having the same scalar value are detected and the isosurfaces are then propagated.

The skeleton of another volume, Dataset no. 2, is shown in Figure 9, where the color of a cell represents not its adjacency but its scalar value. 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, the cycle of a through-hole is contained by a line of cells passing between the legs. Figure 11 shows

an example of generated isosurfaces.

The volume thinning method and the extrema graph method [10] were compared in terms of the number of cells and the performance of each process in generating cell lists.

Table 1: Numbers of registered cells and performance in generating cell lists.

Dataset	1	2	3	4	5
N_c	20736	61680	346644	557868	458664
N_{gp}	4002	11624	62107	97473	80468
N_{ep}	21	46	135	540	5986
N_{c1}	2516	6480	20158	28086	121492
N_{c2}	436	1365	3757	10967	55536
T_{ft} (sec.)	0.19	0.52	2.75	4.76	3.66
T_{st} (sec.)	1.15	4.12	47.95	52.29	63.60
T_{eg} (sec.)	0.58	1.53	7.46	14.51	76.78
T_{vt} (sec.)	1.13	3.22	16.93	26.86	23.22

In Table 1,

- N_c is the number of tetrahedral cells (including boundary cells).
- N_{gp} is the number of grid points.
- N_{ep} is the number of extremum points.
- N_{c1} is the number of registered cells in the extrema graph method, i.e., the total numbers of cells in an extrema graph and on the boundary.
- N_{c2} is the number of registered cells in the volume thinning method.
- T_{ft} is the cost of generating cell lists in the filtering method [1].
- T_{st} is the cost of generating cell lists in the sorting method [2].
- T_{eg} is the cost of generating cell lists in the extrema graph method [10].
- T_{vt} is the cost of generating cell lists in the volume thinning method.

These processes for generating cell lists can be treated as pre-processes of isosurface generation. The cost of pre-processing in our two methods is smaller than in Giles's method but larger than in Gallagher's method in many cases.

The results show that the number of registered cells obtained by volume thinning is much smaller than in the extrema graph method. They also show that the cost of generating the skeleton in volume thinning is approximately proportional to the number of cells in a volume.

Here we note that the cost of the extrema graph method is not directly proportional to the number of cells. The cost for Dataset no. 5 is very much higher than for the other datasets, although there are fewer cells than in Dataset no. 4. This result is caused by the enormous number of extremum points. It is inconvenient for users, since it makes impossible for them to know the number of extrema points without counting them. The volume thinning method generally gives better results than the unstably performing extrema graph method.

Next, the connectivity of cells in the skeletons is analyzed.

Table 2: Number of cells in each classification.

Dataset	1	2	3	4	5
C_1	21	45	124	491	3208
C_2	397	1259	3453	9552	20574
C_3	17	61	178	875	15312
C_4	1	0	2	49	16436

The results in Table 2 show that the number of C_1 cells is almost equal to the number of extremum points, and that most cells are classified as C_2 cells. This indicates that skeletons radiate from the centers of volumes and that there are extremum points at the ends of the skeletons.

Finally, the performance of methods for generating isosurfaces is discussed. In the benchmark tests, a series of 20 isosurfaces were generated for each volume, with various scalar values.

Table 3: Performance in generating isosurfaces.

Dataset	1	2	3	4	5
N_t	67875	80995	135358	1164616	494480
N_v	34921	43158	71358	588796	251506
T_{ft} (sec.)	5.78	8.36	25.51	107.89	59.31
T_{st} (sec.)	5.69	6.85	15.24	100.92	53.48
T_{eg} (sec.)	3.31	4.18	7.56	57.72	26.49
T_{vt} (sec.)	3.27	3.96	7.22	57.12	25.21

In Table 3,

- N_t is the number of patches in 20 isosurfaces.
- N_v is the number of vertices in 20 isosurfaces.
- T_{ft} is the total time in the filtering method [1].
- T_{st} is the total time in the sorting method [2].
- T_{eg} is the total time in the extrema graph method [10].
- T_{vt} is the total time in the volume thinning method.

These results show that our two methods are more efficient than other methods in most cases. They also show that the volume thinning method is more efficient than the extrema graph method. The reason for this is that the number of registered cells in the volume thinning method is smaller that in the extrema graph method, as shown in Table 1.

In Table 4, Table 4 shows the cost of searching for intersected cells by visiting the registered cells.

Table 4: Performance of searching for intersected cells.

Dataset	1	2	3	4	5
T_{ft} (sec.)	3.13	5.95	22.59	65.08	40.20
T_{st} (sec.)	2.24	2.86	8.65	42.65	26.86
T_{eg} (sec.)	0.71	0.98	2.07	12.14	6.35
T_{vt} (sec.)	0.65	0.74	1.76	11.58	5.27

- T_{ft} is the time spent searching for intersected cells in the classified cell lists, in the filtering method.
- T_{st} is the time spent searching for intersected cells in the sorted cell lists, in the sorting method.
- T_{eg} is the time spent searching for intersected cells in an extrema graph and boundary cell lists, and traversing adjacent intersected cells, in the extrema graph method.
- T_{vt} is the time spent searching for intersected cells in a skeleton, and traversing adjacent intersected cells, in the volume thinning method.

Table 5 shows the numbers of visited cells.

Table 5: Numbers of visited cells.

Dataset	1	2	3	4	5
N_{ft}	89548	179088	786220	1927026	1278742
N_{st}	51348	61766	101951	879735	378712
N_{eg}	54420	66618	115639	907319	550605
N_{vt}	51762	63089	105762	890134	436658

In Table 5,

- N_{ft} is the number of visited cells in the filtering method.
- N_{st} is the number of visited cells in the sorting method.
- N_{eg} is the number of visited cells in the extrema graph method.
- N_{vt} is the number of visited cells in the volume thinning method.

These results show that the volume thinning method reduces the cost of searching for cells intersected by an isosurface.

6 Conclusion

This paper has proposed a volume thinning method for generating a skeleton that can be used in searching for intersected cells in isosurface propagation. It is more efficient than other methods, since the number of registered cells used in searching for intersected cells is regarded as $O(n^{1/3})$. The cost of pre-processing is regarded as $O(n)$. The volume thinning method is more stable than the extrema graph method, since it is not heavily dependent on the number of extremum points.

Acknowledgments

We would like to thank K. Shimada, manager of Graphics Applications at Tokyo Research Laboratory (TRL), IBM Japan, and K. Shimizu, manager of Advanced Graphics at TRL, for their encouragement in this work.

References

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

[2] Giles, M., and Haimes, R.: "Advanced Interactive Visualization for CFD," Computer Systems in Engineering, Vol. 1, No. 1, pp. 51-62, 1990.

[3] Welhelms J., and Gelder A. Van, "Octrees for Fast Isosurface Generation," ACM Transactions on Graphics, Vol. 11, No. 3, pp. 201-227, 1992.

[4] Livnat Y., Shen H., and Johnson C. R.,: "A Near Optimal Isosurface Extraction Algorithm Using the Span Space," IEEE Transactions on Visualization and Computer Graphics, Vol. 2, No. 1, pp. 73-84, 1996.

[5] Wyvill G., McPheeters C., and Wyvill B.: "Data Structure for Soft Objects," The Visual Computer, Vol. 2, No. 4, pp. 227-234, 1986.

[6] Bloomenthal J.: "Polygonization of Implicit Surfaces," Computer Aided Geometric Design, Vol. 5, No. 4, pp. 341-355, 1988.

[7] Speray, D., and Kennon, S.: "Volume Probe: Interactive Data Exploration on Arbitrary Grids," Computer Graphics, Vol. 24, No. 5, pp. 5-12, 1990.

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

[9] Silver, D., and Zabusky, N. J.: "Quantifying Visualization for Reduced Modeling in Nonlinear Science: Extracting Structures from Data Sets," Journal of Visual Communication and Image Representation, Vol. 4, No. 1, pp. 46-61, 1993.

[10] Itoh, T., and Koyamada, K.: "Automatic Isosurface Propagation by Using an Extrema Graph and Sorted Boundary Cell Lists," IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 4, pp. 319-327, 1995.

[11] Pavlidis, T.: "Algorithms for Graphics and Image Processing," Computer Science Press, 1982.

[12] Munkres J. R.,: "Topology: A First Course," Prentice-Hall, 1975.

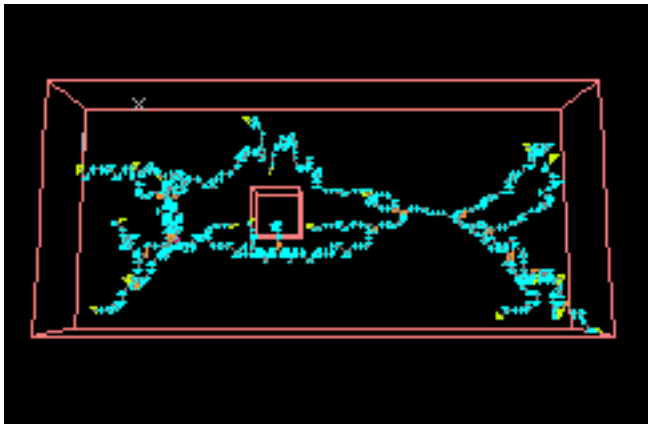

Figure 8: Image (1)

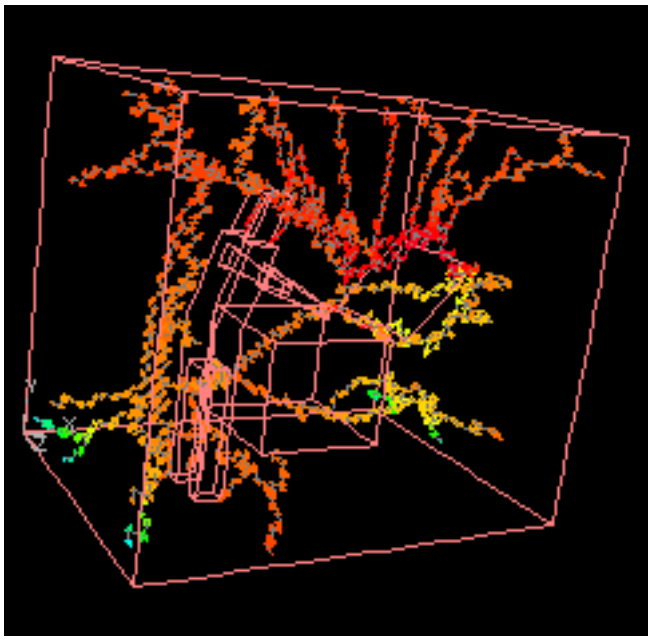

Figure 9: Image (2)

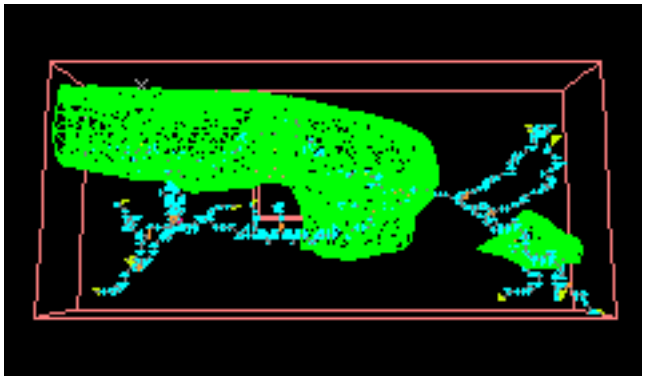

Figure 10: Image (3)

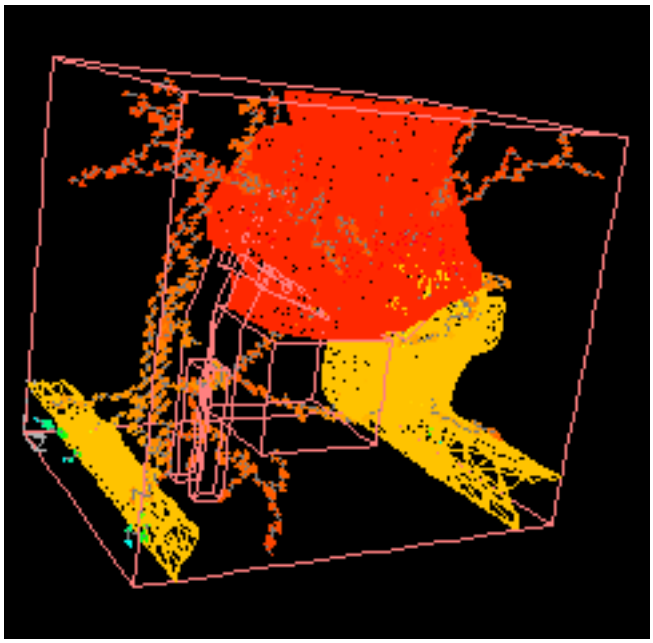

Figure 11: Image (4)