Fast Isosurface Generation Using the Cell-Edge Centered Propagation Algorithm

Takayuki ITOH Yasushi YAMAGUCHI Koji KOYAMADA

IBM Research, Tokyo Research Laboratory,
1623-14 Shimotsuruma, Yamato-shi, Kanagawa, 242-8502, JAPAN
itot@trl.ibm.co.jp
yama@graco.c.u-tokyo.ac.jp
koyamada@soft.iwate-pu.ac.jp

Abstract. Isosurface generation algorithms usually need a vertex-identification process since most of polygon-vertices of an isosurface are shared by several polygons. In our observation the identification process is often costly when traditional search algorithms are used. In this paper we propose a new isosurface generation algorithm that does not use the traditional search algorithm for polygon-vertex identification. When our algorithm constructs a polygon of an isosurface, it visits all cells adjacent to the vertices of the polygon, and registers the vertices to polygons inside the visited adjacent cells. The method does not require a costly vertex identification process, since a vertex is registered in all polygons that share the vertex at the same time, and the vertex is not required after the moment. In experimental tests, this method was about 20 percent faster than the conventional isosurface propagation method.

1. Introduction

Isosurface generation is one of the most effective techniques for extracting features of a scalar field in a volume data, such as the results of numerical simulation or medical measurement. Discussion of efficient isosurfacing methods has therefore been very active. Many approaches have been reported for the acceleration of isosurface generation, such as parallelization [1], graphics acceleration by generating triangular strips [2], and geometric approximation [3]. The most popular approach is to skip non-isosurface cells. Many reported algorithms sort or classify cells according to their scalar values [4,5,6,7]. Other algorithms that use the spatial-subdivision algorithms have been also proposed [8,9]. The present authors have proposed extrema-based algorithms [10,11] that efficiently search for isosurface cells. Starting from the extracted isosurface cells, an isosurface is generated by recursively traversing adjacent cells [12].

The above-mentioned algorithms have drastically reduced the unnecessary cost of visiting non-isosurface cells. Table 1 shows the cost of generating 20 isosurfaces with different iso-values in an unstructured volume consisting of tetrahedral cells. The experimental test compares a straightforward algorithm (ST) that visits all cells and the volume thinning algorithm (VT) [11]. Here,

-- N_c and N_n denote the numbers of cells and nodes in a volume.

-- N_t and N_v denote the total numbers of triangular polygons and vertices in the 20 isosurfaces.

-- T_1 denotes the computational time of visiting non-isosurface cells.

-- T_2 denotes the computational time of visiting isosurface cells and constructing the topology of polygons.

-- T_3 denotes the computational time of calculating polygon-vertex data, such as positions and normal vectors.

-- T_{total} denotes the total time of generating isosurfaces.

Table 1. Computational times of processes in generating isosurfaces.

Dataset	1	1	2	2
Method	SF	VT	SF	VT
N_c	61680	61680	346644	346644
N_n	11624	11624	62107	62107
N_t	80995	80995	135358	135358
N_v	43158	43158	71358	71358
T_1 (sec.)	8.30	0.09	42.23	0.57
T_2 (sec.)	3.80	3.45	6.25	5.88
T_3 (sec.)	0.76	0.75	1.14	1.15
T_{total} (sec.)	12.86	4.29	49.62	7.60

The results indicate that above-mentioned acceleration algorithm archived the great reduction of T_1. In other words, other approaches that reduce T_2 or T_3 are needed to develop more efficient isosurfacing algorithms.

In our observations, a polygon-vertices identification process occupies the largest part of the computational time in constructing the topology of polygons. Though it would be possible to implement the isosurfacing algorithm without the polygon-vertex identification process, the process is desirable, because it reduces the amount of polygon-vertex data calculation and the memory-space. In Table 1, the number of vertices N_v would be $3\,N_t$ --- about six times greater --- without the identification process. In this case, the computational time T_3 would be greater than the polygon construction time T_2, and the memory-space would be about three times greater. Moreover, the identification process is necessary if isosurfaces are used for applications that require the topology of polygons, such as parametric surface reconstruction, mesh compression, or mesh simplification.

An example of implementation of the vertex identification process is described in Doi and Koide [13]. Their implementation uses a hash-table to search shared vertices, however, such traditional search algorithm occupies the large computation time in our observation. Isosurfacing process would be accelerated if the vertex-identification process could be implemented without the costly search algorithm.

In this paper we propose an isosurface propagation algorithm that efficiently identifies shared polygon-vertices. When our algorithm constructs a polygon of an isosurface, it visits all cells adjacent to the vertices of the polygon, and registers the vertices to polygons inside the visited adjacent cells. The method does not require a costly vertex identification process, since a vertex is registered in all polygons that share the vertex at the same time, and the vertex is not required after the moment.

2. Related Work

2.1 Polygon-Vertex Identification

When polygons in an isosurface are generated by the conventional Marching Cubes method [14], all their vertices lie on cell-edges, and mostly shared by several polygons. If a volume data structure contains all cell-edges data, the shared vertices are immediately extracted. However, cell-edge data is not usually preserved in a volume data structure, owing to the limited memory-space.

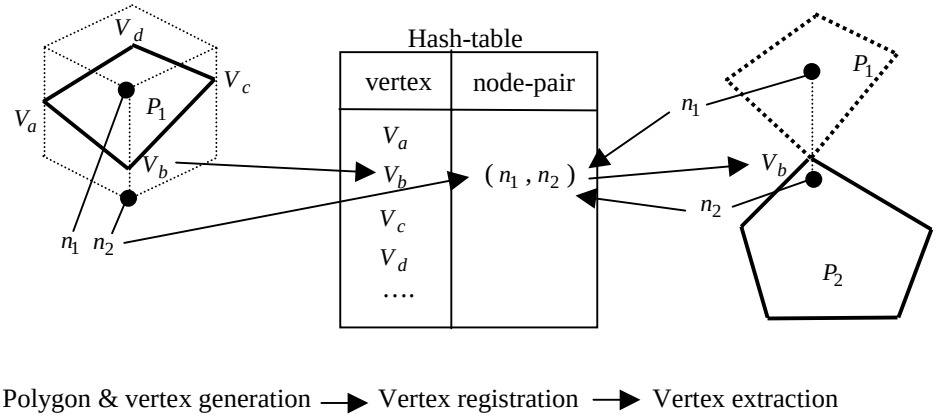

Fig. 1. Polygon-vertex identification using a hash-table.

An example of a vertex identification process is described in Doi and Koide [13]. Given an iso-value of an isosurface C, the implementation first determines the sign of $S(x,y,z)-C$ at nodes of a cell, where $S(x,y,z)$ denotes the scalar value at a node. If all the signs are equal, the cell is not an isosurface cell and the calculation of the action value is skipped. Otherwise, the process extracts isosurface cell-edges of a cell. Here, a cell-edge is represented as a pair of nodes. When a polygon-vertex is generated on a cell-edge, it is registered to the hash-table with the pair of nodes that denotes the cell-edge. When another isosurface cell that shares the same cell-edge is visited, the polygon-vertex is extracted from the hash-table, by inputting the pair of nodes. In the implementation, all isosurface cell-edges are registered to the hash-table with the polygon-vertices of an isosurface.

Fig. 1 shows an example of this process. When polygon P_1 is first constructed, vertices V_a, V_b, V_c, and V_d are registered in a hash-table with pairs of nodes. For example, vertex V_b is registered with a pair of nodes, n_1 and n_2, that denotes a cell-edge that V_b lies on. When polygon P_2 is then constructed, vertex V_b is extracted from the hash-table, by inputting the pair of nodes, n_1 and n_2.

2.2 Isosurface Propagation

An isosurface is efficiently generated by recursively visiting adjacent isosurface cells. Such recursive polygonization algorithms were originally proposed for efficient polygonization of implicit functions [15,16], and have been then applied to a volume datasets [12].

In a typical isosurface propagation algorithm, isosurface cells are extracted by a breadth-first traverse. In the algorithm, several isosurface cells are first inserted into a FIFO queue. They are then extracted from the FIFO, and polygons are generated inside them. Isosurface cells adjacent to the extracted cells are then also inserted into the FIFO. This process is repeated until the FIFO queue becomes empty, and finally the isosurface is constructed.

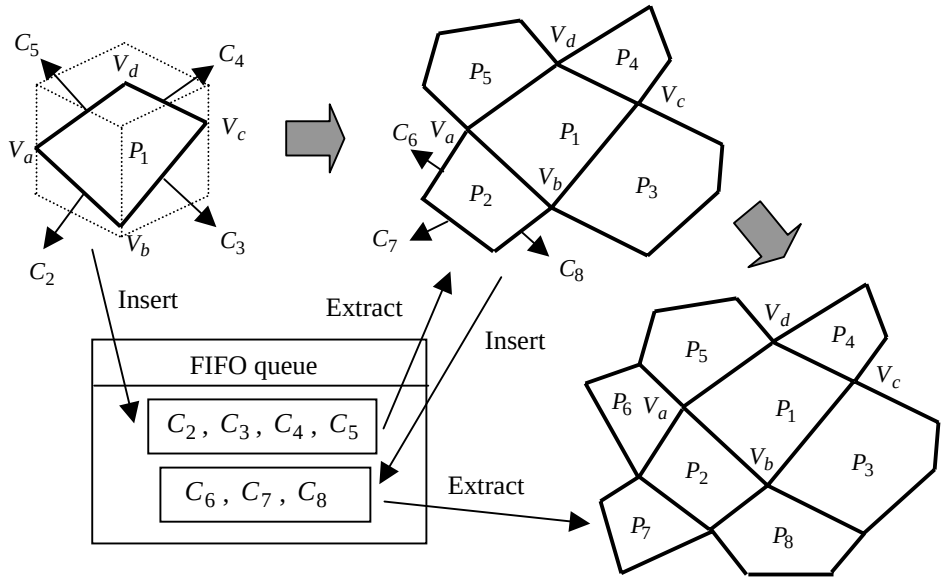

Fig. 2. Isosurface propagation.

Fig. 2 shows an example of a typical isosurface propagation algorithm. When polygon P_1 is first constructed, four adjacent isosurface cells, C_2, C_3, C_4, and C_5 are inserted into the FIFO. When these cells are extracted from the FIFO, four polygons, P_2, P_3, P_4, and P_5, are constructed. When P_2 is constructed, adjacent isosurface cells, C_6, C_7, and C_8 are similarly inserted into the FIFO. Polygons P_6,

P_7, and P_8 are then similarly constructed when C_6, C_7, and C_8 are extracted from the FIFO.

The propagation algorithm has the great advantage of reducing the number of visiting non-intersecting cells. However, it also has a problem that the starting isosurface cells must first be specified. Efficient automatic extraction of the starting cells was previously difficult, especially when the isosurface was separated into many disconnected parts.

The authors have proposed a method for automatically extracting isosurface cells in all disconnected parts of an isosurface [10,11]. The method first extracts extremum points of a volume, and then generates a skeleton connecting all extremum points. The skeleton consists of cells, and every isosurface intersects at least one cell in the skeleton. The method efficiently generates isosurfaces by searching for isosurface cells in the skeleton and then applying the isosurface propagation algorithm [12].

Our method [10,11] requires less than $O(n)$ computational time for isosurfacing process, since the cost of searching for isosurface cells is regarded as $O(n^{1/3})$ on average, unless the number of extremum points is enormous. The computational time of pre-processing in the volume thinning method [11] is always regarded as $O(n)$.

Remark that the vertex-identification process in the isosurface propagation algorithm still needs a vertex search algorithm. For example, polygon-vertices of P_1, V_a, V_b, V_c, and V_d, are registered into a hash-table when P_1 is generated. The polygon-vertex V_b is then extracted from the hash-table when P_2, P_3, and P_8 are generated.

3 Cell-edge Centered Isosurface Propagation

3.1 Algorithm Overview

In this paper we propose an isosurface generation algorithm that does not need a search algorithm in its vertex identification process. Fig. 3 shows the overview of the new method.

The method assumes that at least one isosurface cell is given. It first generates a polygon P_1 inside the given cell, and allocates its polygon-vertices, V_a, V_b, V_c, and V_d. It then visits all cells that are adjacent to polygon-vertices of P_1. In Fig. 3, cells that are adjacent to V_b are visited, and polygons P_2, P_3, and P_4 are generated. Remark that polygon-vertices of the new three polygons are not allocated at that time. It then assigns V_b to the three polygons. V_b is no more required in this algorithm, because all polygons that share V_b have been generated at that time. It means that the search algorithm is not necessary for the vertex-identification in the method. Similarly, in Fig. 3, cells that are adjacent to V_a are then visited. Polygons P_5 and P_6 are generated at that time, and V_a is assigned to them.

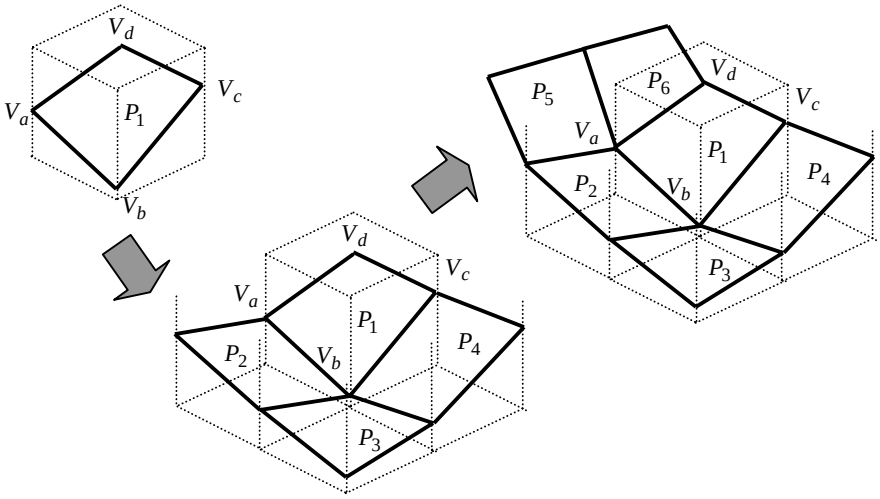

Fig. 3. Overview of the new method proposed in this paper.

3.2 Combination with the volume thinning method

The new method assumes that at lease one isosurface cell is given, so the method should be combined with an isosurface cell extraction method. We applied the volume thinning method [11] in order to extract isosurface cells in all disconnected parts of an isosurface.

The volume thinning method first generates an extrema skeleton, consists of cells, in a pre-processing. The extrema skeleton has a feature that every isosurface intersects it, so isosurface cells can be always extracted by traversing the extrema skeleton.

Fig. 4 shows the pseudo-code of our implementation. The implementation first extracts isosurface cells from the extrema skeleton, and inserts them into a FIFO queue. It then extracts an isosurface cell C_i from the FIFO, and constructs the polygon P_i inside C_i. At the moment, though the number of polygon-vertices of P_i is specified, each polygon-vertex is not allocated. The implementation then extracts the isosurface cell-edges of C_i. If a polygon-vertex V_n is not allocated on an isosurface cell-edge E_n at that time, the implementation allocates V_n and registers to the polygon P_i, and visits all cells that share the cell-edge E_n by using the connectivity of cells. If a polygon is not constructed in the visited cell C_j, the implementation constructs a polygon P_j in C_j. The polygon-vertex V_n on E_n is registered into the polygon P_j. If the visited cell C_j has not been inserted into the FIFO, the implementation also

inserts C_j into the FIFO at that time. The above process repeats until the FIFO becomes empty.

In this algorithm, most of isosurface cells are several times visited by cell-edge-centered process (the for-loop (3) in Fig. 4), and a polygon is constructed at the first visit. The cells are also visited when they are extracted from the FIFO (the for-loop (1) in Fig. 4), and all polygon-vertices of the polygons inside the extracted cells are set at the moment. The method processes a cell several times; however, our experimental tests show that its computational time is less than the conventional methods.

```
void Isosurfacing() {

  for(each cell Ci in an extrema skeleton) {
    if(Ci is an isosurface cell) { insert Ci into FIFO; }
  }

  /* for-loop (1) */
  for (each cell Ci extracted from FIFO ) {
    if(polygon Pi in Ci is not constructed) { Construct Pi in Ci ; }

    /* for-loop (2) */
    for(each intersected edge En ) {
      if(a polygon-vertex Vn on En is not added into Pi ) {
        Allocate Vn on En ;
        Register Vn into Pi in Ci ;

        /* for-loop (3) */
        for(each cell Cj which share En ) {
          if( Pj in Cj is not constructed) { Construct Pj in Cj ; }
          Register Vn into Pj in Cj ;
          if ( Cj has never been inserted into FIFO) { insert Cj into FIFO; }
        } /* end for-loop(3) */
      } /* end if(there is not Vn ) */
    } /* end for-loop(2) */
  } /* end for-loop(1) */

  for(each polygon-vertex Vn ) { Calculate position and normal vector; }
```

Fig. 4. Algorithm of the cell-edge-centered isosurface propagation method.

4. Experimental Results

This section compares the experimental results given by the cell-edge centered propagation method with those given by the conventional propagation method. The experiments were carried out on an IBM PowerStation RS/6000 (Model 560). Four datasets for unstructured volumes consisting of tetrahedral cells, which contain the results of numerical simulations, were used for the experiments.

Table 2 shows the results of experiments in which a series of 20 isosurfaces were generated for each volume, with various scalar values. Here,

-- N_c and N_n denote the numbers of cells and nodes in a volume.

-- N_t and N_v denote the total numbers of triangular polygons and vertices in the 20 isosurfaces.

-- T_1 denotes the computational time of generating 20 isosurfaces by the conventional propagation method.

-- T_2 denotes the computational time of generating 20 isosurfaces by the cell-edge centered propagation method.

-- T_{p1} and T_{p2} denote the computational times of the polygon construction processes of the two propagation methods.

In these experiments, the volume thinning method [11] extracts the starting cells of the propagation.

Table 2. Computational times of processes in generating isosurfaces.

Dataset	1	2	3	4
N_c	61680	346644	458664	557868
N_n	11624	62107	80468	97943
N_t	80995	135398	494480	1164616
N_v	43158	71358	251506	588796
T_{p1} (sec.)	3.45	5.88	21.35	49.80
T_{p2} (sec.)	2.53	4.01	15.72	36.20
T_1 (sec.)	4.29	7.60	26.65	60.81
T_2 (sec.)	3.35	5.52	20.61	46.80

The results show that the polygon construction process in the cell-edge centered propagation method is about 25 percent faster than the conventional propagation method, and the total isosurfacing process is about 20 percent faster.

5. Conclusion

In this paper we proposed an isosurface generation algorithm that does not use a vertex search algorithm for the vertex-identification process. The algorithm visits all cells sharing an isosurface cell-edge at the same time, and the vertex that lies on the

cell-edge is registered to all the polygons inside the visited cells. The vertex is no more required in the process, and the vertex search algorithm is not therefore necessary in our method. Our experimental tests showed that the method is about 20 percent faster than the conventional implementation.

In future, we would like to implement this method for hexahedral cells, and to measure the computational time of isosurfacing processes.

References

[1] Hansen C. D., and Hinker P., Massively Parallel Isosurface Extraction, Proceedings of IEEE Visualization '92, pp. 77-83, 1992.

[2] 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.

[3] Durkin J. W., and Hughes J. F., Nonpolygonal Isosurface Rendering for Large Volume, Proceedings of IEEE Visualization '94, pp. 293-300, 1994.

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

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

[6] 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.

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

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

[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] Itoh T., Yamaguchi Y., and Koyamada K., Volume Thinning for Automatic Isosurface Propagation, Proceedings of IEEE Visualization '96, pp. 313-320, 1996.

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

[13] Doi A., and Koide A., An Efficient Method of Triangulating Equi-valued Surfaces by Using Tetrahedral Cells, IEICE Transactions, Vol. E74, No. 1, pp. 214-224, 1991.

[14] Lorensen W. E., and Cline H. E., Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Vol. 21, No. 4, pp. 163-169, 1987.

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

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