Journal of Visualization, Vol. 11, No. 3 (2008)189-196

Regular Paper

Numerical Visualization by Rapid Isosurface Extractions Using 3D Span Spaces

Hirano, M.*[1], Itoh, T.*[2] and Shirayama, S.*[3]

*1 School of Engineering, University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba 277-8568, Japan.
E-mail: micoo@race.u-tokyo.ac.jp
*2 Graduate School of Humanities and Sciences, Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan.
*3 RACE, University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba 277-8568, Japan.

Received 2 May 2007
Revised 29 March 2008

Abstract : In this paper, we focus on visualization of a 3D scalar field composed of volume data by means of semi-transparent multiple isosurface extraction. A new method for displaying semi-transparent multiple isosurfaces will be proposed to fulfill real-time rendering of scalar fields. The proposed method is considered to be an extension of ISSUE, which is a known algorithm to quickly generate isosurfaces. ISSUE uses a 2D span space, where minimum values of computational cells are assigned to the horizontal axis, and maximum values of cells are assigned to the vertical axis. In the proposed algorithm, computational cells are placed onto a 3D span space. The third axis is the distance from the viewpoint. The proposed algorithm can quickly search for the cells intersecting with isosurfaces using the span space in back-to-front order and can realize representation of semi-transparent multiple isosurfaces.

Keywords : Numerical visualization, Large-scale 3D data, Scalar field, Isosurface, Real-time rendering.

1. Introduction

Numerical visualization techniques have been developed in a few decades (e.g., Liu and Moorhead II, 2005; Bruno et al., 2006; Kobayashi et al., 2008). Isosurfacing is one of the most representative methods to visualize 3D data. Enhancement of the isosurfacing method can be achieved by adopting a semi-transparent surface. If multiple isosurfaces that correspond to multi-isovalues are generated with transparency, this enables the scalar field to demonstrate local and global characteristics, as volume rendering does. Figure 1 shows examples of semi-transparent multiple isosurfaces and volume rendering in the case of visualization of a pressure field around a circular disk placed vertically in the streamwise direction.

Historically, volume rendering emerges after the function of isosurfacing is equipped for the visualization system. The recent development of the Graphical Processing Units (GPU) has shown that several visualization techniques can be accelerated (Stegmaier and Ertl, 2005). In the case of volume rendering technique, GPU volume rendering visualizes the scalar field more rapidly than isosurfaces. It has been reported that volume rendering can be used in place of isosurfacing. However, it is difficult to obtain an effective result from volume rendering owing to the dependencies of rendering parameters such as the opacity transfer function. As shown in Fig. 1, volume rendering

can visualize the entire pressure field. However, the resultant image becomes vague, and we cannot always derive useful information from volume rendering, because most of the pressure field is occupied by a fluid with a certain pressure value (0.0 in this figure). Generally, volume rendering of flow visualization for an external flow has the same problem that occurs when dealing with a wide region, and some artifices in visualization should be required.

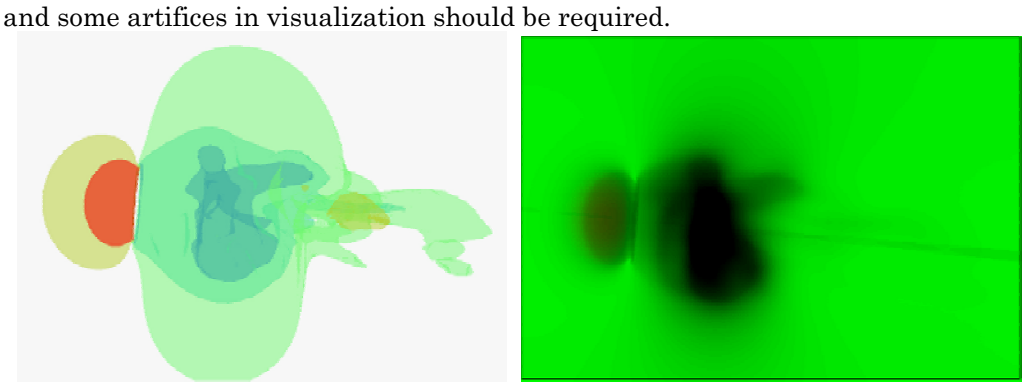

Fig. 1. Pressure distribution around a circular disk placed vertically in the streamwise direction shown by semi-transparent multiple isosurfaces (left) and volume rendering (right).

Therefore, research concerning the extraction of isosurfaces remains active. Isosurfaces have also advantages in computational cost and memory requirements. The computational cost of extracting isosurfaces is less than $O(N)$, and the amount of polygon data is less than $O(N)$, where N is the total number of computational cells in volume data. On the other hand, in the case of volume rendering using tomographic reconstruction for the warped cells (Cabral et al., 1994), the computational cost is $O(N^{4/3})$ and the amount of polygon data is $O(N^{4/3})$. In addition, the time required to render the polygons is greater than proportional to $N^{4/3}$. In this way, the extraction of isosurfaces is regarded as becoming more valuable as a numerical visualization method as the volume data becomes larger.

Isosurface extraction is basically implemented in a two-stage process. As shown in Fig. 2, the cells that intersect with the isosurface are searched (left and middle figures in Fig. 2), and then, the isosurface inside each such cell is locally approximated (right figure in Fig. 2). The isosurface is composed of a collection of geometric primitives. We call this geometric primitive a fragment of the isosurface. In the latter process, the marching cubes algorithm (Lorensen and Cline, 1987) is often utilized. In this case, the fragment of isosurface is a polygon.

In these two-stage processes, many accelerated isosurface extraction approaches have been proposed. In the former process, approaches are aimed at speeding up the search of the cells that intersect isosurfaces. Searching these cells using a span space is a common archetype (Shen et al., 1996; Livnat et al., 1996; Cignoni et al., 1997; Rymon-Lipinski et al., 2004). The span space is the value space composed of the maximum and minimum values in a cell. Other approaches related to the former process include a method for octree-based volume decomposition (Wilhelms and Van Gelder, 1992) and a technique that utilizes a topological structure generated from the local maxima and minima of a scalar field (Kreveld et al., 1997; Itoh et al., 2001).

Concerning the latter process, the fast methods are classified into a speed-up process of polygonizing isosurfaces (Howie and Blake, 1994) and drawing points, rather than polygons, on the isosurfaces (Rymon-Lipinsky et al., 2004; Livnat and Tricoche, 2004; Christopher et al., 2003). From the observation of huge datasets, Livnat and Hansen (1998) proposed an accelerated isosurfacing. Their method is called view-dependent isosurface extraction. In the abovementioned two-stage process, the visibility of cells from a fixed viewpoint is considered for cost reduction of searching and drawing processes.

In the present paper, a new fast semi-transparent multiple-isosurfaces extraction method is proposed. We focus on the following two points in developing the present method:
(1)Extracting multiple isosurfaces simultaneously.

(2)Rendering the fragments of isosurfaces in back-to-front order.

In order to satisfy these two requirements, we propose a 3D span space (hereinafter, the term 3D span space is abbreviated as 3D SS). In the 3D SS, we search the computational cells that intersect multiple isosurfaces simultaneously. The 3D SS gives a semi-transparent property to the isosurfaces efficiently.

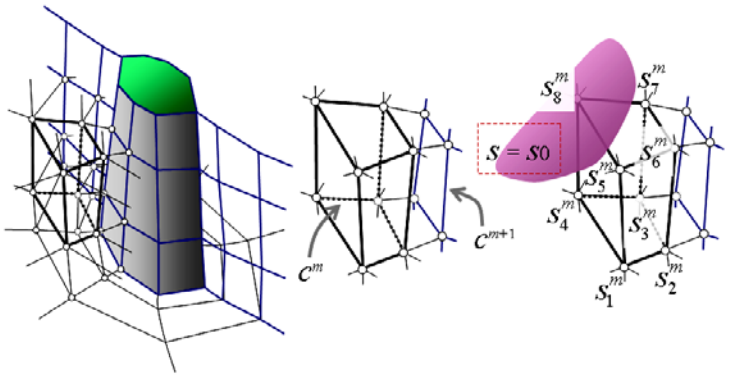

Fig. 2. Two-stage process of isosurfacing.

2. Related Work

2.1 Span Space

Shen et al. (1996) proposed *Isosurfacing in Span Space with Utmost Efficiency* (ISSUE) for accelerating isosurface extraction. After that, several methods have been proposed using the span space for extracting isosurfaces (Livnat et al., 1996; Cignoni et al., 1997; Rymon-Lipinski et al., 2004).

Let s denote a scalar field, and volume data consists of a collection of computational cells. In addition, let c^m represent the m-th computational cell. The cell has several vertices. The scalar s is distributed at the vertices in the cells (Fig. 2). Furthermore, s_l^m represents a scalar value at the vertex, where the superscript m indicates the cell number and the subscript l denotes the local number. A span space is constructed, and the cells intersecting an isosurface are searched as follows.

First, the local minimum and maximum values of the scalar are computed for each cell. For example, in Fig. 2,

$$s_{\min}^m = \min(s_1^m, s_2^m, s_3^m, s_4^m, s_5^m, s_6^m, s_7^m, s_8^m),$$
$$s_{\max}^m = \max(s_1^m, s_2^m, s_3^m, s_4^m, s_5^m, s_6^m, s_7^m, s_8^m) \quad (1)$$

are computed. The component of the span space is $(s_{\min}, s_{\max})$.

Second, all cells are mapped onto the span space, as shown in Fig. 3(left).

Third, the cells intersect an isosurface ($s = s0$) are searched. If c^m is the intersecting cell, then the following equation is satisfied.

$$s_{\min}^m \le s0 \le s_{\max}^m . \quad (2)$$

Therefore, the cells intersecting the isosurface are located in the upper-left region of the span space, as shown in Fig. 3(middle).

2.2 Implementation of ISSUE

The span space is an excellent concept to accelerate the extraction of the isosurface. However, for the case in which a huge number of computational cells exist, searching the cells that intersect the isosurface is time-consuming. Shen et al. (1996) solved this problem by subdividing the span space into L x L lattice. A lattice is denoted by (i,j) as shown in right figure of Fig. 3. Implementation of ISSUE is as follows.

[Preprocessing]
Let $A_{\min}(i)$ and $A_{\max}(j)$ denote the values of $s_{\min}$ and $s_{\max}$, respectively, at the lattice (i,j). A certain computational cell m is stored in the lattice (i,j), if

$$A_{\min}(i) \le s^m_{\min} < A_{\min}(i+1), \quad A_{\max}(j) \le s^m_{\max} < A_{\max}(j+1). \tag{3}$$

[Searching Cells]
A lattice (i,j) contains the cells intersecting the isosurface, if

$$s0 > A_{\min}(i), \quad s0 < A_{\max}(j), \tag{4}$$

where $s0$ is the isovalue of the isosurface.

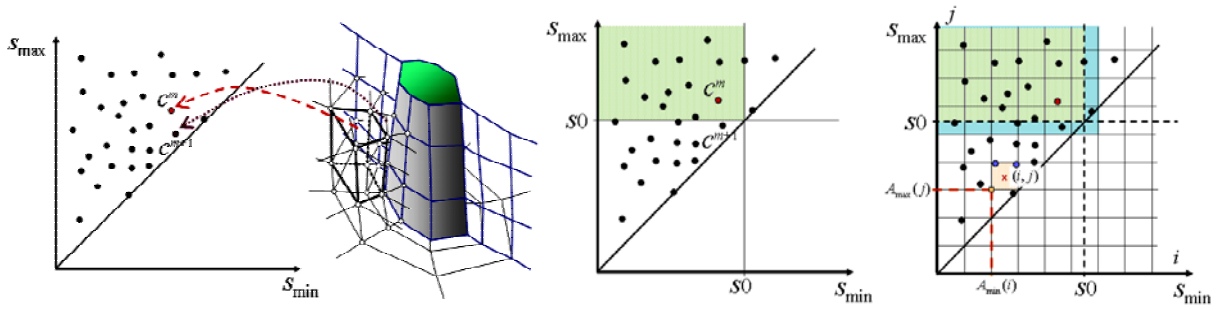

Fig. 3. 2D span space(left), search over the span space (middle) and a subdivided span space (right).

3. Proposed Approach

3.1 Three-Step Algorithm

If we obtain multiple isosurfaces using ISSUE, the four steps described below are needed:
(1) Computational cells intersecting an isosurface are searched, and stored temporarily.
(2) The isosurface inside the cells is extracted.
Repeat step (1) and (2) for each isosurface.
(3) Sort all fragments of the isosurfaces in back-to-front order.
(4) Render the fragments.

We try to reduce these four steps to the following three steps:
(1) Computational cells intersecting multiple isosurfaces are simultaneously searched and sorted in back-to-front order.
(2) The isosurfaces inside the cells are extracted.
(3) Render the fragments.
The details of these algorithms are described in the following subsections.

3.1.1 Algorithm to search the cells intersecting multiple isosurfaces simultaneously

In order to extract multiple isosurfaces simultaneously, we attempt to extend the ISSUE concept. For example, extracting three isosurfaces for which the isovalues are $s1$, $s2$ and $s3$ is considered:

$$(s^n_{\min} \le s1 \le s^n_{\max}) \cup (s^n_{\min} \le s2 \le s^n_{\max}) \cup (s^n_{\min} \le s3 \le s^n_{\max}). \tag{5}$$

Using the span space and Eq.(2), a cell c^n that satisfies Eq.(5) is one of the cells intersecting the three isosurfaces. The region containing the cells is indicated by the hashed region in Fig. 4(left). According to the ISSUE concept, this region is subdivided into a lattice as shown in Fig. 4 ((A), (B) and (C) in the right-hand figure). In this way, we can find the cells that intersect the isosurfaces, and three isosurfaces are extracted simultaneously.

3.1.2 Algorithm for sorting the cells in back-to-front order

In order to cut out the step of sorting all fragments in back-to-front order, the third axis is added to the span space. The distance from the viewpoint is assigned to the third axis, as shown in Fig. 5. This

space is referred to as the 3D span space (3D SS).

In the proposed algorithm, we first divide the third axis into several intervals and prepare a 3D lattice denoted by (i,j,k). Second, the coordinates of the centroid of each cell are computed, and the distance d^m from the viewpoint is obtained. Third, all of the cells $c^m : ((s^m_{\min}, s^m_{\max}, d^m), m = 1, \cdots, N)$ are stored in the 3D lattice, where N is the total number of computational cells. In this way, the cells intersect the isosurfaces are simultaneously sorted in back-to-front order.

3.1.3 Extracting isosurfaces and rendering

The isosurfaces inside the cells are locally approximated by some geometric primitive. In this study, we have two alternatives to approximate the isosurfaces. One is points in the cell. The other is triangles obtained by the marching cubes algorithm. In this paper, the former is mainly used for the approximation.

For rendering, a point rendering technique is utilized. Points at the centroids of the cells which sorted in back-to-front order are drawn in the same order of the cells. The color and opacity are defined according to the isovalues. We use GL_POINTS in OpenGL at the implementation level. This is one of the simplest point rendering. In the case that multiple isosurfaces intersect the cell, one of the isosurfaces is randomly selected. An approach that the cell is subdivided until the isosurfaces can be distinguished will be considered in the future work.

The hole-filling problem occurs in this point rendering. To avoid this problem, the latter approximation which the isosurfaces are extracted by the marching cubes algorithm should be used. However, we should remark that the computational cost of the marching cubes algorithm is higher than that of obtaining the centroid of the cell, the triangles in the cells consume at least three times more memory compared to the points, and the rendering cost has to be taken into account.

In the case of a large-scale visualization, the issue about the huge computational cost and memory requirements has become critical. In such visualization, we consider that the former approximation and the simplest point rendering can be the better alternative.

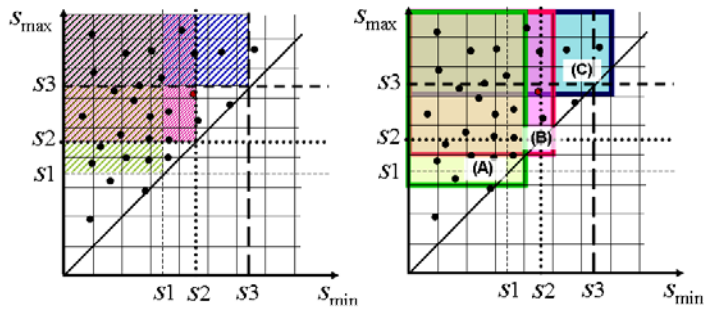

Fig. 4. Fast extraction of multiple isosurfaces simultaneously.

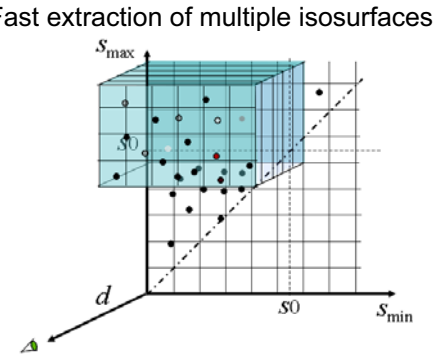

Fig. 5. Generation of the point groups in back-to-front order.

3.2 Computational Cost and Memory Usage

In this section, we estimate the computational cost of searching the cells. Let M be the number of isosurfaces, and let K be the average number of cells intersecting one isosurface. The span space is composed of L x L lattice.

In the case that multiple isosurfaces are extracted by a non-accelerated method and the fragments of the isosurfaces are rendered in back-to-front order, the computational cost for searching the cells intersecting each isosurface from N cells is $O(N)$. It takes $O(MK\log MK)$ time to sort the cells intersecting M isosurfaces if a standard sorting algorithm is used. The total amount of computational cost becomes $O(MN+MK\log MK)$. In the case of ISSUE, Shen et al. (1996), estimated the computational cost of searching the cells intersecting one isosurface is $O(\log(N/L)+\sqrt{N}/L+K)$. For M isosurfaces, adding the cost of sorting to that of searching, order of the total amount of the cost becomes $O(M\log(N/L)+\sqrt{N}M/L+MK+MK\log MK)$. In contrast, the computational cost of the proposed method for M isosurfaces is estimated by $O(\log(N/L)+\sqrt{N}M/L+K_2)$, where K_2 is the number of cells intersecting at least one isosurface. Although K_2 is larger than K, $O(K_2)$ becomes smaller than $O(MK)$ as M becomes larger. In addition, we do not need to sort the cells intersecting isosurfaces. Therefore, the proposed method has a great advantage in computational cost, as M or N becomes larger.

We then estimate the memory usage. Both ISSUE and 3D SS use an array for the span space. In the case of ISSUE, the memory usage is estimated as $O(L^2+N)$ (Shen et al., 1996). In the case of 3D SS, the additional memory should be needed for the third axis. This is related to the 3D lattice, and the memory usage is $O(L^2L_3)$, where L_3 is the number of subdivision on the third axis for the distance from the view point. All of the cells are assigned to a certain position in the 3D lattice. Therefore, the proposed method consumes $O(L^2L_3+N)$ memory.

4. Result

We demonstrate the proposed method by visualizing the flow past a circular disk placed vertically in the streamwise direction. Figure 6 shows the grid system. The number of computational cells is 99 x 99 x 101. The ratio of minimum and maximum cells is approximately 6 x 10^6. This visualization is performed for the warped cells.

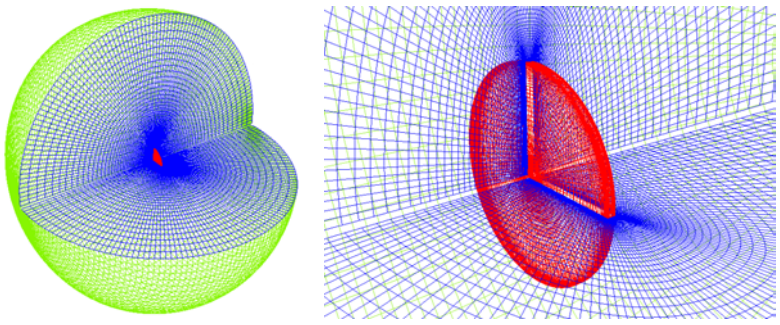

Fig. 6. A grid system for a flow past a circular disk placed vertically in the streamwise direction.

The results for the pressure field are presented in Fig. 7. The number of isosufaces is three to nine. The resultant images have many holes on the isosurfaces because of the point rendering (this is the hole-filling problem). With respect to image quality, an image obtained by the conventional method, like that shown in Fig. 1(left), may be superior to that by our method.

However, if the user has some information or knowledge about the field to be visualized, he/she can complement missing isosurfaces in the holes. In this case, such information that a pressure field in an axisymmetric incompressible fluid flow around a bluff body is visualized enables the user to complement the surfaces. And if user's viewpoint is fixed at a certain position, the following approach will help the user to recognize the image in which the hole-filling problem occurs. In the point rendering, the centroids of the cells $c^n:(s^n_{\min},s^n_{\max},d^n)$ that intersect the multiple isosurfaces are drawn. Computational cost of the distance d^n from the viewpoint is the same order of that of coordinate transformation in a graphics system. Since our method with the point rendering is much faster than the conventional method as described below, several images from slightly different viewpoints can be generated as fast as one image by the conventional method. The user will be able to complement missing isosurfaces in the holes. In this way, it may be said that the user can get

almost the same amount of information compared to the image obtained by the conventional method.

With respect to speed, the proposed method is significantly faster than the conventional method that multiple isosurfaces are extracted using the marching cubes algorithm without any acceleration techniques, and the triangles are rendered in OpenGL.

This is confirmed by two different datasets. The number of computational cells is 50 x 50 x 52 in Data 1, and 100 x 100 x 102 in Data 2. Tables 1 and 2 show the processing time of searching the cells for Data 1 and Data2, respectively. The computational costs are measured by averaging 100 experiments. The unit is milliseconds. The runtime environment was a IBM ThinkPad T60 (Pentium M 1.8 GHz, 1 GB memory), GNU gcc 3.4 and Windows XP. We note that, although the processing time of sorting the cells is not included, in the proposed method, the point groups are generated in back-to-front order.

Accordingly, the larger the number of data, the higher the performance of the proposed method becomes.

Table 1. Processing time of searching the cells for Data 1 (unit: milliseconds).

Number of isosurfaces	3	5	10
Proposed method	6.7	10.8	22.2
ISSUE	8.9	12.8	27.0
Conventional method	21.7	40.2	71.9

Table 2. Processing time of searching the cells for Data 2 (unit: milliseconds).

Number of isosurfaces	3	5	10
Proposed method	44.4	68.9	123.9
ISSUE	62.8	94.5	248.0
Conventional method	164.7	269.5	540.3

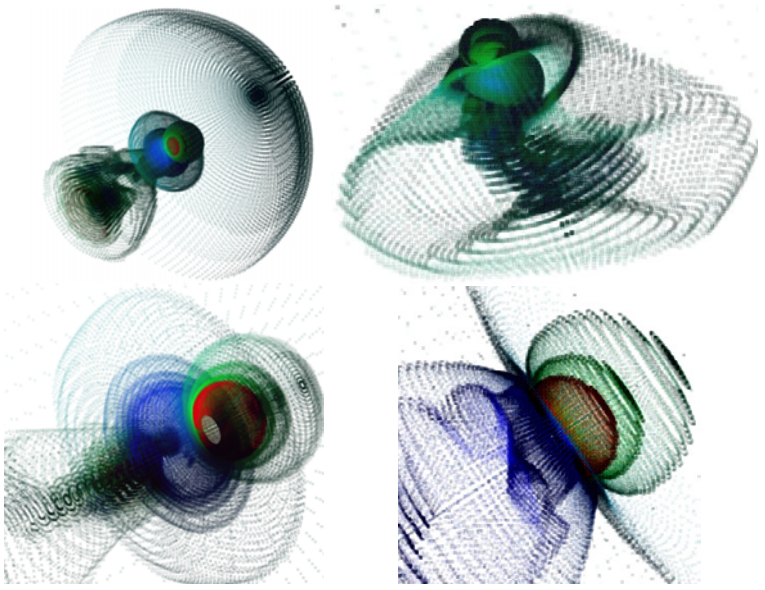

Fig. 7. Resultant images of the distribution of pressure around a circular disk placed vertically in the streamwise direction.

5. Conclusions

In order to speed up the generation of semi-transparent multiple isosurfaces, we considered the extension of the ISSUE concept. Our idea is that computational cells intersecting multiple isosurfaces are simultaneously searched and sorted in back-to-front order. This is realized by a new algorithm, in which all of the computational cells are mapped onto the 3D span space. The multiple

isosurfaces are approximated by points or triangles inside the cells intersect the isosurfaces. In this paper, the points represent the isosurfaces, and the isosurfaces are drawn by a simple point rendering technique.

A theoretical estimation of the computational cost and memory requirements and several numerical experiments showed that the proposed method accelerates the generation of semi-transparent multiple isosurfaces.

References

Bruno, F., Caruso, F., De Napoli, L. and Muzzupappa, M., Visualization of Industrial Engineering Data in Augmented Reality, Journal of Visualization, 9-3 (2006), 319-330.

Cabral, B., Cam, N. and Foran, J., Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, Proc. of Symposium on Volume Visualization, (1994), 91-98.

Christopher, S. Co, Bernd, H. and Kenneth, I. Joy, Reading in Iso-splatting: A Point-based Alternative to Isosurface Visualization, Pacific Graphics '03, (2003), 325.

Cignoni, P., Marino, P., Montani, C., Puppo, E. and Scopigno, R., Speeding Up Isosurface Extraction Using Interval Trees, IEEE Transactions on Visualization and Computer Graphics, 3-2 (1997), 158-170.

Howie, C. T. and Blake, E. H., The Mesh Propagation Algorithm for Isosurface Construction, Computer Graphics Forum (Eurographics), 13-3 (1994), C-65-74.

Itoh, T., Yamaguchi, Y. and Koyamada, K., Fast Isusurface Generation Using the Volume Thinning Algorithm, IEEE Transactions on Visualization and Computer Graphics, 7-1 (2001), 32-46.

Kobayashi, T., Tsubokura, M. and Oshima, N., High-Performance Computing and Visualization of Unsteady Turbulent Flows, Journal of Visualization, 11-1 (2008), 23-32.

Kreveld, M., Oostrum, R., Bajaj, C. L., Pascucci, V. and Schikore, D. R., Contour Trees and Small Seed Sets for Isosurface Traversal, Proceedings of 13th ACM Symposium of Computational Geometry, (1997), 212-219.

Liu, Z. and Moorhead II, R. J., A Texture-Based Hardware-Independent Technique for Time-Varying Volume Flow Visualization, Journal of Visualization, 8-3 (2005), 235-244.

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, 2-1 (1996), 73-84.

Livnat, Y.and Hansen, C. D., View Dependent Isosurface Extraction, Visualization '98, (1998), 175-180.

Livnat, Y. and Tricoche. X., Interactive Point-Based Isosurface Extraction, IEEE Visualization '04, (2004), 457-464.

Lorensen, W. E. and Cline, H. E., Marching cubes: A high resolution 3D surface construction algorithm, Computer Graphics, 21-4 (1987), 163-169.

Rymon-Lipinski, B., Hanssen, N., Jansen, T., Ritter, L. and Keeve, E., Efficient Point-Based Isosurface Exploration Using the Span-Triangle, IEEE Visualization '04, (2004), 441-448.

Shen, H.-W., Hansen, C. D., Linvat, Y. and Johnson, C. R., Isosurfacing in Span Space with Utmost Efficiency, IEEE Visualization '96, (1996), 287-294.

Stegmaier, S. and Ertl, T., On a Graphics Hardware-Based Vortex Detection and Visualization System, Journal of Visualization, 8-2 (2005), 153-160.

Wilhelms, J. and Van Gelder, A., Octrees for Fast Isosurface Generation, ACM Transactions on Graphics, 11-3 (1992), 201-227.

Author Profile

Miku Hirano: She received her B.Sc .(Science) in Information Science in 2006 from Ochanomizu University. She is a student in the Department of Environmental & Ocean Engineering, School of Engineering, University of Tokyo since 2006. Her research interests are Complex Network Analysis and Scientific and Information Visualization.

Takayuki Itoh : He received his M.Sc. (Eng) in 1992 from Waseda University. He also received his Ph.D in 1997 from Waseda University. He worked in IBM Research, Tokyo Research Laboratory as a research staff member. He works in Department of Information Sciences, Ochanomizu University as an Associate Professor since 2005. His research interests are in the area of Geometric Modeling, Computer Graphics, CAD/CAE, Scientific and Information Visualization, Web Services, and XML Security.

,

Susumu Shirayama: He received his M.Sc .(Eng) in Aeronautics in 1984 from University of Tokyo. He also received his Ph.D. in Aeronautics in 1987 from University of Tokyo. He worked in Institute of Computational Fluid Dynamics and Softek Systems Inc. as a director. He works in Reserch into Artifacts, Center for Engineering, University of Tokyo as an associate professor since 2002. His research interests are Intelligent Visualization, Complex Network Analysis, Digital Content Creation.