MATEC Web of Conferences **44**, 01067 (2016)
DOI: 10.1051/matecconf/20164401067
© Owned by the authors, published by EDP Sciences, 2016

A new type of mass structured data duplicate data check

Wen Qi Huang[1,a], Hua Jun Chen[1], Xiao Bin Guo[1], Li Peng[1]

ElectricPowerResearchInstitute,ChinaSouthernPowerGrid,China, 510080.

Abstract. In this study, the data processing method of a large-scale isoface is used to reduce the storage space of internal memory, improve data structure, and achieve predictability. The design of an algorithm mainly includes the following tasks: increasing the operation speed by improving the parallel granular of the GPU, finishing the segmentation of the adaptive tetrahedral octree, calculating the dual points through the quadric error function in four-dimensional space, searching for the minimum edge as well as finishing the data structure of the edge and the design of the octree node, and proposing that the algorithm be constantly performed until all levels of the minimum edge are found. The result of the algorithm design for the large-scale data is that the speed of parallel algorithms improves and the effect becomes more obvious. Research on large-scale data access high-speed ratio and structure improvement have experimental and theoretical reference value.

0 Introduction

The mode of computer storage and data organization is called data structure. Data structure refers to a set of one or more relationships between data elements. Generally, the selected data structure must lead to a higher operating or storage efficiency. The data structure is related to retrieval algorithm and index technology. The structure experience of large-scale systems verify that the degree of difficulty in achieving the quality and system configuration depends largely on the optimality of the data structure. In most cases, the algorithm is easy to obtain after the data structure is determined Occasionally, data structures are chosen in accordance with the specific algorithm. In other words, choosing the data structure that fits is crucial. In this study, a large-scale mesh simplification and an adaptive isoface-generating algorithm are investigated. Rossignac,

First, we consider how to obtain a manifold without a self-intersecting isoface from an adaptive octree data. In an adaptive octree data, every node has eight child nodes or none. The parent nodes are divided equally by all child nodes. The vertex of every node corresponds to a sampling point of the original volume data. All nodes are either a unit cubic voxel of the original data or contain it. All child nodes are filled

Borrel, and Kok-Lim proposed a mesh resampling method that deals with any form of grid data. DeFlorianie et al. proposed that simplifying a triangle set with the same boundary by replacing it with a triangle set in an adjacent grid. Lindstrom proposed an algorithm that processes triangular patches by batches. In the current study, we propose the use of a large-scale isoface based on streaming simplified thought and saving on the global optimal queue.

1 Design of adaptive octree algorithm

Most adaptive isoface algorithms are not guaranteed to produce a mesh manifold. To solve this problem, we propose a sorting method that simplifies and reduces the error produced by the volume data function. This method can be improved by exploiting the parallel granular and operating speed dynamic increase.

1.1 Tetrahedral generation of adaptive octree

with the entire volume data. We use the quadtree in a two-dimensional plane to explain this problem. This quadtree includes three-layer nodes. The lower right of the first layer has no child nodes. The other has 4 child nodes and constitutes the second-layer node, which includes 12 nodes. Four nodes in the second-layer node have child nodes and constitute a third layer, which includes 16 nodes. The third layer is a voxel of the

[a] Corresponding author: xuyunfei902@163.com

This is an Open Access article distributed under the terms of the Creative Commons Attribution License 4.0, which permits unrestricted use distribution, and reproduction in any medium, provided the original work is properly cited.

original volume data and cannot be divided. This quadtree has 25 child nodes, the first layer has 1, the second layer has 8, and the third layer has 16. Where nodes intersect at different layers of volume data, the edge of the node of the high layer adjacent to the lower nodes is split, so that higher-layer nodes no longer take a quadrilateral form.

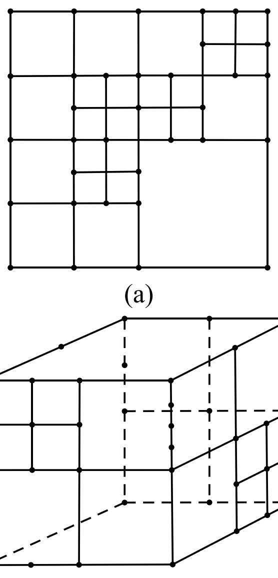

Figure 1: (a) adaptive quadtree in two-dimensional space and (b) divided higher-layer nodes of the octree

The octree in three-dimensional space is based on this intersection. Where the octree nodes intersect at different layers, the edge of the high-layer nodes is split, thereby resulting in a polygon. At the same time, the face is split and the high-layer nodes become a polyhedron. Figure 1(b) shows the splitting of the high-layer nodes of the octree. In the figure, this node becomes a polyhedron and a portion of the face becomes a polygon. The separation of the face and that of the edge both begin from this node adjacent to the lower-layer nodes. When the face is split, it produces a dual point. This dual point consists of four vectors (x, y, z, and w), where x, y, and z are the coordinates of the three-dimensional space, and w is the density value of the volume data. The dual points are connected to each edge of the face and form multiple triangles. The triangular segmentation of the outside surface of Figure 1(b) is presented in Figure 2.

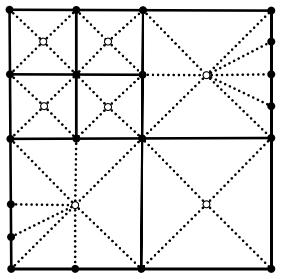

Figure 2: A divided triangle facing outside (white color point indicates the dual point, and dotted lines connect the octree and the dual vertex)

The splitting of the body follows a similar method. Each polyhedron generates a dual point. The triangle is obtained from the dividing face connected with this dual point.

1.2 Search for minimum edge

In the octree, the minimum edge cannot be divided and is significant for the formation of the tetrahedral. The storage structure of the minimum edge is similar to that of an octree. The data structure of the edge includes the direction of edge and the level and index of the octree node. To obtain it, we divide the edge into x, y, and z axes. Figure 3 illustrates the data structure of the edge.

```
struct OctEdge {
    EdgeDir dir;
int level1, level2, level3,
      level4;
int node1, node2, node3,
      node4;
        }
```

Figure 3 Data structure of edge

node1	*node2*
node4	*node3*

Figure 4 Position of adjacent nodes

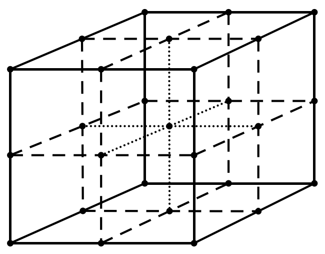

Figure 5 New edge introduced from the octree nodes (dashed line indicates the new edge on the top and dotted line indicates the edge of the nodes)

Figure 4 shows the position of the nodes. These four nodes are grouped into four faces: (node1, node2), (node2, node3), (node3, node4), and (node4, node1). If we assume that all edges of one layer are known, the edge is called the split edge. The process in which the current split edge produces the subsequent split edge may be described as follows. The number of segmentations of the current split edge is checked. The edge with zero segmentation and the edge with non-vanishing number of segmentations are considered as a parallel array segmentation. Then, the minimum edge is copied into the minimum edge array. At the same time, we store the edge with non-vanishing number of segmentations into the next layer array. The edge of the next layer to be split includes the new edge introduced when the octree nodes of the current layer are expanded. These new edges are included in the octree nodes and in the face (see Figure 5).

For the new edges of the octree nodes, the algorithm is parallel to the achieved number of new edges, in which each octree node has expanded. Then, new edges are written into the array to be split.

```
struct OctFace {
FaceDir dir;
int level1, level2;
int node1, node2;
}
```

Figure 6 Data structure of face

We have to solve the edges of every layer face. This face is the face to be segmented. The data structure of the face includes the direction of the face and the layers and index of the nodes adjacent to the face. The direction of the face includes the xy, yz, and xz planes. Figure 6 shows the data structure of the face. As it is similar to the edge to be segmented, we have to know the number of the next layer segmentation. We add the results into the array of the next layer face. The next face to be segmented contains the new face expanded from the nodes of the current layer. The method of adding new edges into the face involves computing the number of new edges and offsetting it through parallel scanning. Then, we add the new edge into the next layer array to be segmented, which includes three sections. These sections are the segmented edge of the last layer, the expanded edge of the last layer nodes, and the expanded edge of the last layer face. To obtain the pseudocode of every segmented layer, we conduct the following procedure. The segmented number of the current layer edge is obtained in parallel. The edge array is segmented in parallel according to the segmented number of the minimum edge, which is zero at the front. The minimum edge is stored into the current layer minimum edge array. The non-vanishing segmented number undergoes SCAN in parallel and is offset. The number of the new parallel edges and faces is obtained. The new edges from the expanded nodes undergo SCAN and are offset in parallel. The new faces from the expanded face undergo SCAN in parallel and are offset in parallel. The number of new edges and faces from every segmented parallel face is obtained and offset. The segmentation number of the surface undergoes SCAN and is offset. The next layer array of the edge to be segmented is entered in the graphics memory. The three parts of the edges to be segmented are added into the array. The next layer array of the face to be segmented is entered in the graphics memory. The tow parts of the edges to be segmented are added into the array.

At the start, we produce the edge and the face of the first layer. Through continuous and iterative execution of the algorithms, the minimum edges are found.

2 Execution results of algorithm design

For different models, we use the adaptive isosurface and large-scale isosurface. We test the speed of the parallel algorithm, implementation results, and execution time,see table 1 and Fig 7. In the test, the simplified algorithm of the unified cache size is set to 20,000.

Table 1 the serial algorithm and implementation time speedup of different models

Model	Size	Iso	Sl	Et	Sides
Teapot	256x256x178	60	5	4000	293042
Engine	256x256x110	127.5	5	1000	1044735
CT_	256x256x106	100	6	103500	183630
HKugel	64x64x64	100	3	990	32592
blunt	256x128x64	90	3	5000	9206

Teapot	0.0803	0.0547	0.166	0.426	13.198	30.9
Engine	0.118	0.125	0.523	0.866	16.614	19.18
CT_	0.102	0.0574	0.191	0.443	4.352	9.82
HKugel	0.0259	0.0106	0.0242	0.141	0.406	2.88
blunt	0.0345	0.0376	0.0673	0.219	0.343	1.57

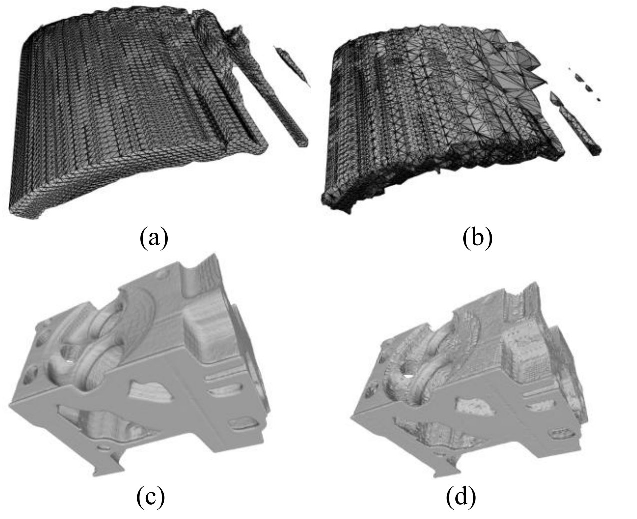

Figure 7 Isosurface of the model: (a) isosurface of a Blunt model using a marching tetrahedron, (b) isosurface of a Blunt model using this algorithm, (c) isosurface of an engine model using a marching tetrahedron, and (d) isosurface of an engine model using this algorithm.

The results indicate that for the large-scale data, the speed of the parallel algorithm is larger and its effect is more obvious. In addition, the simplified model shows that the effect of this algorithm is better on smoother surfaces.

3 Summary

Most adaptive isosurface algorithms cannot be guaranteed to produce a grid manifold without self-intersection. However, a sequential algorithm based on volume data simplex segmentation can solve this problem. This study is based on sequential simplex segmentation. The method of segmentation is simplified to reduce the approximate error of the volume function. Furthermore, the method is improved and is made to fully exploit the use of the parallel granular of GPU, which increases the operating speed. We propose a complete set minimum edge search method. According to the minimum edge, the nodes of the octree undergo tetrahedral segmentation, and the tetrahedral uses a marching tetrahedron to obtain the isosurface of the volume data. In practical tests, the algorithm can produce a high-quality adaptive isosurface.