ICAT 2004

Real-time Marching-cube-based LOD Surface Modeling of 3D Objects

Hasup Lee
KAIST, Daejeon, 305-701 Korea
hasups@kaist.ac.kr

Hyun S. Yang
KAIST, Daejeon, 305-701 Korea
hsyang@cs.kaist.ac.kr

Abstract

The marching cube octree data structure is a scheme for representing and generating the mesh of various level-of-details (LODs) and was proposed in [15]. We suggest a solution to a problem happened when modeling partially complex objects by improving the original LOD model. The marching cube octree is based on the data structure of the Marching Cube algorithm [1], which is used to generate the mesh from the range data and the octree, this last widely used in computer graphics. Our LOD model can support adaptive simplification, compression, progressive transmission, view dependency rendering and collision detection. Our LOD mesh generation algorithm is faster than previous methods because it directly references the marching cube octree.

Key words: Surface Representation, Hierarchy Transformation, Level-of-detail Modeling

1. Introduction

Previous methods of LOD modeling are mostly concentrated on a similar description of the original object. Creations of representations are very complex or expensive, however, and mesh generation is too slow to be used in practice. Thus, most commercial real-time systems have their own model. In a real-time system, user interactiveness is more important than the precision of the description.

This paper proposes a new data structure improved from marching cube octree [15]. We used this data structure to make the new LOD model. Using the sampling paradigm, our algorithm is faster than most of previous methods. If the Marching Cube algorithm is used to generate the mesh, the proposed method can make the "LOD-controllable 3D model" directly from the range data.

1.1. Related Work

1.1.1. Adaptive Subdivision

An adaptive subdivision and analysis method that applies wavelet-based multi-resolution analysis to an arbitrary topology surface was proposed [6][7]. This method can perform smooth parameterization at any LOD and can be applied to adaptive simplification, compression, progressive transmission and editing [8][9][10]. The wavelet-based method makes some of these advantages possible. Nevertheless, this method renders the making of the base mesh expensive and slow. Too many triangles are needed and generated when resolving small local features.

To overcome these drawbacks, a new algorithm, MAPS, was proposed [11]. The MAPS algorithm uses hierarchical simplification, defined by vertex removal, flattening and retriangulation, to induce a parameterization of the original mesh over a base mesh. Although this method can reduce the complexity of the base mesh formulation and resolve small features well, it cannot support view dependency rendering and collision detection, which are important in computer graphics systems.

1.1.2. Geometry Removal

Another algorithm called Progressive Mesh was proposed that makes the new mesh by defining the edge collapse and the vertex split operation, and applying these to the detailed mesh [3]. In addition, a new format was developed for saving and transmitting the triangulated geometric model [4]. The Progressive Mesh method can be applied to adaptive simplification, compression, progressive transmission and view dependency rendering [5]. The model generation is relatively slow, however, because the simplification is based on the energy function. This method also slightly supports collision detection.

1.2. Features of Marching Cube Octree

Our algorithm was designed to rapidly construct the LOD model and generate the LOD mesh. We approximated the 3D object conceptually with sampling range data in many resolutions. We used the octree and the marching cube to represent the LOD model. The operation necessary for the construction of the marching cube octree is relatively simple. We used the octree that naturally supports progressive transmission, view dependency rendering and collision detection. We did not implement these features yet but octree-based view-defendant rendering has been carried out efficiently by [13]. We can construct

the LOD model directly from the range data by using the marching cube data structure, if the Marching Cube algorithm is applied for the mesh generation. Our algorithm directly generates the LOD mesh by referencing only the needed nodes of the tree.

2. Marching Cube Octree Representation

2.1 Overview

The Marching Cube mesh-generation algorithm for medical images like MRI and CT was proposed [1]. The cube, which includes the in/out configuration of each of the 8 vertices, is classified into 14 distinct cases. Triangles are created automatically for each case. The vertices of the triangles are at the midpoints of the cube's edges. The Marching Cube algorithm can also be used to generate the mesh from the range data.

To generate triangles, we use the sign and the ratio converted from the signed distance of the cube's vertices. The signed distance was proposed for surface reconstruction from unorganized points [2]. It is defined as the distance between the vertex P and the closest range point multiplied by +-1, depending on which side of the surface P is. The sign of the vertex is defined as the sign of the signed distance. The ratio of the edge is defined as the ratio of the absolute value of one vertex's signed distance to that of the others. We can generate triangles naturally using a low resolution by choosing the proportional point instead of the midpoint for the vertex of the triangles.

Using an octree for the representation of 3D object is not new idea. The isosurface generation using marching cubes and octree traversal was proposed in [14]. This paper describes efficient creation of octree based representation. But we applied modified marching cube configuration to octree structure, thus can construct hierarchies and implement level-of-detail.

We define a marching cube in this paper as the set of signs and colors for each of the 8 vertices, and of ratios for the 12 edges. A marching cube octree is defined as a spatial octree whose nodes are marching cubes. Since the dimension of the root node is known, we can determine the relative position and the size of any node in the octree. A null node is defined for all edges of the corresponding marching cube that do not intersect with the surface. Thus, all vertex signs are the same. The null node has no child nodes and null points. Finally, the nodes of the marching cube octree correspond only to the region of the surface.

2.2. Creation of the Marching Cube Octree

The marching cube octree is created from the marching cubes of the Marching Cube algorithm [15]. The creation algorithm consists of two operations: parent node creation and marching cube conversion. The spatial octree is created using a bottom-up approach. We make the parent node from adjacent nodes and then convert it into the marching cube.

Creation of the marching cube octree starts with marching cubes of the most detailed resolution from the Marching Cube algorithm. Let the most detailed marching cube's level be 0. Eight or fewer marching cubes that are adjacent to a certain vertex are grouped and become child nodes of the new parent node. We convert the parent node into the marching cube, and then make the next parent nodes successively until all level 0 nodes are covered. Then we repeat this process from level 1 to level 2 and so on, until a marching cube is created in level N. If the child node's corresponding marching cube does not exist, the parent node has the null node for that child node.

A parent node can be made into the marching cube by referencing its child nodes. The decision on the sign of the vertex is classified into four cases [15]:

(1) if a corresponding vertex exists in the child nodes, the sign is the same as that of vertex A;

(2) if an adjacent vertex exists in the child nodes, the sign is the same as that of vertex B;

(3) if a diagonal vertex exists in the child nodes, the sign is the same as that of vertex C; and

(4) if no corresponding, adjacent and diagonal vertex exists in the child nodes, the sign is the same as that of the center vertex (vertex D).

In case (2), several adjacent vertices can exist in the child nodes, but their signs will all be the same. The color value is copied in the same manner. The ratio is calculated easily by extending the vertex that has the triangle's vertex in it (Figure 6). The ratio of the vertex that does not have the triangle's vertex in it is unnecessary.

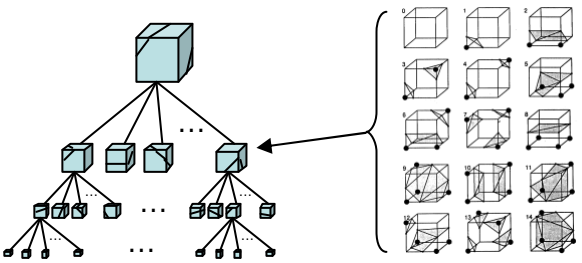

Figure 1. Marching Cube Octree

The intermediate data structure of our algorithm shows in (Figure 1). The right part of the (Figure 1) is the cube

configuration figure proposed in [1].

2.3. Node Priority Numbering

To detail the LOD of the model, we assign a priority number to all the nodes. The priority numbering determines the order of the node expansion. We use the area difference as the LOD metric. The area difference is defined as the difference between the area of the triangles generated from one node and the area of the triangles generated from the child nodes. The higher the area difference is, the more detailed is the description of the local feature and the higher the priority is. To describe the 3D object with fewer triangles, the higher priority node expands first in the same level.

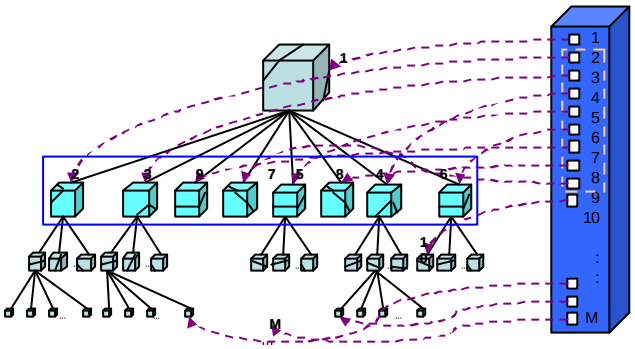

Figure 2. Marching Cube Octree with LOD Array

To generate the LOD mesh rapidly, we save this priority number in a referencing array, the LOD array. The number N means the Nth node to be expanded. Thus, its child nodes are triangulated, as shown in (Figure 2). When the LOD of the mesh is 1, the child nodes of the 1st node (the root) are triangulated. When the LOD of the mesh is 2, the 2nd node' s child nodes and the rest of the 1st node' s child nodes—the 3rd to the 7th nodes—are triangulated.

Cracks are generated at the interfaces of nodes with varying levels (left-hand side of Figure 3 is from [12]). This is a common problem with adaptive subdivision algorithms. Crack patching algorithm was proposed in [12] and we use that in our algorithm. This algorithm can apply the case that adjacent node' s level difference is 1. When priority numbering, we check all adjacent nodes' level. If only all of that are 1 or all are 0, we can expand this node. This case (node A) is illustrated conceptually by 2-dimesion in (right-hand side of Figure 3).

The level difference between node B and the child node expanded from node A is more than 1 is remarkable. This method is depth-first manner but the original priority numbering algorithm is breadth-first one. The modeling of partially complex object is efficient by this method.

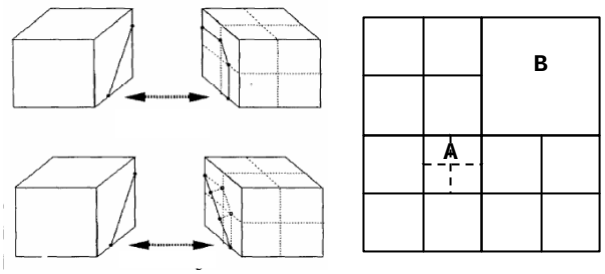

Figure 3. Crack Patching

3. Direct Generation of Level-of-detail Mesh

Using the algorithm mentioned above, we constructed the marching cube octree presentation of a certain mesh. In this section, we consider generating the LOD mesh using this representation. We generate the LOD mesh rapidly and efficiently by referencing the LOD array.

The LOD of the LOD mesh is controlled by using the LOD array. In the LOD array, there is a triangulated node sequence of all nodes in the marching cube octree. When the next node is triangulated, the number of triangles in the mesh is increased by d (0= d= 4). We check each node already expanded in its subtree on the reverse order sequence from given priority number' s node. The pseudo code is written as follows:

```
For all node n (n' s priority number is i, i-1, ... , 2, 1)
    If all n' s child nodes are ' expanded' ,
        exit loop;
    Otherwise,
        triangulate n' s child nodes
        except ' expanded'
        mark n as ' expanded'
end of loop;
```

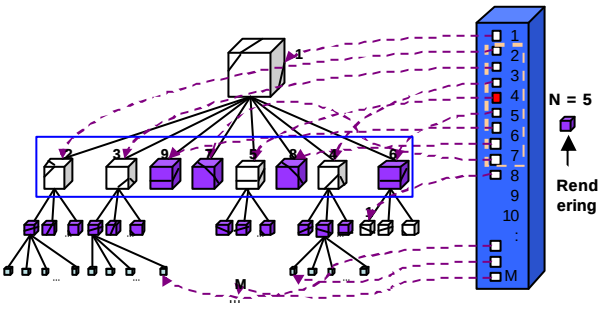

Figure 4. Direct Generation of LOD Mesh

If the number of triangles in the mesh is given, we convert this input to the priority number by using the accumulation table. The accumulation table contains a triangle increment by the priority order triangulation. Thus, we can get the approximated priority inversely from

the number of triangles. The final data structure of our algorithm shows in (Figure 4).

4. Results

The numbers of the triangles are 119262, 45857, 25038, 5585 in male body and 1705, 23184, 59083, 114236 in Venus. The meshes of the upper part are wire-frames and of the lower part, rendered meshes (Figure 5). If the rendered models are far from user position, their appearances are indistinguishable. So these results show our model is feasible for LOD representation.

Our algorithm directly generates the LOD mesh by referencing only the needed nodes of the tree. Let N is the triangle number of the most detailed mesh and C is of the coarsest. If m is the triangle number of an arbitrary LOD mesh (m ? [C,N]), The time complexity of our algorithm is O(m), because it refers only needed cubes from the LOD array. The time complexities of previous models are the same O(m^2) because of the serial accumulation manners. The space complexity of our model is the same as the other models. The additional space of the LOD array used for fast mesh generation occupies only small space - just pointing(indexing) - and needed only in the processing time.

5. Conclusion

In this paper, we propose the improved method of LOD modeling using the marching cube octree. We create the representation easily and efficiently by using the marching cube features. We can take advantage of the octree representation in a 3D graphics system. Our LOD model can support adaptive simplification, compression, progressive transmission, view dependency rendering and collision detection. By using the sampling paradigm, our LOD mesh generation algorithm becomes faster than previous methods. We can construct the LOD model directly from range data by using the marching cube data structure, if the Marching Cube algorithm is applied for the mesh generation. We improved the feasible method of 3D object LOD modeling.

References

1. William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. SIGGRAPH 87 Conference Proceedings, Vol. 21(4), 163-170.

2. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. 1992. Surface Reconstruction from Unorganized Points. Computer Graphics (SIGGRAPH 92 Proceedings), Vol. 26(2), 71-78.

3. Hugues Hoppe. 1996. Progressive Meshes. Computer Graphics, Vol. 30, Number Annual Conference Series, 99-108.

4. Jovan Popovic and Hugues Hoppe. 1997. Progressive simplicial complexes, Computer Graphics, Vol. 31, Number Annual Conference Series, 217-224.

5. Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. In Computer Graphics (SIGGRAPH 97 Proceedings), 189-198, 1997.

6. Lounsbery, M., Derose, T., and Warren, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. Transactions on Graphics 16, 1 (January 1997), 34-73.

7. Lounsbery, M. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. PhD thesis, Department of Computer Science, University of Washington, 1994.

8. Eck, M., Derose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution Analysis of Arbitrary Meshes. In Computer Graphics (SIGGRAPH 95 Proceedings), 173-182, 1995.

9. Certain, A., Popovic, J., Derose, T., Duchamp, T., Salesin, D., and Stuetzle, W. Interactive Multiresolution Surface Viewing. In Computer Graphics (SIGGRAPH 96 Proceedings), 91-98, 1996.

10. Zorin, D., Schroder, P., and Sweldens, W. Interactive Multiresolution Mesh Editing. In Computer Graphics (SIGGRAPH 97 Proceedings), 259-268, 1997.

11. Aaron W. F. Lee, Wim Sweldens, Peter Schroder, Lawrence Cowsar and David Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. SIGGRAPH 98 Conference Proceedings, Annual Conference Series, pp. 95-104, Addison Wesley, July 1998.

12. Raj Shekhar, Elias Fayyad, Roni Yagel, J. Fredrick Cornhill. Octree-Based Decimation of Marching Cubes Surfaces. Proceedings of the Conference on Visualization, pp. 335-344, IEEE, October 27- November 1 1996.

13. David Luebke and Carl Erikson. View-Dependent Simplification of Arbitrary Polygonal Environments. Proceedings of SIGGRAPH 97, ACM Press, August 1997.

14. Wilhelms, Jane and Allen Van Gelder. Octrees for Faster Isosurface Generation, ACM Transactions on Graphics, Vol. 11, No. 3, pp. 201 - 227, July 1992.

15. Hasup Lee, Juho Lee and Hyun S. Yang, Real-time LOD: Marching-cube-and-octree-based 3D Object Level-of-detail Modeling, The 8th International Conference on Virtual Systems and MultiMedia (Proceeding), pp 634 - 643, September 2002.

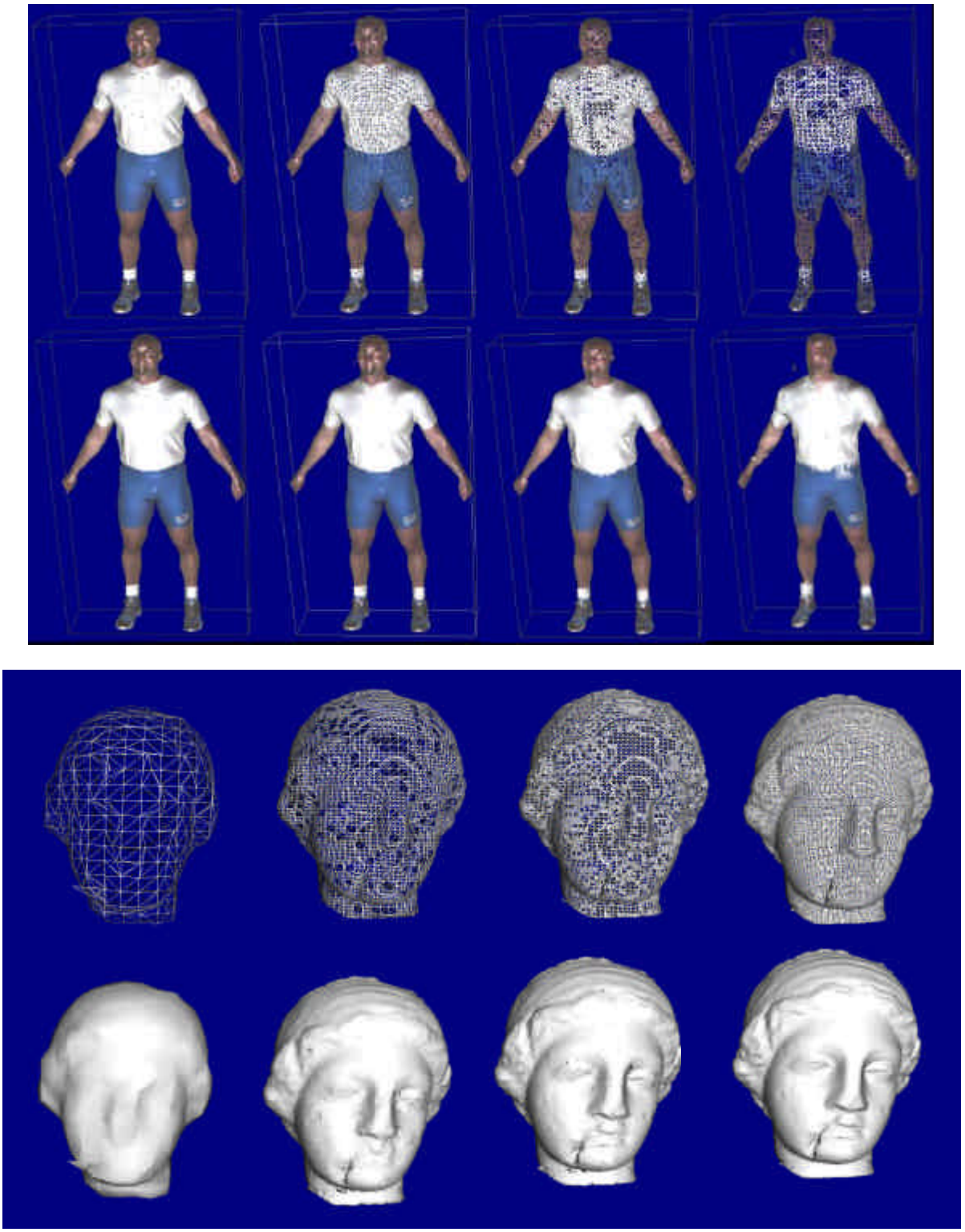

Figure 5. Examples of male bodies and Venus (Dataset courtesy of Cyberware)

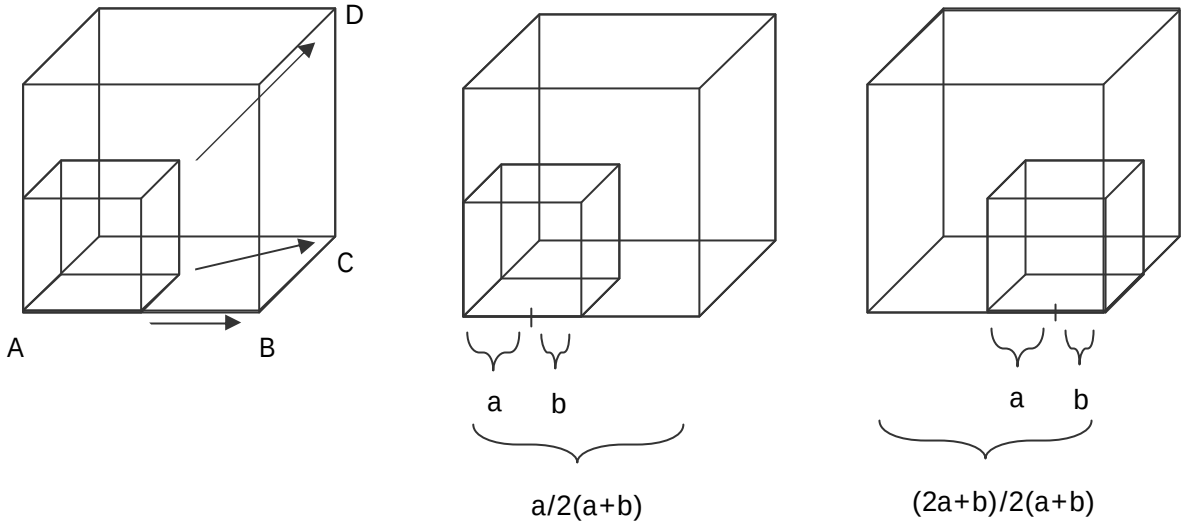

Figure 6. Conversion of the marching cube