Surface Level-Of-Detail Modeling of 3D Objects Using Marching-Cube Octree

Hasup Lee
KAIST
hasups@kaist.ac.kr
http://hasups.kaist.ac.kr

Hyun S. Yang
KAIST
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 (LOD). The marching cube octree is based on the data structure of the Marching-Cube algorithm which is used to generate the mesh from the range data and on the octree that is widely used in computer graphics. In this paper, the feasible LOD model using the marching cube octree is constructed and applied to the partially complex 3D objects. This LOD model can support adaptive simplification, compression, progressive transmission, view dependency rendering and collision detection.

Keywords: Virtual Reality, Mesh, Surface Representation, Multi-Resolution Modeling, Level-of-detail Modeling

1. Introduction

In 3D graphics systems like virtual reality, mixed reality, arcade games, objects are represented by a set of triangles and the set is called a mesh. Objects can be described in detail when many triangles are used but the rendering process is slow. Objects can be rendered faster when small triangles are used but they are described coarsely. We should generate and use meshes that fit the performance of the rendering system because the performance is changed continuously by the number of objects or the complexity of the background scene changed. If we can control the LOD of the object, we can construct an effective computer graphics system.

For given point set P, normal set N and some LOD number l, the problem is defined as finding a certain function Fl that returns triangle sets satisfying LOD degree l.

$$P = \{p \mid p \text{ is input point from range scanner}\}$$
$$N = \{n_p \mid n_p \text{ is normal vector of } p, p \in P\}$$
$$l = [n_0, n_1]\, n_0, n_1 \in \mathrm{N}, n_0 < n_1$$

$$Fl(l, P, N) = \{t \mid t \text{ is triangle satisfying LOD } l \text{ generated from } P, N\}$$

The LOD modeling consists of the representation (data structure) of the mesh and the algorithm to generate the mesh of a certain LOD to implement the function Fl.

Previous methods of LOD modeling are mostly concentrated on a precise description of the original object. The creations of the representations are very complex or expensive and the mesh generation is too slow to be used in practice. 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 called marching cube octree, which is based on the data structure of a Marching-Cube algorithm used to generate mesh from range data. We

used this data structure to make the new LOD model. Using the sampling paradigm, our algorithm turned out to be faster than previous methods. The proposed method can make the "LOD-controllable 3D model" directly from the range data.

1.1 Related Works

There are two main categories of researches on LOD modeling by geometric criteria: 1) algorithms that adaptively subdivide an existing mesh with multi-resolutional approaches, [2][3] and 2) algorithms that remove geometry features such as vertices, edges, etc., approximate the most detailed mesh, and construct parameterization [4][5].

1.1.1 Adaptive Subdivision

Many adaptive subdivision methods have been researched since the fundamental work by [1]. In the field of the LOD modeling, an adaptive subdivision and analysis method that applies wavelet-based multi-resolution analysis to an arbitrary topology surface was proposed [2][3]. This method can perform smooth parameterization at any LOD and can be applied to adaptive simplification, compression, progressive transmission and editing [6][7][8]. 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 [9]. 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 [4]. In addition, a new format was developed for saving and transmitting the triangulated geometric model [5]. The Progressive Mesh method can be applied to adaptive simplification, compression, progressive transmission and view dependency rendering [10]. 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 [11]. 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 initial 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 Marching Cube Node

The Marching-Cube algorithm for medical images like MRI and CT was proposed [12]. 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 [13].

For given point set P, normal set N and some cube C containing vertex $v_1, \ldots, v_8$, we define the closest point p, distance d and sign s for the vertex configuration vc and cube node CN as follows:

$$p_n = \text{the closest point to } v_n,\ p_n \text{ is in } C,\ p_n \in P$$
$$d_n = \|v_n - p_n\| \text{ (euclidian distance)}$$
$$s_n = \frac{(v_n - p_n) \bullet n_{p_n}}{\left|(v_n - p_n) \bullet n_{p_n}\right|},\ n_{p_n} \text{ is normal of } p_n,$$
$$n_{p_n} \in N(+1 : \text{outside}, -1 : \text{inside})$$
$$CN = \{vc_m \mid vc_m = s_m \times d_m, m \in [1,8]\}$$

By the Marching-Cube algorithm, triangles can be generated easily using CN and look-up table.

2.2 Creation of the Marching Cube Octree

The cube data structure of the Marching-Cube algorithm is basis of the marching cube octree. Instead of signed distance [13], the marching cube octree utilize the ratio on each edge for generating triangles.

For given position vector w, vertex sign s_n and edge ratio er_m and the marching cube node MC containing vertex $v_1, \ldots, v_8$ are defined as follows:

$$s_n = \begin{cases} true & \text{if } v_n \text{ is inside.} \\ false & \text{if } v_n \text{ is outside.} \end{cases}$$

$$er_m = \begin{cases} 0 & \text{if adjacent vertex's sign is same.} \\ i, i \in (0,1) & \text{otherwise} \end{cases}$$

$$MC = \{w\} \cup \{s_n \mid n = 1,\cdots,8\}$$
$$\cup \{er_m \mid m = 1,\cdots,12\}$$

Then for given some marching cube node r, the following recursive definition is the marching cube octree T_r.

$$T_r = \begin{cases} r & \text{if } r \text{ is leaf node.} \\ r \cup \bigcup_{s \text{ is } r\text{'s child}} T_s & \text{otherwise} \end{cases}$$

The creation of the marching cube octree starts with marching cube nodes of the most detailed resolution (leaf node) and merged into parent's level as Figure 1. For leaf marching cube nodes, vertex sign is the same as corresponding cube node and edge ratio is obtained from simple divide calculation.

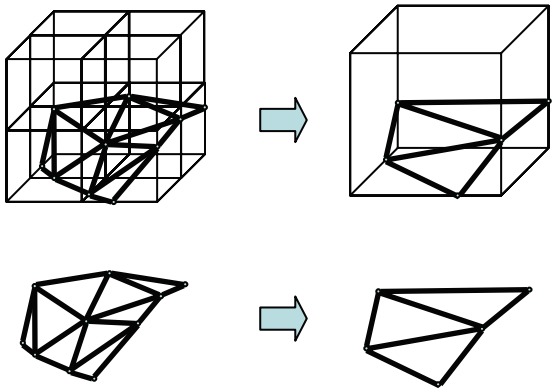

Figure 1 : Creation of the parent node

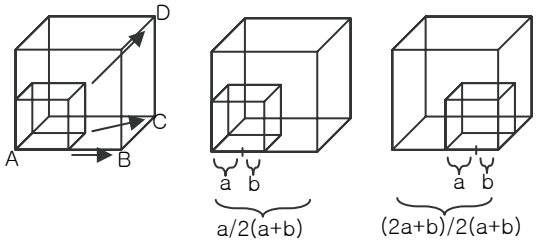

Figure 2 : Conversion of the marching cube

For internal nodes, a parent node can be constructed by referencing its child nodes. The decision on the sign of the vertex is classified into four cases (Figure 2).

1) if a corresponding vertex exists in the child node, 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 2). The ratio of the vertex that does not have the triangle's vertex in it is unnecessary.

The advantage of containing the ratio on edges instead of the distance of vertices is simplifying crack patch problems by sharing the same vertex of triangle generated from the different level marching cube node. Using the origin and the size of node, the parent-children relationship can be calculated but the pointer data structure is used for the efficiency of implementation. The intermediate data structure of our algorithm shows in Figure 3. The right part of the Figure 3 is the cube configuration figure proposed in [12].

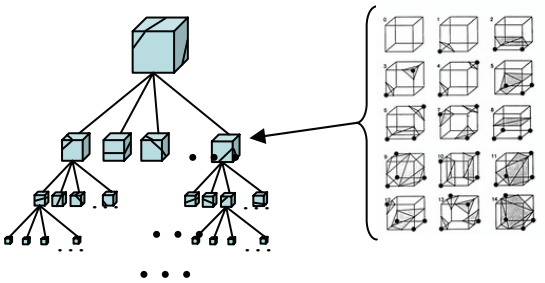

Figure 3 : Marching cube octree

2.3 Node Priority Numbering

To control the LOD of the mesh, we assign a priority number to all the nodes. The mesh generated from all nodes of one level has exactly twice the resolution of the mesh from of the parent nodes' level. Thus, only the LOD of the transition from one level mesh to next is needed to be considered. The priority numbering determines which node expands first in the same level in the previous work [14].

We use the area difference as the LOD metric. For given marching cube node q and r, the area difference M_r and the triangulation function $F_{MC}(q)$ by Marching-Cube algorithm are defined as follows:

$$F_{MC}(q) = \{t \mid t \text{ is triangle generated using Marching - Cube algorithm from } q\}$$

$$M_r = \sum_{s \text{ is } r\text{'s child}} \sum_{area} F_{MC}(s) - \sum_{area} F_{MC}(r)$$

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.

But to describe the complex parts of the object in detail, we improve the previous algorithm. This algorithm starts from the root node. The expanded nodes are marked as 'expanded'. When the level differences between some node and its all surrounding nodes are less than 2, that node can be expanded. The node of the biggest the area difference - LOD metric - in the expandable nodes set is inserted in LOD array. Then this node is marked as 'expanded'. This algorithm can be implemented effectively using the priority queue.

To generate the LOD mesh rapidly, we save this priority number in a referencing array, the LOD array. The number N means the N-th node to be expanded. Thus, its child nodes are triangulated, as shown in Figure 4 in the end of this paper. When the LOD of the mesh is 1, the child nodes of the node 1 (the root) - node 2~9 - are triangulated. When the LOD of the mesh is 2, the node 2's child nodes and the rest of the 1st node's child nodes - node 3~9 - are triangulated.

2.4 Crack Patching

Cracks are generated at the interfaces of nodes with varying levels (the left figure of Figure 5 is from [15]). This is a common problem with adaptive subdivision algorithms. Crack patching algorithm was proposed in [15] and we use that in our algorithm. This algorithm can apply the case that adjacent node's level difference is 1. For fast generation of the LOD mesh, thus we calculate all compensation points before the mesh is generated. Only the compensation points of crack-happen nodes are calculated and saved. In the right

figure of Figure 5, the compensation point R of $P3$ is a cross point of $P1P2$ and a perpendicular line to $P1P2$ containing $P3$.

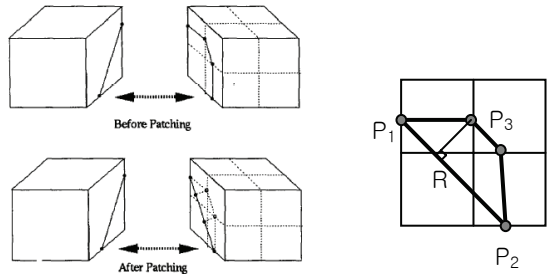

Figure 5 : Crack patching

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 ($0 \leq d \leq 4$). We check each node already expanded in its sub-tree 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;
```

The flag 'expanded' are no related to that of node priority numbering.

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 as Figure 6 in the end of this paper

4. Results

Table 1: Numbers of triangles, vertices in examples of ball joint and Venus

	Ball joint		Venus	
LOD (%)	triangles	vertices	triangles	vertices
20	4212	3077	6838	4961
40	7967	5578	12635	8757
60	11565	7422	18216	11102
80	14735	8228	23152	12835
100	18159	9106	28560	14335

Recently most of range scanners provide the polygon format data as the results like 'PLY' files. So we generate the range image data from them. The vertices are used as the points and the normal vectors are calculated from neighbor triangles' normal.

We use the 'ball joint bone' and the 'Venus statue' meshes as experimental data downloaded from Cyberware™ homepage. The results are showed by 20% of LOD. The meshes of the left side are wire frames and of the right side, rendered meshes in Figure 7 in the end of this paper. The numbers of triangles and vertices is showed in Table 1. These results show our model is feasible for LOD representation.

LOD metric – the difference between the sum of a node's triangles area and the sum of its all child node's triangles area – data are used for the priority numbering. The graph in Figure 8 in the end of this paper is LOD metric data after the priority numbering. The upper graph is the case of ball joint and the lower is Venus. The grey line is data of proposed approach and

the black line is of previous level-based algorithm [14]. They mean our method is beyond level constraint and more continuous model.

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 ($m \in [C,N]$), 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.

About the qualities of surface, the sampling model may have some regularities (step-appearance) or discontinuities between the levels. By using the ratio of the edges' distance, our model doesn't have the regularities and the discontinuity problem is solved by the improved priority numbering scheme. 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 initial mesh generation. It also saves the time of making the polygon-format data of the range scanner. Our model is very useful when it is applied to fast LOD model generation of a large number of data like the exhibits of a ancient museum.

6. Future Work

In our model, the marching cube is divided into halves in the next level. If we control the size of the next level's cube and formulate it, we can construct a more detailed continuous LOD model. The fastness and simplicity of the modeling may be lost.

Acknowledgements

This work was supported by the Ministry of Science and Technology(MOST) Korea Science and Engineering Foundation(KOSEF) through the Advanced Information Technology Research Center(AITrc) and Brain Science Research Center.

References

[1] E. Catmull. J. Clark. Recursively generated B-spline surfaces on arbitrary topological surfaces. Computer-Aided Design, 10(6), 350-355, 1978.

[2] M. Lounsbery. T. Derose, J. Warren. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. Transactions on Graphics, 16(1), 34–73. 1997.

[3] M. Lounsbery. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. doctoral diss.. University of Washington, Seattle, WA, 1994.

[4] H. Hoppe. Progressive Meshes. Computer Graphics, 30, pages 99-108. 1996.

[5] J. Popovic, H. Hoppe. Progressive simplicial complexes. Computer Graphics, 31, pages 217-224. 1997.

[6] M. Eck, T. Derose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. Computer Graphics, 29, pages 173–182. 1995.

[7] A. Certain, J. Popovic, T. Derose, T. Duchamp, D. Salesin, W. Stuetzle. Interactive Multiresolution Surface Viewing. Computer Graphics, 30, pages 91–98. 1996.

[8] D. Zorin, P. Schroder, W. Sweldens. Interactive Multiresolution Mesh Editing. Computer Graphics, 31, pages 259–268. 1997.
[9] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. Computer Graphics, 32, pages 95-104. 1998.
[10] H. Hoppe. View-Dependent Refinement of Progressive Meshes. Computer Graphics, 31, pages 189–198. 1997.
[11] D. Luebke, C. Erikson. View-Dependent Simplification of Arbitrary Polygonal Environments. Computer Graphics, 31, pages 199-208. 1997.
[12] W. E. Lorensen, H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, 21(4), pages 163-170. 1987.
[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle. Surface Reconstruction from Unorganized Points. Computer Graphics, 26(2), pages 71-78. 1992.
[14] H. Lee, J. Lee, H. 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 Proceedings, pages 634 – 643. GyeongJu, ROK., 2002.
[15] R. Shekhar, E. Fayyad, R. Yagel, J. Cornhill. Octree-Based Decimation of Marching Cubes Surfaces, IEEE Visualization, pages 335-342. 1996.

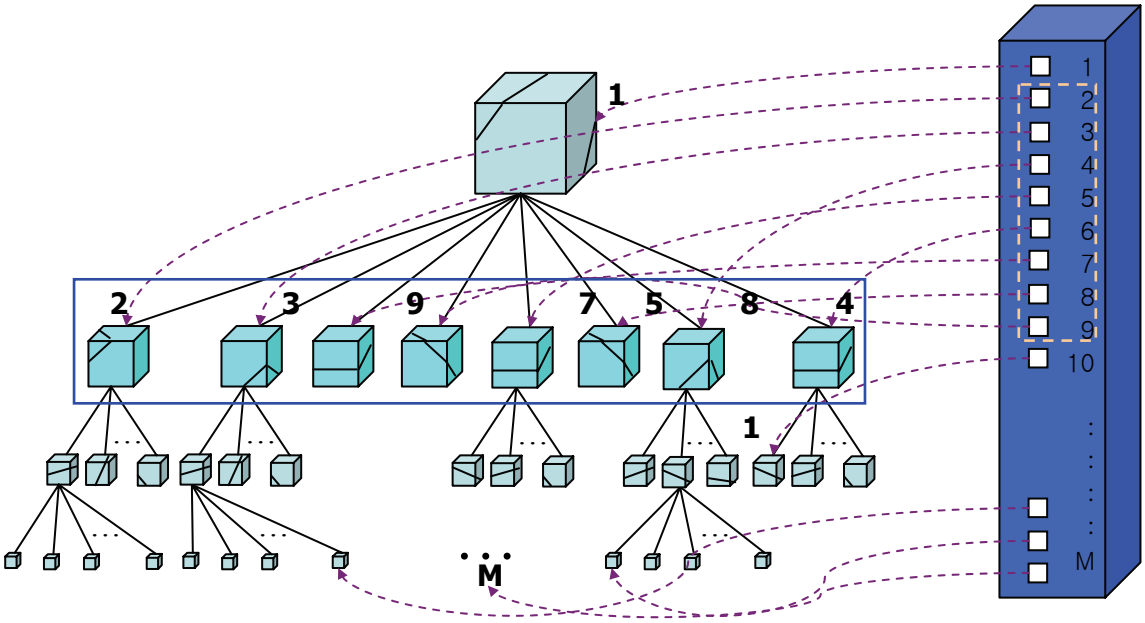

Figure 4 : Marching cube octree with LOD array

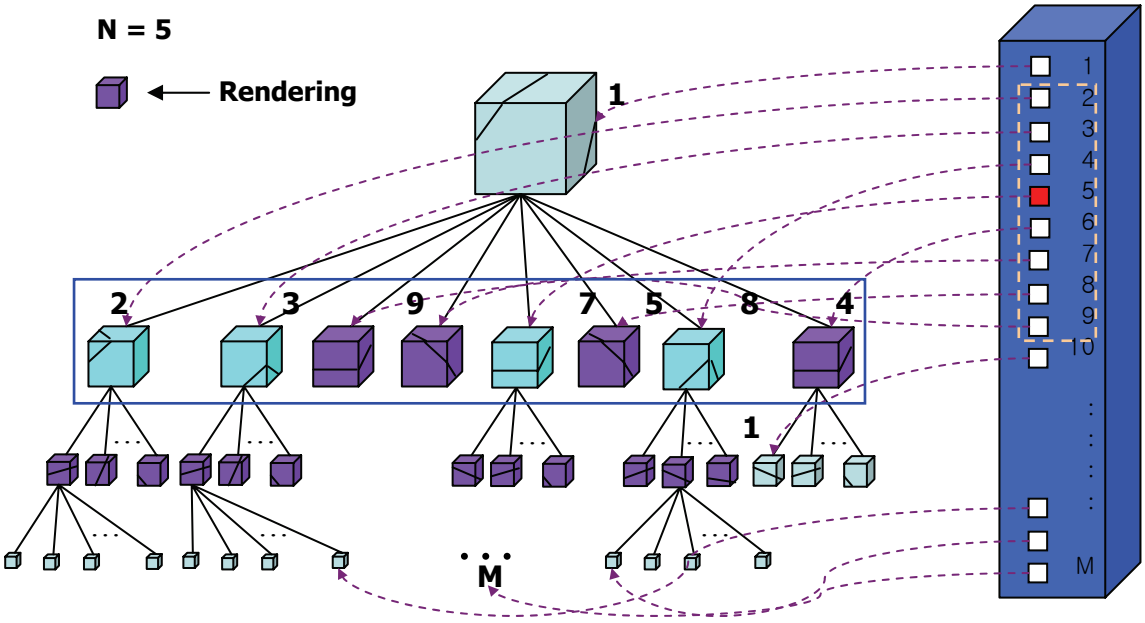

Figure 6 : Direct generation of the LOD mesh

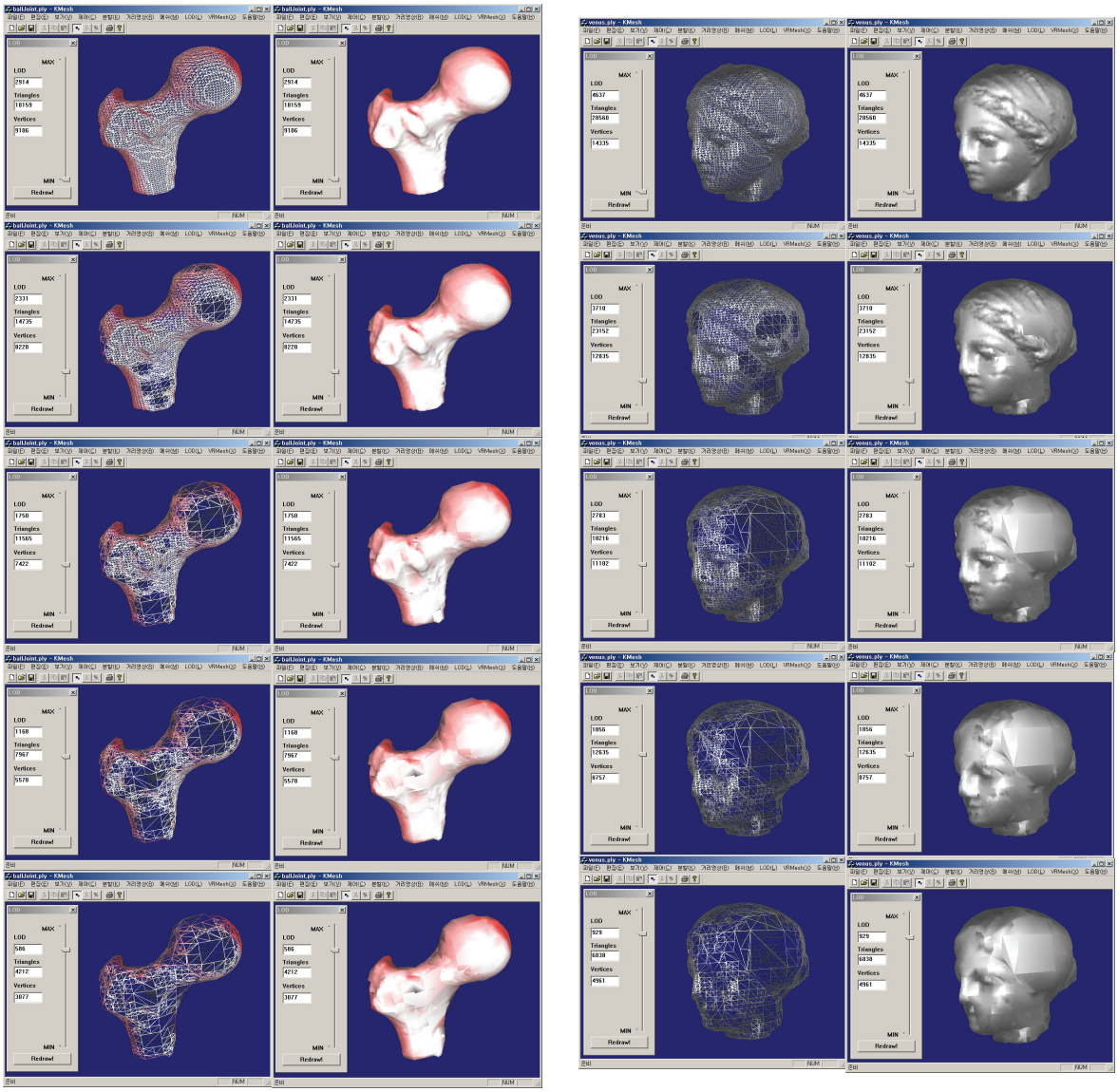

Figure 7 : Examples of ball joint and Venus (dataset courtesy of Cyberware)

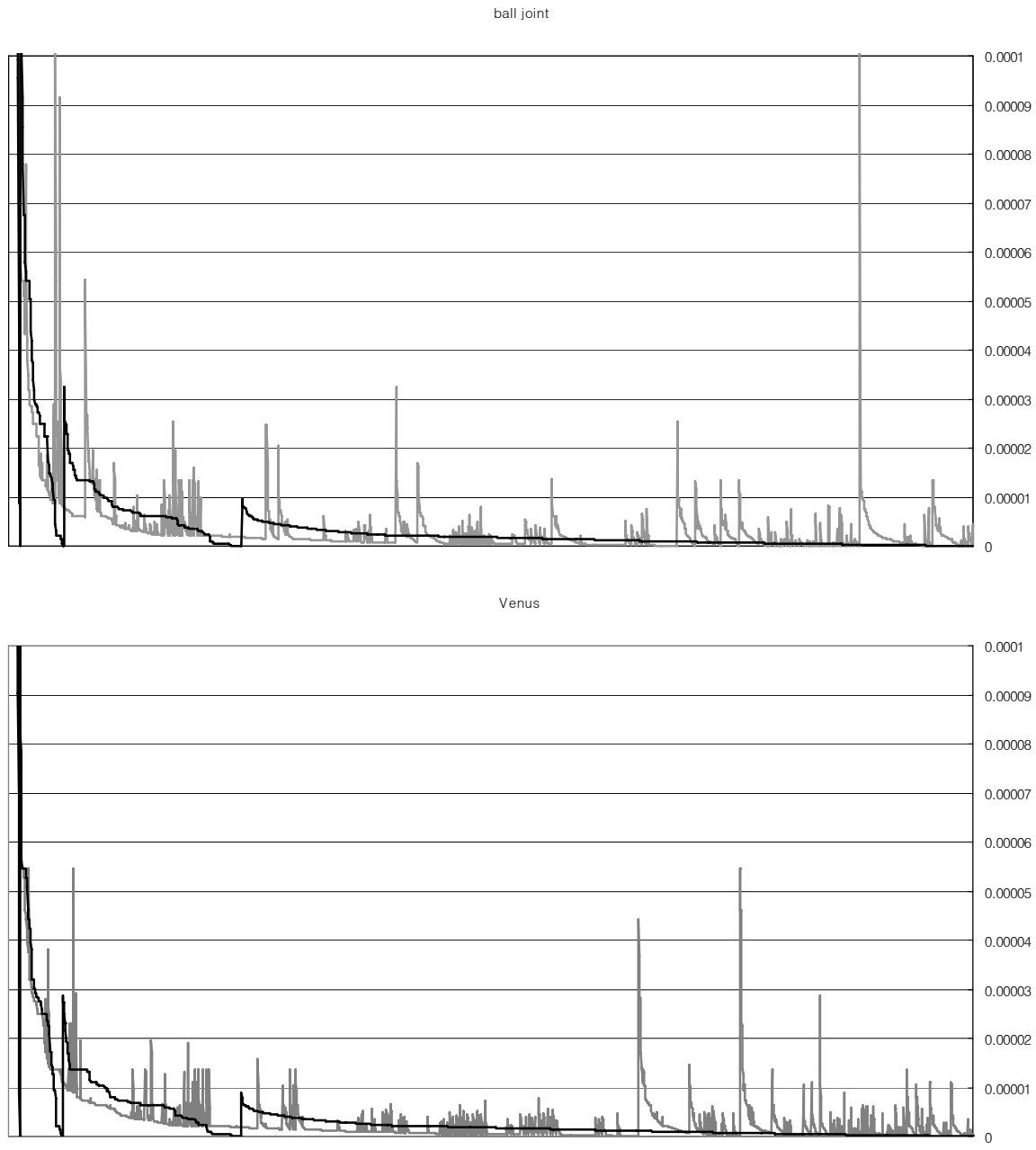

Figure 8 : LOD metric data (black lines are data of previous level-based algorithm)