Entropy-based Mesh Simplification

Article *in* Computer-Aided Design and Applications · August 2013

DOI: 10.3722/cadaps.2010.911-918

CITATION
1

READS
46

2 authors:

Lianping Xing
The Chinese University of Hong Kong
7 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Kin-Chuen Hui
The Chinese University of Hong Kong
85 PUBLICATIONS **561** CITATIONS
SEE PROFILE

Entropy-based Mesh Simplification

Lian Ping Xing and Kin Chuen Hui

The Chinese University of Hong Kong, lpxing@mae.cuhk.edu.hk
The Chinese University of Hong Kong, kchui@mae.cuhk.edu.hk

ABSTRACT

Complex mesh models usually affect the performance of algorithms for CAD applications. Therefore, it is often desirable to use simplified models and at the same time preserves the general features of the shape. In this paper, we present an entropy based surface mesh simplification algorithm which can rapidly and efficiently produce high quality approximations of triangle mesh models. The algorithm uses iterative contraction of edges to simplify models. It first exploits the geometric feature of model surface by estimating vertex curvature entropy and then iteratively contracts edges with relatively lower simplification cost. The simplification cost is measured by a combination of vertex curvature and curvature entropy of the two vertices compositing the edge. The goal is to simplify regions that are relatively flat while preserving salient features of the shape. Experiments show promising results of the algorithm.

Keywords: mesh simplification, entropy, vertex curvature, edge decimation.
DOI: 10.3722/cadaps.2010.911-918

1 INTRODUCTION

In CAD and computer graphics applications, highly detailed three-dimensional (3-D) mesh models are usually required to provide a convincing level of realism. However, these highly detailed models introduce severe difficulties in transmission, storage, processing and rendering. And the full complexity of models is not always required in many circumstances. A simplified version of the complex mesh model is thus preferred which can be generated automatically and at the same time preserve characteristic features of the model.

In general, the input of a mesh simplification algorithm is a surface mesh consisting of triangles only. After a series of processing, the output is a simplified version of the original model consisting of less faces and vertices which satisfy some given target criterion. In this paper, we present an entropy based vertex decimation mesh simplification algorithm. The algorithm first exploits the geometric features of the triangle mesh surface by estimating vertex curvature entropy. The process then iteratively contracts edges with relatively lower simplification cost which is measured by a combination of vertex curvature and curvature entropy of the two vertices compositing the edge. The primary advantages of this algorithm is that it can automatically generate simplified versions of complex mesh models efficiently with high quality of approximation.

The rest of this paper is organized as follows. Section 2 reviews related work in this area and discusses some mesh simplification techniques. In section 3, we briefly introduce entropy theory. Section 4 presents the details of our algorithm. Results are shown in section 5. Finally, we conclude the paper in section 6.

2 RELATED WORK

There are a large number of different techniques on triangle mesh simplification. We classify these studies mainly as vertex decimation, vertex clustering and iterative edge contraction.

Vertex Decimation. The classical algorithm is described in [11], where the method iteratively selects a vertex for deletion, removes all adjacent edges and retriangulates the resulting hole. The technique is also used in [2] where any modification made to the model is restricted within the simplification envelopes. [12] presents a more complex, but essentially a vertex decimation method. While all of these methods provide reasonable efficiency and quality, they put more emphasis on topology of mesh models which somewhat limits their performances.

Vertex Clustering. In vertex clustering process described in [10], a bounding box divided into an equally spaced grid is placed around the original model. The vertices of the model lie within the same cell are clustered together into a single vertex, and the corresponding model faces are updated accordingly. The technique can generate simple versions of the original mesh models fast and gets rid of topology limitation. However, the quality of the output model in this algorithm depends on the organization of the grids composing the bounding box which makes the control of desirable output difficult. Although [9] improves the above algorithm, it still can not give promising results as desired.

Iterative Edge Contraction. So far, several algorithms using iterative edge contraction to simplify triangle mesh models have been proposed. The performance of such techniques relies on two aspects: one is how to choose an edge for contraction, the other is where to locate the new position of the contracted edge. [5] presents an approach to minimize an energy function for choosing an edge. [6] introduces the progressive mesh scheme based on [4] and other related work. The algorithm described in [8] selects the position of the new vertex and minimize the per-triangle change in volume of the tetrahedra swept out by those triangles that are moved. One of the notable work in this area is presented in [3] which uses a QEM (quadric error metric) based method to simplify triangle mesh models and to contract arbitrary vertex pairs in addition to edges. This method can rapidly and efficiently produce high quality approximations of triangle mesh models without maintaining the topology of the original meshes. Recently, [1] presents a fuzzy based approach to mesh simplification and [13] uses a modified QEM based approach to determine contracted edges. [7] describes a greedy algorithm to select initial vertex contraction pairs and then iteratively removes them.

Although almost all the algorithms described above can generate simplified meshes fast and efficiently, sometimes they do not consider the essential feature of triangle mesh models and therefore may not produce optimal results. In this paper, we fully exploit the feature of mesh surfaces by finding their vertex curvatures and appropriately define the edge decimation cost for edge contraction. Experimental results show that our method performs better than classic QEM based approach in [3].

3 INTRODUCTION TO ENTROPY

In 1948, Shannon C. E. introduced entropy into information theory and defined entropy as a measure of the uncertainty associated with a random variable. Considering a random event experiment **A**, assume it has approximately **n** types of results: $a_1, a_2, \ldots, a_n$; the probabilities of every result are: $p_1, p_2, \ldots, p_n$. They satisfy: $0 \le p_i \le 1\ (i = 1, 2, \ldots, n)$ and $\sum_{i=1}^{n} p_i = 1$. The main property related to random events is: the appearance or disappearance of every result is uncertain which is inherent in probability experiments. In order to measure the uncertainty of experiment **A**, Shannon defined entropy.

Entropy is one of the most important definitions in information theory. Assume $X = \{x_1, \ldots, x_n\}$. is a set of discrete random variables, its entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} p_i log p_i \quad (1)$$

where $p_i = pr\{X = x_i\}, i \in \{1, \dots, n\}$ denotes the probability that the random variable X takes on value x_i, the base of the logarithm is 2, and $0log0 = 0$. The term $-logp_i$ represents the self-information of random variable x_i, and the entropy measures the average amount of information stored in the whole random variables. The entropy indicates the global statistical property of the information source and is the measurement of medial uncertainty of the entirety. There is only one entropy related to a certain source and the entropy may vary as its statistical properties change.

The important properties of entropy are:

1. $0 \leq H(X) \leq logn$

   $H = 0 \Leftrightarrow p_j = 1 \; and \; p_i = 0, i \in \{1, \dots, j-1, j+1, \dots, n\}$

   $H(X) = logn$ when $p_i = \frac{1}{n}, i \in \{1, \dots, n\}$

2. The entropy increases when the probabilities of all variables are prone to be equal.

3. $H(p_1, p_2, \dots, p_n) = H(p_1 + p_2, p_3, \dots, p_n) + (p_1 + p_2)H(\frac{p_1}{p_1+p_2}, \frac{p_2}{p_1+p_2})$

4 ENTROPY BASED APPROACH TO MESH SIMPLIFICATION

In the edge contraction process, it is important to select an appropriate edge for decimation and at the same time preserve the general features of the original shape. The vertex curvature of the shape surface maintains the elementary characteristics of the mesh model. The larger the vertex curvature, the more sharp the feature is in the vertex area. However, a purely curvature-based metric may not be a good metric of perceptual importance [14]. For example, a flat region in the middle of densely repeated high-curvature bumps will be perceived to be important. With a purely curvature-based metric, the flat region may be simplified at a faster rate resulting in an inaccurately simplified shape. Garland and Heckbert [3] demonstrated that well defined error of the selected vertex with respect to a set of planes associated with it is a good method. As entropy is a good tool to measure the characteristics of a set, we adopted entropy in our algorithm. We use it to find the feature of the surface region associated with every vertex, and the larger the value is, the more flat is the region the vertex is located in. This region is hence a candidate for simplification. Intuitively, the concept of vertex curvature entropy represents a measure of regional feature that are different from their surrounding context and is an evaluation of surface curvatures surrounding a vertex. The smaller its value is, the more feature the region contains.

In the proposed algorithm, we first set the contraction error at each vertex as the vertex curvature entropy. Then the contraction cost of every edge is defined as a combination of the vertex curvature and curvature entropy of the two vertices compositing the edge.

4.1 Approximating Vertex Curvature

The first step in the proposed algorithm involves computing surface curvatures. There are a number of approaches that generalize differential-geometry-based definition of curvatures to discrete meshes [4, 15, 16]. In this paper, the method in [16] is adopted.

In order to have a continuous tensor field over the whole surface, a curvature tensor can be defined at each point along an edge, and then this line density of tensors is integrated over an arbitrary region by summing the different contributions from region B, leading to the simple expression:

$$T(v) = \frac{1}{|B|}\sum_{edges\, e} \beta(e)|e \cap B|\vec{e}\vec{e}^T \quad (2)$$

Where v is an arbitrary vertex on the mesh, $|B|$ is the surface area around v over which the tensor is estimated, $\beta(e)$ is the signed angle between the normals to the two oriented triangles incident to edge e, $|e \cap B$ is the length of $e \cap B$ which is always between 0 and $|e|$, and $\vec{e}$ is a unit vector in the

same direction as e. The principle curvatures of the vertex are given by the eigenvalues of Eq(2) except for the minimum magnitude of the eigenvalue. Finally, the mean curvature of the vertex is given as:

$$K_M = \frac{1}{2}(\mathrm{K_{max}} + \mathrm{K_{min}}) \quad (3)$$

4.2 Edge Decimation Cost

As we have described above, the performance of the edge decimation techniques depends on two factors. The first one is how to choose an edge for contraction. This can also be interpreted as the magnitude of the edge decimation cost. The second factor is where to locate the new position of the contracted edge to make the operation error small enough.

For the edge decimation operation $(v_1,\ v_2) \rightarrow v$, the new position of vertex v can be anywhere on the edge or simply the midpoint or v_2. For simplicity, we adopted the contraction $(v_1,\ v_2) \rightarrow v_2$. The effect of the location of v will be analysis in our future work.

To estimate the edge decimation cost, we use the following equations:

1) First, we calculate the vertex curvature entropy for every vertex. For every vertex v_i, we define a set $X = \{v_{i_0},\ v_{i_1} \ldots, v_{i_n}\}$ (where $v_{i_0} = v_i$) which is composited of the vertex in consideration and its associated vertices. The vertex curvature of this set can be $K = \{K_{i_0},\ K_{i_1} \ldots, K_{i_n}\}$. Then the curvature entropy is defined as:

$$\mathrm{H_i} = -\sum_{\mathrm{j=0}}^{\mathrm{n+1}} p_j log p_j \quad (4)$$

Where $p_j = \frac{k_i}{\sum_{i=0}^{n+1} k_i}$ and its normalized form is:

$$C_i = \frac{H_i}{H_{i,max}} = \frac{-\sum_{\mathrm{j=0}}^{\mathrm{n+1}} p_j log p_j}{\log(n+1)} \quad (5)$$

2) Define the decimation error of this vertex as: $C_i = 1 - C_i$.

3) Finally, for contraction $(v_i,\ v_j) \rightarrow v_j$, we construct the cost as:

$$cost_i = C_i \times (k_i + k_j)/2 \quad (6)$$

4.3 The Algorithm

The algorithm can be summarized as follows:

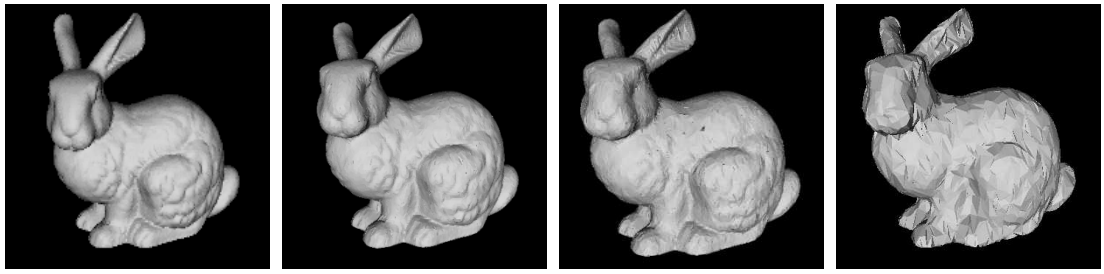
1) For every vertex v_i of the original mesh model, calculate its vertex curvature k_i.
2) Compute the vertex contraction error C_i.
3) Compute the edge contraction cost $cost_i$ and place all the contraction edges in a heap sorted by cost with the minimum cost edge at the top.
4) Contract the edge with the least cost from the heap, and update all affected information.
5) If the heap is not null, go to step (4), otherwise, go to step (6).
6) Stop.

5 RESULTS

We have implemented the proposed algorithm using C++ language. All the experiment results are obtained by running the algorithm on a 1.4GHz machine with 512 MB RAM. We compare our results with those generated using the classical QEM-based method [3] and the Saliency-based method [14]. In Saliency-based method, we use different scales of 2ϵ, 3ϵ and 4ϵ to calculate the mesh saliency where ϵ is set to be 0.3 times the length of the diagonal of the bounding box of the model and also set λ to be 1. The outcome shows that our technique efficiently produces simplified mesh model.

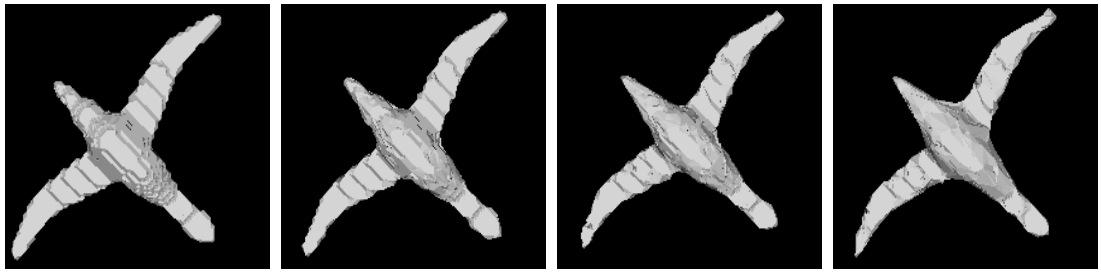
Fig. 1, Fig. 2 and Fig. 3 demonstrate the results of our algorithm. In Fig. 1, the bunny model is simplified to 30%, 60% and 90% of the original bunny mesh model, the simplified shapes still look

similar to the original one. The same results are obtained with the simplification of the bird and dinosaur model as shown in Fig. 2 and Fig. 3. Besides, it can also be observed that the important perceptual features of the models are well preserved under low-level simplification.



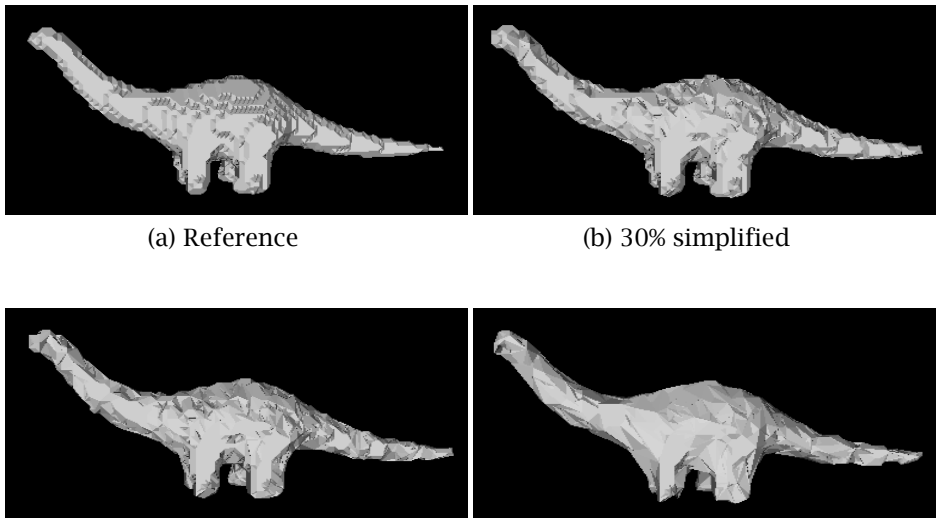
(a) Reference (b) 30% simplified (c) 60% simplified (d) 90% simplified

Fig. 1: Simplification of the bunny model (69451 triangles).



(a) Reference (b) 30% simplified (c) 60% simplified (d) 80% simplified

Fig. 2: Simplification of the bird model (87197 triangles).

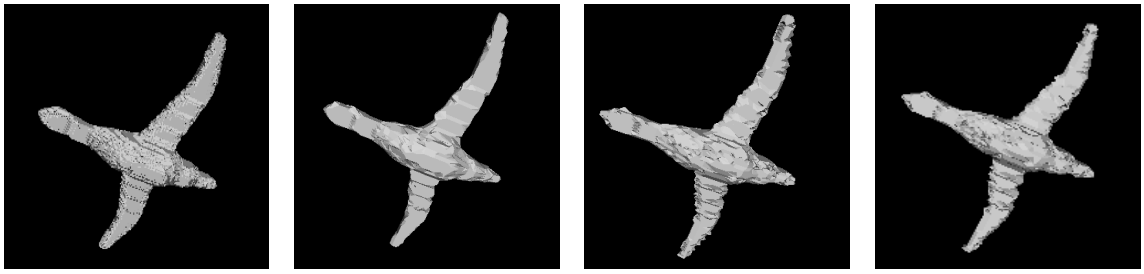


(a) Reference (b) 30% simplified

(c) 50% simplified (d) 80% simplified

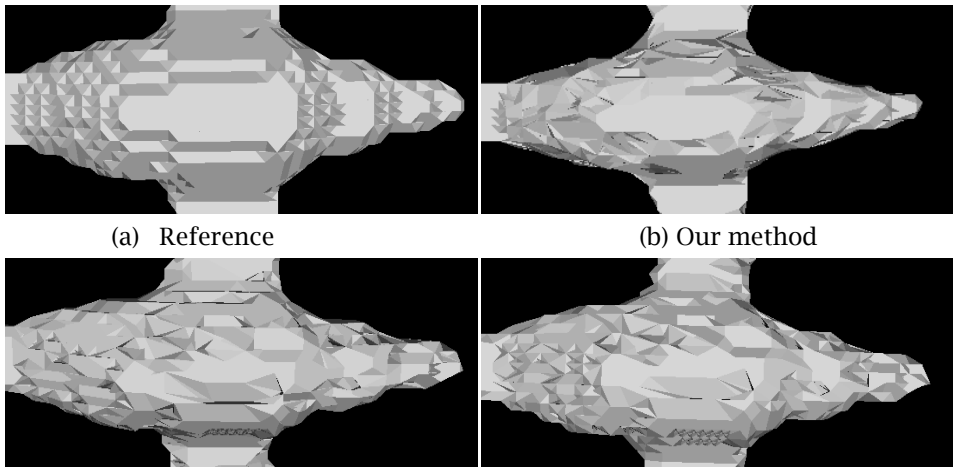
Fig. 3: Simplification of the dinosaur model (111241 triangles).

A comparison of the proposed entropy-based method with the classical QEM-based approach and the Saliency-based method is shown in Fig. 4, Fig. 5, Fig. 6 and Fig. 7. It can be observed that under the same simplification level, all of the three techniques can produce good results. However, the performance of our method is better than the other two methods. We can see from the pictures that our algorithm produces meshes that are smoother, e.g. the abdomen of the bird with more details shown in Fig. 5, the neck area, backside with more details also shown in Fig. 7 and tail of the dinosaur. In the bird model, the perceptually important features are better preserved than the other two, e.g. the wing of the bird.



(a) Reference (b) Our method (c) QEM-based method (d) Saliency-based method

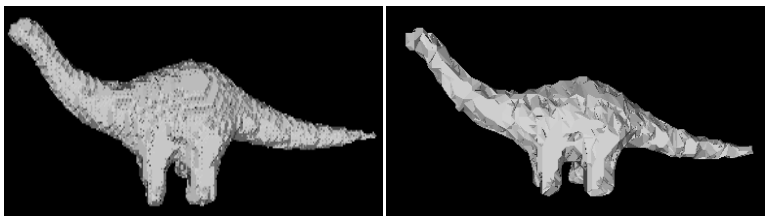
Fig. 4: Comparison of our algorithm with the QEM-based and Saliency-based methods on the bird model (45% simplified).



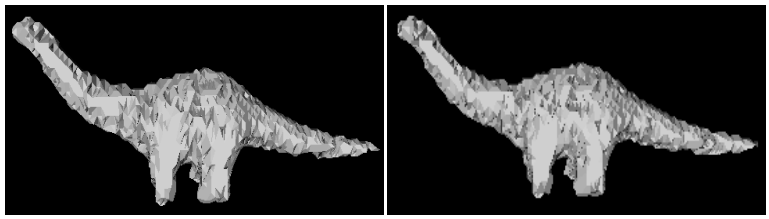
(a) Reference (b) Our method

(c) QEM-based method (d) Saliency-based method

Fig. 5: Comparison of our algorithm with the QEM-based and Saliency-based methods on the bird abdomen (45% simplified).

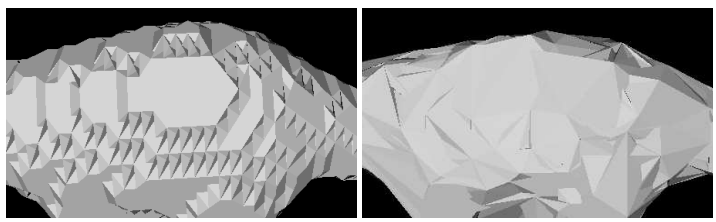


(a) Reference (b) Our method



(c) QEM-based method (d) Saliency-based method

Fig. 6: Comparison of our algorithm with the QEM-based and Saliency-based methods on the dinosaur model (50% simplified).



(a) Reference (b) Our method

(c) QEM-based method (d) Saliency-based method

Fig. 7: Comparison of our algorithm with the QEM-based and Saliency-based methods on the dinosaur back (85% simplified).

Table 1 shows a comparison of the running time consumed by our method with the QEM-based approach for simplifying different mesh models at different simplification level. Both of the two algorithms are fast and the proposed algorithm performs a little better than the QEM method.

Mesh Model	Primitives	Simplified (%)	Time	
			Our Method (s)	QEM Method (s)
Bunny	69451	30%	2.04	2.08
		60%	5.56	6.05
Bird	87197	40%	5.19	6.33
Dinosaur	111241	50%	10.23	10.35

Tab. 1: Comparison of the running time of our algorithm with the QEM-based method.

6 CONCLUSION

In this paper, we present an entropy-based mesh simplification algorithm. We exploit the feature of model surface by estimating vertex curvature entropy and then iteratively contract edges with relatively lower simplification cost. The simplification cost is measured by a combination of vertex curvature and curvature entropy of the two vertices compositing an edge. Experimental results show that our method can generate simplified models efficiently.

ACKNOWLEDGEMENT

This work is partially supported by a grant from the Innovation and Technology Commission of the Government of the Hong Kong Special Administrative Region (No. ITS/057/08) and a grant from the University Grant Council of the Hong Kong Special Administrative Region (No. 412508).

REFERENCES

[1] Chang, C.; Yang, S. K.; Duan, D. Z.; Lin, M. F.: A Fuzzy Based Approach to Mesh Simplification, Journal of Information Science and Engineering, 18, 2002, 459-466.
[2] Cohen, J.; Varshney, A.; Manocha, D.: Simplification Envelopes, Computer Graphics (SIGGRAPH' 96 Proceedings), 1996, 119-128.
[3] Garland, M.; Heckbert, P.: Surface Simplification Using Quadric Error Metrics, Computer Graphics (SIGGRAPH' 97 Proceedings), 1997, 209-216.
[4] Hamann, B.: Curvature Approximation for Triangulated Surfaces, Computing Supple, 13(8), 1993, 139-153.
[5] Hoppe, H.; DeRose, T.; Duchamp, T.; McDonald, J; Stuetzle, W.: Mesh Optimization, In SIGGRAPH' 93 Proceedings, 1993, 19-26.
[6] Hoppe, H.: Progressive Meshes, In SIGGRAPH' 96 Proceedings, 1996, 99-108.
[7] Hussain, S.; Grahn, H.; Persson, J.: Feature-preserving Mesh Simplification: A Vertex Cover Approach, International Conferences on Computer Graphics and Visualization, 2008, 270-275.
[8] Lindstrom, P.; Turk, G.: Fast and Memory Efficient Polygonal Simplification, Proceedings of IEEE Visualization, 1998, 279-286.
[9] Luebke, D.; Erikson, C.: View-dependent Simplification of Arbitrary Polygonal Environments, In SIGGRAPH'97 Proceedings, 1997.
[10] Rossignac, J.; Borrel, P.: Multi-resolution 3D Approximations for Rendering Complex Scenes, Modeling in Computer Graphics: Methods and Applications, 1993,455-465.
[11] Schroeder, W. J.; Zarge, J. A.; Lorensen, W. E.: Decimation of Triangle Meshes, Computer Graphics (SIGGRAPH' 92 Proceedings), 26(2), 1992, 65-70.
[12] Soucy, M.; Laurendeau, D.: Multiresolution Surface Modeling based on Hierarchical Triangulation, Computer Vision and Image Understanding, 63(1), 1996, 1-14.
[13] Wu, Y.; He, Y.; Cai, H.: QEM-based Mesh Simplification with Global Geometry Features Preserved, Computer Graphics (SIGGRAPH' 04 Proceedings), 2004, 50-57.
[14] Lee, C. H..; Varshney, A.; Jacobs, D.: Mesh Saliency, ACM Trans. Graph. 24, 3, 659-666.
[15] Meyer, M.; Desbrun, M.; Schoroder, P.; Barr, A. H.: Discrete Differential- geometry Operators for Triangulated 2-manifolds, In Visualization and Mathematics Ш (Proceedings of VisMath 2002), Springer Verlag, Berlin, Germany, 35-54.
[16] Alliez, P.; Cohen, S.; Devillers, O.; Levy, B.; Desbrun, M.: Anisotropic Polygonal Remeshing, Proceedings of ACM Siggraph 2003, 3(22), 485-493.