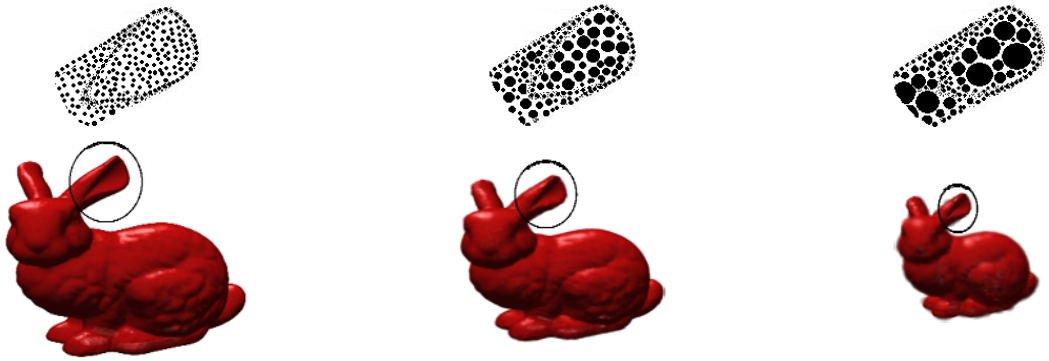
芸術科学会論文誌　Vol. 9　No. 3　pp.140-153

Simplification of Point Set Surfaces using Bilateral Filter and Multi-Sized Splats

Batchimeg SOSORBARAM　　Tadahiro FUJIMOTO　　Norishige CHIBA
Iwate University
E-mail: {chimeg@cg., fujimoto@, nchiba@}cis.iwate-u.ac.jp

バイラテラルフィルタと多重サイズスプラットによるポイント表現された曲面の簡易化手法

ソソラバラム　バトゥチメグ　藤本忠博　千葉則茂
岩手大学



(a) Original model　(b) Simplified model after one iteration　(c) Simplified model after several iterations

Figure 1. Point-rendering results of multiresolution point set surfaces

Abstract

We have developed a new algorithm that addresses the two major issues that are critical for the use of point-based rendering in real-world applications: rendering performance and rendering quality. The proposed algorithm improves rendering performance by reducing the number of points to be rendered. We generated points with different sizes in order to improve the quality of the point-splatting-type rendering. In this paper, we propose a feature-preserved simplification algorithm for point-sampled surfaces that generates models with different levels of details. The proposed method automatically balances the sampling density and point sizes. Our algorithm iteratively reduces the number of points using a bilateral filtering algorithm. We validate our method on the basis of the rendering results of different models with different resolutions.

Keywords: Point-based graphics, Level of details

概要

本論文では，ポイントベースレンダリングにとって重要であるレンダリングのパフォーマンスと品質を改善する新しいアルゴリズムを提案する．提案する手法では，ポイントの数を減らすことによりレンダリングパフォーマンス，また，異なったサイズのポイントを用いることによりレンダリングの品質も改善することが可能である．具体的には，バイラテラルフィルタを用いて，特徴点を保持しながらポイントの数を減らす．この手法により，ポイントの大きさとポイントの分布が自動的にバランスをとるためレンダリングの品質が改善される．また，レベル・オブ・ディテールによる異なる詳細度レベルに対応するモデルの生成も可能である。最後に，いくつかの実験例により本手法の有効性を示す．

キーワード：ポイントベースのグラフィクス，レベル・オブ・ディテール

1. Introduction

Three-dimensional (3D) points are the simplest and most fundamental geometry-defining primitives. They are a popular computational tool in particle simulation and in the 3D scanning industry. Points are popular in procedural modeling because the lack of connectivity makes it easier to deal with dynamically changing objects. Points have long been considered to be modeling tools; however, point-based rendering (PBR) has received increased attention recently because of the widespread use of 3D scanning technology and advanced physical modeling. PBR is an attractive option because of its conceptual simplicity and generality. One apparent drawback of PBR, in comparison with polygonal representations, is that in order to sketch a simple 3D shape, we are required to use millions of sample points rather than a few polygonal faces. Therefore, it can be concluded that the complexity of PBR is independent of the shape simplicity. As a consequence, studies on PBR primarily aim at efficient and flexible handling of a large 3D point cloud. One of the key challenges in point rendering is the reduction of the complexity of such large data sets. In this paper, we propose a feature-preserved simplification algorithm for point-sampled surfaces that generates models with different levels of details (LODs). This research is an extension of our previous work [Sosorbaram *et al*., 2009]. We have added a more detailed description and provided more result images. Multiresolution point set surfaces are generated by carrying out the following steps: First, the algorithm finds the *k*-nearest neighbors for a selected point. Second, the variations in normal vectors within the *k*-nearest neighborhood points are summarized. Third, the points are decimated using a bilateral filtering operator, and finally, the size of a new point is calculated.

Our contributions can be summarized as follows:

1) We used a hierarchical volumetric partitioning method in order to accelerate the *k*-nearest neighbor search process. Two or three volume grids were used for precise partitioning. One volume grid was used for the approximate partitioning, and the other volume grids were used for fine grade partitioning. In order to partition the 3D space into ($N \times N \times N$) partitioning, we used two volume grids with ($\sqrt{N} \times \sqrt{N} \times \sqrt{N}$) voxels. This hierarchical volumetric partitioning enabled us to reduce memory consumption while improving the partitioning resolution. Searching for the neighborhood points in the nearest voxels significantly improved the performance of the *k*-nearest neighborhood search algorithm. The advantages of our hierarchical volumetric partitioning algorithm include the following:
   - **Performance.** Our algorithm uses a hierarchical hashing technique. Previous space partitioning methods mainly use binary space partitioning trees (BSP trees), bounding volume hierarchies (BVHs), and octree structures. BVHs, BSP trees, and octree structures all use some sort of tree as their basic data structure. Hash tables are faster than tree structures in accessing data. Searching a hash table is easy and extremely fast: a typical search carried out using a hash function takes $O(1)$ time.
   - **Memory efficiency.** Our algorithm uses a relatively small amount of extra memory for high-resolution partitioning. For example: in order to partition the 3D space into $10^6 \times 10^6 \times 10^6$ voxels, the algorithm uses two hash tables with $10^3 \times 10^3 \times 10^3$ voxels or three hash tables with $10^2 \times 10^2 \times 10^2$ voxels.
   - **Adaptation for non-uniformly distributed point sets**. At each hierarchy level, the algorithm skips the empty voxels in order to adapt to the point distribution.

2) The feature-preserved point set simplification operator uses the concept of bilateral filtering proposed by Tomasi and Manduchi [1998]. Our filtering operator takes the distance value, the variation coefficient, and the threshold value as the arguments in order to determine the point elimination criteria. If a point has a variation lower than the threshold value and if it is located close to the selected point, it is removed from the point set. The following reasons have inspired us to use bilateral filtering for the simplification of point set surfaces:
   - **Feature-preserved simplification.** Recently, the bilateral filtering algorithm has been successfully used in many feature-preserved operations in computer graphics, e.g., image processing and high dynamic range (HDR) techniques. In the feature-preserved simplification of point sets, we have to consider two important characteristics of local surfaces the geometrical closeness of neighborhood points and the surface curvature near the selected point. The bilateral filtering algorithm was chosen as a well-suited filtering operator to combine these two important features.
   - **Conceptual simplicity.** Bilateral filtering produces a doubly weighted local average. The calculation of two Gaussian weighted functions is simple and fast. Several very fast versions of bilateral operators require $O(1)$ time to run.
   - **Suitability for point set processing.** The proposed algorithm operates directly on point sets. Most other point surface simplification algorithms reconstruct the local surface (MLS and progressive mesh) or graphs (Voronoi

diagram and point repulsion) to evaluate the simplification criteria.
- **Iterative behavior.** It can iteratively produce the multi-resolution models in the form of levels of details (LODs).

3) The generation of points with different sizes improves the quality of rendering while reducing the number of points. After removing the points, the algorithm calculates the position of a new point and calculates its size. Its position is calculated by locating the center of the removed points. The size of a new point is computed by determining the longest distance between the position of the new point and the positions of the removed points. Our contributions in generating the point model with multi-sized splats are as follows:

   - We describe the algorithm to define the position and the size of new points. Although some research works demonstrate the result images with multi-sized points [Pauly *et al*., 2002], they do not provide a detailed algorithm for defining the position and the size of new points.
   - Our algorithm generates not only the multi-resolution LODs but also the multi-sized points within the LOD models.

Our method can be used in various applications such as LOD-based rendering and feature extraction in point-based models. Three resultant images are shown in Figure 1. These demonstrate that our algorithm is useful for the visualization of point set surfaces with different LODs.

2. Related Works

The proposed algorithm builds on a long sequence of earlier studies, which we briefly review here.

Point-Based Rendering: Points were first considered to be rendering primitives in the work of Levoy and Whitted [1985]; subsequently, they were rediscovered by Grossman and Dally [1998] and then improved with the introduction of surfels in the study by Pfister *et al*. [2000]. Recently, several researchers have introduced high-quality techniques using point splatting, differential points, and hardware acceleration. [Botsch *et al*., 2002; Botsch and Kobbelt, 2003; Kalaiah and Varshney, 2001].

Level of Details (LOD): There have been a number of approaches to speed up the rendering of complex models. One approach uses the LOD method. An automatic method for reducing the geometric complexity of surfaces by triangle decimation was first developed by Hoppe [1996]. The cost of inserting and deleting points in point-based methods is less than that in the case of polygon-based methods because of the absence of connectivity information in the point models. Therefore, LOD methods have been successfully used in PBR. Point rendering systems such as QSplat [Rusinkiewicz and Levoy, 2000] and Surfel [Pfister *et al*., 2000] have introduced a hierarchical structure like LOD. Further, research using LOD techniques in point rendering has resulted in efficient LOD representations and has considered issues such as the combination of point and triangle primitives in an LOD-based rendering approach [Cohen *et al*., 2001; Chen and Nguyen, 2001; Dey and Hudson, 2001]. The challenge in the generation of an efficient LOD representation lies in the efficient processing of large point sample data sets [Boubekeur, 2005]. Our algorithm differs from these algorithms in that it generates not only the multiresolution LODs but also the multi-sized points within the LOD models.

Point Surface Simplification: The works that relate the most to our methods are point simplification methods. The simplification techniques used in some of the most significant related works are summarized in Table 1.

The proposed algorithm differs from the algorithms listed in Table 1 in the following ways:
- We use a bilateral filtering algorithm to evaluate the point decimation criteria. This results in the advantages of bilateral filtering mentioned earlier in the introduction section.
- The proposed neighborhood searching, bilateral filtering, point elimination, and new point generation algorithms are straightforward and easy to implement. Many of the related works deal with additional processing to simplify the model such as mesh reconstruction [Alexa *et al*., 2001, Hoppe 1996, Rossignac and Borrel 1993, Jianhua Wu et al., 2005, Fleishman et al., 2003], solving a system of linear equations [Turk 1992], and the construction of additional structures [Moenning and Dodgson 2003].
- In the proposed algorithm, the surface variation is calculated by evaluating the weighted differences of neighboring normal vectors, which is a more intuitive approach than methods based on principal component analysis [Pauly *et al*., 2002, Jianhua Wu et al., 2005].
- A direct comparison of performance and quality of results with previously published works is difficult because of the uncommon computational environments, differences among the point models, and distinctive rendering tools used for producing the resulting images. We provide different rendered images to evaluate the basic characteristics of our simplification algorithm. Our algorithm works in linear time. The performance of each step is shown in Table 2. The graph of the performance of other algorithms shows [Pauly *et al*., 2002] logarithmic or quadratic time. On the basis of the available

information, we can conclude that our algorithm is faster than the other algorithms.

Table 1. Works related to point surface simplification

Related Work	Simplification Method
Efficient simplification of point-sampled surfaces. [Pauly *et al*., 2002]	Conversion of various mesh simplification techniques into point simplification
Point set surfaces. [Alexa *et al*., 2001]	Error metric for moving least square (MLS)-based local surfaces
Point cloud representation [Linsen 2001]	Entropy evaluation
Multiresolution 3D approximations for rendering complex scenes. [Rossignac and Borrel 1993]	Clustering by region growing
Progressive meshes. [Hoppe 1996]	Iterative edge collapsing
Re-tiling polygonal surfaces. [Turk 1992]	Particle relaxation
A new point cloud simplification algorithm. [Moenning and Dodgson 2003]	Voronoi diagram
Progressive splatting [Jianhua Wu et al., 2005]	Greedy algorithm PCA
Progressive point set surfaces. [Fleishman et al., 2003]	MLS based refinement operator

Nearest Neighbor Search: Point simplification algorithms are heavily dependent on the use of neighborhoods of points. A considerable amount of effort has gone into the development of an efficient nearest neighbor search method [Jagan *et al*., 2007]. Point neighborhoods are used for computing variations and point decimation and for removing noise. On the basis of the previously proposed hierarchical bucket sorting method [Zorig *et al*., 2007], we used a hierarchical volumetric partitioning method to accelerate the *k*-nearest neighbor search algorithm. The advantages of the hierarchical volumetric partitioning method have been described in the introduction part of this paper.

3. Overview of New Multiresolution Point Generation Algorithm

3.1 Formulation of Problems and Solution

Problem Formulation: Let *S* be a surface defined by a point cloud *P*. We assume that the discrete point samples *P* satisfy the necessary sampling criteria such as the Nyquist condition, and that they completely define the surface geometry and its features. Furthermore, it is assumed that each point is associated with attributes needed for point rendering such as the normal, size, and color. Our algorithm aims to generate several point sets with different resolutions. The requirements for a multiresolution surface generation algorithm are as follows:

- Newly generated approximations should resemble the original surface as closely as possible;
- The surface generation process should be controllable through configuration parameters in order to achieve the best results; and
- The performance of the algorithm should be fairly fast, since it will be necessary to apply the algorithm to interactive simulations involving dynamically changing point models.

Proposed Solution: With regard to the proposed algorithm, we focused on two important aspects of point elimination: the importance of a point for model description and the existence of the nearest points that can cover the removed point on the surface. We selected bilateral filtering as the most suitable solution for the feature-preserved simplification of point models when the normal of each point was predefined. In addition to the feature-preserved point elimination, we attempted to find a new algorithm for the *k*-neighborhood search algorithm. We applied a two-step hierarchical voxelization in order to accelerate the neighborhood search algorithm.

3.2 Algorithm Details

3.2.1 Steps of Proposed Algorithm

The multiresolution point set surfaces are generated by following the steps shown in Figure 2. Initially, the algorithm reads the original point set, analyzes the data set, determines the maximum and minimum sizes of points, and calculates the size of the modeled object in three dimensions. The initial values of the configuration parameters are assigned according to the information derived from the data analysis. Then, the algorithm iteratively generates simplified models. Each simplified model approximates the model of the previous iteration.

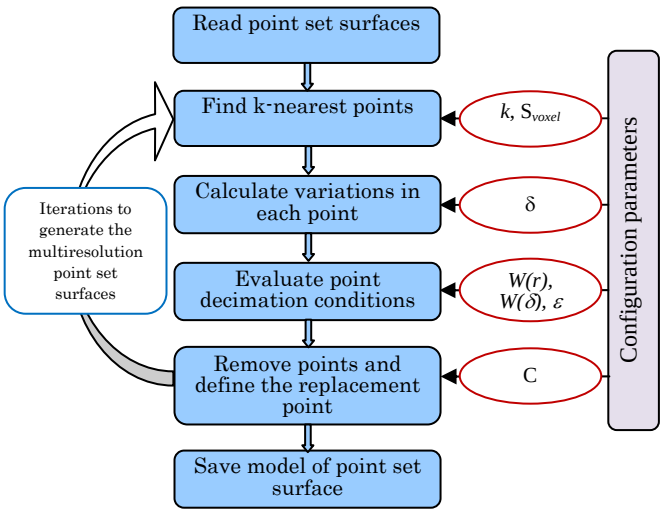


Figure 2. Algorithm steps.

Throughout an iteration, the algorithm reads all points of the existing simplified model several times and performs point decimation. The point decimation operator performs five tasks: it locates the nearest points, calculates the variations, evaluates the point decimation conditions, removes points, and generates a new replacement point. Each step in the iteration takes $O(n)$ time to perform.

3.2.2 Nearest Neighbor Searching Algorithm

The proposed algorithm makes considerable use of the neighborhoods of points. Upon analyzing different partitioning methods, we found that the two-step hierarchical voxelization was more effective than the other partitioning methods. In order to develop high-resolution voxelization in linear time, we used two volume grids with dimensions $N \times N \times N$. One of the volume grids was used for first-level approximate voxelization, and the other was used for accurate voxelization.

The size of the volume grid was defined by using the minimum size of the considered points, as shown in the following equation (Eq. 1):

$$N = \max\left(\sqrt{\frac{X_{\max} - X_{\min}}{S_{\min}}}, \sqrt{\frac{Y_{\max} - Y_{\min}}{S_{\min}}}, \sqrt{\frac{Z_{\max} - Z_{\min}}{S_{\min}}}\right) \quad (1)$$

where $\boldsymbol{X_{max}}$, $\boldsymbol{Y_{max}}$, $\boldsymbol{Z_{max}}$ and $\boldsymbol{X_{min}}$, $\boldsymbol{Y_{min}}$, $\boldsymbol{Z_{min}}$ are the maximum and minimum coordinates in the X, Y, and Z directions, respectively. $\boldsymbol{S}$ is the minimum size of the considered points; $\boldsymbol{N}$ is one dimension of the volume grid with dimensions $N \times N \times N$.

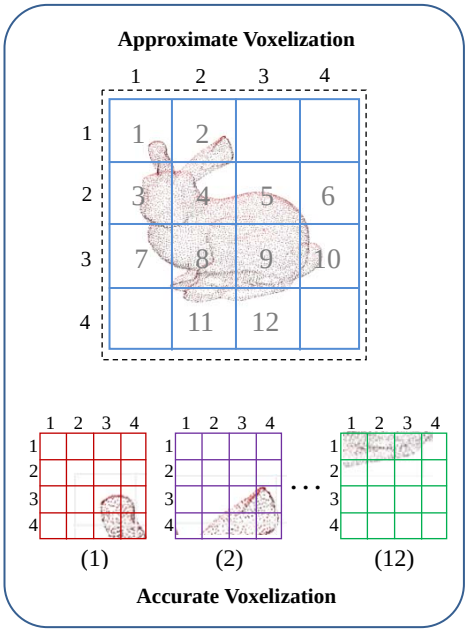


Figure 3. Outline of two step hierarchical voxelization

Figure 3 shows a high-level overview of our hierarchical voxelization procedure. First, the algorithm traverses all points and distributes them in the first-level approximation volume grid. The points are assigned to voxels by using a simple calculation that is based on the position of the points. Then, the non-empty voxels of the approximated bucket list are distributed into the accurate volume grids. A hash function for accurate voxelization works in a manner similar to the first-level voxelization. Only the range values of the voxels need to be changed.

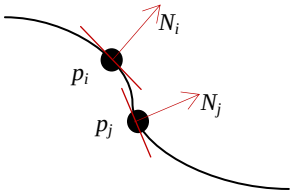


Figure 4. Normal vectors for selected point and nearest point

The main advantages of our voxelization algorithm are as follows:

- It works in linear time $O(n)$. The algorithm reads the entire point set twice to produce the required voxels in order to find the nearest neighborhood points. During accurate voxelization, the algorithm skips all empty voxels of the approximate volume grid.
- It generates high-quality voxelization using a small amount of memory. For example, in order to generate a large voxelization of dimension 10,000 × 10,000 × 10,000, we required two volume grids of dimensions 100 × 100 × 100; one volume grid was used for the first-level approximation, and the other volume grid was used for accurate voxelization.

3.2.3 Calculation of Variations

The variation in points is calculated on the basis of the differences in normal vectors. We define the variation of point (δ_i) as the average of the dot products of the selected point and the k-nearest neighborhood points; $\boldsymbol{\delta_i}$ is calculated on the basis of the following equation (Eq. 2):

$$\delta_i = \frac{\sum_{j=1}^{k} (N_i \cdot N_j)}{k} \quad (2)$$

where $\boldsymbol{k}$ is the number of neighborhood points and $\boldsymbol{N_i}$ and $\boldsymbol{N_j}$ are the normals of the selected point $\boldsymbol{i}$ and the neighborhood point $\boldsymbol{j}$, respectively.

3.2.4 Evaluation of Point Decimation Criteria

We determine the points that need to be removed by using the bilateral filtering operator. The evaluation function determines the elimination criteria of a selected point by taking into account the variation in and the distance of the neighborhood points. The point elimination criteria C_e^i are defined by the following equation (Eq. 3):

$$C_e^i = \begin{cases} 1; & W_i > \varepsilon \\ 0; & W_i \le \varepsilon \end{cases} \tag{3}$$

where W_i is the weight function of a selected point and ε is the threshold value. The threshold value must be $0 < \varepsilon < 1$. The selected point has a high elimination probability if the point elimination criteria are equal to 1. We calculate W_i as the bilateral function by using the following equation (Eq. 4):

$$W_i = \frac{\sum_{j=1}^{k} W_r(i,j) \cdot W_\delta(i,j)}{\sum_{j=1}^{k} W_r(i,j)} \tag{4}$$

where W_r and W_δ are the distance and variation based weight functions, respectively. They are calculated by using the following equations (Eqs. 5 and 6):

$$W_r = e^{-\alpha \cdot d_{ij}} \tag{5}$$

$$W_\delta(i,j) = \begin{cases} e^{-\beta \cdot |\delta_i - \delta_j|} & |\delta_i - \delta_j| \le \Psi \\ 0 & |\delta_i - \delta_j| > \Psi \end{cases} \tag{6}$$

where d_{ij} is the distance between the point i and the neighborhood point j; δ_i and δ_j are the variations in the selected point i and the neighborhood point j, respectively; β is the multiresolution-level-based coefficient; and Ψ is the threshold parameter for variation.

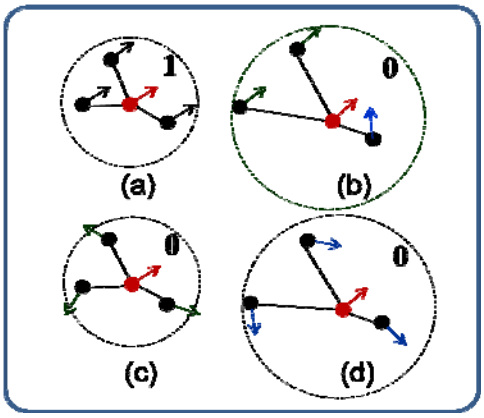


Figure 5. *Illustration of point elimination criteria*

Figure 5 illustrates the point elimination criteria. The point is removed when the neighbors are near and the normal vectors are in approximately the same direction, as shown in Figure 5(a). In other conditions such as when the neighbors are far (Figure 5(b) and 5(d)) or normal vectors are directed into different directions (Figure 5(c) and 5(d)), the point will not be removed.

3.2.5 Removal of Points and Generation of New Point

The generation of points with different sizes improves the quality of rendering, while reducing the number of points. After removing the points, the algorithm determines the position of a new point and computes its size. The position of new point p_{new} is determined by locating the center of the removed points (Eq. 7).

$$p_{new} = \left(\frac{\sum_{i=1}^{m} x_i}{m}; \frac{\sum_{i=1}^{m} y_i}{m}; \frac{\sum_{i=1}^{m} z_i}{m} \right) \tag{7}$$

Here, x_i, y_i, z_i are the coordinates of the removed points and m is number of the removed points. The size of a new point is defined by the following equation (Eq. 8):

$$r_{new} = max(d_j + r_j) \tag{8}$$

Here, r is the size of the new point, r_j is the size of the neighboring point, and d_{ij} is the distance between the new point and the neighboring points.
Equation 8 demonstrates that the size of a new point is computed by finding the greatest distance between the position of the new point and the positions of the removed points, as shown in Figure 6.

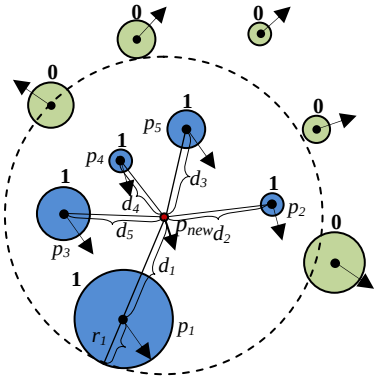


Figure 6. *Calculation of size of new point*

4. Experimental Results

The algorithms introduced in this paper were implemented in C++ and the CG shading language. Images were rendered on a computer with a 2.4 GHz

Intel Core 2Quad Q6600 processor, 2GB RAM, and an NVIDIA GeForce 8600GT graphics card. We used OpenGL and its extensions for the implementation of the vertex texture, multi-target rendering, and the Shader 3.0 model of the programmable vertex and fragment processing. In order to demonstrate the potential of our algorithm, we selected several simple polygon models. For the experiment, we used the Stanford Bunny, Female, Male, Beethoven, Ball-Joint, and Dragon polygon models available from the public domain. Point models were generated by taking the vertices and the normal vectors of polygon models. Three of the result images illustrating the basic features of our algorithm are shown in Figure 1. Through the following experiments, we demonstrated different aspects of the multiresolution generation algorithm.

LOD and Point Models. Figure 8 shows the results obtained by changing point models on the basis of the LODs. We used simple models as models that are located far from the view point. The images in the first row show the point distributions of models in each LOD. The other images illustrate the LOD in the case of different models.

Table 2. Performance of multiresolution point set generation algorithm

Model	*Dragon*	*Bunny*	*Female*
Number of Points	437,645	139,122	302,948
Neighborhood Search (ms)	110	34	76
Calculation of Variations (ms)	82	26	57
Evaluation of Point Elimination (ms)	96	31	66
Removal of Points and Generation of New Point (ms)	135	43	93
Total (ms)	423 (2.3 fps)	133 (7.5 fps)	292 (3.4 fps)

Weight Functions for Feature-Preserved Simplification. Variation coefficient and distance are the two main components of the bilateral elimination operator. In Figure 9(a), we show the effect of weight functions in the case of feature-preserved simplification. As shown in Figure 9(b), the number of removable points is inversely proportional to the threshold ε and parameter β. In other words, when the value of ε is decreased, the number of removable points will increase.

Multi-sized Splats. The effect of multisized splats on the rendering quality is shown in Figure 10.

Other Applications and Simplified Models. The proposed algorithm can be used in various applications such as LOD-based rendering and feature extraction in point-based models. We used the multiresolution models in algorithms other than LOD. We applied our algorithm to the feature-line extraction algorithm and to the generation of models for laser projection (Figure 11).

The performance of our algorithm is summarized in Table 2. We evaluated the performance by processing three different types of point clouds.

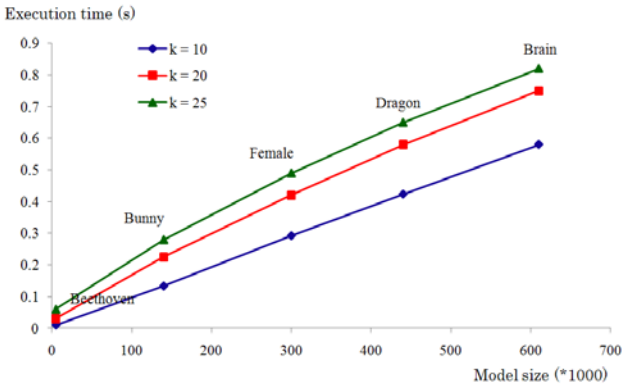


Figure 7. *Execution time for different k-nearest values*

Overall, our algorithm works in linear time. Figure 7 shows the graph of the computation time for different *k* values for neighbor selection.

5. Conclusion and Future Works

We have presented a new multiresolution point model generation algorithm. This algorithm works in linear time and requires a small amount of memory. The experimental results reveal that the algorithm can generate several feature-preserved simplified models, which can be used in rendering LOD, feature-line extraction, and laser-projection systems. Multi-sized splats contributed to the improvement in the quality of the model. In the future, we intend to make the following improvements to the algorithm:

- Optimization of the evaluation operator for point elimination. Improve the quality according to an error metrics [Jianhua Wu et al., 2005]
- Improvement in algorithm for using of progressive multi-resolution point surfaces by applying efficient data structures [Dachsbacher et al., 2003], [Gobbetty and Marton 2004] and compression schemes. [Fleishman et al., 2003]
- Implementation of our algorithm on a graphics processing unit (GPU)
- Application of our algorithm to volumetric point clouds

Acknowledgment
This work was partially supported by a Grant-in-Aid

for Scientific Research (B) 19300022 of the Ministry of Education, Science, and Culture.

References

[1] ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, T. (2001) Point set surfaces. In *Proceedings of the IEEE Visualization 2001*, pp.21-28.

[2] BATCHIMEG, S., FUJIMOTO, T., and CHIBA, N. (2009) Generation of multiresolution point set surfaces using multisized splats, In *Proceedings of the Nicograph International 2009*.

[3] BOTSCH, M., AND KOBBELT, L. (2003) High-quality point-based rendering on modern GPUs. In *Proceedings of the Pacific Graphics IEEE*, Computer Society Press, pp.335-343.

[4] BOTSCH, M., WIRATANAYA, A., AND KOBBELT, L. (2002) Efficient high-quality rendering of point-sampled geometry. *Rendering Techniques 2002, Eurographics Workshop on Rendering, Springer-Verlag*, pp.53-64.

[5] BOUBEKEUR, T., DUGUET, F., AND SCHLICK, C. (2005) Rapid visualization of large point-based surfaces. In *Proceedings of the VAST 2005*, pp.75-82.

[6] CHEN, B., AND NGUYEN, M. X. (2001) POP: A hybrid point and polygon rendering system for large data. In *Proceedings of the IEEE Visualization 2001*, pp.45-52.

[7] COHEN, J. D., ALIAGA, D. G., AND ZHANG, W. (2001) Hybrid simplification: Combining multiresolution polygon and point rendering. In *Proceedings of the IEEE Visualization 2001*, pp.37-44.

[8] DACHSBACHER, C., VOGELGSANG, C., AND STAMMINGER, M. (2003) Sequental point trees. *ACM Transactions on Graphics*, Vol.22, No.3, pp.657-662.

[9] DEY, T. K., AND HUDSON, J. (2002) PMR: Point to mesh rendering, a feature-based approach. In *Proceedings of the IEEE Visualization 2002*, Computer Society Press, pp.155-162.

[10] FLEISHMAN, S., COHEN-OR, D., ALEXA, M., AND SILVA, T. (2003) Progressive point set surfaces. *ACM Transactions on Graphics*, Vol.22, No.4, pp.997-1011.

[11] GARLAND, M., AND SHAFFER, E. (2002) A multiphase approach to efficient surface simplification. In *Proceedings of the IEEE Visualization 02*, pp.117-124.

[12] GOBBETTI, E., AND MARTON, F. (2004) Layered point clouds: A simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics*, Vol.25, No.1, pp.815-826.

[13] GROSSMAN, J., AND DALLY, W. (1998) Point sample rendering. *Rendering Techniques '98. Springer*, pp.181-192.

[14] HOPPE, H. (1996) Progressive meshes. In *Proceedings of the SIGGRAPH '96, Computer Graphics*, pp.99-108.

[15] HÜBNER, T., ZANG, Y., AND PAJAROLA, R. (2006) Multi-view point splatting. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques*, pp.285-294.

[16] JAGAN, S., SAMET, H., AND VARSHBEY, A. (2007) A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers & Graphics*, Vol.31, No.2, pp.157-174.

[17] KALAIAH, A., AND VARSHNEY, A. (2001) Differential point rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques. Springer-Verlag*, pp.139-50.

[18] LEVOY, M., AND WHITTED, T. (1985) The use of points as display primitives. *Technical Report TR 85-022*, Univ. North Carolina Chapel Hill, Computer Science Department.

[19] LINSEN, L. (2001) Point cloud representation. *Technical Report 2001-3*, Faculty of Computer Science, University of Karlsruhe.

[20] MARINOV, M., AND KOBBELT, L. (2005) Automatic generation of structure preserving multiresolution models. *Computer Graphics Forum*, Vol.24, No.3, pp.479-486.

[21] MOENING, C., AND DODGSON, N.A. (2003) A new point cloud simplification algorithm. In *Proceedings of the 3rd IASTED International Conference on Visualization, Imaging, and Image Processing*, Benalmadena, Spain, pp.1027-1033.

[22] NICK, R., KELLY, S., AND VINCENT, F. (1995) Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp.71-79.

[23] PAULY, M., GROSS, M., AND KOBBELT, L. (2002) Efficient simplification of point-sampled surfaces. In *Proceedings of the IEEE Visualization 2002*, pp.163-170.

[24] PAULY, M., KOBBELT, L., AND GROSS, M. (2006) Point–based multiscale surface representation. *ACM Transactions on Graphics*, Vol.25, No.2, pp.177-193.

[25] PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. (2000) Surfels: Surface elements as rendering primitives. In *Proceedings of the Computer Graphics SIGGRAPH 2000*, pp.335-342.

[26] REUTER, P., JOYOT, P., TRUNZLER, J., AND BOUBEKEUR, T. (2005) Point set surface with sharp features. *Research Report RR-1355-05*.

[27] ROSSIGNAC, J., AND BORREL, P. (1993) Multiresolution 3D approximations for rendering complex scenes. *Modeling in Computer Graphics: Methods and Applications, Springer-Verlag*, pp.455-465.

[28] RUSINKIEWICS, S., AND LEVOY, M. (2000) QSplat: A multiresolution point rendering system for large meshes. In *Computer Graphics Proceedings, Annual Conference Series* (*SIGGRAPH '00*), Addison Wesley, pp.343-352.

[29] TOMASI, C., AND MANDUCHI, R. (1998) Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision*, pp.839-846.

[30] TURK, G. (1992) Re-tiling polygonal surfaces. *SIGGRAPH '92*, pp.55-64.

[31] WU, J., ZHANG, Z., and KOBBELT, L. (2005) Progressive splatting. *Eurographics Symposium on Point-Based Graphics*, pp.25-32.

[32] ZORIG, G., FUJIMOTO, T., and CHIBA, N. (2007) Point splatting based on translucent shadow mapping and hierarchical bucket sorting. *The Journal of the Society for Art and Science*, Vol.6, No.1, pp.21-36.

Authors' biographies

Batchimeg SOSORBARAM is currently a Ph.D. candidate in computer science at Iwate University. Her research interests include computer graphics and point-based graphics. She received the BE in software engineering from Mongolian University Science and Technology, Mongolia and the ME in computer science from Mongolian University Science and Technology, Mongolia in 1999 and 2006, respectively.

Tadahiro FUJIMOTO is currently an associate professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics, geometric model, fractal theory, and mathematics for shape description in general. He received the BE in electrical engineering, and the ME and Ph.D. in computer science from Keio University in 1990, 1992, and 2000, respectively. He worked at Mitsubishi Research Institute from 1992 to 1995. He was a research associate in the Department of Computer and Information Sciences at Iwate University from 1999 to 2002, and a lecturer from 2002 to 2005. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.

Norishige CHIBA is currently a professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics, algorithm theory and science on form. He received the BE in electrical engineering from Iwate University and the ME and DE in information engineering from Tohoku University in 1975, 1981 and 1984, respectively. He worked at Nippon Business Consultant Co., Ltd. from 1975 to 1978. He was a research associate in the Department of Communication Engineering at Tohoku University from 1984 to 1986, an associate professor of computer science at Sendai National College of Technology from 1986 to 1987 and an associate professor of the Department of Computer and Information Sciences at Iwate University from 1987 to 1991. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.

LOD and Point Models

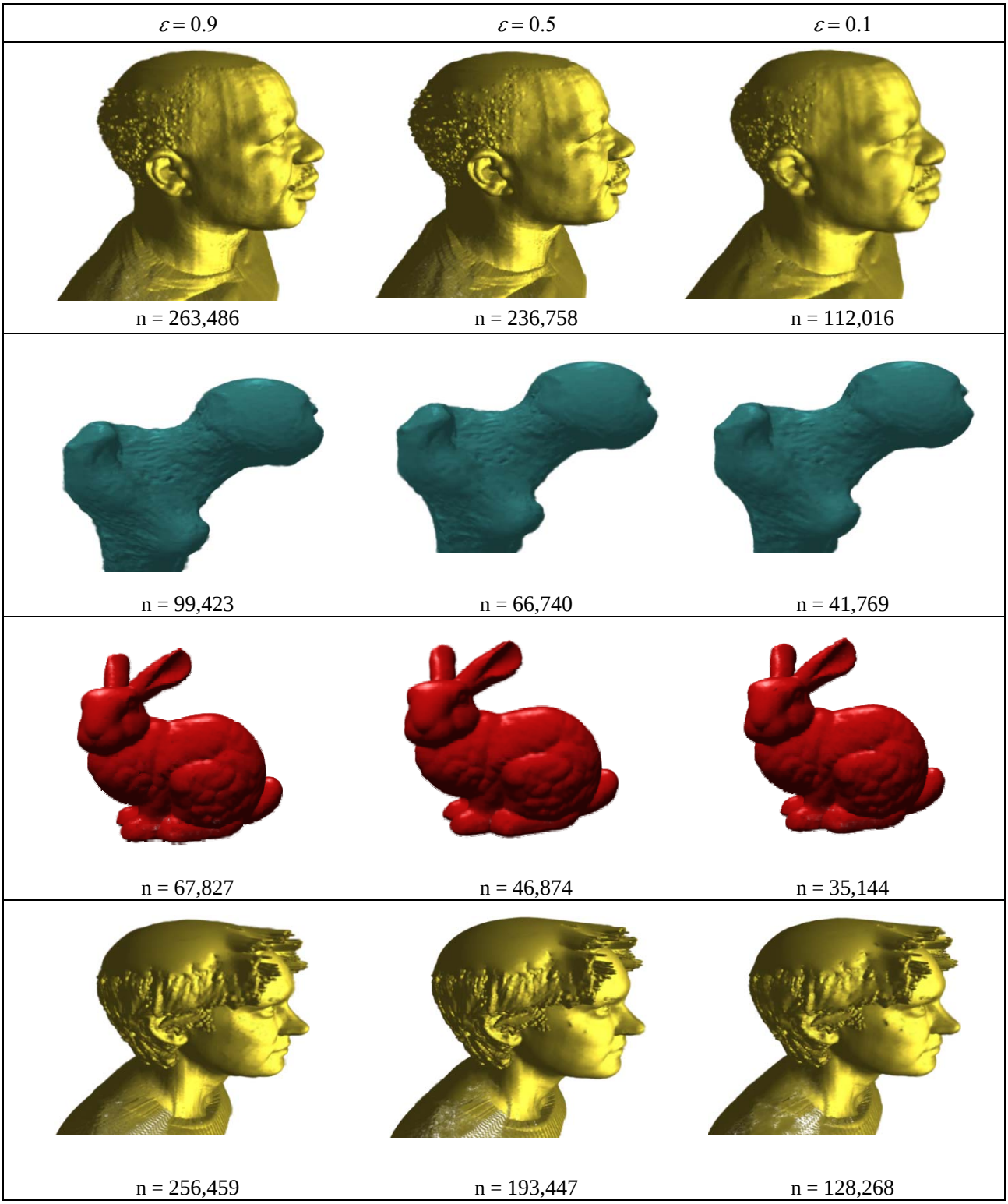


Figure 8. *LOD models ($\alpha = 0.1$, $\Psi = 0.01$, $\beta = 0.9$)*

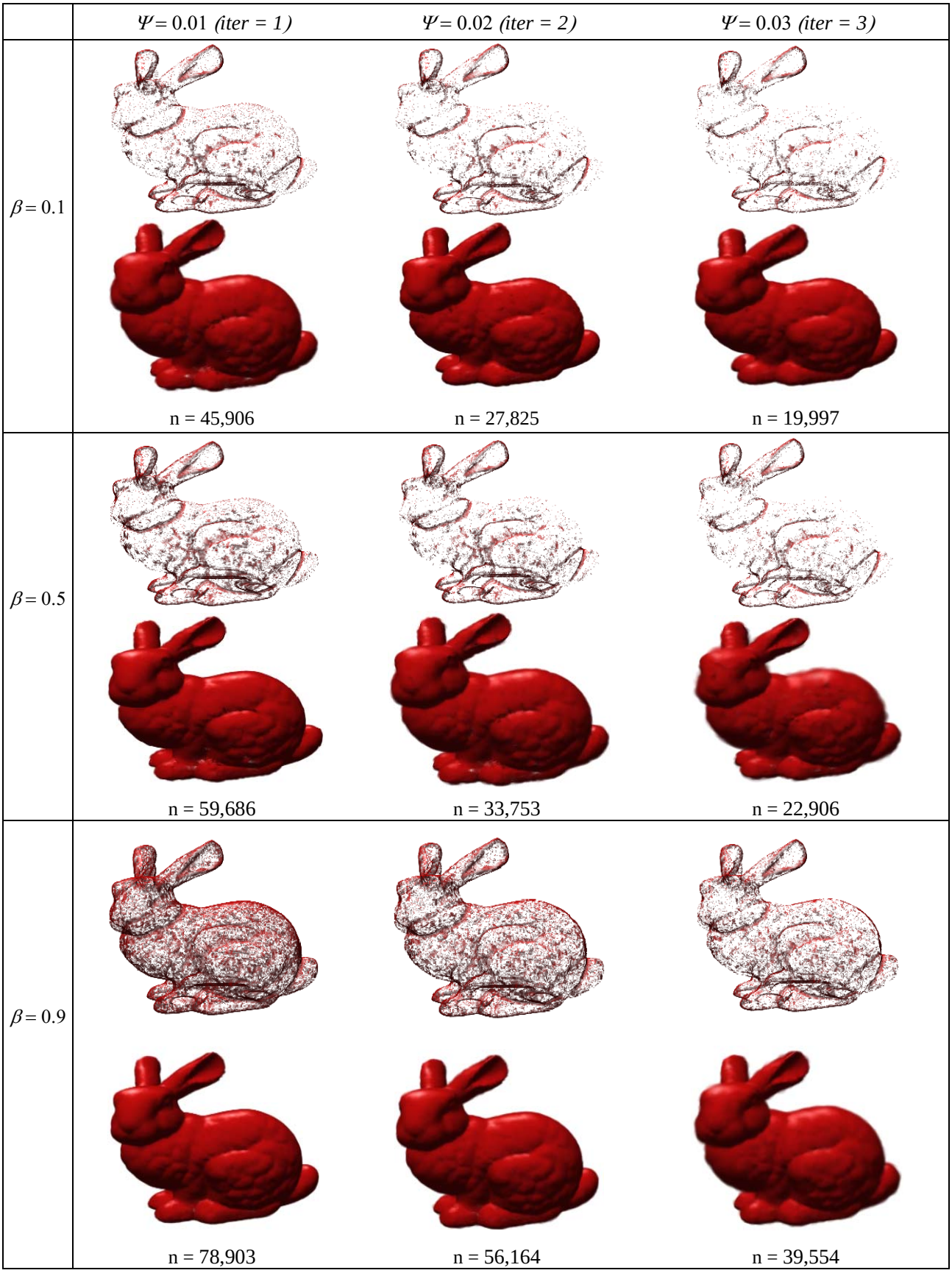


Figure 9(a). *Feature-preserved simplification with different parameters* β *and* Ψ $(\alpha = 0.1, \varepsilon = 0.99)$

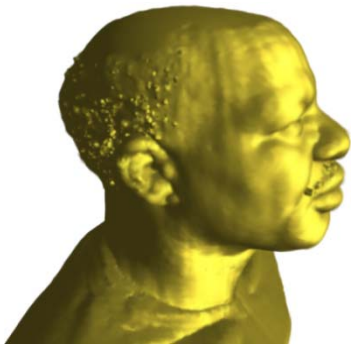
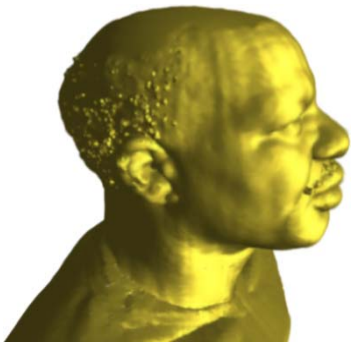
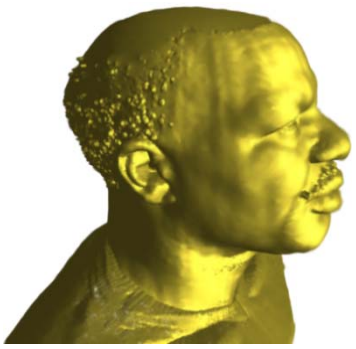
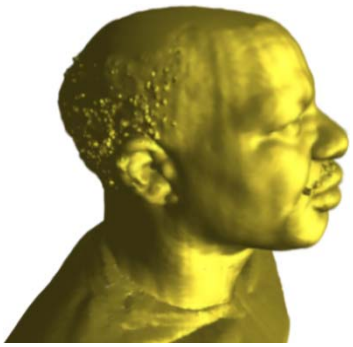
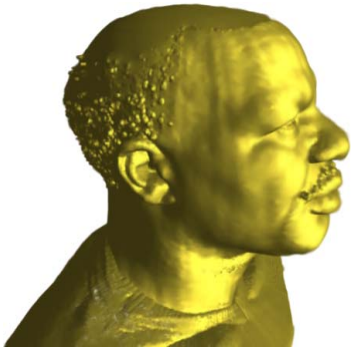
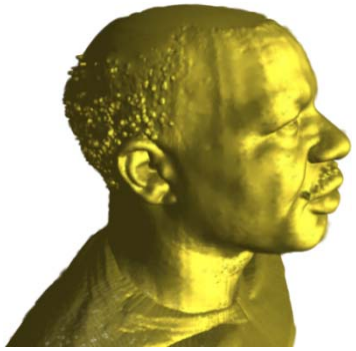
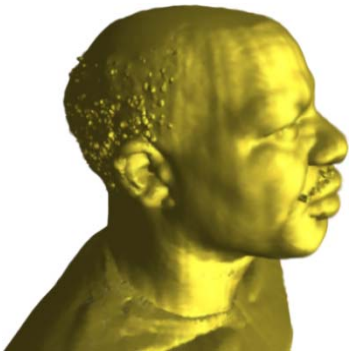
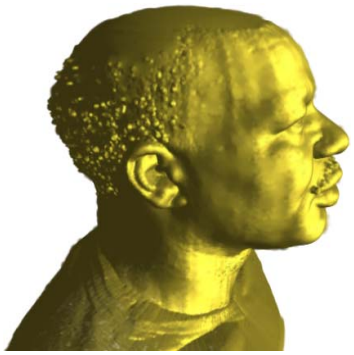
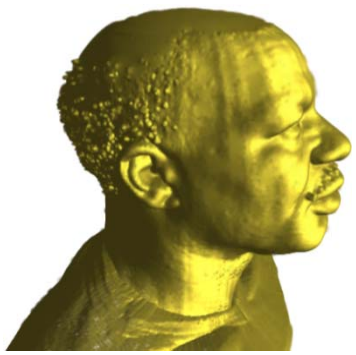
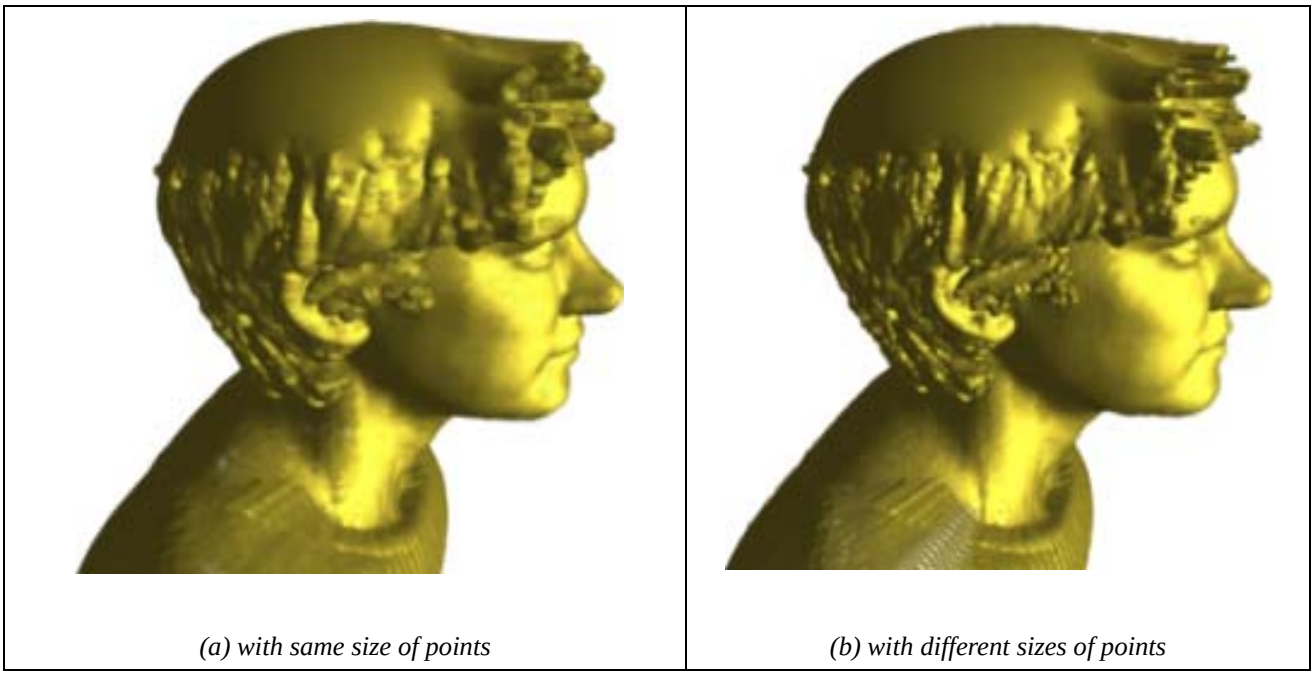
	$\varepsilon = 0.1$	$\varepsilon = 0.5$	$\varepsilon = 0.9$
$\beta = 0.1$	 n = 102,233	 n = 106,278	 n = 173,867
$\beta = 0.5$	 n = 106,278	 n = 154,696	 n = 236,751
$\beta = 0.9$	 n = 112,016	 n = 236,758	 n = 263,486

Figure 9(b). *Simulation results with different parameters β and ε ($\alpha = 0.1$, $\Psi = 0.01$)*



(a) with same size of points　　*(b) with different sizes of points*

Figure 10. *Effect of multisized splats (n = 184,312)*

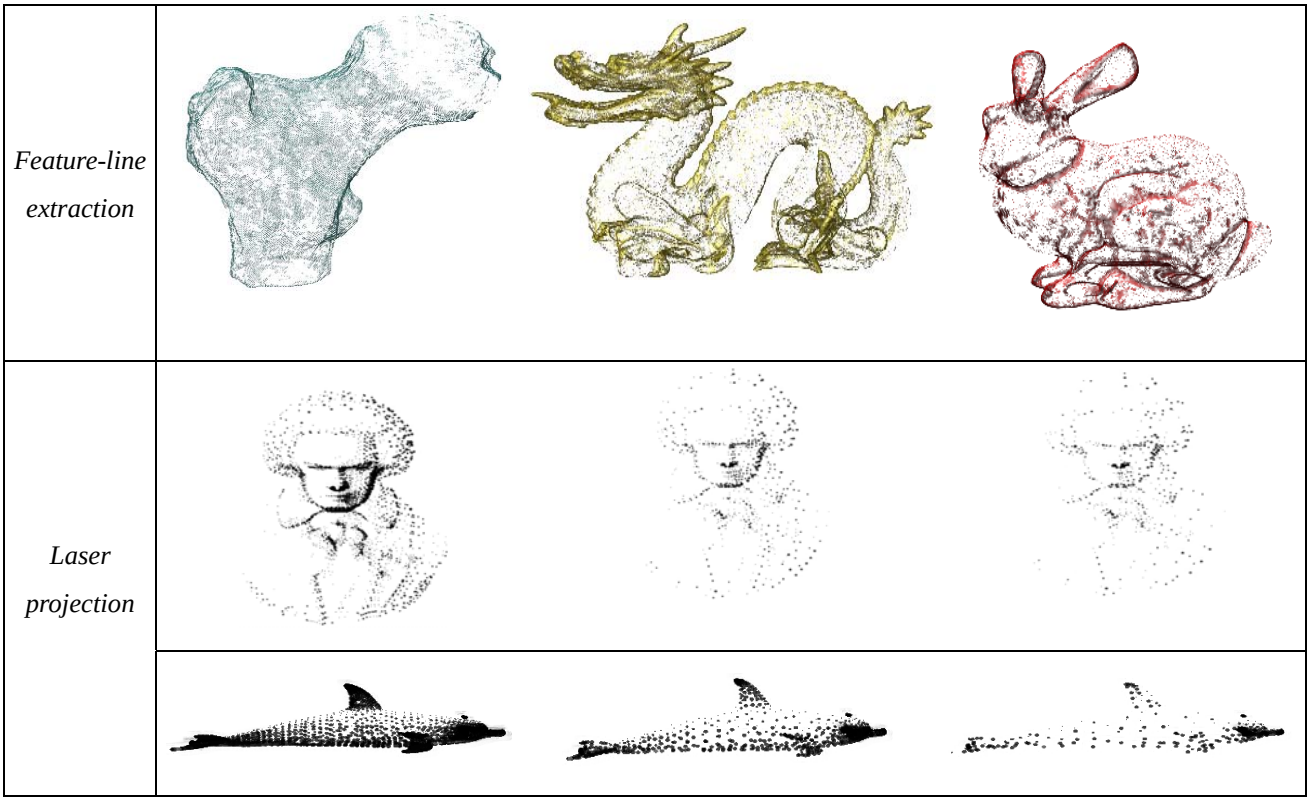


Figure 11. *Feature-line extraction and models for laser projection*