Adaptive Selection of Rendering Primitives for Point Clouds of Large Scale Environments

Takashi Maeno[1], Hiroaki Date[2] and Satoshi Kanai[3]
Graduate School of Information Science and Technology, Hokkaido University, Japan
[1] t_maeno@sdm.ssi.ist.hokudai.ac.jp, [2] hdate@ssi.ist.hokudai.ac.jp, [3] kanai@ssi.ist.hokudai.ac.jp

Abstract:
Recently, with the progress of laser scanning technology, it has become possible to easily acquire point clouds of large scale environments from several scanning platforms, and these point clouds are used in several fields such as simulation analysis, city planning, and plant management. Viewing the scenes acquired by laser scanning is necessary for checking scanned environments. However, it is difficult to understand the scanned environments only by displaying points as a rendering primitive. There are several existing rendering methods for point-sampled objects, such as methods using splats or surface mesh models in computer graphics field. However, it is difficult to achieve an effective view of the scenes of large scale environments with the existing rendering methods because the data have extremely non-uniform point density and spatial distribution and also include various kinds of objects with different scales and shape complexity. In this paper, in order to realize effective views of the scanned large scale environments, we describe a method for generating rendering models and point hierarchies of scanned large scale environments, as well as a method for LOD rendering using them.

Keywords: Laser Scanning, Point Cloud, Rendering, Splat, Mesh

1. Introduction

Laser scanning technology has become more and more common, and the development of technology has enabled the easy acquisition of point clouds of large scale environments from indoor to outdoor scenes. These point clouds are used in several fields such as simulation analysis, city planning, and plant management. There are 3 typical scanning types: TLS (Terrestrial Laser Scanning), MMS (Mobile Mapping System), and ALS (Airborne Laser Scanning). For example, MMS can acquire point clouds of large area, such as urban areas, using a sensor mounted mobile vehicle, and point clouds acquired from MMS are used for simulating a landscape of a city, periodic checkup of utilities such as roads, tunnels, etc.

Viewing the scenes acquired by laser scanning is necessary for checking the scanned environments. However, it is difficult to understand the scanned environments only by displaying points as a rendering primitive, which do not have surface information. There are typically two types of point cloud rendering methods, polygon-based rendering and point-based rendering. A rendering method using triangular mesh models is one of the most major polygon-based rendering methods. The surface of point-sampled objects can be reconstructed by generating mesh models. On the other hand, the point-based rendering method, as represented by splatting, has a simple data structure and does not need to construct topological information because a model is constructed from each point. Moreover, point-based rendering can easily sample points in suitable density for a specific image resolution, and it is suitable for LOD (Level of Detail) processing, which is necessary in case of handling large scale data sets. However, it is difficult to achieve an effective view for understanding the scanned large scale environments using these methods because the data have extremely non-uniform point density and spatial distribution and also include various kinds of objects with different scales and shape complexity.

In this paper, we describe a method for generating rendering models and point hierarchies appropriate for effective views of scanned large scale environments and a method for LOD rendering using them. A rendering model consists of three types of primitives, i.e. line segment, quadrilateral splat, and triangular mesh, and it is generated by adaptive selection of them based on dimensional analysis of local point distribution. Point hierarchy is created using octree structure and quantization.

The rest of this paper is organized as follows. Related works are described in section 2. In section 3, an overview of our rendering method is explained. In section 4, a method for rendering model generation is described. Creation of point hierarchy based on octree is mentioned in section 5. In section 6, LOD rendering using a rendering model and point hierarchy is described. Results and evaluations are shown in section 7, and finally, conclusions and future works are discussed in section 8.

2. Related Works

The triangular mesh is often generated from point clouds for rendering application. There are various studies on surface reconstruction methods from point clouds [1-3]. However, mesh quality (whether the mesh represents a correct geometry of the objects) may vary depending on the properties of the point clouds. For

example, mesh models cannot be always generated successfully for point clouds of large scale environments that have extremely non-uniform point density and include objects of complicated shapes. Moreover, topological information is not necessarily required for rendering applications.

Point-based rendering is one of the rendering methods for several geometric models. In point-based rendering, models are regarded as a set of points, and rendering primitives are defined at each point individually. Points were first used as universal rendering primitives for rendering geometric models by Levoy and Whitted [4]. Splatting [5] is one of the point-based rendering techniques. Splatting defines finite disks or ellipses in object space instead of points and renders them in image-space by projecting them onto a screen. Nakagawa [6] developed a point-based rendering application which can generate spatially interpolated virtual reality data called LiDAR VR. LDI (Layered Depth Image) [7] is also one of the point-based rendering methods using splatting. In this method, each pixel of a given screen holds a list of all color and depth values of the objects that intersect with a given sight line. In order to generate an image from several viewpoints, LDC (Layered Depth Cube) [8] is developed and consists of 3 LDI which are associated with 3 axes. Additionally, LDI tree [9] enables an appropriate sampling of a LDI pixel according to the position and the resolution of the reference image by constructing hierarchical structure using an octree structure, whose nodes are associated with LDI.

One of the problems of point-based rendering is to generate hole-free rendering on screen. Pfister, et al. [10] proposed the point-based rendering method using surface elements as rendering primitives, called Surfels, in order to close the holes and gaps between sample points. This method represents an object based on points by hierarchical structure using LDI. Surface splatting [11] renders object-space disks or ellipses instead of points for a hole-free rendering in image-space. This method proposes an effective rendering of point-sampled objects by texture mapping using a screen space EWA (Elliptical Weighted Average) filter.

Rusinkiewicz et al. [12] proposed a method for efficient rendering of large scale 3D mesh data using the multi-resolution point rendering technique. For the multi-resolution rendering, the data is converted into a tree structure. The nodes are laid out in breadth-first order, and during rendering, the appropriate resolution can be loaded progressively depending on the viewpoint. Wand et al. [13] described a new out-of-core multi-resolution data structure for real-time visualization and editing of large scale point clouds. To achieve efficient rendering, multi-resolution data structure is created by converting the point clouds into an octree structure and creating the quantized points by hierarchical down sampling at each inner node of the octree.

Usually, point based rendering methods using anisotropic disks or ellipses as rendering primitives require a normal for each point. However, it is difficult to derive a correct normal for each point from point clouds of large scale environments because they have extremely non-uniform point density and spatial distribution. In our method, several types of primitives were used in rendering simultaneously to get effective views of such environments, and correct normal for each point is not required.

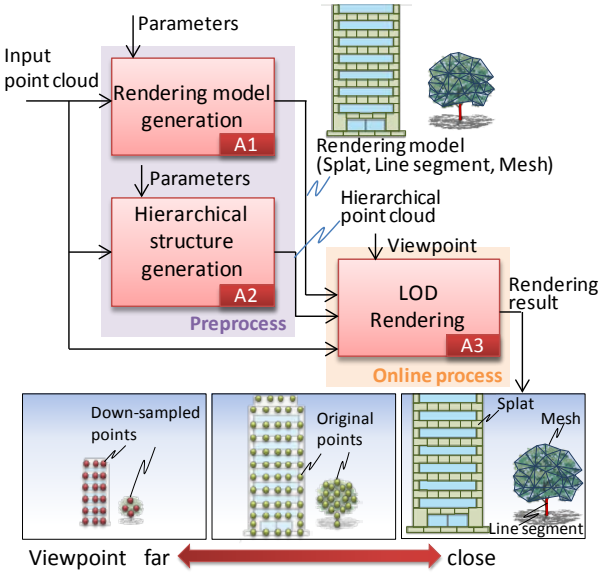


Figure 1. Proposed view-dependent LOD rendering method

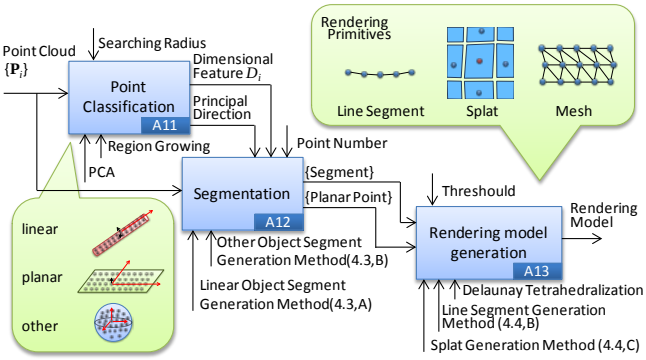


Figure 2. Proposed rendering model generation method

3. Rendering of Scanned Large Scale Environments using Adaptive Primitive Selection and LOD

In the rendering of scanned large scale environments, using adaptive primitives is useful because the environment includes several types of objects, such as pole like objects, buildings, power lines, cars, trees, roads, and so on. For example, compared with polygons or points, it is better to use linear type primitives (line segment or cylinder) for rendering power lines or pole like objects. In addition, LOD rendering of scanned environments is necessary for efficient rendering because the environments are viewed from several viewpoints. Therefore, the construction of a rendering model created by adaptive primitive selection and LOD technique using them are proposed in this paper.

In our method, the scanned environment (scene) is rendered effectively by view-dependent LOD using

hierarchical point cloud representations and a rendering model with several primitives (line segment, splat, mesh) as shown in Fig. 1. When the viewpoint is far from the scene, quantized points which are hierarchically down sampled points based on an octree structure are used. When the viewpoint is moved closer to the scene, original points are rendered. In addition, when the viewpoint is moved closer and closer to the scene, a rendering model is rendered, which is generated by adaptively selecting rendering primitives. The rendering model and point hierarchies are generated in preprocess in our method as shown in Fig. 1.

4. Rendering Model Generation

4.1 Overview

In rendering model generation, first, each point is classified into a linear object point, a planar object point, or an other object point. Then, a line segment is selected as a rendering primitive for linear object points, and a quadrilateral surface (splat) and a triangular mesh are selected as rendering primitives for planar object points as well as other object points respectively.

Figure 2 shows an overview of the rendering model generation method. The model is generated in preprocess. First, using PCA (Principal Component Analysis) and region growing, each point is classified into a linear object point, a planar object point, or an other object point. Next, linear object segments and other object segments are generated based on the result of point classification. Finally, the rendering model is generated from the result of point classification and segmentation.

4.2 Point Classification (Fig. 2 (A11))

Dimensional analysis of point clouds using PCA is commonly used for classification and segmentation purposes. Vandapel, et al. [14] classify points based on the dimensional analysis using PCA. However, they mention that it is difficult to classify points based on only a result of PCA because the result varies considerably depending on the type of terrain and the sensor. Jerome, et al. [15] classify points by adaptively tuning a parameter (search range for neighboring points) of PCA based on point distribution and entropy. In our research, we also use PCA for point classification.

First, planar regions are recognized because they can be found a lot in point clouds of large scale environments, such as buildings and roads in urban environments. In our method, PCA is applied to point clouds for initial point classification, and points are reclassified based on region growing in order to correctly classify planar object points. Then, linear objects and other objects can be segmented with high accuracy in subsequent segmentation process by classifying planar points correctly.

PCA is used in order to investigate local point distribution. Variance-covariance matrix $\mathbf{M}_i$ of neighboring points of point i is expressed by Eq. (1):

$$\mathbf{M}_i = \frac{1}{|i^*|}\sum_{j\in i^*}(\mathbf{p}_j - \bar{\mathbf{p}}_i)(\mathbf{p}_j - \bar{\mathbf{p}}_i)^T, \quad (1)$$

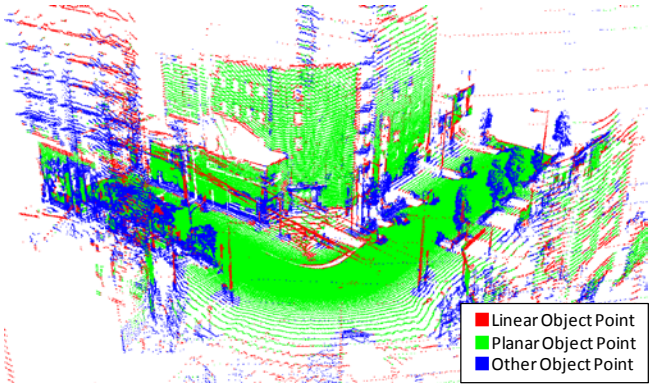


Figure 3. Point classification result based on PCA

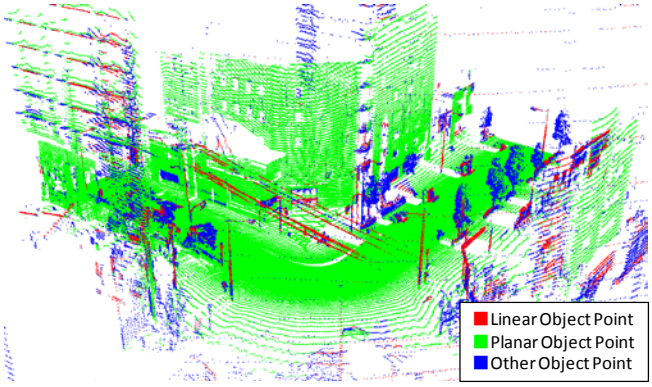


Figure 4. Point classification result after reclassification

where $\mathbf{p}_i$ is a coordinate value of point i, i^* is a set of neighboring points of point i, $\bar{\mathbf{p}}_i$ is a barycenter of i^*.

Eigenvalues $\lambda_1^i, \lambda_2^i, \lambda_3^i$ ($\lambda_1^i \geq \lambda_2^i \geq \lambda_3^i$) of $\mathbf{M}_i$ are derived by eigen analysis, and distribution feature values S_1^i, S_2^i, S_3^i are calculated by Eq. (2):

$$S_1^i = \lambda_1^i - \lambda_2^i, \quad S_2^i = \lambda_2^i - \lambda_3^i, \quad S_3^i = \alpha\lambda_3^i, \quad (2)$$

where α is a coefficient in order to correctly recognize 3D objects from point clouds of the object surfaces, and we set α=10 experimentally. Dimensionality feature [15] D_i is derived by Eq. (3):

$$D_i = \arg_{d\in\{1,2,3\}} \max(S_d^i). \quad (3)$$

According to the D_i, each point i is classified into a linear object point (D_i=1), a planar object point (D_i=2), and an other object point (D_i=3). Figure 3 shows a result of point classification of MMS point cloud based on PCA.

Points near the boundary of objects may be misclassified as other object points in point classification based only on PCA, even if they are actually planar object points (Fig. 3). Therefore, by using region growing, reclassification of planar object points is performed in order to correctly classify them. Region growing is done using the following conditions:

- Seed point: a point i which satisfies D_i=2 and has maximum S_2^i.
- Condition of growing (adding a point to the region): the distance between the point and the region boundary point (initially, seed point) is smaller than a threshold, and distance between the point and the

plane is also smaller than a threshold. The plane is defined using the normal of the seed point i (eigenvector corresponds to λ_3^i).

Dimensionality feature D_i of points in the resulting region are newly set to D_i=2. Using region growing, misclassified planar object points near the object boundary can be reclassified correctly, as shown in Fig. 4.

4.3 Segmentation (Fig. 2 (A12))

A) Linear Object Segmentation

Segments of linear objects, such as utility poles and power lines, are generated from point clouds without planar object points. First, a point i which satisfies D_i=1 and has maximum S_1^i is selected as a seed point of a segment. Then, for seed point i (segment boundary point), a point k which satisfies the conditions

$$\begin{aligned} &(\mathbf{p}_i - \mathbf{p}_k)\cdot \pm\mathbf{e}_i^1 > \delta_\theta, \\ &\min_k \|\mathbf{p}_i - \mathbf{p}_k\|, \\ &\|\mathbf{p}_i - \mathbf{p}_k\| < \delta_d, \end{aligned} \tag{4}$$

is added to the segment in sequence as shown in Fig. 5(a). Where $\mathbf{e}_1^i$ is an eigenvector of point i corresponding to the largest eigenvalue, δ_θ and δ_d are thresholds.

In case of thick linear objects, such as utility poles, multiple parallel line segments can be generated, and therefore, neighboring segments are integrated. As shown in Fig. 5(b), two segments are integrated if they have similar principle directions and the shortest distance between fitted straight lines is less than a certain threshold.

B) Other Object Segmentation

Segments of other object such as vegetation, cars, etc. are generated from point clouds without planar object points and linear object points using region growing. A point i with D_i=3 is selected as a seed point, and a point satisfying the condition that the distance from region boundary point (initially, seed point) and the point is within a certain threshold is added to the region.

4.4 Rendering Model Generation (Fig. 2 (A13))

A) Splat

Quadrilateral splat is generated for planar object points. According to regularity of laser scanned point clouds, quadrilateral shapes are selected because it is suitable for filling gaps in object space. First, adjacent points $\mathbf{p}_1^i \ldots \mathbf{p}_4^i$ of point i are searched along the principal directions in order to create splats. As shown in Fig. 6(a), a point satisfying the conditions

$$\begin{aligned} &(\mathbf{p}_i - \mathbf{p}_k)\cdot \mathbf{d} > \delta_\theta, \\ &\min_k \|\mathbf{p}_i - \mathbf{p}_k\|, \\ &\|\mathbf{p}_i - \mathbf{p}_k\| < \delta_d, \end{aligned} \tag{5}$$

is selected as an adjacent point of point i related to direction vector $\mathbf{d}$.

By using $\pm\mathbf{e}_1^i$, $\pm\mathbf{e}_2^i$ as $\mathbf{d}$ in above conditions, four

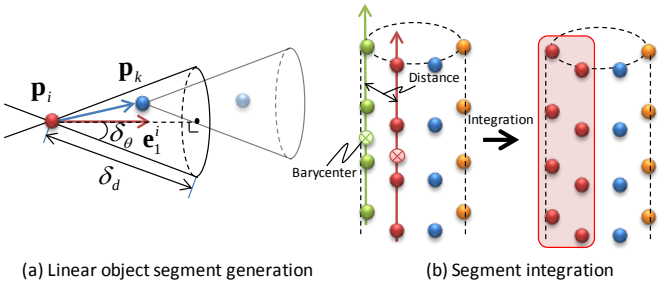


Figure 5. Linear object segment generation

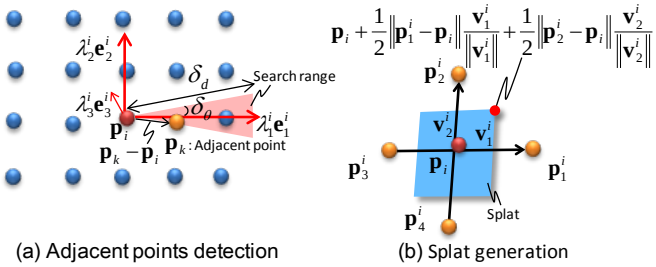


Figure 6. Splat generation

adjacent points can be obtained. Next, two difference vectors of adjacent points $\mathbf{v}_1^i$=$\mathbf{p}_1^i - \mathbf{p}_3^i$, $\mathbf{v}_2^i$=$\mathbf{p}_2^i - \mathbf{p}_4^i$ are generated. Then, corner points are represented by the sum of two difference vectors weighted by half of the distance between adjacent points and point i, as shown in Fig. 6(b).

B) Line Segment

For linear object segments, first, a line is created so that the line passes the barycenter of segment points and has an average direction of eigenvectors $\{\mathbf{e}_1^i\}$ of segment points as a directional vector. Then, segment points are projected onto the line, and line segments are generated connecting projected points in sequence.

C) Mesh

Triangular mesh is generated for other object segments. There have been various studies on surface reconstruction from point clouds [1-3]. In our implementation, a method based on tetrahedralization is used [1].

5. Hierarchical Point Cloud Representation

5.1 Octree

When the viewpoint is far from the scene, view-dependent LOD of point clouds is used in rendering. LOD is based on quantized points and octree structure.

An octree is a hierarchical representation of given point cloud and is adopted for efficient rendering of a scene by points. The hierarchy is generated by space subdivision. In Fig. 7, a quadtree is illustrated instead of an octree for simplicity. An octree is a tree data structure which is used to partition a 3D space by recursively subdividing it into eight sub-spaces. Each node has eight children nodes. The root node is associated with a minimum axis-aligned bounding cube of the given point

clouds. Each children node is associated with each uniformly subdivided cube of the parent node. The leaf node has points in the associated cube. Each node is recursively subdivided until the number of points stored in it becomes less than a certain number, $n_{\max}$. For example, Fig. 7 shows the case of $n_{\max}$=10. An octree structure enables efficient rendering and determining LOD as described in a latter section.

5.2 Quantization

LOD techniques [12,13] are necessary for efficient rendering of large scale scenes. In order to achieve an LOD representation, we use quantized points. As shown in Fig. 8, first, we create a quantization grid in each inner node of an octree by uniformly dividing its cube into quantization grid cells. If there are one or more points in each quantization grid cell, only one representative point is selected randomly and is stored in the node. In this way, uniformly down-sampled points can be stored in each inner node.

6. LOD Rendering

As mentioned in section 3, hierarchical point rendering using octree structure and quantized points is done when the viewpoint is far from the scene. When the viewpoint is moved closer, a rendering model generated by several primitives is rendered. Proposed LOD rendering is done as follows.

When the viewpoint is far from the scene, a depthfirst search of the octree is performed during rendering. Depth traversal is done until the node satisfies the condition $s/d < \delta$, and the points of the node are used for rendering, where s is the side length of quantization grid cell, d is the distance from viewpoint to barycenter of the points of each octree node, and δ is a threshold (Fig. 9). When the viewpoint is moved away from the scene, value d becomes large and more down-sampled points in upper level nodes are rendered. When the viewpoint is moved closer to the scene and the value s/d exceeds the threshold, more detailed points in deeper nodes are rendered. In addition, when the viewpoint is moved closer and closer, and the value s/d' exceeds the threshold, where d' is the distance between the viewpoint and the barycenter of the nearest node, a rendering model generated from several primitives is rendered. As a result, real-time view-dependent LOD can be achieved. In current implementation, a rendering model which represents the scene including all objects is used for rendering (local rendering of the model is not done).

7. Results and Evaluations

7.1 Test Data Set and Implementation

The data set used in this research was scanned by MMS in urban area which included 1,585,985 points. Our proposed method is implemented on a standard PC with Intel Core i7 2.93GHz, 8GB RAM, and GeForce GTX 470 graphics board using OpenGL for rendering.

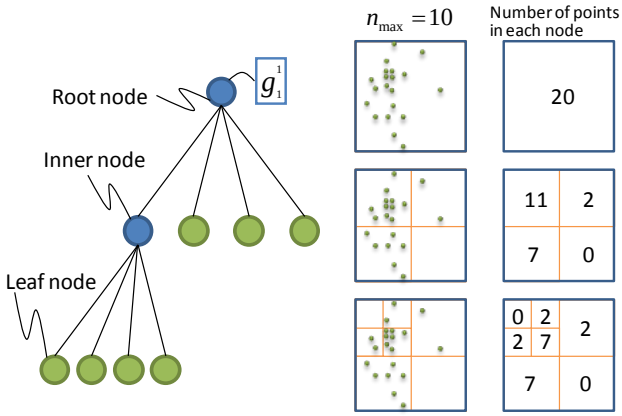


Figure 7. Hierarchical point cloud representation

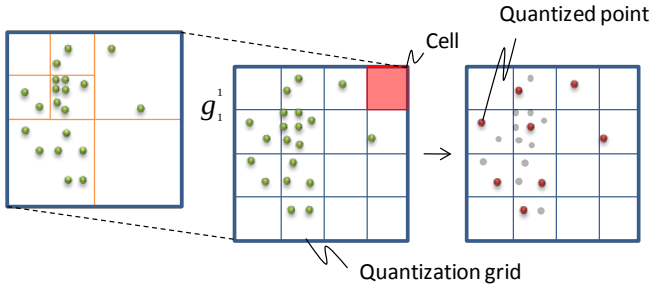


Figure 8. Quantization

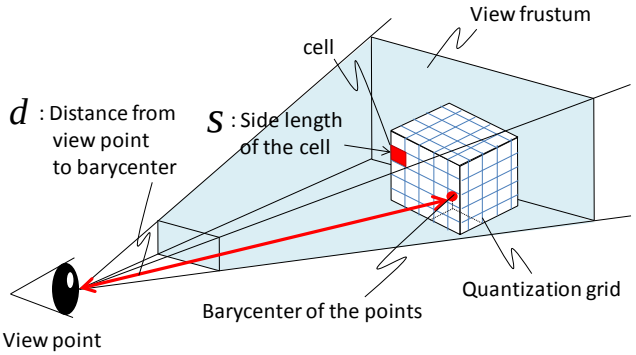
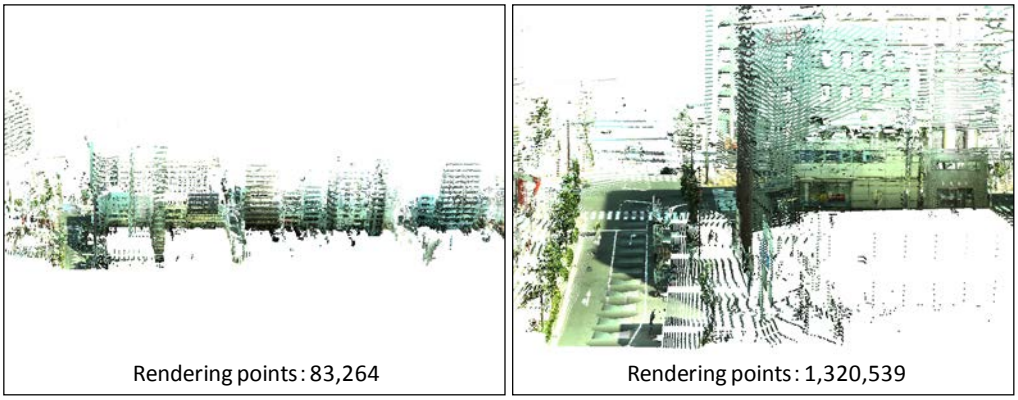


Figure 9. Each parameters used for LOD rendering

7.2 Point Cloud Rendering Results

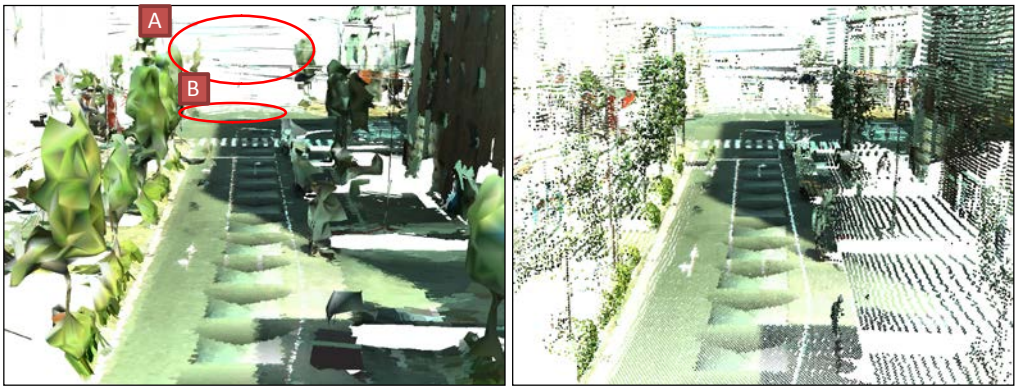
Figure 10 shows the rendering results using our method and other methods. When a viewpoint is far from the scene, it can be rendered with fewer points by LOD rendering while maintaining a certain FPS around 60. Results of point rendering at different viewpoints are shown in Fig. 10(a)-(b). Efficient rendering is achieved using LOD rendering of points.

For viewpoints close to the scene, the rendering model is used in rendering. The rendering model generated by our method is shown in Fig. 10(c). The number of splats, line segments, and mesh models are 1,256,277, 145,868, and 413 (consist of 558,789 triangles) respectively. Figures 10(d), (e), (f) show rendering results using points, splats, and mesh respectively. Compared with the rendering result using only points (Fig. 10(d)), our result (Fig. 10(c)) gives easier understanding of the scanned environment,



(a) Viewpoint: far, Primitive: point

(b) Viewpoint: middle, Primitive: point

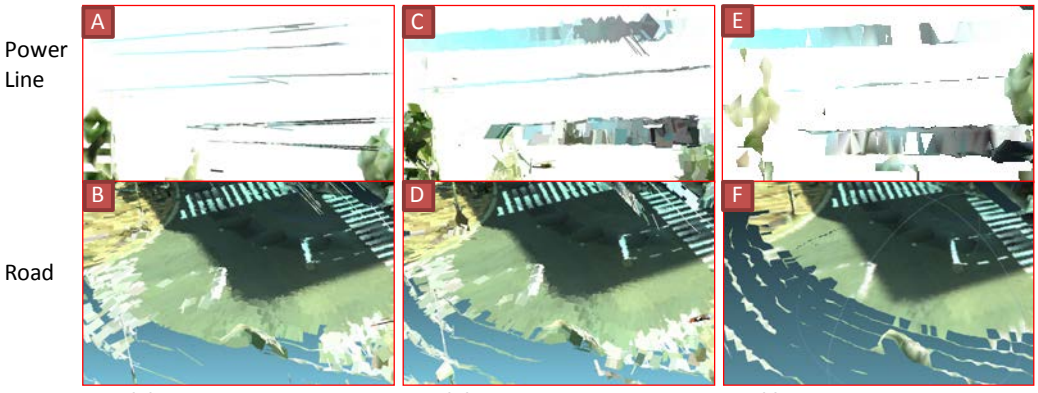


(c) Viewpoint: close
Primitive: point, line segment, splat, mesh

(d) Rendering by points

(e) Rendering by splats without point classification

(f) Rendering by mesh without point classification (Ball Pivoting [2])

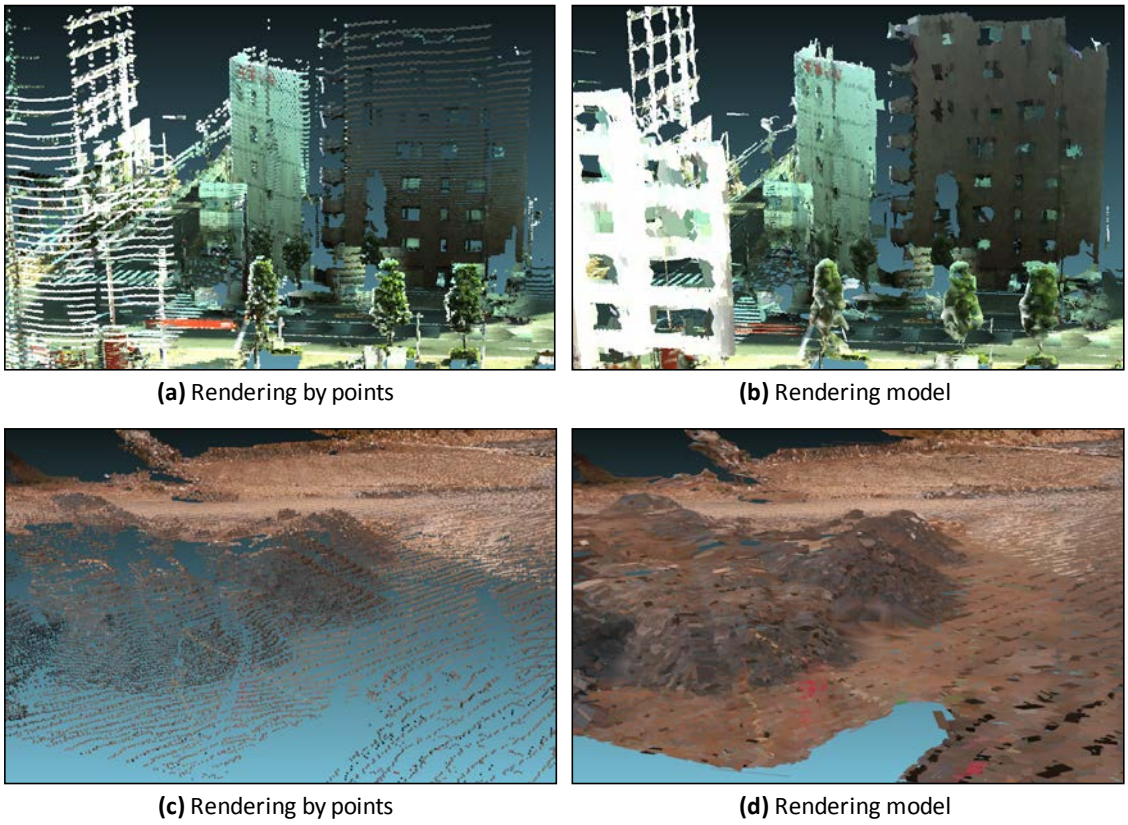


(g) Another view of (c)

(h) Another view of (e)

(i) Another view of (f)

Figure 10. Result of point cloud rendering



(a) Rendering by points **(b)** Rendering model

(c) Rendering by points **(d)** Rendering model

Figure. 11 Other results of point cloud rendering

especially at the region near the viewpoint, because the gaps at the ground and the vegetations are filled by splats and meshes. Unnatural vegetations can be seen in the rendering result using only splats (Fig. 10(e)) because it is difficult to reconstruct complex 3D objects using planar primitives from coarse points. Moreover, in the result, splats are generated at power lines (Fig. 10(h)). On the other hand, in our results, undesired splats are not generated at the power lines, because the line segment is used for rendering them with the help of point classification. Rendering result using only mesh (Fig. 10(f)) also shows unnatural views at power lines (Fig. 10(i)). Moreover, mesh is not generated at low point density areas such as the edge of roads (Fig. 10(i)) and building surfaces. In such areas, the view is improved by splats (Fig. 10(g)). Figure 11 shows other rendering results from different viewpoints of the scene (Fig. 11(a), (b)) and rendering results of another data set (Fig. 11(c), (d)) scanned by TLS. Compared with the rendering results using only points, the results using the rendering models give easier understanding of the scene. However, unnatural view around the boundary of the buildings and windows can be seen in Fig. 11(b). Therefore, quality improvement of the model using the recognition of object boundaries is required.

The processing time for generating the rendering model from the MMS data set was 462 [s]. FPS for rendering using point hierarchy was around 60, and for rendering model was 1-3. Improvement of a rendering speed while using rendering model is one of our future works.

8. Conclusions and Future Works

In this paper, we developed a method for rendering model generation by adaptive selection of rendering primitives (point, line segment, splat, mesh) in order to support visual understanding for point clouds of large scale environments. Our method is based on the dimensional analysis of local point distribution. First, each point is classified as either a linear object point, a planar object point, or as an other object point based on PCA and region growing. Next, linear object segments and other object segments are generated based on the results of point classification. Finally, a rendering model consisting of a set of line segments, splats and meshes is generated from the result of point classification and segmentation. In addition, hierarchical representation of point clouds based on octree structure and quantization is described for LOD of point clouds. We confirm that our method provides better viewing of scanned large scale environments compared to simply using a single type of primitive.

In the future, we will introduce blending techniques for splats, other types of primitives to render a scene more effectively, and LOD of a rendering model.

Acknowledgements

The data is provided from TOPCON Corporation, Koishi Corporation, and The Japan Society for Precision Engineering, Cyber Field Construction Technique Research Sectional Committee.

References

[1] J. D. Boissonnat, 1984, Geometric Structures for Three-Dimensional Shape Representation, ACM Transactions on Graphics, Vol.3, No. 4, pp.266-286

[2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, 1999, The Ball-Pivoting Algorithm for Surface Reconstruction, IEEE Transactions on Visualization and Computer Graphics, Vol.5, No. 4, pp.349-359

[3] N. Amenta, M. Bern, and M. Kamvysselis, 1998, A New Voronoi-Based Surface Reconstruction Algorithm, SIGGRAPH '98 Proceedings of the 25th annual conference on Computer Graphics and interactive techniques, pp.415-421

[4] M. Levoy, and T. Whitted, 1985, The Use of Points as Display Primitives, Technical Report TR 85-022

[5] L. Westover, 1990, Footprint Evaluation for Volume Rendering, SIGGRAPH '90, pp.367-376

[6] M. Nakagawa, 2010, LiDAR VR Generation with Point-based Rendering, UDMS2011 (28th Urban Data Management Symposium), pp.223-230

[7] J. Shade, S. J. Gortler, L. He, and R. Szeliski, 1998, Layered Depth Images, SIGGRAPH '98, pp.231-242

[8] D. Lischinski and A. Rappoport, 1998, Image-Based Rendering for Non-Diffuse Synthetic Scenes, Rendering Techniques '98, pp.301-314

[9] C. F. Chang, G. Bishop, and A. Lastra, 1999, LDI Tree: A Hierarchical Representation for Image- Based Rendering, SIGGRAPH '99, pp.291-298

[10] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, 2000, Surfels: Surface Elements as Rendering Primitives, SIGGRAPH 2000, pp.335-342

[11] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, 2001, Surface Splatting, SIGGRAPH 2001, pp.343-352

[12] S. Rusinkiewicz, and M. Levoy, 2000, QSplat: A Multiresolution Point Rendering System for Large Meshes, SIGGRAPH 2000, pp.343-352

[13] M. Wand, A. Berner, M. Bokeloh, A. Fleck, M. Hoffmann, P. Jenke, B. Maier, D. Staneker, and A. Schilling, 2007, Interactive Editing of Large Point Clouds, Eurographics Symposium on Point-Based Graphics, pp.37-46

[14] N. Vandapel, D. F. Huber, A. Kapuria, and M. Hebert, 2004, Natural Terrain Classification Using 3-D Ladar Data, IEEE International Conference on Robotics and Automation, pp.5117-5122

[15] J. Demantke, C. Mallet, N. David, and B. Vallet, 2011, Dimensionality Based Scale Selection in 3D LiDAR Point Clouds, Proceedings of ISPRS Workshop Laser Scanning 2011