www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Reconstruction of Polygonal Faces from Large-Scale Point-Clouds of Engineering Plants

Hiroshi Masuda[1], Takeru Niwa[2], Ichiro Tanaka[3] and Ryo Matsuoka[4]

[1]The University of Electro-Communications, h.masuda@uec.ac.jp
[2] The University of Electro-Communications, takeru.niwa@uec.ac.jp
[3]Tokyo Denki University, tanaka@cck.dendai.ac.jp
[4]The University of Tokyo, matsuoka@nakl.t.u-tokyo.ac.jp

ABSTRACT

The recent progress of mid-range and long-range laser scanners makes it possible to capture dense point-clouds of manufacturing plants. 3D models of manufacturing plants are useful for simulating the reorganizing of production lines. Since point-clouds are not structured and very large, it is often required to convert point-clouds into more concise models. So far, researchers have studied shape reconstruction of pipe structures by detecting cylindrical surfaces and estimating the lengths of the cylinders. On the other hand, few researches have discussed to extract polygonal faces from large-scale point-clouds. Planar faces may have very complicated boundaries, because point-clouds are noisy and often occluded by other surfaces in manufacturing plants. In this paper, we propose an efficient method for constructing simplified polygonal faces from large-scale point-clouds. In our method, we map each point-cloud onto a 2D image and detect bounded planar faces in the 2D image. Our method allows us to construct polygonal faces in a practical calculation time.

Keywords: shape reconstruction, point cloud processing, reverse engineering.

1. INTRODUCTION

The recent progress of mid-range and long-range laser scanners makes it possible to capture dense point-clouds of engineering plants. Phase-based laser scanners are typically used to survey engineering plants, because they can capture tens of millions points within a few minutes. When a engineering plant is surveyed at intervals of 6.3 mm at the distance of 10 m, the number of points is about fifty millions per each scan. Since a manufacturing plant has to be measured at dozens of places for reducing unmeasurable areas, the total amount often reach billions of points.

3D models of manufacturing plants are useful for simulating reorganization of production lines. The engineers of process planning have to evaluate various plans. Then as-is 3D models are very useful, because they can precisely predict feasibility of planes by estimating collisions. As-is 3D models are also useful for inventory management of engineering plants.

Although point-clouds are promising for supporting reorganization and maintenance tasks, raw point-clouds are not convenient, because they are not suitable for shape manipulation and their data sizes are too large to handle with common PCs.

It is often required to convert point-clouds to simplified surface models. Recently, it is becoming common to survey large-scale fields using laser scanners, and researchers have been developing methods that can process huge point-clouds. While the computer graphics community tends to roughly generate 3D models of building, cities and indoor scenes [3][8,9][12], the CAD communities have studied methods for generating 3D models as precisely as possible. When 3D models are used for estimating renovation plans or inventory management, requirements for precision of as-is models are typically 3 mm–10 mm.

So far, researchers in the field of CAD have proposed shape reconstruction of pipe structures

Computer-Aided Design & Applications, 12(5), 2015, 555–563, http://dx.doi.org/10.1080/16864360.2015.1014733

Taylor & Francis
Taylor & Francis Group

from large-scale point-clouds [1,2][4,5][7][13]. In these researches, the essential topic is how to extract cylindrical surfaces precisely and reliably. When a cylindrical surface can be extracted, the shape of a pipe can be reconstructed by only estimating the length of the cylinder.

On the other hand, planar faces may have complicated boundaries, and surfaces are often be occluded by other surfaces. While planar surfaces can be extracted using the RANSAC method or the region-growing method, the estimation of their boundaries is not a trivial problem.

In this paper, we discuss methods for efficiently and reliably constructing polygonal faces from large-scale point-clouds. We assume that point-clouds are captured at several positions, and each point-cloud contains tens of millions points. In our experiments, a common PC with 16GB RAM can process only a single point-cloud at a time. Therefore, we detect polygonal faces from each point-cloud, store them in a concise format, and merge them into unified polygonal faces. In addition, boundaries of polygonal faces may be partly missing because of occlusions. We discuss how to make up for missing parts so that the resultant shapes are consistent with measured point-clouds.

2. OVERVIEW

We efficiently generate polygonal faces from large-scale point-clouds. In our method, we first convert each point-cloud into a 2D development image for efficiently processing large-scale point-clouds. Then we search for planar faces in the image. Since it is very time-consuming and unstable to extract all planes from a large-scale point, we segment a point-cloud into continuous point-sets by examining the proximity of neighbor points. When planar faces are detected, their boundaries are detected and stored. When all faces are detected from point-clouds, they are merged into unified polygonal faces. Finally we discuss how to recover missing portions when the original surfaces are estimated as rectangle planes.

3. RECONSTRUCTION OF POLYGONAL FACES

3.1. Development Image of Point-Cloud

The kd-tree and octree are typically used for estimating the connectivity of points, but these methods are time-consuming when they are applied to very large point-clouds. In stead, we represent the connectivity using the adjacency on the 2D lattice.

In our method, we map a point-cloud onto a 2D lattice for maintaining adjacent relationships of points. This mapping is possible because mid/long-range laser scanners output organized points based on the mechanism of the measurement. Mid/long range laser scanners emit a laser beam to an object and measure the round-trip travel time of the laser beam. Laser scanners sample points on objects while rotating laser beams with an equal angle spacing. Fig. 1 shows a mechanism of a laser scanner. Laser beams are rotated vertically by spinning a miller and horizontally by rotating the body of the laser scanner.

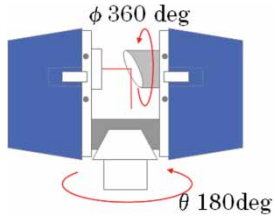

Fig. 1: Rotation angles of a laser scanner.

The directions of laser beams can be represented using the azimuth angle θ and the zenith angle ϕ. Then (x, y, z) coordinates can be converted to spherical coordinates (r, θ, ϕ), when the origin of the coordinate system is placed at the source of laser beams. When the angle spacing is constant while measurement, points are regularly ordered in the angle space and they can be developed onto a 2D rectangle image defined by two rotation angles.

We map each point (x, y, z) in a point-cloud onto (θ, ϕ) in a development image, and quantize the position using the sampling interval of a laser scanner. Fig. 2 shows a development view of a point-cloud captured at a fixed position. In this figure, a reflected intensity of a laser beam is drawn at each pixel as a brightness value. In this image, (x, y, z) is also described at each pixel.

Neighbor points can be obtained by traversing adjacent pixels on a 2D image. In our experiments, our neighbor search is about 10 times faster than the search by the kd-tree and the octree when a point-cloud consists of tens of millions points. We evaluated the computation time of the kd-tree and the octree using the Point Cloud Library (PCL) [10].

3.2. Segmentation to Continuous Regions

In this paper, we detect planar regions using the RANSAC method. In the RANSAC method, three points are randomly selected and a plane equation is calculated using the three points. Then we count the number of points on the plane. This process is iterated many times and the plane equation with the maximum number of points is recorded. When the maximum number exceeds a certain threshold, the detected region is regarded as a planar face.

Although the RANSAC method is powerful, it is very time-consuming when it is applied to a large-scale point-cloud. Schnabel, et al. [11] estimated the number of iterations T when the RANSAC method can detect a plane with the probability p:

$$T \geq \frac{\log(1-p)}{\log\{1-(n/N)^3\}}, \tag{3.1}$$

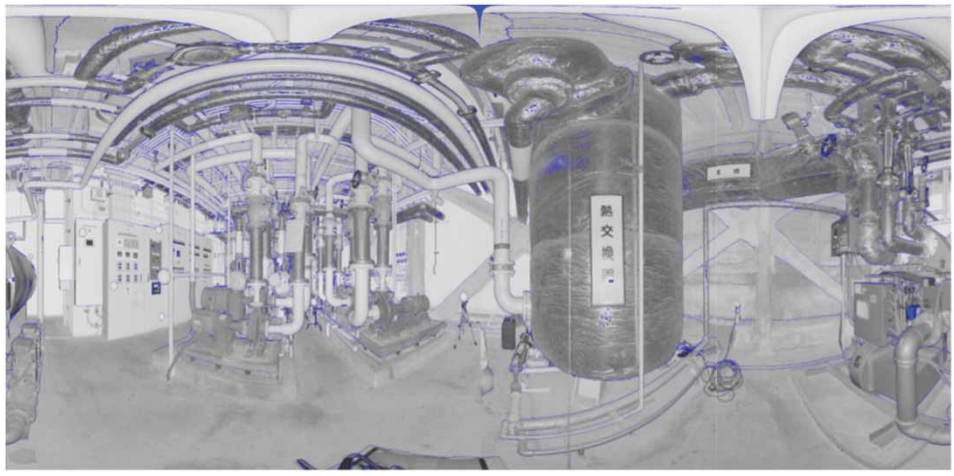

Fig. 2: Development image of a point-cloud.

where N is the number of points in a point-cloud; n is the number of points on a plane. In this paper, we handle tens of millions points of a manufacturing plant. When we suppose $N = 50,000,000$, $n = 10,000$, and $p = 0.99$, the value of T becomes more than 57 billions only for detecting a single plane. It is absolutely necessary to segment a point-cloud into many point-sets without dividing each plane.

For reliably segmenting a point-cloud, we detect continuous surfaces in a development image. We estimate distances between neighbor points when the two points are on the same surface. As shown in Fig. 3, the distance of two neighbor points on the same plane can be estimated as:

$$s = \frac{|\mathbf{p}|^2 \Delta\phi}{|(\mathbf{p}, \mathbf{n})|}, \tag{3.2}$$

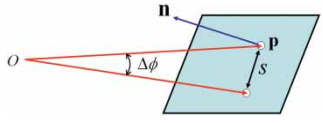

Fig. 3: Estimation of distance between neighbor points.

where $\mathbf{p}$ is a coordinate; $\mathbf{n}$ is the normal vector of the plane; $\Delta\phi$ is the interval of the azimuth and zenith angles. Since $\Delta\phi$ is very small in a dense point-cloud, we can approximate a small region on a continuous surface as a plane. The normal vector can be estimated using neighbor points on the development image [1][6].

We evaluate whether two neighbor points are on a continuous surface. Suppose the actual distance between two neighbor points is calculated as d. We regard that the two points are on a continuous surface only when $d < k \cdot s$. The variable k is a constant value. We specified $k = 1.2$ in this paper.

We detect each continuous surface using the region-growing method. We select a point and grow the region when the distance of adjacent points is less than $k \cdot s$. We repeat this process until each point belongs to one of continuous regions. In our implementation, we discard continuous regions with less than M points. The threshold M can be specified by the user.

Fig. 4 shows an example of segmentation. In this example, 2972 continuous regions were generated. We specified threshold M as 300. In our experiments, the processing time was about 5 sec for 40 million points.

3.3. Detection of Planes

Planes are detected in each continuous region using the RANSAC method. The performance of the RANSAC method is determined by the value of (n/N) in Eqn. (3.1). When (n/N) is very small, the number of iterations may become prohibitively large. In the example of Fig. 4, the largest region consists of 7 million points, and six regions have more than a million points.

Fortunately, very large continuous regions in a manufacturing plant include large planar floors or walls. Since (n/N) is relatively large in these cases, the floor and wall planes can be detected during a reasonable number of iterations. Fig. 5 (a) shows a planar floor in the largest continuous region. Since (n/N) is equal to 0.54 in this case, only 27 iterations are enough to detect the floor plane with a probability of 99%.

In the RANSAC method, planes tend to be detected in the order of areas of planes from the largest to the smallest. When the largest plane is detected and eliminated from a continuous region, the remaining region becomes disconnected in most cases. In the case of Fig. 5(a), the elimination of the floor plane causes three large continuous regions and many small ones.

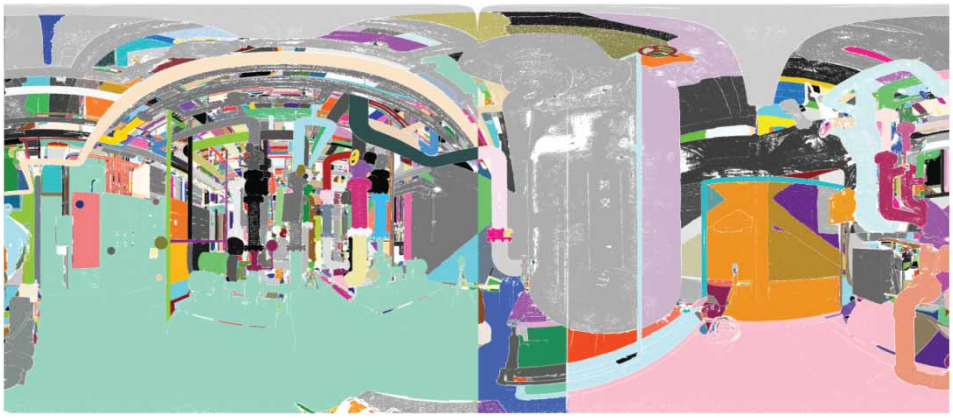

Fig. 4: Segmentation into continuous surfaces.

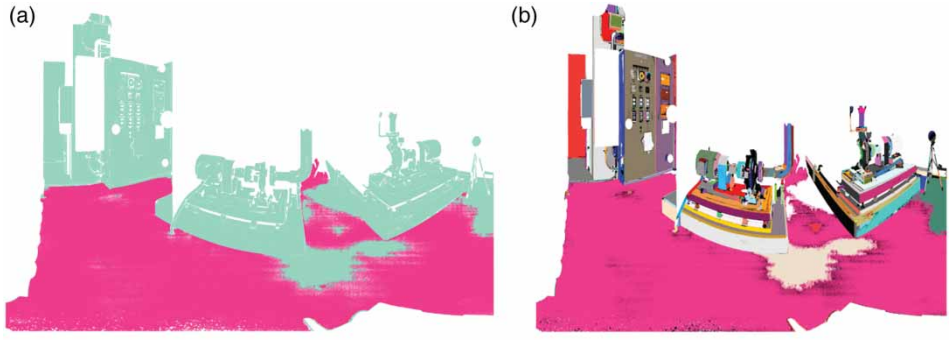

Fig. 5: Plane detection in the largest continuous region.

In each subdivided region, planes can be detected in a fewer number of iterations, because N in Eqn. (3.1) becomes smaller.

For reducing the number of iterations, we apply segmentation based on Eqn. (3.2) each time a plane is detected and removed from a continuous region. The continuous region is recursively subdivided so that the search areas become gradually smaller. This algorithm allows us to detect even small planes from a large-scale point-cloud. Fig. 5(b) shows all detected planes in the largest continuous region. In this example, the largest plane consists of 3.8 million points, and the smallest one consists of 307 points. Small planes are detected after the original continuous region is subdivided into considerably small regions.

The size of detectable planes is determined by distances and sampling intervals, as shown in Fig. 3. In our experiments, we could calculate a plane equation when a certain number of points were sampled from a planar region. In this paper, we specified a threshold for the plane size as the number of points on a plane. In this paper, we used 300 points as the threshold.

We note that cylindrical surfaces are also included in manufacturing plants. When the plane detection is applied to a cylindrical surface, it generates many strip-like planes. For avoiding such false planes, we apply the cylinder detection concurrently with the plane detection. When multiple adjacent planes fit to a cylinder, the points are regarded as a part of the cylinder and removed from the list of planes.

We apply plane detection to each continuous region while segmenting the region until no planes can be detected. Fig. 6 shows planes and cylinders detected from a point-cloud. The computation time was 268 second using Intel Core i7 CPU with 12 GB RAM.

Each detected plane consists of a lot of points. In the example of Fig. 6, the number of points on all planes is 24 million points. Since very large memory space is required to store all planes in multiple point-clouds, each plane should be represented concisely.

We represent planes using equations and their boundaries for reducing the data size. However, boundaries of detected planes may be complicated because of sampling intervals, noises and occlusions. Fig. 7(a) shows an example of the boundary of a planar region. We simplify the boundaries of detected planes. We detect straight lines on the boundary of each plane and merge collinear segments along polygon boundaries. Since the sampling interval can be estimated using Eqn. (3.2), we use the sampling interval as a threshold of line fitting.

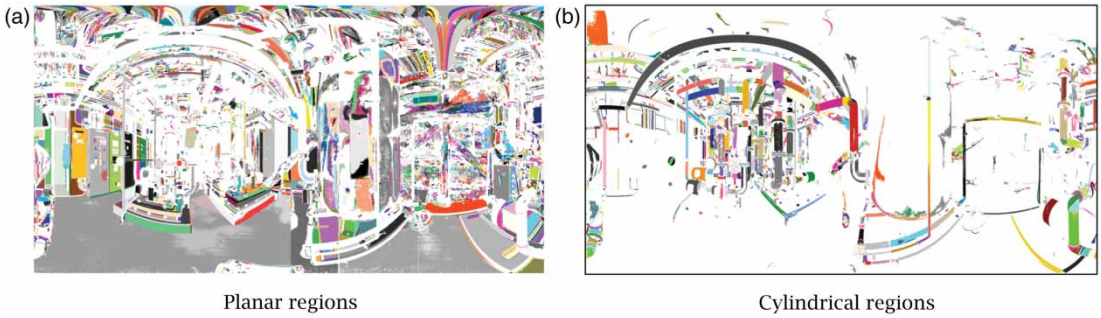

Fig. 6: Regions of continuous surfaces on a development image.

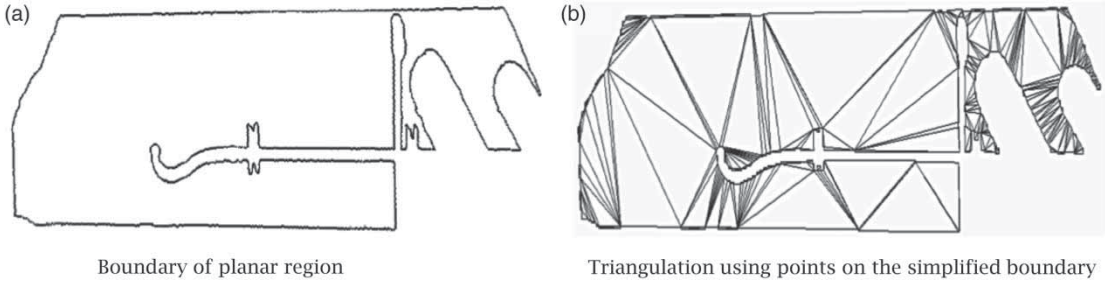

Fig. 7: Representation of a planar face.

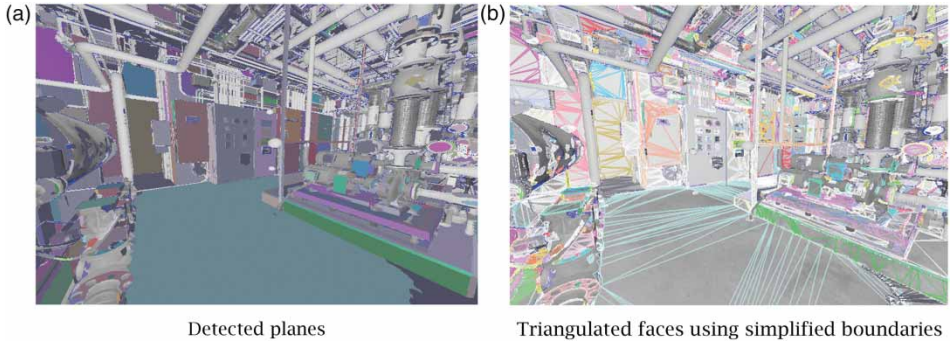

Fig. 8: Simplified plane faces.

Fig. 7(b) shows a simplified boundary and the triangulated mesh model. In the case of Fig. 6, we could reduce the number of points from 24 million to 0.21 million points for 5624 planes. Fig. 8 shows simplified faces on a development image. Fig. 9 shows all planes extracted from a single point-cloud with about 50 million points. Our method can efficiently process a large-scale point-cloud. The computation time was 114 second using Intel Core i7 CPU with 12 GB RAM.

3.4. Merging Polygonal Faces from Multiple Scans

When a manufacturing plant is surveyed, point-clouds are captured at multiple positions in order to reduce occluded regions. Since our method uses development images based on scanner positions, we have to process each point-cloud separately.

We suppose that a registration matrix is assigned to each point-cloud. Then planar faces can be transformed onto the world coordinate system using registration matrices, as shown in Fig. 10. We describe the world coordinate system as S_w, a local coordinate system defined at scanner position i as S_i, and a registration matrix from S_i to S_w as M_i.

When faces from each point-cloud overlap in the world coordinate system, we merge them and refine the boundary. We define a 2D lattice on a detected plane and merge overlapping planar faces on the lattice. Since we can estimate sampling intervals using

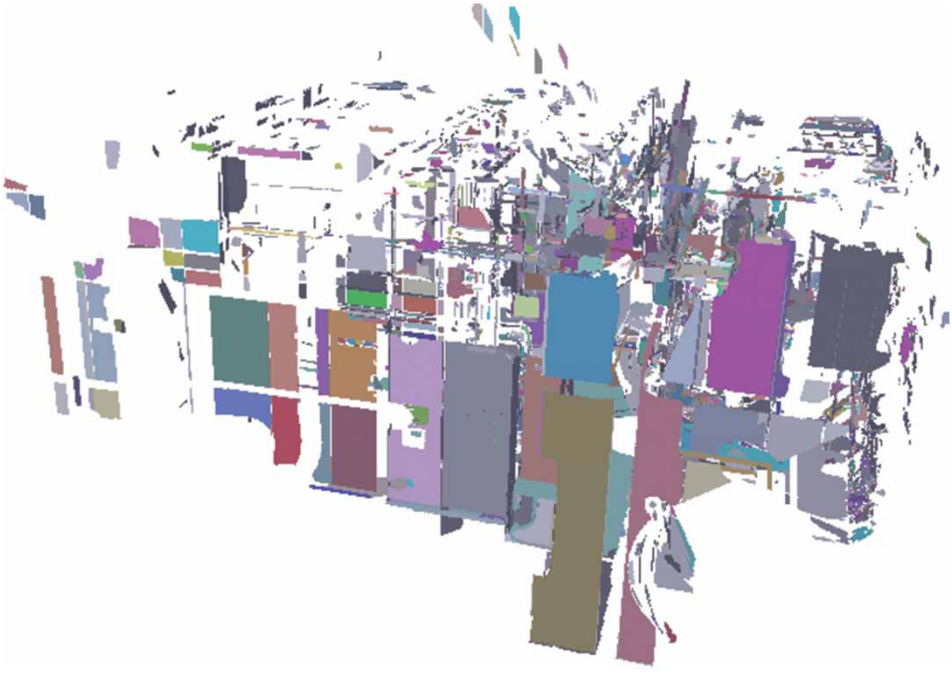

Fig. 9: Planar faces extracted from a point-cloud.

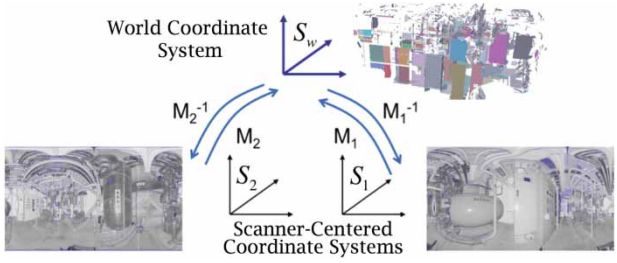

Fig. 10: Transformation between the world coordinate system and the scanner-centered coordinate systems.

Eqn. (3.2), we define the spacing of the lattice as the average of intervals.

We project boundary points of each plane onto the lattice. We generate a closed line of the boundary and fill the inside of the closed region. When all overlapping faces are projected on the lattice, the boundary of the filled region is traversed and it is stored as a merged polygonal face.

Even when planes are merged, some potions of planes are still missing. In Fig. 11, some planes have complicated boundaries because of occlusions. In a manufacturing plant, many planar faces are originally rectangles. For reconstructing rectangles, we search for straight-line segments from the boundary of each plane. When perpendicular and parallel lines are detected from the boundary of a plane, the plane is regarded as a candidate of a rectangle.

When we generate a candidate rectangle, we evaluate the rectangle is consistent with point-clouds. Fig. 12 shows the relationships between the depth

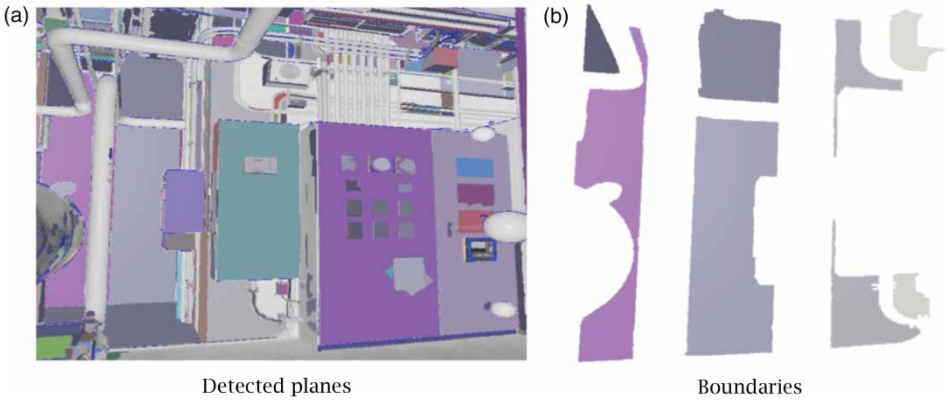

Fig. 11: Boundaries of polygonal faces.

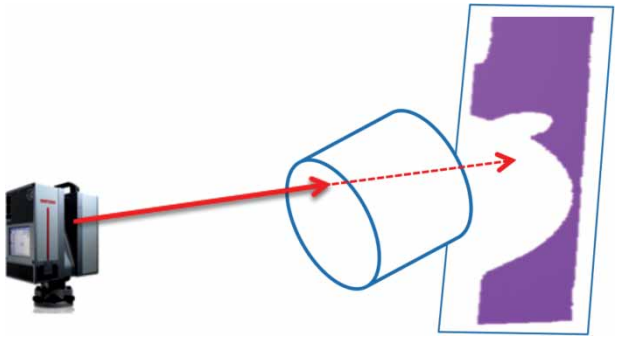

Fig. 12: Consistency check on a development image.

on a development image and the distance to the estimated plane. If a missing portion stems from an occlusion, the depth on a development image must be larger than the distance to the plane. We project candidate rectangle onto each development image and apply the consistency test. If a candidate rectangle passes tests on all development images, it is regarded as a rectangle face. Fig. 13 shows rectangle faces generated from planes with complicated boundaries.

4. EXPERIMENTAL RESULT

We applied our method to point-clouds in Fig. 14. We measured a facility using two different resolutions. Point-clouds 1, 2, and 3 contain about 40 million points, and point-clouds 4, 5, and 6 contain about 10 million points. In this figure, blue pixels show that laser beams are not returned.

Tab. 1 shows the numbers of points, continuous regions generated by segmentation, and planar faces. The computation time of segmentation and plane detection is also shown in this table.

In Tab. 2, we show the computation time for unifying and simplifying polygons from three point-clouds. The total processing time includes loading three ASCII files, removing their noises, and detecting unified planar faces.

Our experimental results show that our method can generate polygonal faces from large-scale point-clouds in a practical time. Our method could process a

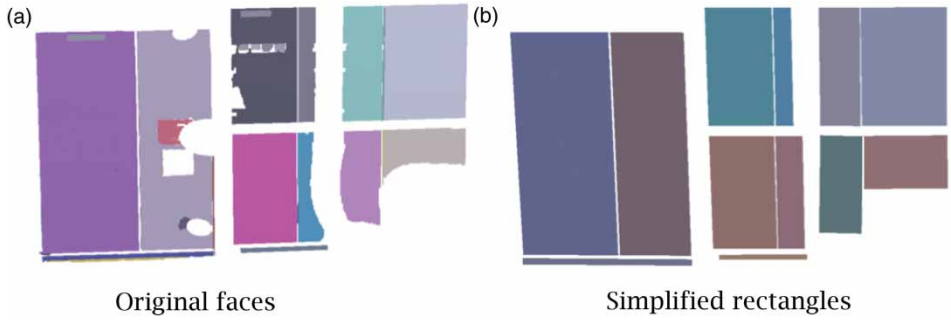

Fig. 13: Generation of simple polygons.

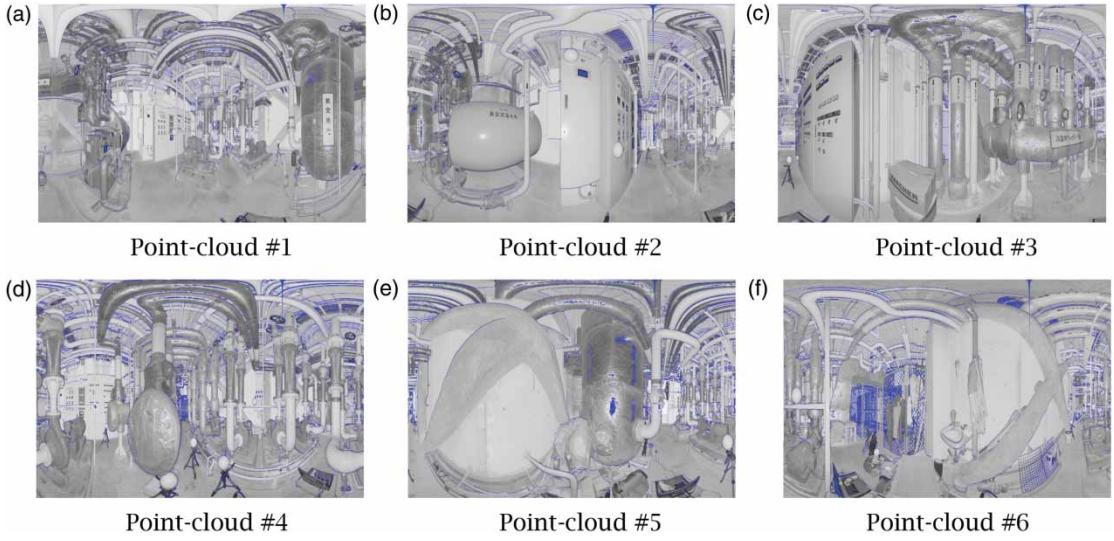

Fig. 14: Examples of point-clouds.

Point-Cloud	Number of Points	Number of Regions	Number of Planes	Timing for Segmentation	Timing for Plane Detection	Average for 1000 Planes
# 1	40,660,717	2,629	5,818	4.7 sec	268 sec	46.1 sec
# 2	40,538,597	3,223	4,037	4.4 sec	167 sec	41.4 sec
# 3	40,705,360	2,611	2,923	4.5 sec	83 sec	28.4 sec
# 4	9,842,495	1,486	1,852	1.1 sec	36 sec	19.4 sec
# 5	10,016,136	898	1,072	1.1 sec	62 sec	57.8 sec
# 6	9,911,855	2,093	1,552	1.0 sec	80 sec	51.6 sec

Tab. 1: Calculation time for point-clouds.

Point-Cloud	Total Number of Polygons	Timing for Unifying Plane faces	Total Processing Time
#1, #2, #3	12,778	48.3 sec	13 min 28 sec
#4, #5, #6	4,476	11.6 sec	4 min 8 sec

Tab. 2: Calculation time for unifying polygons.

dozen of point-clouds with 40 million points in about an hour.

5. CONCLUSION

In this paper, we proposed a method for extracting polygonal faces from large-scale dense point-clouds. We first generated a development image using each point-cloud and segmented it into continuous regions. Then we extracted planes using the RANSAC method. We merged polygonal faces extracted from multiple point-clouds. We also showed a method for simplifying rectangle faces. The experimental results showed that our method could generate polygonal faces in a practical time.

In future work, we would like to reconstruct complete solid objects in manufacturing plants by combining polygonal faces. Since there are a lot of missing portions in point-clouds, it is necessary to make up for missing shapes. In actual surveys, a single plane may be captured as multiple disconnected ones because of occlusions. In such cases, we would like to reconstruct reasonable planar shapes by considering various types of constraints. We observed that very thin cylinders were sometimes detected as planes. Since it is actually hard to identify surface types from sparse noisy points, we would like to investigate knowledge-based approaches for engineering plants. In our current implementation, we can construct simple cuboids with perpendicular polygonal faces. Since there are a variety of objects in manufacturing plants, we would like to develop methods that can be applied to non-trivial shapes.

REFERENCES

[1] Kawashima, K.; Kanai, S.; Date, H.: As-Built Modeling of Piping System from Terrestrial Laser Scanned Point Clouds Using Normal-Based Region-Growing, 2013 Asian Conference on Design and Digital Engineering, 2013.

[2] Lee, J.; Kim, C.; Son, H; Kim, C.: Skeleton-Based 3D Reconstruction of As-Built Pipelines from Laser-Scanned Data, ASCE International Conference on Computing in Civil Engineering, 2012, 245.

[3] Lin, H.; Gao, J.; Zhou, Y.; Lu, G.; Ye, M., Zhang, C.; Yang, R: Semantic decomposition and reconstruction of residential scenes from lidar data, ACM Transactions on Graphics, 32(4), 2013. http://dx.doi.org/10.1145/2461912.2461969

[4] Masuda, H.; Tanaka, I.: Extraction of Surface Primitives from Noisy Large-Scale Point-Clouds, Computer-Aided Design & Applications, 6(3), 2009, 387–398. http://dx.doi.org/10.3722/cadaps.2009.387-398

[5] Masuda, H.; Tanaka, I.; Enomoto, M.: Reliable Surface Extraction from Point-Clouds using Scanner-Dependent Parameters, Computer-Aided Design & Applications, 10(2), 2012, 265–277. http://dx.doi.org/10.3722/cadaps.2013.265-277

[6] Mitra, N., J.; Nguyen, A.: Estimating Surface Normals in Noisy Point Cloud Data, Symposium on Computational Geometry'03, 2003, 322–328.

[7] Mizoguchi, T.; Kuma, T.; Kobayashi, Y.; Shirai, K.: Manhattan-World Assumption for As-Built Modeling Industrial Plant. Key Engineering Materials, 523, 2012, 350–355. http://dx.doi.org/10.4028/www.scientific.net/KEM.523-524.350

[8] Nan, L.; Sharf, A.; Zhang, H.; Cohen-Or, D.; Chen, B.: SmartBoxes for Interactive Urban Reconstruction. ACM Transactions on Graphics (TOG), 29(4), 2010, 93. http://dx.doi.org/10.1145/1778765.1778830

[9] Nan, L.; Xie, K.; Sharf, A: A search-classify approach for cluttered indoor scene understanding. ACM Transactions on Graphics (TOG), 31(6), 2012, 137–1. http://dx.doi.org/10.1145/2366145.2366156

[10] The Point Cloud Library, http://pointclouds.org
[11] Schnabel, R.; Wahl, R.; Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection, Computer Graphics Forum, 26(2), 2007, 214–226. http://dx.doi.org/10.1111/j.1467-8659.2007.01016.x
[12] Sharf, Q.; Sharf, A.; Wan, G.; Li, Y.; Mitra, N., Cohen-Or, D., Chen, B.: Non-local Scan Consolidation for 3D Urban Scenes, ACM Trans. Graph., 29(4), 2010, Article 94.
[13] Tang, P.; Huber, D.; Akinci, B.; Lipman, R.; Lytle, A: Automatic Reconstruction of As-Built Building Information Models from Laser-Scanned Point Clouds: A Review of Related Techniques, Automation In Construction, 19(7), 2010, 829–843. http://dx.doi.org/10.1016/j.autcon.2010.06.007