


Interactive collision detection for engineering plants based on large-scale point-clouds

Takeru Niwa  and Hiroshi Masuda 

The University of Electro-Communications, Japan

ABSTRACT

In this paper, we discuss interactive collision detection for large-scale point-clouds. The state-of-the-art laser scanners enable us to capture dense point-clouds from engineering plants. Dense point-clouds are useful for simulating renovation tasks of engineering plants. However, it is difficult to interactively calculate collisions between 3D models and huge point-clouds. We propose an efficient collision detection method for large-scale point-clouds, and discuss an interactive system for collision detection. In our method, collisions are evaluated on two-resolution depth-maps. Our method allows us to precisely and efficiently detect collisions with large-scale point-clouds. The experimental results showed that our system could interactively detect collisions in large-scale point-clouds.

KEYWORDS

Point-Clouds; Collision Detection; Depth Map; Engineering Plants

1. Introduction

The state-of-the-art laser scanners make it possible to capture dense point-clouds of engineering plants. Typical phase-based scanners can capture one million points in a second in the range of 100 meter. Dense point-clouds are faithful 3D representation of engineering plants. They can be used for simulating whether equipment can be safely moved without collisions when engineering plants are renovated.

However, it is not easy to realize real-time collision detection for large-scale point-clouds, because point-clouds of an engineering plant often consist of hundreds of millions of points. In addition, point-clouds include a lot of missing positions, because occlusion cannot be avoided in most manufacturing plants. Although points are absent in occluded regions, such regions may not be free space. Even if 3D models do not collide with point-clouds in occluded regions, they may collide in actual engineering plants.

So far several researchers have studied collision detection for point-clouds [1–4],[6–8]. Most methods subdivide 3D space using the octree or the kd-tree for quickly detecting collisions [2–4],[6],[8]. However, the octree and the kd-tree are not very efficient when the number of points is tens or hundreds of millions. Other researchers proposed screen-space methods for collision detection [1],[7]. However, they applied their methods only to relatively small point-clouds. In our experiments,

it was difficult to promptly evaluate collisions on a high-resolution depth map with 50 megapixels. In addition, conventional methods may output false-negative collision results when points are missing in occluded regions, although false-negative collisions are dangerous in actual tasks.

In this paper, we propose a real-time point-based collision detection method, which can handle large-scale dense point-clouds and can avoid false-negative evaluations. Our method can be categorized into screen-space methods, but we introduce two-resolution angle-space depth maps for improving the performance of collision detection. We also discuss techniques for developing an interactive collision detection system.

2. Overview

In this paper, we consider collision detection between 3D models and point-clouds. We suppose that the positions of laser scanners are known, because typical laser scanners output coordinates that are defined on the scanner-centered coordinate system, in which the origin (0, 0, 0) is the scanner position. When multiple point-clouds are captured at different positions, coordinates of point-clouds are transformed to the world coordinate system. We suppose that coordinates can be mutually converted to the scanner-centered system and the world coordinate system.

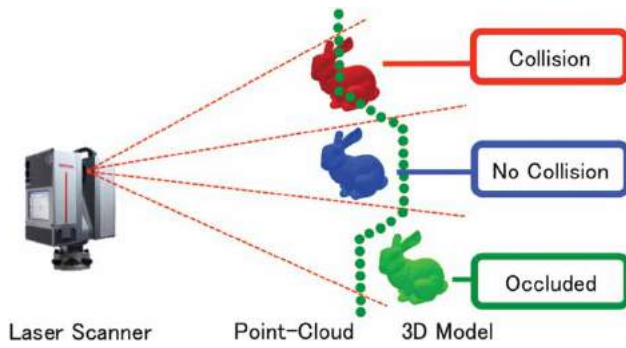


Figure 1. Three statuses of collision detection.

Our method classifies collision statuses into “*collision*”, “*no-collision*”, and “*occluded*”. Fig. 1 shows three statuses for collisions between a 3D model and a point-cloud. When the closed space of a 3D model includes a part of the point-cloud, the status is “*collision*”. When the 3D model is placed in front of the point-cloud, the status is “*no-collision*”. Otherwise, the status becomes “*occluded*”, which means that a 3D model is placed in an occluded region.

In our method, collisions are evaluated on a 2D angle-space depth map for efficiently processing large-scale point-clouds. Our method converts each scanner-centered coordinate (x, y, z) into a spherical coordinate (θ, ϕ, r) and projects the point on the angle-space depth map by quantizing two angles (θ, ϕ) . Each pixel value of the depth map is $r = \sqrt{x^2 + y^2 + z^2}$.

Figure 2 shows depth checks on an angle-space depth map. Each face of 3D models is projected on the depth map, and depth values are compared at each pixel to determine collision statuses.

Our method evaluates collisions using two-resolution depth maps. While the high-resolution depth map is useful for precise collision detection, the low-resolution one can be quickly processed. Our two-resolution depth maps allow us to detect collisions precisely and quickly.

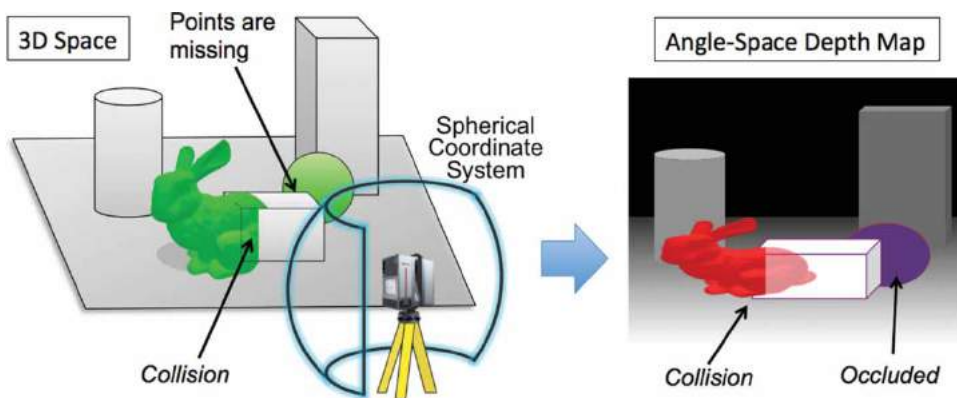


Figure 2. Collision check on angle-space depth map.

The low-resolution depth maps are used for checking collisions in obvious cases and restricting regions to be processed in the high-resolution depth map.

Figure 3 shows the process of collision detection between point-clouds and 3D models. When multiple point-clouds are captured from an engineering plant, they are separately processed on their scanner-centered coordinate systems. When 3D models are concave, they are decomposed into a set of convex shapes. Each face of a convex model is projected onto the angle-space depth map, and the depth values are compared at each pixel.

When the status is *no-collision* or *occluded* on the low-resolution depth map, the collision status can be determined only on the low-resolution depth maps, as described in the next section. The high-resolution depth map is necessary only when the status on the low-resolution depth map is *collision*. When the collision status is *occluded*, collisions have to be evaluated using other point-clouds, because points are missing at the position.

Figure 4 shows our interactive collision detection system. Point-clouds and 3D models are displayed in the main viewer. The user can drag and move 3D models in this window. The whole view of point-clouds is displayed in the bird’s-eye view. The panorama image of a point-cloud is also displayed in the window below. A panorama image is an angle-space color image, which is explained in Section 3.1.

3. Collision detection for large-scale point-clouds

3.1. Generation of angle-space depth map

Directions of laser beams are determined by the azimuth angle θ and the zenith angle ϕ , as shown in Fig. 5. Since the laser scanner measures coordinates in the equal angle intervals, it outputs coherently ordered points on the angle space (θ, ϕ) .

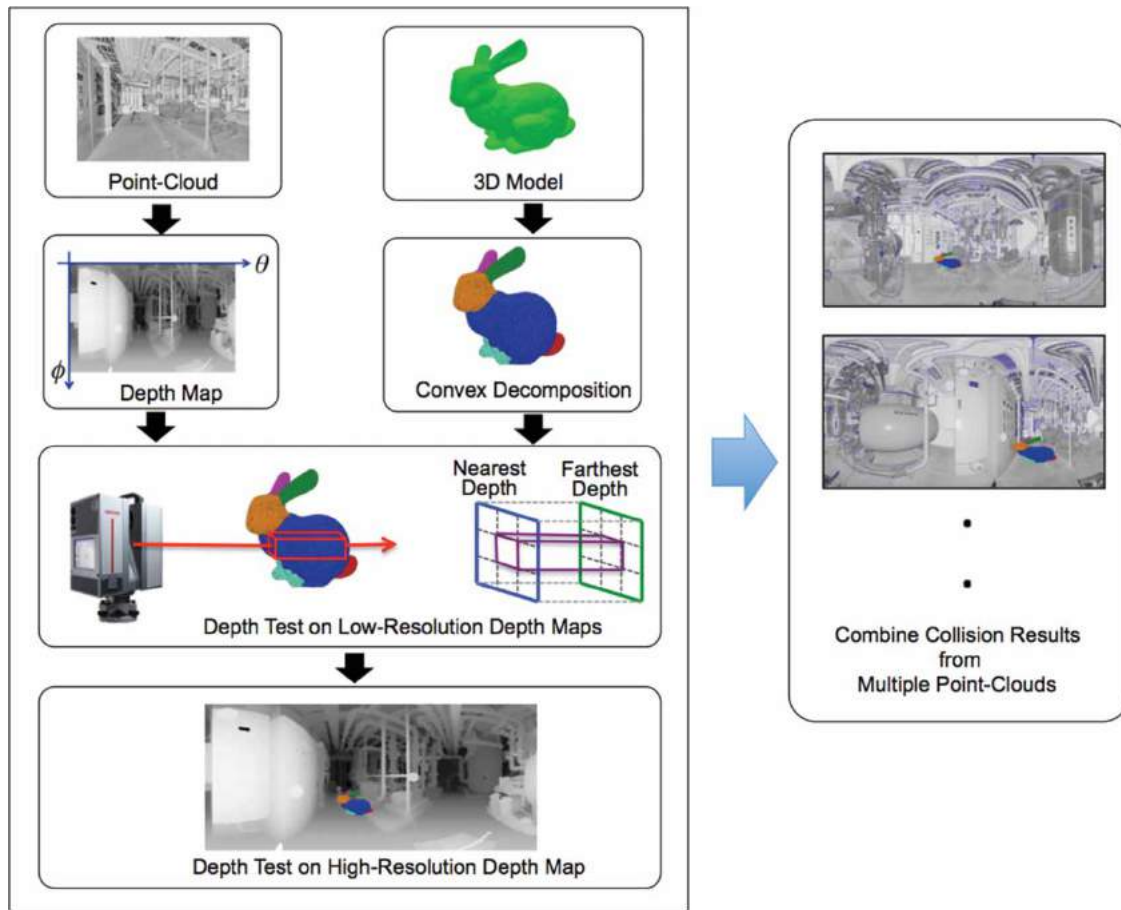


Figure 3. Process of collision detection.



Figure 4. Collision detection system.

An angle-space depth map is created by converting all (x, y, z) coordinates onto spherical coordinates (θ, ϕ, r) . Angles (θ, ϕ) are quantized into an integer coordinate (i, j) and the distance r is described at pixel (i, j) .

3.2. Collision detection by depth test

Our method evaluates collisions by projecting faces of a mesh model onto depth maps. When a 3D model is

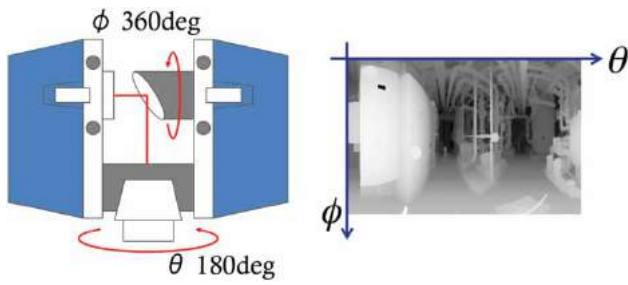


Figure 5. Angle-space depth map.

convex and closed, the line of a laser beam intersects with two points on the 3D model except in tangent cases, as shown in Fig. 6. We describe the smallest distance as d_s , the largest as d_e , and the depth value at the intersecting pixel as r . The 3D model occupies the space between depths d_s and d_e .

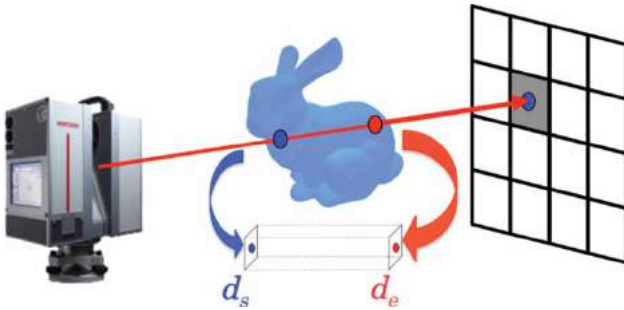


Figure 6. Two intersecting points of a laser beam with a closed mesh model.

The collision status is determined based on the relationships among depths d_s , d_e , and r , as shown in Fig. 7. When depths satisfy $d_s \leq r \leq d_e$ at more than one pixel, the 3D model collides with the point-cloud. When all depths of the model are smaller than the ones of the depth map, the 3D model does not collide with the point-cloud. Otherwise, the status is *occluded*, because the model is partly occluded from the scanner position.

This method is efficient for relatively small point-clouds, but it is time-consuming to compare depths on a huge number of pixels. One solution for improving performance is to reduce the resolution of depth maps, but low-resolution depth maps deteriorate the accuracy of collision detection. For example, if the number of points is reduced to one million, obstacles that are less than 9 cm may be overlooked at the distance of 20 m.

3.3. Collision detection by two-layer depth maps

To accelerate collision detection for large-scale point-clouds, we introduce a new collision detection technique

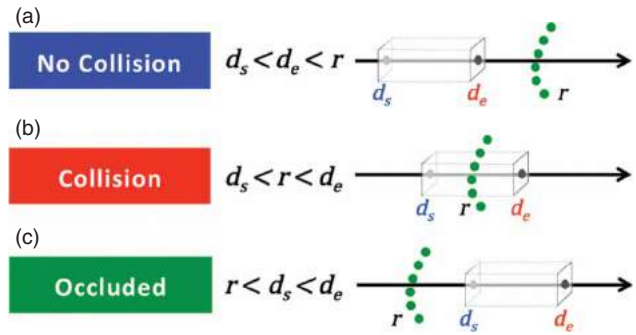


Figure 7. Collision status determined by the relationships among three depths.

based on two-layer depth maps. In this method, we prepare low-resolution depth maps as well as a high-resolution depth map. Low-resolution depth maps are used for detecting obvious *non-collision* and *occluded* cases, and a high-resolution one is used only when a 3D model collides with low-resolution depth maps.

A low-resolution depth map is created by reducing the resolution of a high-resolution depth map, as shown in Fig. 8. A high-resolution depth map is divided into blocks with $n \times n$ pixels, and the nearest and the farthest depths are selected in each block. One low-resolution map maintains the nearest depths, and the other map maintains the farthest depths.

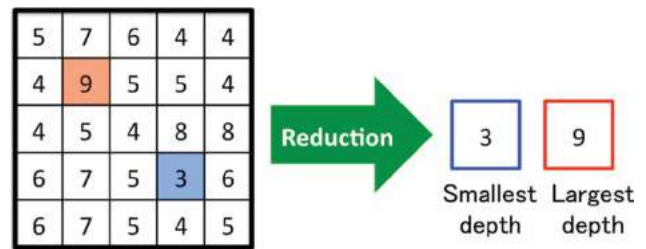


Figure 8. Generation of low-resolution depth maps.

First, the collision is evaluated on low-resolution depth maps. At pixel (i, j) , the two depths of a 3D model are described as d_s and d_e ($d_s \leq d_e$), and the near and far depths on the two low-resolution depth maps are described as r_s and r_e ($r_s \leq r_e$). Fig. 9 illustrates relationships among depths d_s , d_e , r_s , and r_e . When the depth d_e of a 3D model is smaller than the depth r_s , the model does not collide with a point-cloud (Fig. 9(a)), because the depth r_s is the nearest depth in a block on the high-resolution depth map, and therefore all points in the block do not collide with the 3D model. When $d_e \leq r_s$ is satisfied at all pixels on the low-resolution maps, the 3D model does not collide with the high-resolution depth map either.

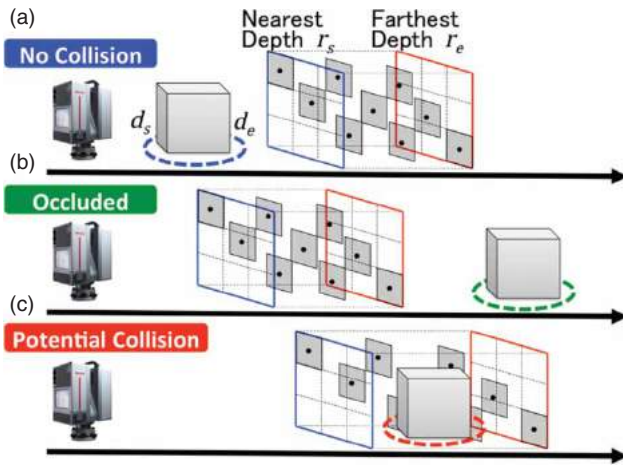


Figure 9. Collision detection on a low-resolution depth map.

When the depth d_s of a 3D model is larger than the depth r_e , the model is placed at the back of point-clouds (Fig. 9(b)). In this case, the status is *occluded*, because the model is partly occluded from the scanner position. When $d_s \geq r_e$ is satisfied at all pixels on the low-resolution maps, the status is also *occluded* on the high-resolution depth map.

When the range $[d_s, d_e]$ overlaps with the range $[r_s, r_e]$, the status cannot be precisely determined on the low-resolution depth maps (Fig. 9(c)). Then the status can be any of *collision*, *no-collision*, or *occluded* on the high-resolution depth map. Therefore, the collision status has to be evaluated using the high-resolution depth map.

In the cases in Fig. 9(a)(b), the collision statuses can be precisely determined only using the low-resolution depth maps. Then the performance can be greatly improved, because the number of pixels in the low-resolution map is much smaller than the ones in the high-resolution map. Even when our method requires a high-resolution depth map for precise collision detection, it calculates collisions only for faces that collide with low-resolution depth maps. Since the number of evaluated pixels is greatly reduced, collision detection based on the two-layer depth

maps is much more efficient than the method that uses only high-resolution depth maps.

3.4. Rough check by bounding box

The bounding box of a 3D model can further accelerate the performance of collision detection. When the collision result of a bounding box is *no-collision* or *occluded* on the low-resolution depth map, the collision detection is completed only using the bounding box.

We suppose that 3D models are placed on a flat floor. The bottom face of a bounding box is defined on the floor, and other two directions of the bounding box are calculated using the principal component analysis. In our method, a concave 3D model is decomposed into a set of convex shapes. We create a bounding box for each convex model. Bounding boxes of decomposed models are created so that their bottom faces are parallel to the floor plane for avoiding collisions with the floor.

4. Interactive collision detection system

4.1. System overview

We implemented a collision detection system based on the proposed method, as shown in Fig. 10. In this system, while the user drags a 3D model on a floor using a mouse, the system promptly changes the color of the 3D model according to the collision status. Floors are extracted in a pre-process phase.

When multiple point-clouds are captured from an engineering plant, they are separately processed for collision detection. At the first step, the user selects a point-cloud using panoramic images, and places a 3D model at the specified position. Once depth maps of a point-cloud is selected, they are continuously used for collision detection until the collision status becomes *occluded*. When the status is *occluded*, the system replaces the current depth map to the one of the next nearest point-cloud. In our system, depth maps are automatically replaced while the user drags a 3D model on a screen.



Figure 10. Collision detection system.

When a 3D model is concave, it has to be subdivided into a set of convex shapes. We implemented a cutting operation to subdivide a 3D model in our system. When the user draws a cutting-plane line on a screen, the system subdivides the 3D model along the line. Holes are automatically filled with triangles using the Delaunay triangulation.

Figure 11 shows a process flow of collision detection in our system. In the step (1), the user selects a point-cloud. Then a 3D perspective view, a top view, and a panoramic image are displayed on the window, as shown in Fig. 10. 3D models are also loaded by the user.

In the step (2)-(4), collision detection is processed on low-resolution depth maps. First, collision is roughly checked using bounding boxes. If the status is *no-collision*, collision detection is completed. If the status is *occluded*, the current depth map cannot answer whether the 3D model collides or not. Then the depth map is replaced with the next ones.

In the step (3), each convex model is tested using its bounding box. Only when the status is neither *no-collision* nor *occluded*, the system goes to the next step.

In the step (4), the system evaluates the collision of each convex model on the low-resolution depth maps. Only when the status is neither *no-collision* nor *occluded*,

the system further evaluates the status on the high-resolution map.

In the step (5), collisions are evaluated only using faces that collide with the low-resolution depth maps. The status is *collision* only when one or more convex shapes collide with a point-cloud. The status is *no-collision* only when the statuses of all convex shapes are *no-collision*. Otherwise, one or more convex models are occluded from the current scanner position. Then the current depth maps are replaced, and occluded convex models are further tested on the next depth maps by following the step (2)-(5).

4.2. Interactive collision detection

Floors are extracted from point-clouds in the pre-process phase. In our previous work [5], we proposed an efficient surface detection method for large-scale point-clouds. We use this method to detect planer regions. Large horizontal planes represent floors or ceiling in most engineering plants. When laser scanners are placed on floors, points on floors have negative z values, because the scanner position is the origin. Therefore, when large horizontal planes have negative z values, they are regarded as floors. Fig. 12(a) shows detected floor planes.

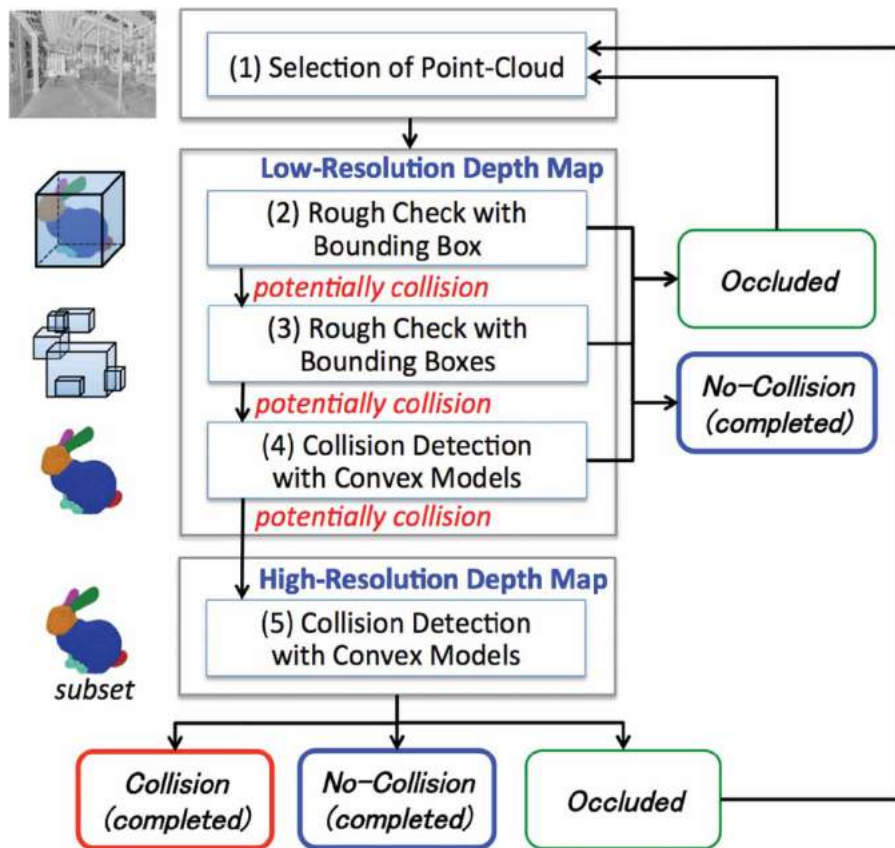


Figure 11. Our collision detection method using multiple point-clouds.

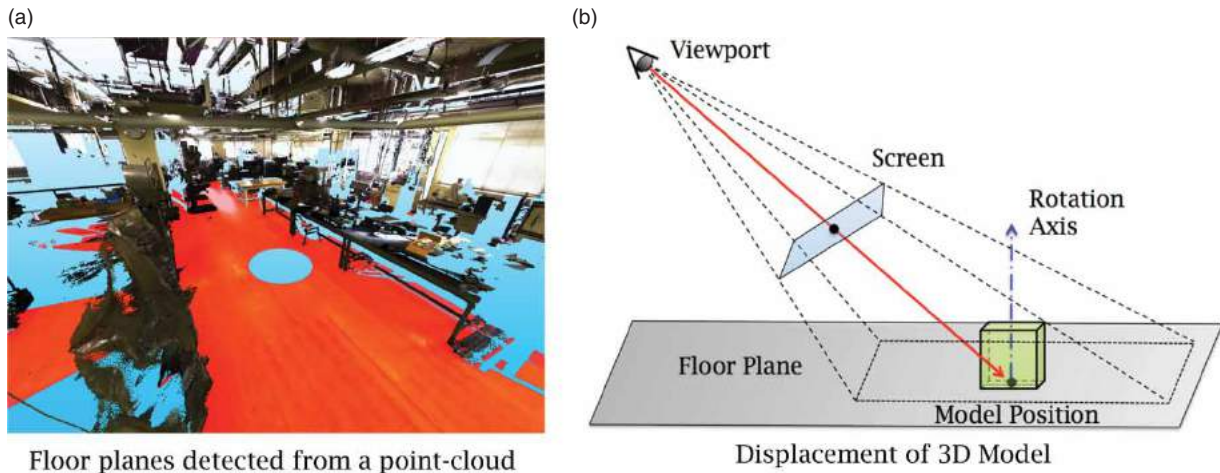


Figure 12. Interactive movement of 3D model.

In our system, the user drags 3D models on floor planes. Fig. 12(b) shows how the position of a 3D model is calculated. When the user specifies a position on a screen by clicking with the mouse, a straight line of sight is calculated. Then a 3D model is placed at the intersection point between the straight line and the floor plane. We implemented the rotation mode as well as the translation mode. In the translation mode, the user can interactively drag a 3D model on the floor. In the rotation mode, the user can rotate 3D models around the axis that is perpendicular to the floor plane, as shown in Fig. 12(b).

5. Experimental results

We evaluated our method using point-clouds of an engineering plant. We captured points using the angle resolution of 360/10000 degree. The number of points in a single point-cloud was about 40 million, and the sample 3D model consisted of 5,000 faces. The resolution of low-resolution depth maps was 720×360 pixels (0.26 megapixels), and the one of high-resolution depth maps was $11,520 \times 5,720$ pixels (66 megapixels).

We selected 121×121 (14,641) positions at the equal intervals in the range of $6 \text{ m} \times 6 \text{ m}$. We placed a 3D model at each position and measured CPU time of collision detection. CPU time was measured using a laptop PC with 2.8 GHz Intel Core i7 and 16.0GB RAM.

We compared our two-layer method with the single-layer method that uses only a high-resolution depth map. The result is shown in Tab. 1. The both methods output the same collision statuses. The ratios of *collision*, *no-collision*, and *occluded* statuses were 20%, 46%, and 34%, respectively. The average calculation time is shown for the single-layer method and the two-layer method. The results show that the two-layer method was about 6 times more efficient than the single-layer method. The frame rates were high enough to follow the mouse pointer.

Table 2 shows CPU time and frame rates when collision could be evaluated only on the low-resolution depth maps. In our experiments, the collision statuses of 72% cases could be very quickly determined only on the low-resolution maps. 27% cases required the high-resolution maps for precise collision detection.

Table 3 shows CPU time when the high-resolution depth maps were required. The performance was also efficient enough to follow the mouse. The ratios of *collision*, *no-collision*, and *occluded* statuses were 75%, 6%,

Table 2. Calculation time on low-resolution depth maps.

Status	Number of Positions	CPU Time	Frame Rate
<i>Collision</i>	4011 (27.4%)	<i>Not determined</i>	<i>Not determined</i>
<i>No Collision</i>	6479 (44.3%)	15.8 msec	63.29 fps
<i>Occluded</i>	4151 (28.3%)	16.3 msec	61.52 fps

Table 1. Comparison of calculation time.

Status	Number of Positions	CPU Time		Frame Rate	
		Single-Layer	Two-Layer	Single-Layer	Two-Layer
<i>Collision</i>	2987 (20.4%)	107.1 msec	21.6 msec	9.3 fps	46.20 fps
<i>No Collision</i>	6725 (45.9%)	105.5 msec	15.9 msec	9.5 fps	62.71 fps
<i>Occluded</i>	4929 (33.7%)	108.3 msec	17.2 msec	9.2 fps	58.20 fps

Table 3. Calculation time on high-resolution depth maps.

Status	Number of Positions	CPU Time	Frame Rate
<i>Collision</i>	2987 (74.5%)	21.64 msec	46.20 fps
<i>No Collision</i>	246 (6.1%)	20.09 msec	49.78 fps
<i>Occluded</i>	778 (19.4%)	22.09 msec	45.26 fps

and 19%, respectively. The result indicates that the statuses of 25% cases were false *collision* on the low-resolution depth maps.

6. Conclusion

In this paper, we proposed a point-based collision detection method, which could process large-scale point-clouds in a real time. In our method, collisions were evaluated very quickly using the two-layer depth maps. We developed an interactive collision detection system based on the proposed method. The experimental results showed that our method could very quickly process collision detection even for large-scale point-clouds.

In future work, we would like to investigate memory management for very large point clouds, because the data size of point-clouds becomes increasingly larger and larger. Our method can detect occluded regions. We would like to study efficient laser scanning methods for compensate missing points. We would also like to develop automatic route search methods based on our collision detection technique.

ORCID

Takeru Niwa  <http://orcid.org/0000-0001-8034-3932>

Hiroshi Masuda  <http://orcid.org/0000-0001-9521-6418>

References

- [1] dos Anjos, R. K; Pereira, J. M.; Oliveira, J. F.: Collision detection on point clouds using a 2.5+D image-based approach, *Journal of WSCG*, 20(2), 2012, 145–154.
- [2] Figueiredo, M.; Oliveira, J.; Araújo, B.; & Pereira, J.: An efficient collision detection algorithm for point cloud models, In 20th International conference on Computer Graphics and Vision, 43, 2010, 44.
- [3] Hermann, A.; Drews, F.; Bauer, J.; Klemn, S.; Roennau, A.; Dillmann, R.: Unified GPU voxel collision detection for mobile manipulation planning, *Intelligent Robots and Systems (IROS 2014)*, IEEE/RSJ International Conference on. IEEE, 2014, 4154–4160, <http://dx.doi.org/10.1109/iros.2014.6943148>
- [4] Klein, J.; Zachmann, G.: Point cloud collision detection, In *Computer Graphics Forum*, Blackwell Publishing, Inc., 23, 3, 2004, 567–576, <http://dx.doi.org/10.1111/j.1467-8659.2004.00788.x>
- [5] Masuda, H.; Niwa, T.; Tanaka, I.; Matsuoka, R.: Reconstruction of polygonal faces from large-scale point-clouds of engineering plants, *The 11th annual International CAD Conference (CAD'14)*, 2014, <http://dx.doi.org/10.14733/cadconf.2014.150-152>
- [6] Pan, J.; Sucas, I. A.; Chitta, S.; Manocha, D: Real-time collision detection and distance computation on point cloud sensor data, In *Robotics and Automation (ICRA)*, 2013 IEEE International Conference on. IEEE, 2013, 3593–3599, <http://dx.doi.org/10.1109/icra.2013.6631081>
- [7] Radwan, M.; Ohrhallinger, S; Wimmer, M.: Efficient collision detection while rendering dynamic point clouds, *Proceedings of the 2014 Graphics Interface Conference*, Canadian Information Processing Society, 2014, 25–33.
- [8] Schauer, J.; Nüchter, A.: Efficient point cloud collision detection and analysis in a tunnel environment using kinematic laser scanning and KD Tree search, *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1, 2014, 289–295, <http://dx.doi.org/10.5194/isprsarchives-xl-3-289-2014>