Interactive Collision Detection for Engineering Plants based on Large-Scale Point-Clouds

Takeru Niwa[1] and Hiroshi Masuda[2]

[1]The University of Electro-Communications, takeru.niwa@uec.ac.jp
[2]The University of Electro-Communications, h.masuda@uec.ac.jp

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, dense point-clouds of an engineering plant are very large, and it is difficult to interactively calculate collisions while the user drags objects in 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, which generated from point-clouds. Our method allows us to precisely and efficiently detect collisions in large-scale point-clouds. The experimental results show that our system can 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, and they can be used for simulating whether equipment can be safely moved without collisions when engineering plants are upgraded or renovated.

However, it is not easy to realize real-time collision detection for large-scale point-clouds. Since point-clouds of an engineering plant often consist of hundreds of millions of points, collision detection methods must handle very large point-clouds efficiently. In addition, point-clouds of engineering plants include a lot of missing potions, because occlusion cannot be avoided when many manufacturing machines are installed. Occluded 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 detecting collisions [2-4],[6],[8]. However, they did not intend to process large-scale point-clouds interactively. 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], because 2D depth tests are more efficient than 3D depth tests in many cases. 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 when the user interactively moves objects in large-scale point-clouds. 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 in occluded regions. Our method can be categorized into screen-space methods, but our depth map is defined in angle-space.

For enormously accelerating performance of collision detection, we introduce two-resolution angle-space depth maps. 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. When a point-cloud is captured using a typical laser scanner, coordinates are described 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 and they are merged onto the world coordinate system, coordinates of point-clouds are transformed using transformation matrices. We suppose that transformation matrices for registration are maintained and coordinates can be mutually converted onto the scanner-centered system and the world coordinate system.

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 points in a point-cloud are included in closed space of a 3D model, the status is "*collision*". When a 3D model is placed in front of a point-cloud, it does not collide with the point-cloud, and therefore the status is "*no-collision*". Otherwise, the status is "*occluded*", because a 3D model is placed in an occluded region, in which no points are captured because of occlusion.

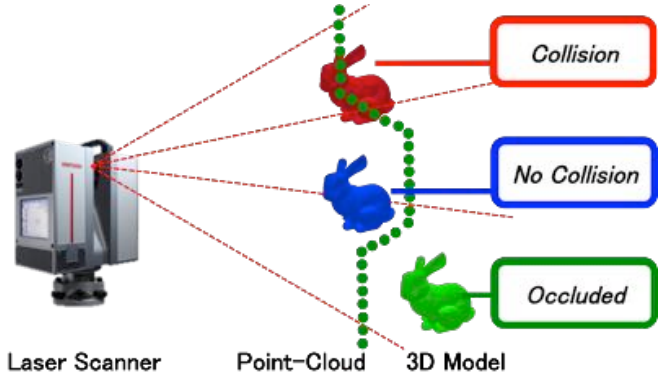


Fig. 1: Three statuses of collision detection

In our method, collisions are evaluated on a 2D angle-space depth map for efficiently processing large-scale point-clouds. Each scanner-centered coordinate (x, y, z) is converted into a spherical coordinate (θ, ϕ, r). Two angles (θ, ϕ) are quantized and an angle-space depth-map is created by projecting all points on the θ-ϕ plane. Each pixel value of the depth map is $r = \sqrt{x^2 + y^2 + z^2}$.

Collision statuses are evaluated on the angle-space depth map. Fig.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. 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.

Fig. 3 shows a process of collision detection between point-clouds and 3D models. When multiple point-clouds are captured from an engineering plant, each point-cloud is separately processed using the scanner-centered coordinates. For efficiency reason, 3D models are decomposed into a set of convex shapes, and collisions of each convex shape are evaluated. Collisions are detected using depth tests. Each face of a convex model is projected onto an angle-space depth map, and the depth values are compared at each pixel.

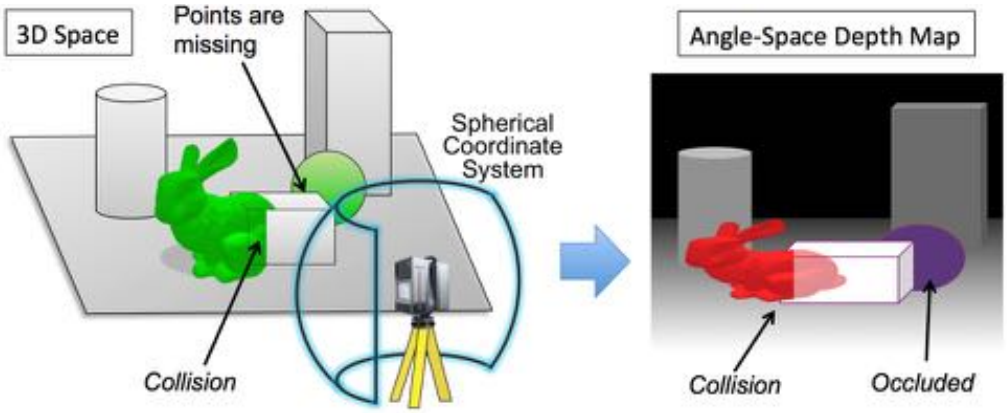


Fig. 2: Collision check on angle-space depth map

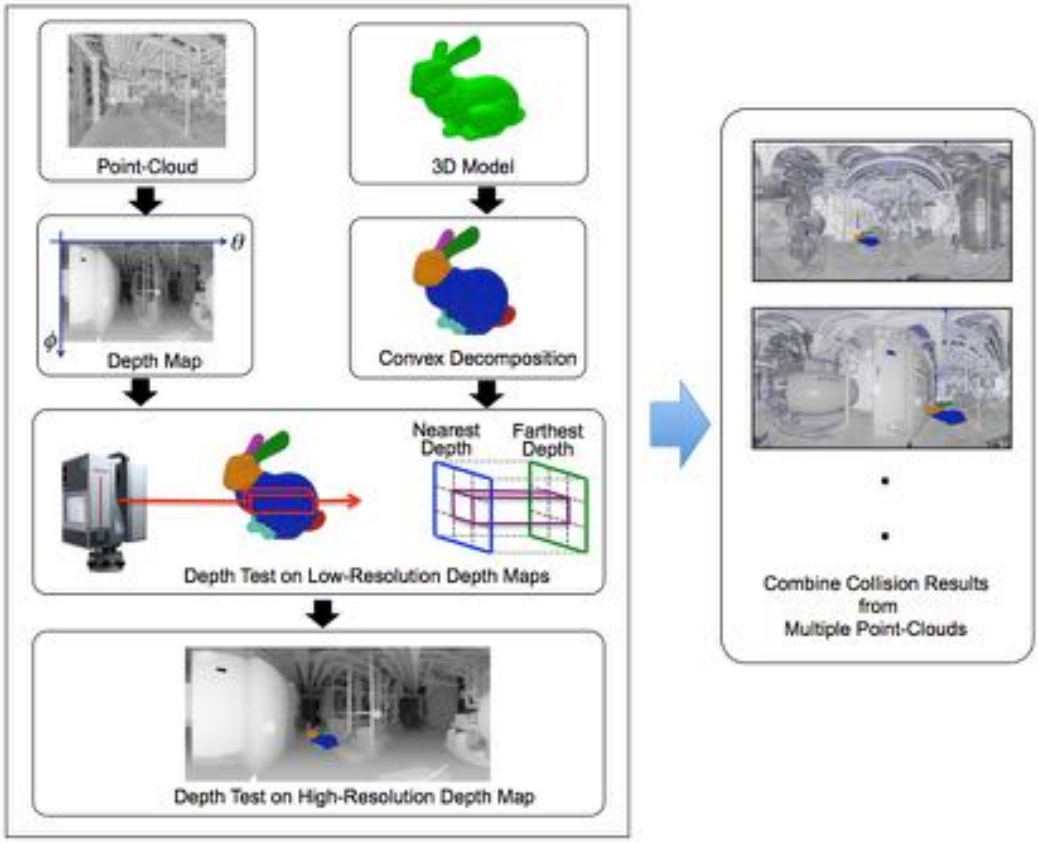


Fig. 3: Process of collision detection

The high-resolution depth map is necessary only when the status on the low-resolution depth maps is *collision*. 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. When the collision status is *occluded*, collisions are evaluated using other point-clouds, because points are missing at the 3D model position.

Fig. 4 shows our interactive collision detection system. In the main viewer, point-clouds and 3D models are displayed. 3D models can be dragged by the user in this window. Since the system automatically extracts floor planes from point-clouds, 3D models move on floors. 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. A panorama image is an angle-space color image, which is explained in Section 3.1. Since our method can very efficiently process large-scale point-clouds, collisions can be promptly evaluated while the user interactively drags a 3D model on a screen.

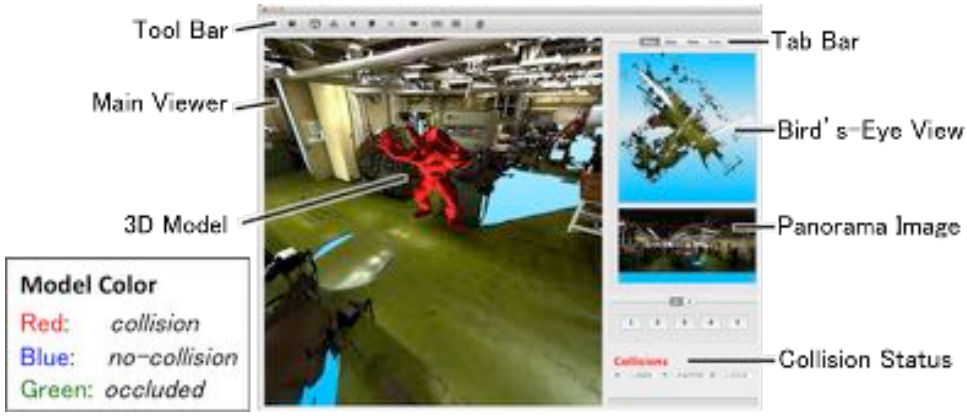


Fig. 4: Collision detection system

3 COLLISION DETECTION FOR LARGE-SCALE POINT-CLOUDS

3.1 Generation of Angle-Space Depth Map

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

An angle-space depth map can be generated by converting (x, y, z) coordinates to 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). We suppose that the width and height of a depth map are known, because points captured by most commercial laser scanners can be output in the PTX format. The first two values in this format indicate the numbers of points along the azimuth and the zenith angles. We use these numbers as the width and height of a depth map.

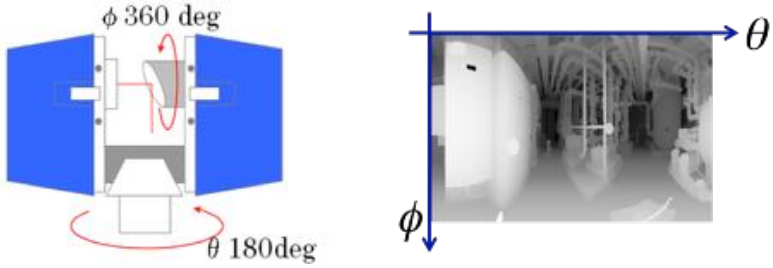


Fig. 5: Angle-space depth map

3.2 Collision Detection by Depth Test

Collisions are evaluated by projecting faces of a mesh model onto depth maps. We suppose that 3D mesh models are closed and convex. When a 3D model is not convex, they are subdivided into a set of convex shapes. Then collision of each convex model is separately evaluated.

When a 3D model is 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 of a depth map as r at the pixel. When two points on a 3D model are projected onto pixel (i,j), space between depths d_s and d_e is occupied by the 3D model.

Collision is evaluated at each pixel on a depth map. 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*, which means the model is partly occluded from the scanner position.

This depth test is efficient for relatively small point-clouds, but it is time-consuming to handle large-scale point-clouds, because depths have to be evaluated on a huge number of pixels. One solution for improving performance is to reduce the resolution of depth maps, but rough depth maps lose details of shapes and 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.

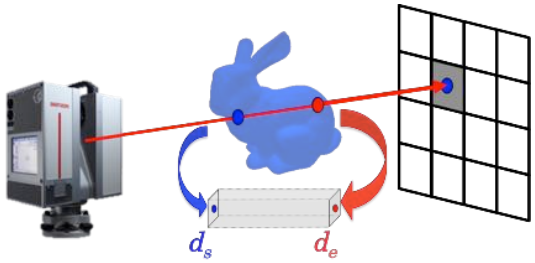


Fig. 6: Depth image defined by two rotating angles.

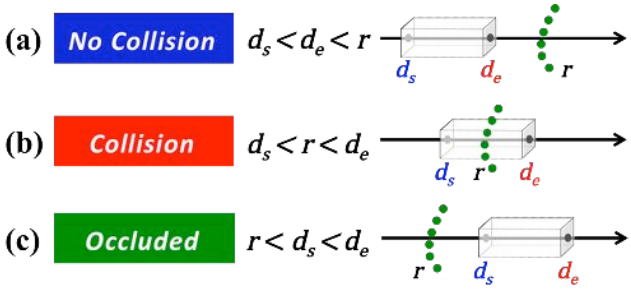


Fig. 7: Collision status determined by the relationships among three depths

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

Low-resolution depth maps are generated using a high-resolution depth map. In Fig. 8, we consider low-resolution depth maps with an $n \times n$ times smaller number of pixels. 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.

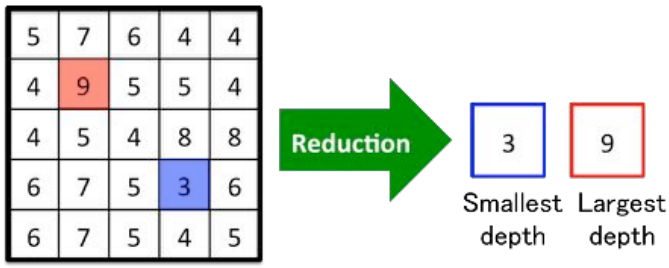


Fig. 8: Generation of low-resolution depth maps

First, collision is evaluated on low-resolution depth maps. The two depths of a 3D model at pixel (i, j) are described as d_s and d_e, and the depths on the low-resolution depth map are described as r_s and r_e, where $d_s \leq d_e$ and $r_s \leq r_e$.

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)). Since the depth r_s is the nearest depth in a block on the high-resolution depth map, all points in the block do not collide with the 3D model. Therefore, if $d_e \leq r_s$ is satisfied at all pixels on the low-resolution maps, the 3D model does not collide with the high-resolution depth map either.

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 model is partly occluded from the scanner position, and therefore the status is *occluded*. If $d_s \geq r_e$ is satisfied at all pixels on the low-resolution maps, the status is *occluded* on the high-resolution depth map also.

When the range $[d_s, d_e]$ overlaps with the range $[r_s, r_e]$, the status cannot be precisely determined on low-resolution depth map (Fig. 9(b)). The status can be any of *collision*, *no-collision*, or *occluded* on the high-resolution depth map. Then the system stores faces that collide with the low-resolution depth maps, and projects them on the high-resolution depth map.

In the cases in Fig. 9(a)(b), collision statuses can be precisely evaluated only using the low-resolution depth maps. Since the number of pixels in low-resolution maps is much smaller than the ones in the high-resolution map, the performance can be greatly improved in these cases. Even when a high-resolution depth map is required for precise collision detection, collisions are evaluated only for faces that collide with low-resolution depth maps. Since the number of evaluated pixels is greatly reduced in all cases, collision detection based on two-layer depth maps is extremely efficient compared to the method that uses only high-resolution depth maps.

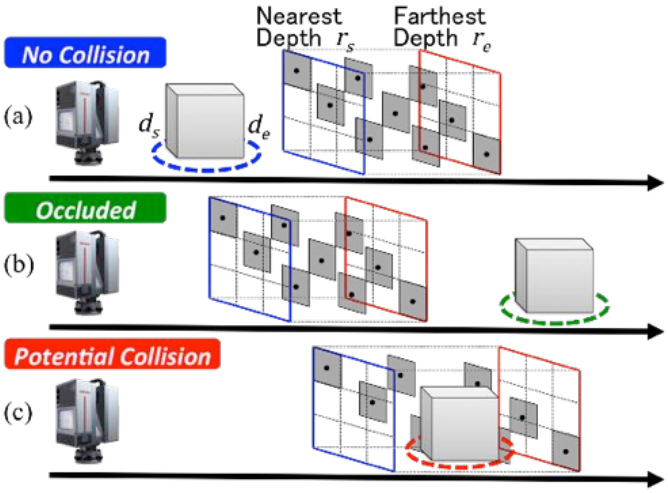


Fig. 9: Collision detection on a rough depth map

3.4 Rough Check by Bounding Box

To further accelerate collision detection, bounding boxes of 3D models can be used for rough check. When a bounding box is evaluated as *no-collision* or *occluded* on the low-resolution depth map, collision detection is completed only using the bounding box.

We suppose that 3D models are placed on a flat floor. For calculating a bounding box, the bottom face of a 3D model is transformed on the x-y plane, and the two directions of the bounding box are calculated as two eigenvectors of the principal component analysis for (x, y) coordinates of vertices in a 3D model. In our method, a concave 3D model is decomposed into a set of convex shapes. We also create bounding boxes for decomposed convex models and apply rough checks using them. Bounding boxes of decomposed models are created so that their bottom faces are parallel to the x-y plane for avoiding collisions between bounding boxes and floors.

4 INTERACTIVE COLLISION DETECTION SYSTEM

4.1 System Overview

We implemented a collision detection system, in which our collision detection method is involved. Fig. 10 shows our implemented system. In this system, when the user selects a position on a screen using a mouse, a 3D model moves on a floor, and then the collision status is promptly indicated as the color of the 3D model and a text. Floors are extracted in pre-process phases and their equations are stored in the system.

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 of point-clouds, and places a 3D model by specifying a position on a floor. 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 maps to the ones of the next nearest point-cloud. In our system, depth maps are automatically replaced while the user drags a 3D model on a screen.

In our method, a 3D model is represented as a set of convex shapes. Therefore we implemented a cutting operation, which subdivides a 3D model in our system. When the user draws a cutting-plane line on a 3D model, the system subdivides the 3D model along the line. Holes are automatically filled with triangles using the Delaunay triangulation. Then the system maintains the relationships between decomposed convex models and the original 3D model.

Fig. 11 shows conditional branching of collision detection in our system.

In the step (1), a point-cloud is specified by the user, and two low-resolution depth maps and a high-resolution depth map are determined. Then a 3D perspective view, a top view, and a panoramic image are displayed on the window, as shown in Fig. 10. A 3D model is also selected by the user.

Fig. 10: Collision detection system

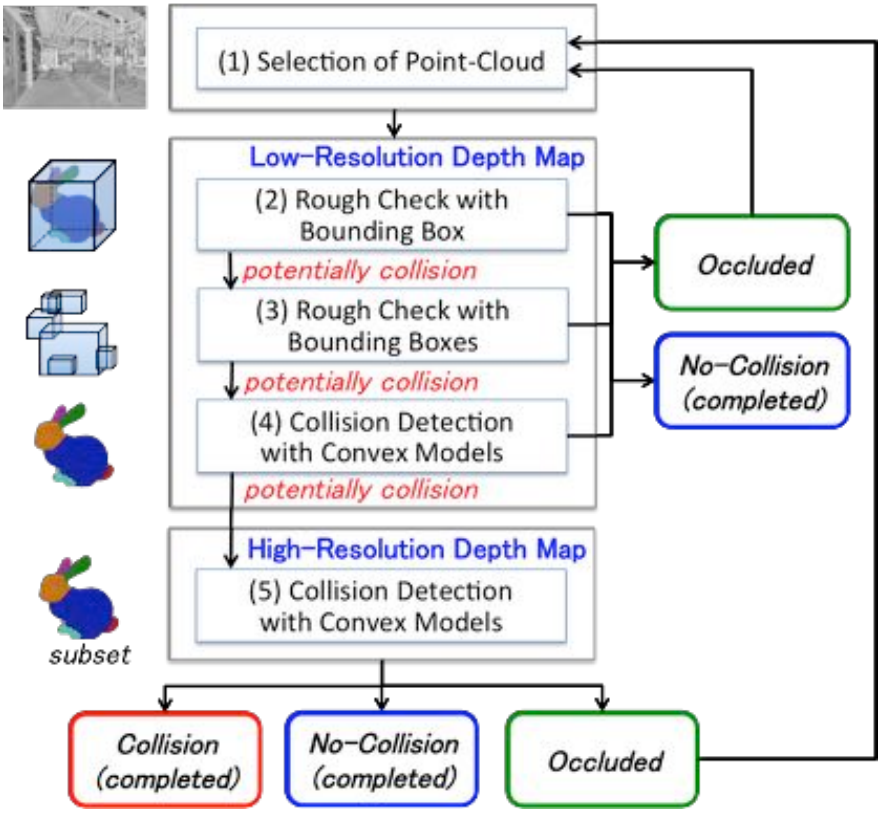


Fig. 11: Our collision detection method using multiple point-clouds

In the step (2)-(4), collision detection is processed on low-resolution depth maps. First, collision is roughly checked using a bounding box of a 3D model. If the status is *no-collision*, collision detection is completed. If the status is *occluded*, depth maps are replaced with the next ones because collisions cannot be evaluated on the current depth maps. Otherwise, the system goes to the next step. In 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 step (4). In the next step, collision of each convex model is evaluated by projecting faces on the low-resolution depth maps. Only when the status of a convex model is neither *no-collision* nor *occluded*, collision is further evaluated using the high-resolution map.

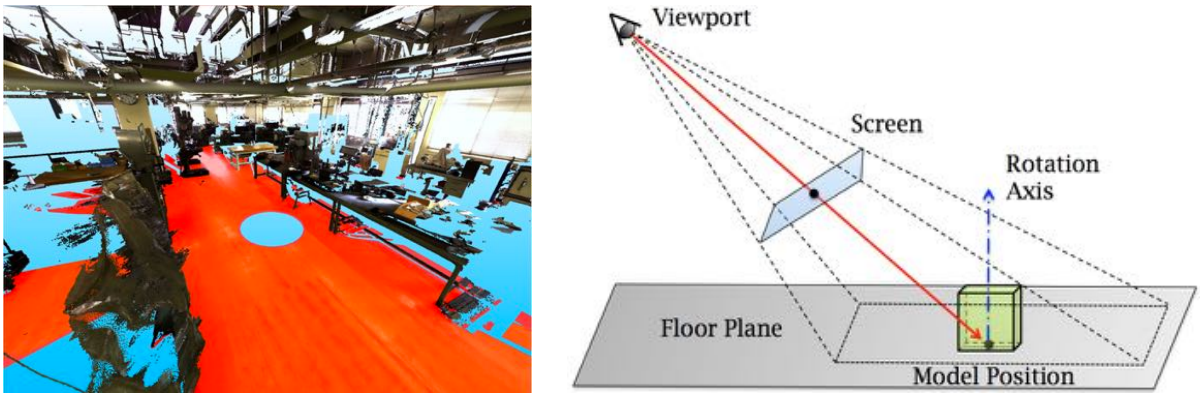
In the step (5), collisions are evaluated only using faces that collide with the low-resolution depth maps. The status is *collision* only when statuses of one or more convex shapes are *collision*. The status is *no-collision* only when statuses of all convex shapes are *no-collision*. Otherwise, one 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

In a pre-process phase of collision detection, floors are extracted from point-clouds. In our previous work [5], we proposed an efficient surface detection method for large-scale point-clouds. Our plane detection is based on this method, which extracts planes using the RANSAC method.

Large horizontal planes are floors or ceiling in most engineering plants. When laser scanners are placed on floors, the z values of points on floors are negative on the scanner-centered coordinate system. Therefore, when a large horizontal plane is detected from a point-cloud, and it has negative z values, the plane is regarded as a floor. When a floor plane is detected, other planes on the plane are regarded as floors. Fig. 12(a) shows detected floor planes.

3D models are moved on floor planes. Fig. 12(b) shows positioning of a 3D model. When the user specifies a position on a screen by clicking with the mouse, a straight line of sight can be calculated. Then a 3D model is placed at the intersection point between the straight line and the floor plane. While the user interactively drags the cursor, the model moves following intersection points. The user can also rotate 3D models by dragging the cursor. A 3D model rotates around the rotation axis that is perpendicular to the floor plane, as shown in Fig. 12(b).



(a) Floor planes detected from a point-cloud (b) Displacement of 3D Model

Fig. 12: Interactive movement of 3D model

5 EXPERIMENTAL RESULTS

We evaluated our method using point-clouds of an engineering plant, which were measured with the angle resolution of 360/10000 degree. The number of points in a single point-cloud was about 40 million. We placed a mesh model with 5000 triangles in the point-clouds. The resolution of low-resolution depth maps was 720×360 pixels (0.26 megapixels), and the one of high-resolution depth maps was 11520×5720 pixels (66 megapixels). CPU time was measured using a laptop PC with 2.8GHz Intel Core i7 and 16.0GB RAM.

For performance evaluation, we placed a 3D model at 15,000 positions on a floor, and applied collision detection at the positions. Tab. 1 shows frame rates of collision detection. In this evaluation, collisions were evaluated using a single point-cloud. When the status was *no-collision* or *occluded* on the low-resolution depth maps, collision detection was completed. Only when the status was *collision* on the low-resolution depth map, the high-resolution depth map was used for collision detection. Since the number of faces for depth check was reduced on the high-resolution depth map, the performance was efficient even on the high-resolution depth map.

The experimental results showed that our method was very efficient even when point-clouds were large-scale. We can conclude that collision detection on large-scale point-clouds can be calculated in real-time while the user drags a 3D model in large-scale point-clouds.

Status	Low-Resolution Depth Map	High-Resolution Depth Map
Collision	59.76 fps	52.39 fps
No Collision	64.17 fps	53.85 fps
Occluded	61.54 fps	56.17 fps

Tab. 1: Calculation time (fps: frame per seconds)

6 CONCLUSION

In this paper, we proposed a point-based collision detection method, which could process large-scale point-clouds in a real time. We also proposed a method for efficiently identifying whether a 3D model was placed in an occluded region. In our method, collisions were evaluated by projecting 3D models on depth maps. We showed that collisions could be precisely and quickly evaluated on two-layer depth maps. In addition, we explained our implementation of an interactive collision detection system. The experimental results showed that our method could very quickly process collision detection even for large-scale point-clouds following the user's dragging of a 3D model.

In future work, we would like to investigate memory management for very large point clouds. We would also like to develop automatic route search methods based on our collision detection technique.

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/cadconfp.2014.150-152

[6] Pan, J.; Sucan, 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