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

Authors:
Takeru Niwa, takeru.niwa@uec.ac.jp, The University of Electro-Communications
Hiroshi Masuda, h.masuda@uec.ac.jp, The University of Electro-Communications

Keywords:
Point-Clouds, Collision Detection, Geometry Processing, Point Processing, Computer-Aided Maintenance

DOI: 10.14733/cadconfP.2015.338-342

Introduction:
The state-of-the-art laser scanners make it possible to capture dense point-clouds of engineering plants. Dense point-clouds are faithful as-is 3D representation of engineering plants. They can be used for simulating whether equipment can be safely moved without collisions when engineering plants are renovated.

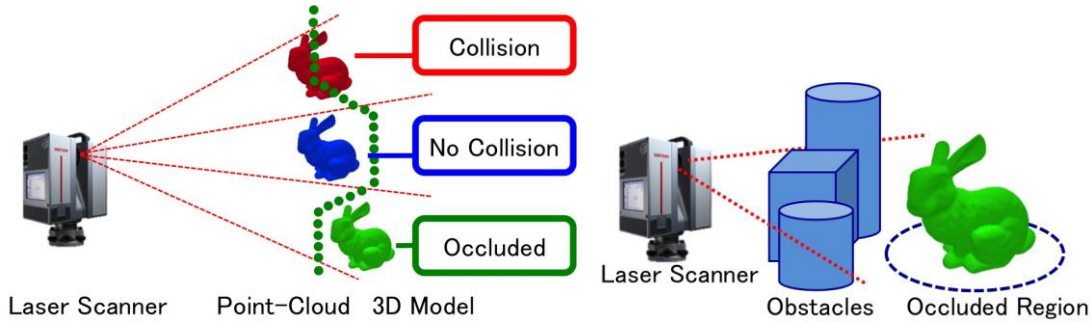
However, it is not easy to realize real-time collision detection for large-scale point-cloud, because point-clouds of an engineering plant often contain hundreds of millions of points. In addition, many places in an engineering plant cannot be measured because of occlusion. Although points are absent in occluded regions, such regions may not be free space. Conventional point-based collision detection methods [1-7] cannot handle very large point-clouds that contain missing points. In this paper, we consider real-time point-based collision detection that can solve these problems.

Collision Detection based on Depth Maps:
We suppose that closed 3D models are placed in large-scale point-clouds. For efficiently handling large-scale point-clouds, we convert point-clouds into angle-space depth maps, which maintain the depth of each point from a laser scanner. Collisions are detected on depth-maps.

Fig. 1(a) 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, as shown in Fig. 1(b).

Our method consists of the following steps.
(1) Generation of Depth Map: We first project all points captured by a single scan onto a plane and create a 2D depth map. Since the directions of laser beams are controlled by the azimuth angle θ and the zenith angle ϕ, (x,y,z) coordinates can be converted to spherical coordinates (r,θ,ϕ) without overlapping, when the origin of the coordinate system is placed at the source of laser beams. Then we can obtain a depth map defined on (θ,ϕ) plane by quantizing θ and ϕ, and describing distance r at each pixel, as shown in Fig 2.

(2) Projection of 3D models: We subdivide a 3D model into a set of convex closed triangular meshes, and project each convex mesh onto the depth map.The line of a laser beam intersects with two faces, as shown in Fig. 3. We define the smallest distance as d_s, the largest as d_e. The depth image of each triangle is calculated by interpolating depth values of three vertices of the triangle. The smallest and the largest depths of each convex mesh are used for collision detection.



(a) Three statuses of collision detection (b) An Object placed in an occluded region

Fig. 1: Collision detection between point-clouds and 3D models.

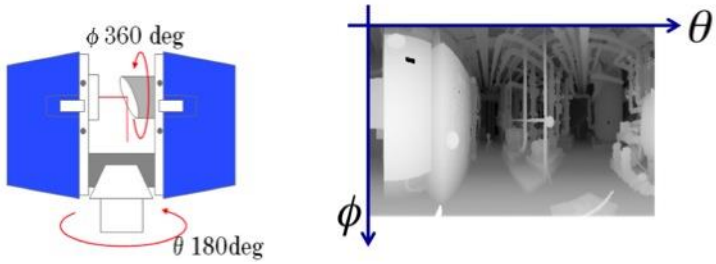


Fig. 2: Depth image defined by two rotating angles.

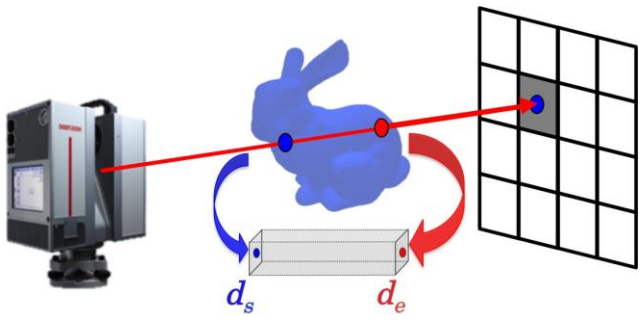


Fig. 3: Two intersecting points of a laser beam with a closed mesh model.

(3) Collision Detection on Depth Map: Collision is evaluated at each pixel of a depth map. Each convex mesh is projected on a depth map, and depths d_s and d_e are calculated at each pixel, as shown in Fig. 3.

When the depth of a point-cloud is r at the projected pixel, collision is determined based on the relationships among three depths, as shown in Fig. 4.

When the condition of *collision* is satisfied at more than one pixel, the 3D model collides with the point-cloud; when all depths of the 3D model are smaller than the ones of the depth map, the status is *no-collision*; otherwise, the status is *occluded*.

This method is efficient for relatively small point-clouds, but it is time-consuming to handle large-scale point-clouds, because a lot of pixels on dense depth maps have to be evaluated. One solution for improving performance is to reduce the resolution of depth maps, but low-resolution depth maps lose

details of shapes and decrease the accuracy of collision detection. Therefore, we consider a new collision detection method based on two-layer depth maps.

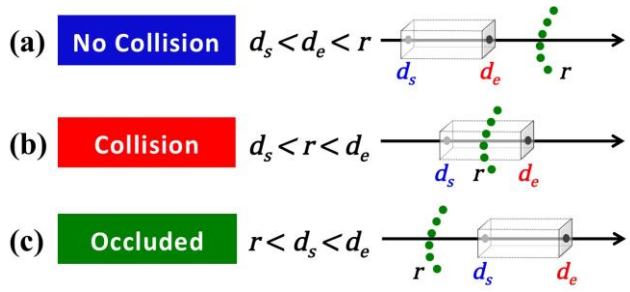


Fig. 4: Relationship among depth values.

Improvement of Performance using Two-Layer Depth Maps:

To improve performance of collision detection, we introduce two-layer depth maps. A low-resolution depth map is used for obvious cases, and a high-resolution one is used only when 3D models collide with low-resolution collision depth maps. A low-resolution depth map is created by reducing the resolution of a high-resolution depth map. Each pixel of a low-resolution depth map has two values, which are the smallest and the largest depths in a rectangular region on a high-resolution depth map, as shown in Fig. 6.

At the first step, collision is evaluated using a low-resolution depth map. We describe the smallest and the largest depths at a pixel on a low-resolution depth map as r_s and r_e. Fig. 7 illustrates the relationships among depths d_s, d_e, r_s, and r_e. When the range $[d_s, d_e]$ of a 3D model is smaller than the depth r_s, the model does not collide with a point-cloud (Fig. 7(a)). When they are larger than the depth r_e, the model is in an occluded region (Fig. 7(b)). These two cases can be precisely evaluated only using a low-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 a low-resolution depth map. Then the pixel is expanded to the original pixels, and the collision is evaluated using a high-resolution depth map.

In two-layer collision detection, many cases can be quickly evaluated on a low-resolution depth map. Even when a high-resolution depth map is required for precise collision detection, our method calculates collisions only for faces that collide with low-resolution depth maps. Since the number of evaluated pixels is greatly reduced in all cases, our two-layer method is extremely efficient compared to the method that uses only high-resolution depth maps.

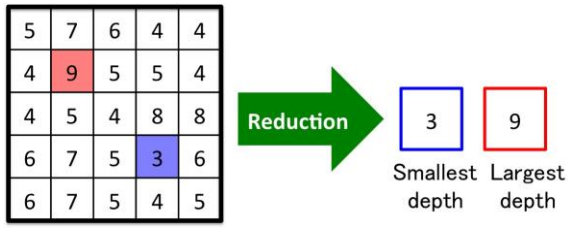


Fig. 6: Generation of low-resolution depth maps.

Experimental Results:

We evaluated our method using actual point-clouds of an engineering plant. We captured points with the angle resolution of 360/10000 degree. The number of points in a single point-cloud was about 40 million. We moved a 3D model with 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 11520×5760 pixels. CPU time was measured using a laptop PC with 2.8 GHz Intel Core i7 and 16.0GB RAM.

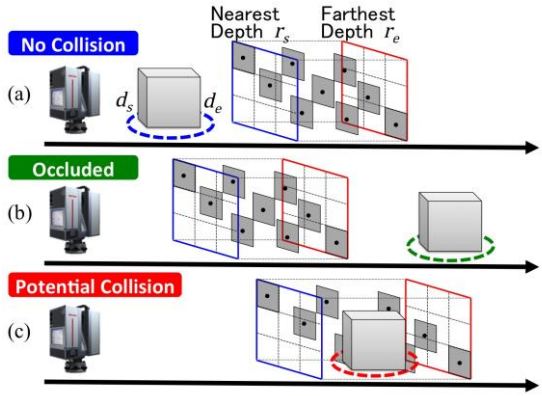


Fig. 7: Collision detection on a low-resolution depth map.

We selected 121 x 121 (14,641) positions at the equal intervals in the range of 6m x 6m, and placed a 3D model at each position. We compared our two-layer method with the single-layer method that uses only a high-resolution depth map. The both methods output the same collision statuses. The 3D model collided at 20% of positions, did not collide at 46% of positions, and was occluded at 34% of positions. The average calculation time is shown in Tab. 1. The results showed that the two-layer method was about 6 times faster than the single-layer method. The frame rates were high enough to follow the mouse pointer.

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

Tab. 1: Comparison of calculation time.

Implementation of Interactive Collision Detection System:
We implemented an interactive collision detection system, in which our proposed method is involved. Fig. 8 shows our system. The user can drag and move 3D models in this window. Since the system automatically extracts floor planes in pre-processing phase, 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. Since our method can very efficiently process large-scale point-clouds, collisions can be promptly evaluated while the user interactively drags 3D models on a screen.

Conclusions:
In this paper, we proposed a method for detecting collisions based on two-layer depth maps. We showed that our method could precisely evaluate *no-collision* and *occluded* cases only on low-resolution depth maps. We also showed our method was very efficient even when high-resolution depth maps were required for precise collision detection. The experimental results showed that out method could very quickly process collision detection even for large-scale point-clouds.

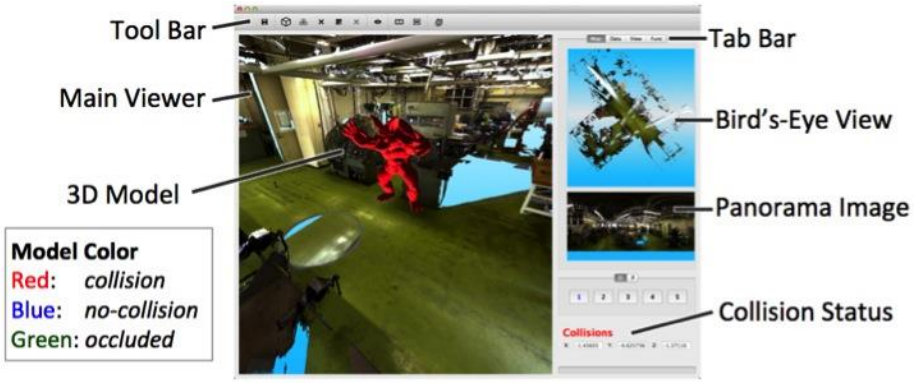


Fig. 8: Collision detection system.

References:

[1] Hermann, A.; Drews, F.; Bauer, J.; Klemn, S.; Roennau, A.; Dillmann, R.: Unified GPU voxel collision detection for mobile manipulation planning, International Conference on Intelligent Robots and Systems, 2014, 4154-4160, http://dx.doi.org/10.1109/iros.2014.6943148

[2] 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

[3] Pan, J.; Sucan, I. A.; Chitta, S.; Manocha, D: Real-time collision detection and distance computation on point cloud sensor data, In 2013 IEEE International Conference on Robotics and Automation, 2013, 3593-3599. http://dx.doi.org/10.1109/icra.2013.6631081

[4] 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.

[5] 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

[6] 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.

[7] 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.