RGB-D SLAM based Incremental Cuboid Modeling

Masashi Mishima[1], Hideaki Uchiyama[1], Diego Thomas[1], Rin-ichiro Taniguchi[1], Rafael Roberto[2], João Paulo Lima[2,3], and Veronica Teichrieb[2]

[1] Kyushu University, Japan
{mishima,uchiyama,thomas}@limu.ait.kyushu-u.ac.jp
rin@kyudai.jp
http://limu.ait.kyushu-u.ac.jp/
[2] Universidade Federal de Pernambuco, Brazil
{rar3,jpsml,vt}@cin.ufpe.br
[3] Universidade Federal Rural de Pernambuco, Brazil
jpsml@cin.ufpe.br

Abstract. This paper present a framework for incremental 3D cuboid modeling combined with RGB-D SLAM. While performing RGB-D SLAM, planes are incrementally reconstructed from point clouds. Then, cuboids are detected in the planes by analyzing the positional relationships between the planes; orthogonality, convexity, and proximity. Finally, the position, pose and size of a cuboid are determined by computing the intersection of three perpendicular planes. In addition, the cuboid shapes are incrementally updated to suppress false detections with sequential measurements. As an application of our framework, an augmented reality based interactive cuboid modeling system is introduced. In the evaluation at a cluttered environment, the precision and recall of the cuboid detection are improved with our framework owing to stable plane detection, compared with a batch based method.

Keywords: Geometric shape, cuboid, incrementally structural modeling, point cloud

1 Introduction

Owing to the advance of visual odometry and simultaneous localization and mapping (SLAM), the automated control of cars, drones and robots has been achieved by generating a point cloud based map. Although the localization can be performed by using the map, the map does not represent semantics in the environment. For 3D scene understanding, it is important to convert a point cloud into an object-level representation. Planes, cylinders and spheres are examples of a parametric object representation. The recognition of such primitive shapes is an important process for obstacle avoidance and object grasping [21].

A cuboid is also considered as an informative shape representation because there exist many cuboids in our environment. For instance, delivery boxes used in logistics and product packages in markets can be represented by cuboids. To achieve automated robot manipulation in such environments, the techniques to recognize cuboid objects

are often required. In the literature, the cuboid detection has been performed by using an RGB image [6, 7, 1, 26, 8] or a point cloud generated with an RGB-D image or LIDAR [18, 20, 13, 9, 10, 19, 5, 16]. Generally, these methods are based on an off-line batch processing such that the recognition is performed only with a single observation. Because of noisy observations, they often suffer from both false positives and false negatives. To suppress the false detections, an on-line sequential approach is investigated in our framework because it can incorporate multiple observations with temporal filtering.

In this paper, we propose a framework for incremental cuboid modeling combined with RGB-D SLAM. At every frame, planes are incrementally reconstructed from points clouds acquired from an RGB-D SLAM based approach [17], and used as input to our framework. Then, the planes are clustered to compose a cuboid by analyzing three plane positional relationships; orthogonality, convexity, and proximity. To accurately reconstruct a cuboid, a cluster of three perpendicular planes is first selected, and their intersection is computed [4]. By determining three perpendicular cuboid edges from both the intersection and the normal vectors of each face, the width, height and depth are finally computed. Since the plane parameters are incrementally updated [17], the positional relationships are analyzed in every frame not only for newly-detected planes but also for previously-detected cuboids. False detections can be suppressed with this sequential processing such that a falsely-detected cuboid face can be replaced with a correct one. Also, a new cuboid face can be assigned to a previously-detected cuboid as a forth one. As another advantage of our incremental approach, we introduce an interactive cuboid modeling system to assist users to reconstruct cuboids with augmented reality(AR) based affordance. In the evaluation, the accuracy of our framework was quantitatively evaluated by using some boxes with their ground truth sizes. Also, the comparison between a batch based method and our incremental one was investigated to show the effectiveness of our incremental approach at a cluttered environment. Finally, the computational cost was investigated to show that our framework can run in real time at a room-scale environment. The contributions of our paper are summarized as follows. ra

- A cuboid reconstruction is performed by searching three perpendicular planes and computing the intersection of the planes.
- A framework for incremental cuboid modeling based on cuboid detection and mapping is proposed.
- An application for AR based interactive cuboid modeling is presented.

2 Related work

The cuboid detection has been investigated in semantic 3D scene understanding. In this section, we review the literature from the aspects of devices used for the detection.

Recognizing cuboids from a single RGB image has been proposed [6, 7, 1, 26, 8, 3]. Hedau *et al*.reconstructed a cuboid based room layout by using vanishing points [6, 7]. First, wall, ceiling and furniture contours were extracted from an input image, and then vanishing points were estimated from orthogonal three straight lines. Finally, a bounding box was aligned to a rectangular area to recognize a cuboid object. Del *et al*.proposed to use the Manhattan world property such that many surfaces in a room

were parallel to three principle ones [1]. This assumption is valid only when cuboids are placed on a floor and parallel to walls. Xiao *et al*.proposed to first detect vertices on a cuboid based on histograms of oriented gradients, and then detect a cuboid by finding connected edges [26]. Hejrati and Ramanan investigated the performance of several feature representations for categorizing cuboid objects [8]. Dwibedi *et al*.proposed a deep learning based region proposal method for the cuboid detection [3]. Basically, the cuboid detection using a RGB image is an ill-posed problem, and the accuracy is largely degraded under occlusions.

A point cloud acquired from RGB-D images or LIDAR has also been used for the cuboid detection [18, 20, 15, 13, 9, 10, 19, 5, 16]. Shape descriptors for arbitrary 3D objects were proposed for object classification including cuboids [18, 20, 19]. For indoor environments, the prior knowledge of a room layout was incorporated to globally optimize the object arrangement including cuboids in the room [13, 19]. To detect buildings as cuboids, a closed polyhedral model is searched from planes detected in a point cloud [15]. An optimization based approach was proposed by designing a cost function with surfaces, volumes and their layout to detect cuboids in an RGB-D image [9, 10, 5]. Compared with the approach using a RGB image, the one using a point cloud can provide the size and pose of a cuboid in a scene. However, it still suffers from the false detections in the presence of sensor noises and registration error. To improve the stability and accuracy of the cuboid detection, we propose an incremental approach by fusing multiple measurements captured from different viewpoints without using any constraint on the object arrangement.

3 Overview

We start by explaining the main steps of our algorithm. First, a plane map, which is composed of oriented planes, is incrementally generated from point clouds, and used as input to our framework. This reconstruction process is based on an existing method [17] that applies a shape detection method [22] to incoming point clouds acquired from RGB-D SLAM [12]. This method can incrementally reconstruct accurate parametric shapes including planes, and largely contribute to our stable cuboid modeling.

From the plane map, a cuboid map, which is composed of cuboids with positions, poses, and sizes, is generated. As described in Section 4, cuboid faces are detected among a group of planes by cuboid check based on analyzing plane positional relationships. A cuboid is a convex polyhedron comprising six quadrilateral faces. Also, the adjacent faces of a cuboid are perpendicularly connected. By analyzing these relationships, cuboids can be detected. The procedure of the cuboid detection is illustrated in Figure 1. First, the orthogonality of all the pairs of two planes in the plane map is investigated by brute force searching. Next, a pair of two perpendicular planes is selected to search their third plane by using the cross product of the two plane normals. Finally, the proximity between the planes is checked. When a set of these three planes passes the cuboid check, the planes are classified as composing a cuboid. By computing the intersection of the planes, the position, pose, and size of a cuboid are determined.

To generate an accurate cuboid map, an incremental reconstruction process is proposed, as described in Section 5. At every frame, the status of the planes in the plane

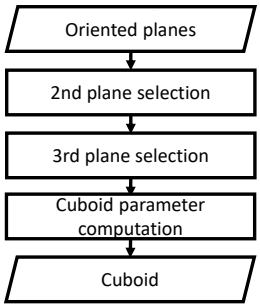


Fig. 1: Cuboid detection. From oriented planes, two planes are first selected as a plane pair by checking their positional relationships. Then, the third plane perpendicular to the pair is searched by using the cross product of the two plane normals. Finally, the cuboid shape parameters are computed from these three planes.

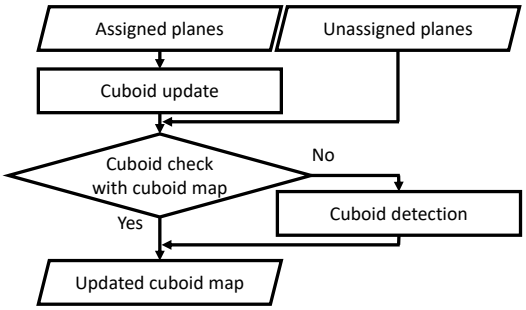


Fig. 2: Cuboid mapping. The status of the planes in the plane map is classified into planes assigned to cuboids or not. For the assigned planes, the cuboid check is performed for the cuboids in the map in every frame, as cuboid update. For the unassigned planes, the cuboid check with the cuboids in the map is first performed, and then the cuboid detection is performed if necessary.

map is classified into planes assigned to cuboids and unassigned ones, as illustrated in Figure 2. The cuboid check is performed for the cuboids in the cuboid map to check the positional relationships of the cuboid faces in every frame because their parameters are incrementally updated [17], This process is specifically referred to as cuboid update. For the unassigned planes, the cuboid check with the cuboid faces in the cuboid map is first performed so that the faces in the cuboid map can be replaced with new planes or an undetected cuboid face such as a fourth plane can be assigned to a cuboid in the cuboid map. Then, the cuboid detection is performed for the remaining unassigned planes. This incremental process allows users to make modeling succeed with AR based affordance.

4 Cuboid detection

Next, we explain the detail of detecting a cuboid from planes. The first process is to select two perpendicular planes by analyzing the positional relationships. The second process is to search the third plane by using the cross product of the two plane normals. After three perpendicular faces are determined, cuboid shape parameters are also determined by computing the intersection of the three planes.

4.1 Second plane selection

In this process, sets of two planes composing a cuboid are searched in a brute force manner. An i-th plane in the plane map is parameterized with the center of mass $\boldsymbol{p}_i$ and the normal vector $\boldsymbol{n}_i$ [17]. First, the inner products between a target plane and all of the other planes are computed, as orthogonality check. A plane is selected if the angle computed from the inner product is perpendicular with an error tolerance (e.g. 5 degrees). However, the orthogonality is not sufficient for the cuboid detection because there are two possibilities of the positional relationship between two planes; concave and convex. Also, a plane can be selected even if it is far from the target plane and does not compose a cuboid in the environment. Therefore, in the latter processes, these criteria are considered to select an appropriate plane.

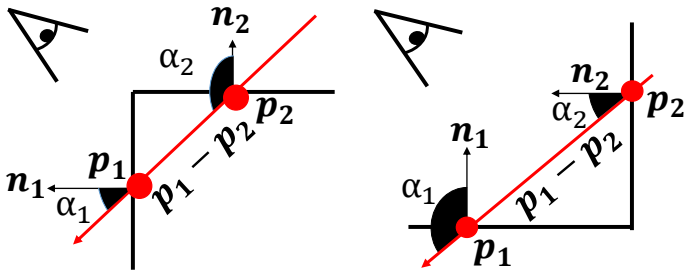
For the plane selected by the orthogonality check, the convexity with the target plane is analyzed by using [25], as convexity check. If the relationship between two planes is concave, they do not compose a cuboid because we assume that only outer cuboid faces are captured. Since each plane has the center of mass and the normal vector, the convexity can be computed from them. In Figure 3, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are the normal vectors, and $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are the centers, and α_1 and α_2 are the angles between a vector $\boldsymbol{p}_1 - \boldsymbol{p}_2$ and each normal vector $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$, respectively. When α_1 is smaller than α_2, the relationship between two planes is convex. Otherwise, the relationship is concave. Therefore, a plane is selected when it satisfies this convex condition: $\alpha_1 < \alpha_2$.

After performing both orthogonality and convexity checks, there may be multiple candidates for the second plane of a cuboid. In this case, the plane closest to the target plane is finally selected. The distance between each candidate and the target plane is computed by using the center of mass, as proximity check. All of these checks between two planes are referred to as cuboid check.

4.2 Third plane selection

From the two perpendicular planes, it is possible to infer a cuboid by using the bounding box for both planes. However, the inference may be incorrect because 3D edge regions of a cuboid cannot normally be degraded in depth images. To accurately reconstruct a cuboid, three perpendicular planes are used to determine cuboid shape parameters.

First, the normalized normal vectors for all of the planes in the plane map are indexed by using a kd-tree, as a plane normal space for fast approximated nearest neighbor searching. Then, the cross product of the two perpendicular planes is computed, and is used as a query to the kd-tree to search the third plane of a cuboid. In other words,



(a) convex($\alpha_1 < \alpha_2$) (b) concave($\alpha_1 > \alpha_2$)

Fig. 3: Convexity check. The convexity is analyzed by using the centers of mass and the normal vectors [25].

planes orthogonal to both of the two planes are searched in the space. By using the radius search in the kd-tree with a threshold (e.g. 0.1 for L2 norm between two vectors), the candidates for the third plane are retrieved. Then, the convexity with each of the two planes is checked for each candidate. Finally, the third plane is selected from the candidates according to the proximity check.

4.3 Cuboid parameter estimation

To reconstruct an accurate cuboid shape, the shape parameters are computed from the three perpendicular planes. In our framework, the parameters are the origin vertex position, three edge directions from the origin, and their lengths. In Figure 4, π_i is an ith plane, $\boldsymbol{n}_i$ is the normal vector of π_i, and $\boldsymbol{p}_i$ is the center of mass, and $\boldsymbol{p_o}$ is the intersection of the three perpendicular planes. First, $\boldsymbol{p_o}$ is computed by using [4] as follows.

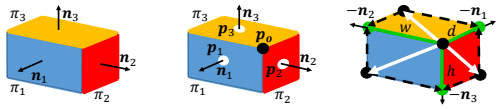
$$\boldsymbol{p_o} = \frac{(\boldsymbol{p_1} \cdot \boldsymbol{n_1})(\boldsymbol{n_2} \times \boldsymbol{n_3})}{(\boldsymbol{n_1} \times \boldsymbol{n_2}) \cdot \boldsymbol{n_3}} + \frac{(\boldsymbol{p_2} \cdot \boldsymbol{n_2})(\boldsymbol{n_3} \times \boldsymbol{n_1})}{(\boldsymbol{n_1} \times \boldsymbol{n_2}) \cdot \boldsymbol{n_3}} + \frac{(\boldsymbol{p_3} \cdot \boldsymbol{n_3})(\boldsymbol{n_1} \times \boldsymbol{n_2})}{(\boldsymbol{n_1} \times \boldsymbol{n_2}) \cdot \boldsymbol{n_3}} \quad (1)$$

The intersection can be used as the origin of a cuboid to describe the shape parameters. After determining the intersection, the edge directions from the intersection can automatically be determined because they correspond to the plane normal vectors.

To determine the size of a cuboid, the length of each edge is computed by projecting the points on a plane onto the edge as

$$length = \max_i \{(\boldsymbol{x}_i - \boldsymbol{p}_o) \cdot \boldsymbol{n}\} \quad (2)$$

where $\boldsymbol{n}$ is the normalized edge direction vector, and $\boldsymbol{x}_i$ is an ith point in the plane. This equation represents that the points on the plane sharing the edge are projected onto the edge, and the furthest point from the intersection on the edge is selected to compute the length. Since each edge is shared by two planes, the average of two lengths is used as a final result. It should be noted that the plane numbers i can be arbitrary determined for the three perpendicular planes.



(a) Three planes (b) Intersection (c) Edge and size computation

Fig. 4: Cuboid parameter estimation. The intersection of three perpendicular planes is first computed by using the centers of mass and the plane normal vectors. Edges of a cuboid are then determined by using the plane normal vectors. The width, height and depth are finally computed from the point projection from a plane to an edge.

5 Cuboid mapping

In the plane map, the status of planes can be divided into two categories; planes assigned to one of the cuboids in the cuboid map, and the rest. The first category is referred to as assigned planes, and the other is unassigned planes. When a new plane appears in the plane map, it is first considered as an unassigned plane. As illustrated in Figure 2, the process for each plane is different according to the status. In this section, we explain the detail of the cuboid mapping.

5.1 Cuboid update

To reduce the false detections, the cuboid check is performed for the cuboid faces in the cuboid map. While capturing only a part of a plane, a cuboid may be wrongly detected as a false positive or a false negative at a time due to the incomplete measurement. Therefore, in every frame, the cuboid check is applied to the cuboids in the map, and cuboid shapes are updated such that some cuboids disappear or others are refined.

This cuboid update is useful for the visual feedback to users. Normally, users do not understand the best way to capture a scene and when to finish capturing it. By using an incremental approach, false positives and false negatives are visualized in an on-line manner. This helps users to complete the modeling because they can understand the progress.

As an alternative approach, it is possible to apply a batch based method to a point cloud in every frame. However, the computational cost at a frame increases according to the size of the point cloud. Also, it is redundant to search cuboids in the map in every frame because the detection result at a frame can be useful at the next frame. In terms of the computational efficiency, the incremental approach is appropriate for on-line systems.

5.2 Cuboid check with cuboid map

The unassigned planes in the plain map contain both newly-detected planes and previously-detected planes that are not assigned to the cuboids. For those planes, the cuboid check with respect to all the cuboids in the map is first performed. A cuboid face in the map

(a) Incomplete (Top is missing) (b) Completed (Top is captured)

Fig. 5: Interactive cuboid modeling. Yellow regions and blue ones represent incomplete and completed, respectively. A box is represented by yellow in (a), and the color becomes blue in (b) after the user completely captures it.

can be replaced with an unassigned plane when an unassigned plane passes the cuboid check, its normal vector is the same as the cuboid face one, and it is more proximate than the cuboid face. Also, the plane is assigned to the cuboid if it passes the cuboid check, and it corresponds to a missing face in the cuboid. After this process, the cuboid shape parameters are updated.

For the remaining unassigned planes, the cuboid detection is performed, as described in Section 4. After a set of three perpendicular planes is detected, a new cuboid is generated and inserted into the cuboid map.

6 Interactive cuboid modeling

Since 3D modeling using a camera is not an easy task for non-experts, interactive techniques have been proposed [24]. For instance, the result of the 3D modeling can be easily modified on user interfaces [23, 27]. Also, the incompleteness of the modeling is visualized by showing a 2D slide of a point cloud for modeling a room [2] or showing an example for modeling an object [11]. Here, we introduce a simple but effective affordance for modeling a cuboid.

As illustrated in Figure 5, the points on planes are overlaid with some colors. Blue regions and yellow ones represent detected cuboids as completed ones and two perpendicular planes as incomplete ones, respectively. In other words, the color represents the modeling progress. To model a cuboid, the user's task is to find yellow regions and then capture the remaining plane where colored points are not overlaid, as illustrated in Figure 5a. This corresponds to the instruction for the users. Since the users are induced to capture the remaining plane from the visualization, this interaction can be regarded as AR based affordance. After the user successfully captures the cuboid, the color of the cuboid becomes blue, as illustrated in Figure 5b. Owing to the incremental approach, it is possible to develop this type of interactive modeling systems.

7 Evaluation

To evaluate the performance of our proposed framework, we first prepared RGB-D image sequences capturing multiple boxes as our dataset because only a dataset for single views was developed in the literature [26] and there is no dataset with RGB-D sequences containing cuboid ground truth annotations. For the camera, a Kinect V1 sensor was used, and therefore the boxes were set up in the indoor environments. The size of each box was measured by a ruler as a ground truth.

For the evaluation criterion, the accuracy of each estimated cuboid shape was investigated by comparing the result with its ground truth. To investigate the effectiveness of our proposed method, a batch based cuboid detection in a point cloud was implemented as a benchmarking method, and its result was compared with our result. Finally, the computational cost of each process was measured.

7.1 Cuboid shape estimation

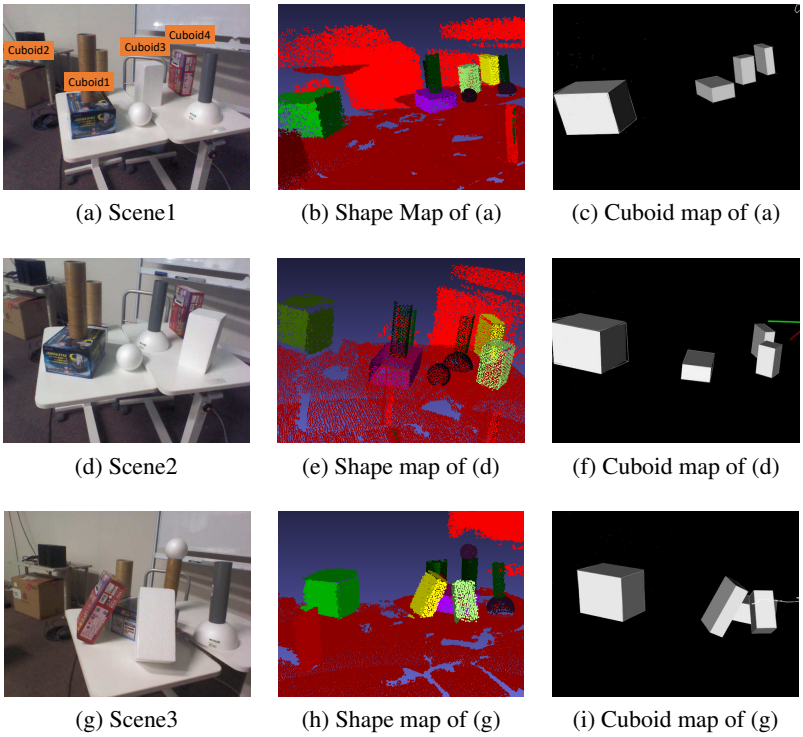
As illustrated in Figure 6, three scenes were designed such that four cuboids with different sizes were arranged on a table and also other objects were placed as obstacles. In the Scene 1, the cuboids were rotated to face to the same direction. In the Scene 2, the cuboids were rotated not to face to the same direction except for the top face. In the Scene 3, two cuboids were inclined onto a cuboid. In all the scene, one cuboid was placed far from the table. In Figure 6, the first column represents an example image of each scene, the second one does the shape map [17] drawn with different colors per cuboid, and the third one does our cuboid map. At each scene, an RGB-D image sequence was freely captured from one side of the table by moving around the table.

In this experiment, all of the cuboids were successfully detected regardless of the cuboid arrangements with some occlusions, and their shape parameters were also computed. The estimated size of each cuboid at each scene was described in Table 1. The error of cuboid 2 was larger than others because this cuboid was located at the furthest position from the table. This results from the accuracy degradation of depth images. Also, the shape of cuboid 2 was not completely measured because an obstacle hid the cuboid 2. In this case, the accuracy was largely decreased. For other cuboids, the error variance was relatively small.

7.2 Cuboid detection at a cluttered environment

To show the effectiveness of our incremental approach, a batch based approach was implemented as follow. A RGB-D SLAM system [12] was applied to an RGB-D image sequence to generate a full point cloud in a scene. Next, a shape detection method [22] was applied to the point cloud to detect planes in the scene. Since each plane normal vector cannot be uniquely determined, two planes having opposite normal vectors were generated from one plane. Then, the cuboid detection in Section 4 was applied to all the oriented planes to detect cuboids.

For this experiment, a challenging scene was designed, as illustrated in Figure 7. In this scene, 19 cuboids were randomly arranged, and many other objects were also placed as a cluttered environment. The scene was captured by freely moving around the



(a) Scene1 (b) Shape Map of (a) (c) Cuboid map of (a)

(d) Scene2 (e) Shape map of (d) (f) Cuboid map of (d)

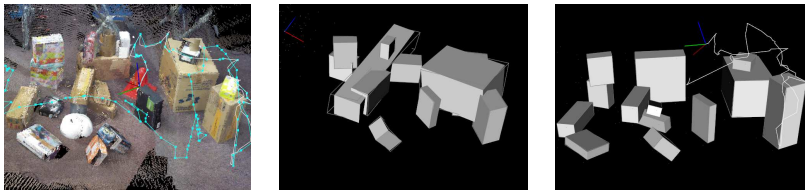
(g) Scene3 (h) Shape map of (g) (i) Cuboid map of (g)

Fig. 6: Cuboid shape estimation at various scenes. Three datasets were designed to investigate the accuracy of estimated cuboid shapes according to the arrangement. The first, the second, and their columns represent a scene, its shape map [17], and its cuboid map, respectively. The size of each cuboid was measured by a ruler as ground truth.

Table 1: Estimated cuboid lengths (cm)

Cuboid1	**Depth**	**Width**	**Height**	**Cuboid2**	**Depth**	**Width**	**Height**
Ground truth	16.0	22.5	10.5	*Ground truth*	46.6	49.8	41.0
Scene1	16.8	23.5	9.8	*Scene1*	46.9	50.3	34.1
Scene2	16.8	23.1	9.6	*Scene2*	40.9	50.7	36.5
Scene3	16.3	24.6	9.8	*Scene3*	39.8	47.1	31.2
Cuboid3	**Depth**	**Width**	**Height**	**Cuboid4**	**Depth**	**Width**	**Height**
Ground truth	9.8	9.8	19.7	*Ground truth*	7.9	16.0	21.8
Scene1	10.3	10.7	19.2	*Scene1*	7.0	17.2	21.3
Scene2	10.7	11.4	19.4	*Scene2*	7.4	19.4	21.9
Scene3	9.9	10.5	19.3	*Scene3*	7.2	16.7	21.2

scene. It should be noted that our visual guidance system was not used to capture the dataset.



(a) A cluttered environment (b) A batch based approach (c) Our incremental approach

Fig. 7: Comparison between a batch based approach with our incremental one. For (a) a scene reconstructed by [12], we applied a batch based cuboid detection to the scene, and had (b) the result. Compared to (c) our result, there were many false positives and false negatives because of noisy point cloud reconstruction. The detail of the accuracy is presented in Table 2.

Table 2: Performance of cuboid detection

	Batch	**Ours**
Precision	0.64	0.92
Recall	0.37	0.63

Batch	*Positive*	*Negative*
True	7	-
False	4	12

Ours	*Positive*	*Negative*
True	12	-
False	1	7

The performance of the method was evaluated in terms of precision and recall based on false positives, false negatives and true positives, as described in Table 2. Also, the results of cuboid maps were illustrated in Figure 7. In the batch based approach, there were more false positives and false negatives, compared with our approach. Since the point cloud from RGB-D SLAM was noisy due to registration error, several false positive and false negative planes were detected. Also, the ambiguity of plane normals caused the wrong clustering of three perpendicular planes. In our approach, cuboids were correctly detected because most of the planes were accurately modeled by avoiding the influence of error accumulation in ICP based D-SLAM [17]. However, the false negatives still occurred in our approach due to the incomplete measurement of the cuboids. Therefore, our AR based guidance system is helpful to complete the modeling.

7.3 Computational cost

The computational cost was measured at the Scene 1 in Figure 6a with 3.70 GHz of Intel (R) Xeon (R) CPU E 5-1620 v2, as illustrated in Figure 8. In [17], the computational cost required for RGB-D SLAM and plane reconstruction was within 100-ms on average. Compared to [17], we focused on measuring the cost for the 3D cuboid mod-

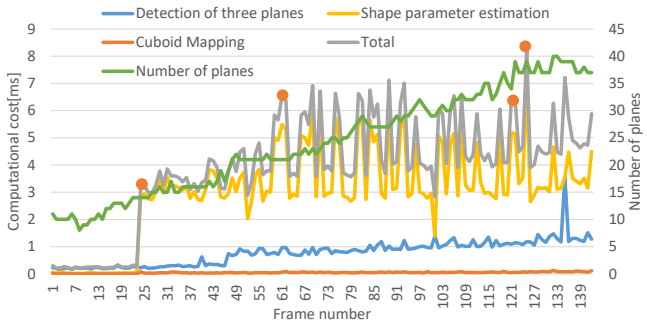


Fig. 8: Computational cost. The orange dots represent the time when a new cuboid is detected. The cost of shape parameter estimation is larger than others. The detection of three planes represents the sum of both 2nd and 3rd plane detections in Section 4. This cost increased as time passes because the number of unassigned planes increased.

eling. In the cuboid detection, the costs of detecting three planes and computing shape parameters were separately measured. In the figure, the orange dots represent the time when a new cuboid is detected.

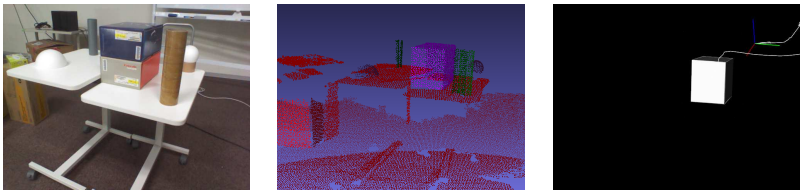
The shape parameter estimation needed most computational cost, especially in the process of the point projection to a line to compute edge lengths. The cost of detecting three planes gradually increased according to the number of planes in the map because planes can be detected from not only cuboids but also non-cuboids such as walls. In this case, the cuboid check was applied to the planes from non-cuboids in every frame. Therefore, this process affected the increase of the cost. Overall, the cost of our framework at a room-scale environment was sufficient for running with RGB-D SLAM.

7.4 Limitation

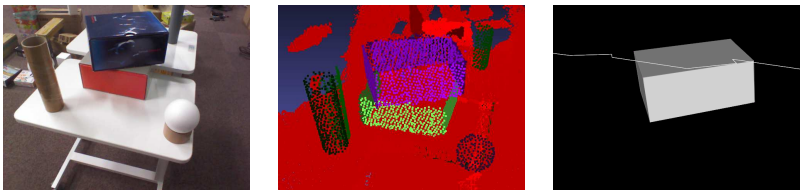
As illustrated in Figure 9, the cuboid detection using three perpendicular planes sometimes failed when boxes were stacked. Basically, the detection accuracy depends on the quality of the plane map. For instance, two stacked boxes can be detected as one cuboid when they are aligned. Since the faces of the two boxes compose a plane, they are detected as one plane. By using an image based segmentation, two boxes can be separately detected. In another case, the lower box cannot be detected even when two stacked boxes are not aligned because the top face of the lower box is still hidden by the upper box. Since three perpendicular planes are required to detect a cuboid in our framework, the detection fails.

8 Conclusions

We presented a framework for generating a cuboid map in an incremental manner. In this approach, a cuboid is first detected by analyzing the positional relationship between oriented planes. Then, it is incrementally updated to suppress false detections.



(a) Aligned case (b) Shape map of (a) (c) Cuboid map of (a)



(d) Unaligned case (e) Shape map of (d) (f) Cuboid map of (d)

Fig. 9: Limitation. At the first row, two stacked boxes are detected as one cuboid when they are aligned. At the second row, even when two stacked boxes are not aligned, the lower box is not detected because the top face of the lower box is not sufficiently captured as a plane. The accuracy of the cuboid detection can be degraded when boxes are stacked according to the arrangement.

An interactive cuboid modeling system was designed to assist the users to reconstruct cuboids.

The evaluation demonstrated that the cuboid modeling with our approach was more accurate than a batch-based method. Also, our method successfully detected the cuboids regardless of their arrangements. However, three perpendicular planes are required to be captured to compute the cuboid shape parameters, as our limitation.

In the future work, the performance of our framework with additional various scenes is investigated. Also, image features from RGB images will be integrated into our framework to increase the accuracy and robustness of the cuboid detection. Since the point cloud is obtained by RTAB-MAP [12] which is relatively inaccurate in terms of reconstruction quality compared with a TSDF based fusion method such as KinectFusion [14]. Our system should be combined with technique to improve the accuracy for primitive shape reconstruction quality. Additionally, the comparison of the state of the art will be done for display the advantage of our proposed method.

Acknowledgment

This work is supported by JSPS KAKENHI Grant Number JP17H01768.

References

1. Del Pero, L., Guan, J., Brau, E., Schlecht, J., Barnard, K.: Sampling bedrooms. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 2009–2016. IEEE

(2011)
2. Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D.B., Seitz, S.M., Fox, D.: Interactive 3d modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th international conference on Ubiquitous computing. pp. 75–84. ACM (2011)
3. Dwibedi, D., Malisiewicz, T., Badrinarayanan, V., Rabinovich, A.: Deep cuboid detection: Beyond 2d bounding boxes. arXiv preprint arXiv:1611.10010 (2016)
4. Goldman, R.: Intersection of three planes. In: Graphics gems. p. 305. Academic Press Professional, Inc. (1990)
5. Hashemifar, Z.S., Lee, K.W., Napp, N., Dantu, K.: Consistent cuboid detection for semantic mapping. In: Semantic Computing (ICSC), 2017 IEEE 11th International Conference on. pp. 526–531. IEEE (2017)
6. Hedau, V., Hoiem, D., Forsyth, D.: Recovering the spatial layout of cluttered rooms. In: Computer vision, 2009 IEEE 12th international conference on. pp. 1849–1856. IEEE (2009)
7. Hedau, V., Hoiem, D., Forsyth, D.: Thinking inside the box: Using appearance models and context based on room geometry. In: European Conference on Computer Vision. pp. 224–237. Springer (2010)
8. Hejrati, M., Ramanan, D.: Categorizing cubes: Revisiting pose normalization. In: Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on. pp. 1–9. IEEE (2016)
9. Jiang, H., Xiao, J.: A linear approach to matching cuboids in rgbd images. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 2171–2178. IEEE (2013)
10. Khan, S.H., He, X., Bannamoun, M., Sohel, F., Togneri, R.: Separating objects and clutter in indoor scenes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4603–4611 (2015)
11. Kim, Y.M., Mitra, N.J., Huang, Q., Guibas, L.: Guided real-time scanning of indoor objects. In: Computer Graphics Forum. vol. 32, pp. 177–186. Wiley Online Library (2013)
12. Labbé, M., Michaud, F.: Online global loop closure detection for large-scale multi-session graph-based slam. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. pp. 2661–2666. IEEE (2014)
13. Lin, D., Fidler, S., Urtasun, R.: Holistic scene understanding for 3d object detection with rgbd cameras. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 1417–1424. IEEE (2013)
14. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. pp. 127–136. IEEE (2011)
15. Nguatem, W., Drauschke, M., Mayer, H.: Finding cuboid-based building models in point clouds. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XXXIX-B3**, 149–154 (2012). https://doi.org/10.5194/isprsarchives-XXXIX-B3-149-2012, https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B3/149/2012/
16. Nguyen, T., Reitmayr, G., Schmalstieg, D.: Structural modeling from depth images. IEEE transactions on visualization and computer graphics **21**(11), 1230–1240 (2015)
17. Olivier, N., Uchiyama, H., Mishima, M., Thomas, D., ichiro Taniguchi, R., Roberto, R., do Monte Lima, J.S., Teichrieb, V.: Live structural modeling using rgb-d slam. In: ICRA. pp. 6352–6358 (2018)
18. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape distributions. ACM Transactions on Graphics (TOG) **21**(4), 807–832 (2002)
19. Ren, Z., Sudderth, E.B.: Three-dimensional object detection and layout prediction using clouds of oriented gradients. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1525–1533 (2016)

20. Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M.: Towards 3d point cloud based object maps for household environments. Robotics and Autonomous Systems **56**(11), 927–941 (2008)
21. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. The International Journal of Robotics Research **27**(2), 157–173 (2008)
22. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. In: Computer graphics forum. vol. 26, pp. 214–226. Wiley Online Library (2007)
23. Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., Guo, B.: An interactive approach to semantic modeling of indoor scenes with an rgbd camera. ACM Transactions on Graphics (TOG) **31**(6), 136 (2012)
24. Sinha, S.N., Steedly, D., Szeliski, R., Agrawala, M., Pollefeys, M.: Interactive 3d architectural modeling from unordered photo collections. In: ACM Transactions on Graphics (TOG). vol. 27, p. 159. ACM (2008)
25. Stein, S.C., Wörgötter, F., Schoeler, M., Papon, J., Kulvicius, T.: Convexity based object partitioning for robot applications. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on. pp. 3213–3220. IEEE (2014)
26. Xiao, J., Russell, B., Torralba, A.: Localizing 3d cuboids in single-view images. In: Advances in neural information processing systems. pp. 746–754 (2012)
27. Zhang, Y., Luo, C., Liu, J.: Walk&sketch: create floor plans with an rgb-d camera. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. pp. 461–470. ACM (2012)