sensors

MDPI

Article

Incremental 3D Cuboid Modeling with Drift Compensation

Masashi Mishima [1,*], Hideaki Uchiyama [2], Diego Thomas [3], Rin-ichiro Taniguchi [3], Rafael Roberto [4], João Paulo Lima [4,5] and Veronica Teichrieb [4]

1 Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan
2 Library, Kyushu University, Fukuoka 819-0395, Japan; uchiyama@limu.ait.kyushu-u.ac.jp
3 Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan; diego_thomas@limu.ait.kyushu-u.ac.jp (D.T.); rin@kyudai.jp (R.-i.T.)
4 Voxar Labs, Centro de Informática, Universidade Federal de Pernambuco, Recife 50740-560, Brazil; rar3@cin.ufpe.br (R.R.); jpsml@cin.ufpe.br (J.P.L.); vt@cin.ufpe.br (V.T.)
5 Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife 52171-900, Brazil
* Correspondence: mishima@limu.ait.kyushu-u.ac.jp; Tel.: +82-92-802-3580

Received: 3 December 2018; Accepted: 28 December 2018; Published: 6 January 2019

Abstract: This paper presents a framework of incremental 3D cuboid modeling by using the mapping results of an RGB-D camera based simultaneous localization and mapping (SLAM) system. This framework is useful in accurately creating cuboid CAD models from a point cloud in an online manner. While performing the RGB-D SLAM, planes are incrementally reconstructed from a point cloud in each frame to create a plane map. Then, cuboids are detected in the plane map by analyzing the positional relationships between the planes, such as orthogonality, convexity, and proximity. Finally, the position, pose, and size of a cuboid are determined by computing the intersection of three perpendicular planes. To suppress the false detection of the cuboids, the cuboid shapes are incrementally updated with sequential measurements to check the uncertainty of the cuboids. In addition, the drift error of the SLAM is compensated by the registration of the cuboids. As an application of our framework, an augmented reality-based interactive cuboid modeling system was developed. In the evaluation at cluttered environments, the precision and recall of the cuboid detection were investigated, compared with a batch-based cuboid detection method, so that the advantages of our proposed method were clarified.

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 3D map. Although localization can be performed by using the map, the map is simply composed of sets of a point, and does not represent semantics in the environment. Toward 3D-scene understanding, it is important to convert a point cloud into an object-level representation as the high-level knowledge of the scene geometry. Planes, cylinders, and spheres are an example of a low-level parametric primitive representation to compose the scene geometry. The recognition of such primitive shapes is an important task for various applications such as obstacle avoidance and object grasping [1].

A cuboid, which is the target shape for the CAD conversion in this paper, is also considered as an informative shape representation because there exist many types of cuboids in our environment. For instance, delivery boxes used in logistics and product packages in markets are practical examples.

To achieve automated robot manipulation in such environments, techniques to recognize cuboids under cluttered conditions are often required. In the literature of this field, cuboid detection has been performed by using a monocular RGB image [2–6] or a point cloud generated with a single RGB-D image [7–14]. For outdoor environments, the approaches for generating cuboid-based building models have been proposed according to the density of a point cloud with Light Detection and Ranging (LIDAR) [15–17]. Generally, these methods are based on offline batch processing such that recognition is only performed with a single observation. Typically, they suffer from both false positives and false negatives due to the noisy 3D map. To solve the problem, we investigated a novel sequential approach to suppress false detection by incorporating multiple observations with temporal filtering in an online manner [18,19].

In this paper, we propose a framework of incremental cuboid modeling by using the mapping results of an RGB-D SLAM system. In our framework, an RGB-D SLAM system was used as an external tracker to incrementally produce a point cloud of the environment so that any RGB-D SLAM system could be applied, as similar to the framework in Reference [20]. We focus on stably modeling cuboids from the incrementally updated point cloud. At every frame, planes are incrementally reconstructed and updated from a point cloud-based map [21], and are used as the input to our framework. Then, planes are clustered to compose cuboids by analyzing three plane-positional relationships, namely, orthogonality, convexity, and proximity. To accurately reconstruct a cuboid, a cluster of three perpendicular planes is first selected, and their intersection is computed [22]. By determining three perpendicular cuboid edges from both the intersection and the normal vectors of each face, the width, height, and depth of a cuboid are finally computed as cuboid-shape parameters. Since the parameters can be incrementally updated based on Reference [21], the positional relationships of the planes are analyzed at every frame. This means that both newly detected planes and previously detected cuboids are analyzed so that the false detection of the cuboids can be suppressed with this sequential processing. For example, a falsely detected cuboid face can be replaced with a correct one, and a new plane can be assigned to a previously detected cuboid as a forth cuboid face [23]. In addition, the drift error in the SLAM system is compensated by using the cuboids. Owing to this drift compensation, the duplicated cuboids caused by the fusion failure that occurred in Reference [23] can be reduced. As an application of our framework, we developed an interactive cuboid modeling system to allow users to reconstruct cuboids with augmented reality (AR)-based affordance. In the evaluation, the accuracy of our framework was quantitatively evaluated by using several boxes with their ground truth sizes. The comparison between a batch-based cuboid detection method and our incremental one was also investigated to show the effectiveness of our approach in a cluttered environment. In addition, the accuracy of the cuboid modeling with drift compensation was investigated. Finally, computational cost was explained to show that our framework can run in real time in a room-scale environment.

In summary, the contributions of our paper are listed as follows:

- Cuboid reconstruction is performed by searching three perpendicular planes and computing the intersection of the planes.
- A novel framework for incremental cuboid modeling based on cuboid detection and mapping is proposed.
- The drift error of the SLAM is accurately compensated by using cuboids.
- An application for AR-based interactive cuboid modeling is presented.

This paper is the extended version of Reference [23], and the main difference is twofold: additional process of the drift compensation with the cuboid matching, and its performance for accurate cuboid detection. Due to the drift compensation process, the accumulated error was further corrected, and the accuracy of the cuboid map was improved, as explained in Section 8.3.

2. Related Work

Cuboid detection has been investigated in the field of semantic 3D-scene understanding. In this section, we review the literature in terms of the devices used for the detection.

Recognizing cuboids from a single RGB image has been proposed [2–6,24]. Hedau et al. reconstructed a cuboid-based room layout by using vanishing points [2,3]. First, wall, ceiling, and furniture contours were extracted from an input image, and then vanishing points were estimated from three orthogonal 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 [4]. 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 [5]. Hejrati and Ramanan investigated the performance of several feature representations for categorizing cuboid objects [6]. Dwibedi et al. proposed a deep-learning-based region proposal method for cuboid detection [24]. Basically, cuboid detection using a RGB image is an ill-posed problem, and accuracy is largely degraded under occlusions.

A point cloud acquired from RGB-D images or LIDAR has also been used for cuboid detection [7–14,25–27]. Shape descriptors for arbitrary 3D objects were proposed for object classification, including cuboids [7,8,12]. For indoor environments, prior knowledge of a room layout was incorporated to globally optimize object arrangement including cuboids in the room [9,12,27]. To detect buildings as cuboids, a closed polyhedral model is searched from planes detected in a point cloud [25]. 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 [10,11,13,26]. Compared with the approach using a RGB image, the one using a point cloud can provide the metric size and pose of a cuboid in a scene. However, it still suffers from the false detection of cuboids in the presence of sensor noises and registration error. To solve this problem and improve the stability and accuracy of cuboid detection, we propose an incremental approach by fusing multiple measurements captured from different viewpoints without using any constraint on 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 created from point clouds, and used as the input to our framework. This incremental mapping process is based on our previous method [21] that applies a shape-detection method [28] to incoming point clouds acquired from an RGB-D SLAM system [29]. In other words, any SLAM system can be used as an external tracker, and incrementally produces a noisy 3D point cloud. From such a point cloud, plane are stably detected, and are used for our stable cuboid modeling.

From the plane map, a cuboid map, which is composed of cuboids with their positions, poses, and sizes, is created. As described in Section 4, cuboid faces are detected among a group of planes by analyzing the positional relationships of the planes, namely cuboid check. 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 normal vectors. 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 as cuboid-shape parameters. These parameters can be used in CAD systems.

To create an accurate cuboid map, an incremental reconstruction process is proposed [23], as described in Section 5. At every frame, the planes in the plane map are classified into planes assigned to cuboids and unassigned ones, as illustrated in Figure 2. First, the cuboid check is performed for the cuboids in the cuboid map to check the positional relationships of the cuboid faces at every frame because their parameters are incrementally updated with [21], 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 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, cuboid detection is performed for the remaining unassigned planes. This incremental process allows users to make the cuboid modeling succeed with AR-based affordance.

In addition, the drift-compensation process is proposed as described in Section 6. Generally, the map from the SLAM is not accurate when the camera trajectory is long. A loop-closing technique is applied to suppress the error when the closure occurs [30]. However, the error cannot be perfectly suppressed, such that the map may still contain the duplicate of the same object because of the registration error. Therefore, we propose to further suppress the remaining drift error by using the cuboids as illustrated in Figure 3. This process can be considered as the suppression of the registration error or the refinement of the map by using the cuboid. First, a local cuboid map is generated with the point cloud acquired from the SLAM system, and the global map is generated as the collection of the local maps. When a new cuboid is detected in the local map, it is matched with one of the cuboids in the global map to merge the local map into the global one. If the match is detected, the transformation from the local map to the global map can be computed by using the matched cuboids. Otherwise, the local map grows until another cuboid is detected.

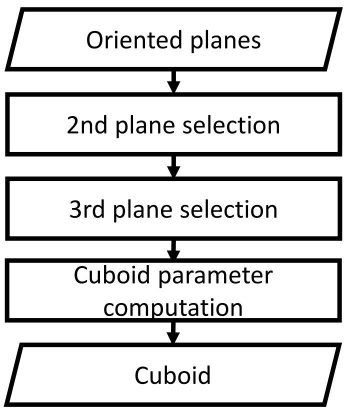


Figure 1. Cuboid detection. From oriented planes, two planes are first selected as a plane pair by checking their positional relationships. Then, the third plane perpendicular to the pair is searched by using the cross product of the two plane normal vectors. Finally, the cuboid-shape parameters are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

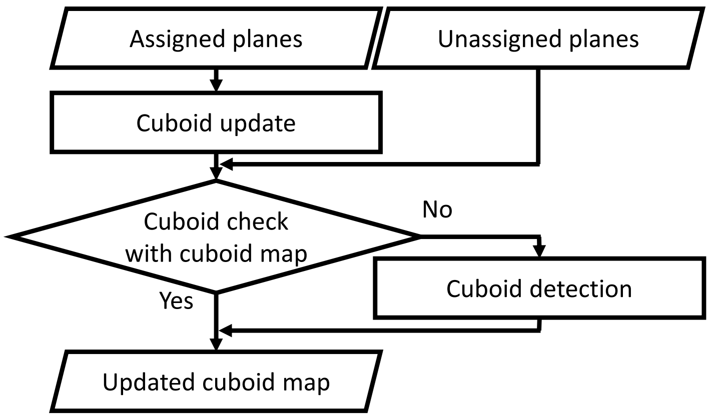


Figure 2. Cuboid mapping. The status of the planes in the plane map is classified into planes assigned to cuboids or not. For the assigned planes, the cuboid check is performed for the cuboids in the map at every frame, as a cuboid update. For the unassigned planes, the cuboid check with the cuboids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

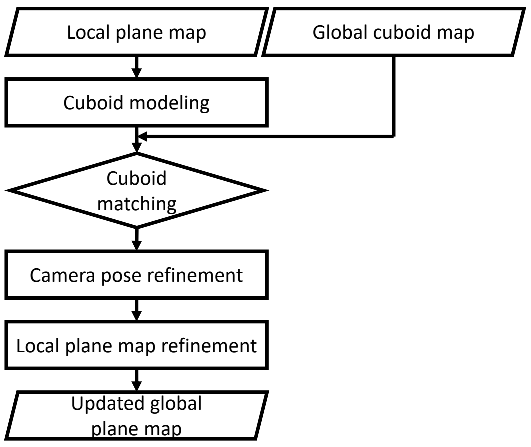


Figure 3. Drift compensation. First, a local plane map is generated with a point cloud from the simultaneous localization and mapping (SLAM), and a cuboid is detected in the local map. Next, the detected cuboid is matched with the one in the global cuboid map in cuboid matching. Then, the transformation from the cuboid in the local map to the one in the global map is computed as the camera pose refinement. Finally, the local map can be merged into the global one with the transformation, as local plane map refinement.

4. Cuboid Detection

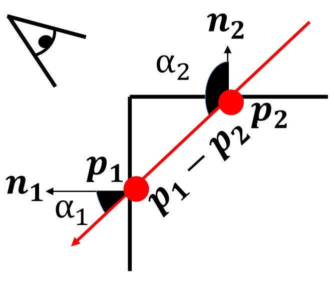
Next, we explain in detail how to detect a cuboid from planes. The first process is to select two perpendicular planes by analyzing the positional relationships. The second process is to search the third plane by using the cross product of the two plane normal vectors. After three perpendicular faces are determined, cuboid-shape parameters are finally determined by computing the intersection of the three planes as cuboid detection.

4.1. Second Plane Selection

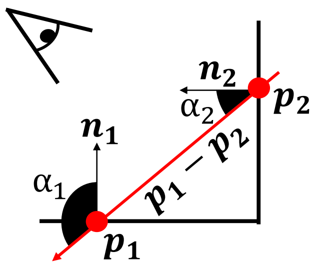
In this process, sets of two planes composing a cuboid are searched in a brute-force manner. An i-th plane in the plane map is parameterized with the center of mass $\boldsymbol{p}_i$ and normal vector $\boldsymbol{n}_i$ [21]. First, the inner products between a target plane and all of the other planes are computed as an orthogonality check. A plane is selected if the angle computed from the inner product is perpendicular within an error tolerance (e.g., 5 degrees). However, orthogonality is not sufficient for cuboid detection because there are two possibilities of the positional relationship between two planes, concave and convex. A plane can also 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 Reference [31] as the 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 visible from a camera. Since each plane has the center of mass and the normal vector, the convexity can be computed from them. In Figure 4, $\boldsymbol{n}_1$ 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 the 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 the proximity check. All of these checks between two planes are referred to as the cuboid check.



(**a**) convex ($\alpha_1 < \alpha_2$)



(**b**) concave ($\alpha_1 > \alpha_2$)

Figure 4. Convexity check. Convexity is analyzed by using the centers of mass and the normal vectors [31].

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 the 3D edge regions of a cuboid are normally degraded in depth images. To accurately reconstruct a cuboid, three perpendicular planes are used to determine cuboid-shape parameters in our framework.

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 in the third plane selection. 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, 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. These parameters are useful for standard 3D CAD systems. In Figure 5, π_i is an i-th 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 [22] as follows.

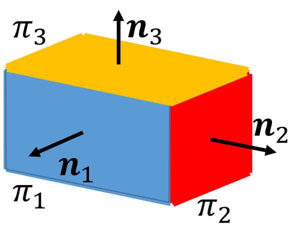
$$\boldsymbol{p}_o = \frac{(\boldsymbol{p}_1 \cdot \boldsymbol{n}_1)(\boldsymbol{n}_2 \times \boldsymbol{n}_3)}{(\boldsymbol{n}_1 \times \boldsymbol{n}_2) \cdot \boldsymbol{n}_3} + \frac{(\boldsymbol{p}_2 \cdot \boldsymbol{n}_2)(\boldsymbol{n}_3 \times \boldsymbol{n}_1)}{(\boldsymbol{n}_1 \times \boldsymbol{n}_2) \cdot \boldsymbol{n}_3} + \frac{(\boldsymbol{p}_3 \cdot \boldsymbol{n}_3)(\boldsymbol{n}_1 \times \boldsymbol{n}_2)}{(\boldsymbol{n}_1 \times \boldsymbol{n}_2) \cdot \boldsymbol{n}_3} \tag{1}$$

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

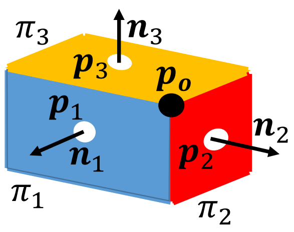
To determine the size of a cuboid, the length of each edge is computed by projecting the points on a plane onto the edge as

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

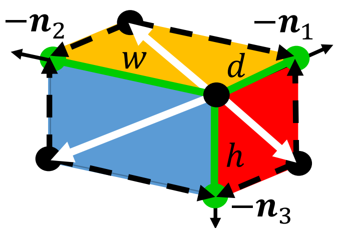
where $\boldsymbol{n}$ is each of the normalized edge-direction vector, and $\boldsymbol{x}_i$ is an i-th point in the plane. This equation represents that the points on the plane sharing the edge are projected onto the edge, and the furthest point from the intersection on the edge is selected to compute 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 plane number i can be arbitrarily determined for the three perpendicular planes.



(**a**) Three planes



(**b**) Intersection



(**c**) Size computation

Figure 5. 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 properties of planes can be divided into two categories, planes assigned to one of the cuboids in the cuboid map, or not. 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 properties. In this section, we explain the details of cuboid mapping.

5.1. Cuboid Update

To reduce false detection of cuboids, the cuboid check is performed for all of 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, at 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 of the cuboid modeling system. Normally, users do not understand the best way to capture a scene and when to finish capturing it. By using an incremental approach, the false positives and false negatives of the cuboids can be visualized in an online manner. This largely helps users complete the modeling because they can understand the progress.

As an alternative approach, it is possible to apply a batch-based cuboid-detection method to a point cloud generated from the past at every frame. However, the computational cost at a frame increases according to the size of the point cloud. It is also redundant to detect the cuboids in a point cloud at every frame because the detection result at a frame can be useful at the next frame. In terms of computational efficiency, the incremental approach is appropriate for online 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 of the cuboids in the map is first performed.

A cuboid face in the map 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. The plane is also 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, 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. Drift Compensation

Any SLAM system cannot avoid the accumulated drift error over time, and always degrades the accuracy of the map. Even though pose graph optimization can reduce the error when the loop is closed [30], the remaining error sometimes still exists. In this section, we explain the process of further suppressing the error with the cuboids.

6.1. Overview

To create an accurate cuboid map, drift error is compensated by using the cuboids as illustrated in Figure 6. We propose individually using two different maps for planes and cuboids: the local map and the global one. First, the point cloud from the SLAM is used for generating the local plane map. Next, cuboid detection is applied to the local plane map to detect a cuboid. Then, cuboid matching between the local cuboid map and the global one is performed to detect the same cuboid in both the maps. If detected, the local map is merged into the global one by computing the transformation from the cuboid in the local map to the one in the global map. Otherwise, the cuboid remains in the local cuboid map, and another cuboid can be detected after finding additional planes in the local plane map. In our system, we assume that the drift error is small after the pose graph optimization-based map refinement so that cuboid matching can be performed, as implemented in Reference [29]. However, there is still the remaining error. Therefore, we further refine the map by using cuboid-parameter comparison. In other words, our method fails if the overlap does not occur because of the large drift error. It should be noted that both the global plane map and global cuboid one are empty at the beginning of the process. As initialization, the local plane map is directly used as the global one when a cuboid is detected for the first time.

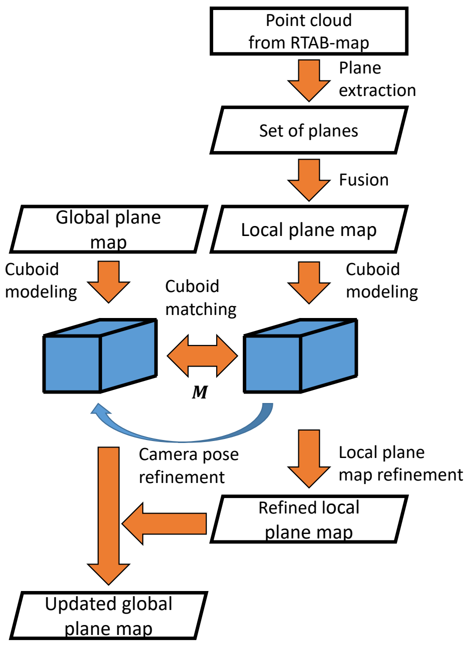


Figure 6. Drift compensation. Cuboids are used to suppress the drift error. First, a local plane map is generated with a point cloud from the SLAM system, and the cuboid is detected in the local map. Then, the detected cuboid is matched with the one in the global map. The transformation matrix is computed by comparing the cuboids in the local map and the global one. Then, the local plane map is refined with the transformation, and is merged into the global one. Once drift compensation is performed, the local plane map is reset.

6.2. Cuboid Matching

The global map is defined in the world co-ordinate system, and the local map generated with the SLAM system is also the same at the beginning. However, the local map can be gradually shifted due to the accumulation of registration error, and there are duplicates if the local map is simply merged into the global one. To solve this problem, the misalignment of the cuboid is detected by computing the overlapping volume between the axis-aligned bounding box (AABB) of the same cuboid in the local map and global one, as illustrated in Figure 7.

At every frame, cuboid detection is applied in the local map. If a new cuboid is detected, it is checked whether the same cuboid exists in the global map or not. First, the local map is directly merged into the global one. Then, the overlap volume between the AABB of the cuboid in the local map and that in the global one is computed. Finally, the transformation matrix is computed based on the cuboid as described in Section 6.3.

The details of cuboid matching are as follows. The furthest and nearest vertices of the AABB from the origin are detected, and are defined as $\boldsymbol{C_f}$ and $\boldsymbol{C_n}$, respectively. For instance, the farthest and nearest vertices for the AABB in the global cuboid map are defined as $\boldsymbol{C_{f_G}}$ and $\boldsymbol{C_{n_G}}$, and those of the local one are defined as $\boldsymbol{C_{f_L}}$ and $\boldsymbol{C_{n_L}}$. The overlap volume between the AABBs is defined as V_O, and is computed as follows:

$$\begin{aligned} V_O =& [\max\{0, \min\{\boldsymbol{C_{f_G}}, \boldsymbol{C_{f_L}}\} - \max\{\boldsymbol{C_{n_G}}, \boldsymbol{C_{n_L}}\}\}]_{xvalue} \\ &\times [\max\{0, \min\{\boldsymbol{C_{f_G}}, \boldsymbol{C_{f_L}}\} - \max\{\boldsymbol{C_{n_G}}, \boldsymbol{C_{n_L}}\}\}]_{yvalue} \\ &\times [\max\{0, \min\{\boldsymbol{C_{f_G}}, \boldsymbol{C_{f_L}}\} - \max\{\boldsymbol{C_{n_G}}, \boldsymbol{C_{n_L}}\}\}]_{zvalue} \end{aligned} \tag{3}$$

where $[\quad]_{ivalue}$ means the value of i axis $(i = x, y, z)$. If V_O is greater than zero, the cuboid is considered as overlapped.

In cuboid matching, two or more duplicate cuboids may appear. In this case, the cuboid with the highest overlap rate is selected. The volume of the AABB in the global cuboid map is defined as V_G, and the overlap rate is defined as *rate*. V_G and *rate* are computed as follows.

$$V_G = [\boldsymbol{C_{f_G}} - \boldsymbol{C_{n_G}}]_{xvalue} \times [\boldsymbol{C_{f_G}} - \boldsymbol{C_{n_G}}]_{yvalue} \times [\boldsymbol{C_{f_G}} - \boldsymbol{C_{n_G}}]_{zvalue} \tag{4}$$

$$rate = V_O / V_G \tag{5}$$

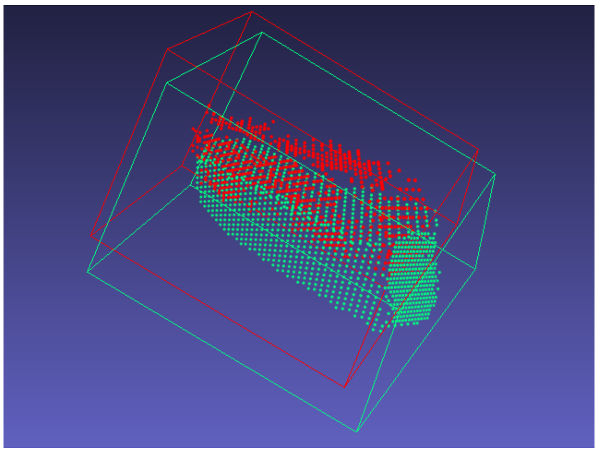


Figure 7. Misalignment between local cuboid (red) and global cuboid (green). Due to the drift error, a duplicate of the same object can arise.

6.3. Camera Pose Refinement

After cuboid matching, the camera pose of the SLAM in the local map is refined with the duplicate of the same cuboid to be consistent with the global map. The relationship among a point in global map $\boldsymbol{X_G}$, a point in the local one $\boldsymbol{X_L}$, and transformation matrix $\boldsymbol{M}$ is

$$\widetilde{\boldsymbol{X}}_G = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \widetilde{\boldsymbol{X}}_L \quad (6)$$

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad (7)$$

where $\widetilde{\boldsymbol{X}}_G$ and $\widetilde{\boldsymbol{X}}_L$ are homogeneous coordinates, $\boldsymbol{R}$ is a rotation matrix, and $\boldsymbol{t}$ is a translation vector. $\boldsymbol{P_G}$ and $\boldsymbol{P_L}$ are the pose of the cuboid in global and local map, respectively, and are defined as:

$$\boldsymbol{P_G} = \begin{bmatrix} \boldsymbol{R_G} & \boldsymbol{t_G} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad (8)$$

$$\boldsymbol{P_L} = \begin{bmatrix} \boldsymbol{R_L} & \boldsymbol{t_L} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad (9)$$

Cuboid rotation is computed from the normal vectors of three perpendicular planes $\boldsymbol{n_i}$, as illustrated in Figure 5. As a notation, the normal of the first plane is $\boldsymbol{n_1}$,the second one is $\boldsymbol{n_2}$, and the third one is $\boldsymbol{n_3}$. Cuboid rotations are formed as follows:

$$\boldsymbol{R_G} = \begin{bmatrix} \boldsymbol{n_{1_G}} & \boldsymbol{n_{2_G}} & \boldsymbol{n_{3_G}} \end{bmatrix} \quad (10)$$

$$\boldsymbol{R_L} = \begin{bmatrix} \boldsymbol{n_{1_L}} & \boldsymbol{n_{2_L}} & \boldsymbol{n_{3_L}} \end{bmatrix} \quad (11)$$

$\boldsymbol{R}$ and $\boldsymbol{t}$ are, finally, computed as follows.

$$\boldsymbol{R} = \boldsymbol{R_G}\boldsymbol{R_L^{\mathrm{T}}} \quad (12)$$

$$\boldsymbol{t} = \boldsymbol{p_{o_G}} - \boldsymbol{R}\boldsymbol{p_{o_L}} \quad (13)$$

where $\boldsymbol{p_{o_G}}$ and $\boldsymbol{p_{o_L}}$ are the origin of local and global cuboids, respectively. $\boldsymbol{t}$ is computed by using the origin point of the cuboid.

6.4. Local Plane Map Refinement

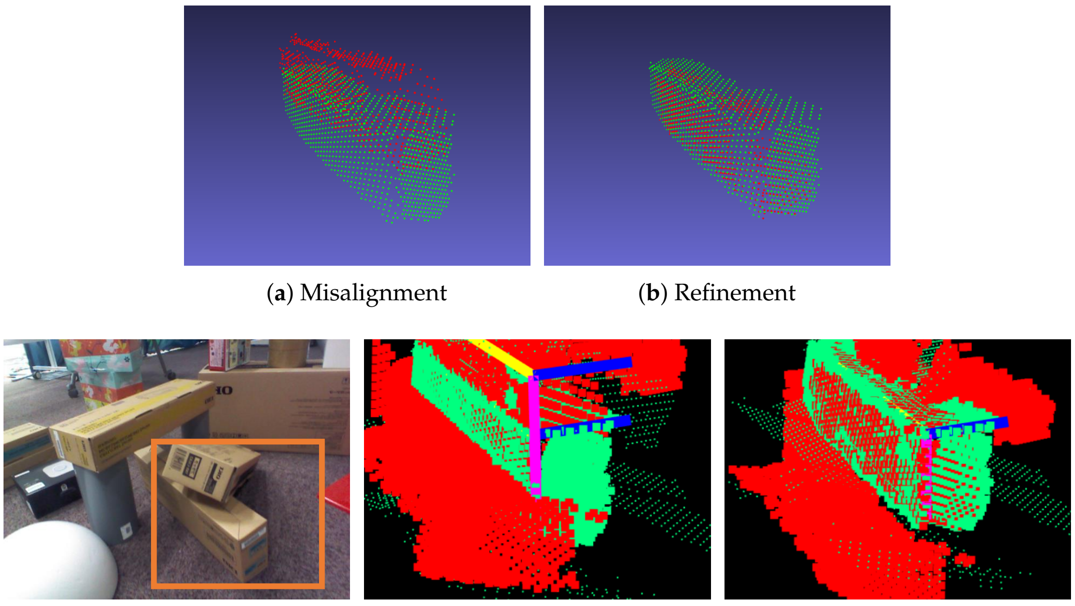
The local plane map is refined by using $\boldsymbol{M}$ computed in the camera pose refinement. The plane parameters are a normal vector $\boldsymbol{n_L}$, and an arbitrary point on plane $\boldsymbol{V_L}$. First, $\boldsymbol{n_L}$ is rotated, and $\boldsymbol{V_L}$ is transformed as

$$\boldsymbol{n_G} = \boldsymbol{R}\boldsymbol{n_L} \quad (14)$$

$$\widetilde{\boldsymbol{V}}_G = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \widetilde{\boldsymbol{V}}_L \quad (15)$$

where $\boldsymbol{n_G}$ is the normal vector of the refined plane, and $\boldsymbol{V_G}$ is an arbitrary point on the refined plane. Then, all points on the plane are projected following the point-projection method [21]. Finally, the refined map is merged into the global one.

Figure 8 illustrates the results of this refinement process. The red and green planes represent detected cuboids in the local and global map, respectively. When directly merging the local map with the global one, as illustrated in Figure 8a, a duplicate may occur. After refinement, the duplicate disappears, as illustrated in Figure 8b. Other planes in the local plane map are also transformed. In the area of the orange rectangle in Figure 8c, cuboid matching and camera pose refinement are performed by using one of the cuboids. As illustrated in Figure 8d, there is misalignment before refinement. For instance, the difference between the positions of the local and global cuboid was 5.8 cm in this case. After refinement, many planes are well-aligned, as illustrated in Figure 8e. The misalignment between cuboid faces and other planes is reduced.


(**a**) Misalignment (**b**) Refinement

(**c**) Example (**d**) Before refinement (**e**) After refinement

Figure 8. Local plane map refinement. The red and green represent the cuboid in the local and global map, respectively. The cuboid faces are not aligned in (**a**) after directly merging the local plane map into the global one. The pose of a cuboid is refined in (**b**) after camera pose refinement. For example, scenes (**c**,**d**) represent the direct merger of the local map into the global map, and (**e**) is the result of the refinement. The edges of each cuboid are visualized, and there is a gap between the cuboids in (**d**), whereas the cuboids are merged in (**e**).

7. Interactive Cuboid Modeling

Since 3D modeling using a camera is not an easy task for nonexperts, interactive techniques have been proposed [32]. For instance, the result of 3D modeling can easily be modified on user interfaces [33,34]. The incompleteness of the modeling is also visualized by showing a 2D slide of a point cloud for modeling a room [35] or showing an example for modeling an object [36]. Here, we introduce a simple but effective affordance for modeling a cuboid.

As illustrated in Figure 9, the points on planes are overlaid with colors. Blue and yellow regions 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 9a. This corresponds to the instruction for the users. Since 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 9b. Owing to the incremental approach, it is possible to develop this type of interactive modeling systems.

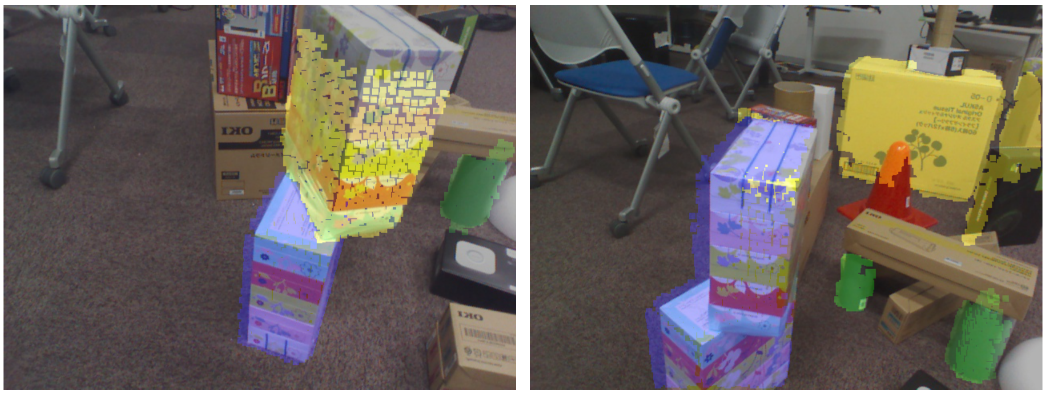

(**a**) Incomplete (Top is missing) (**b**) Completed (Top is captured)

Figure 9. Interactive cuboid modeling. Yellow and blue regions 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.

8. Evaluation

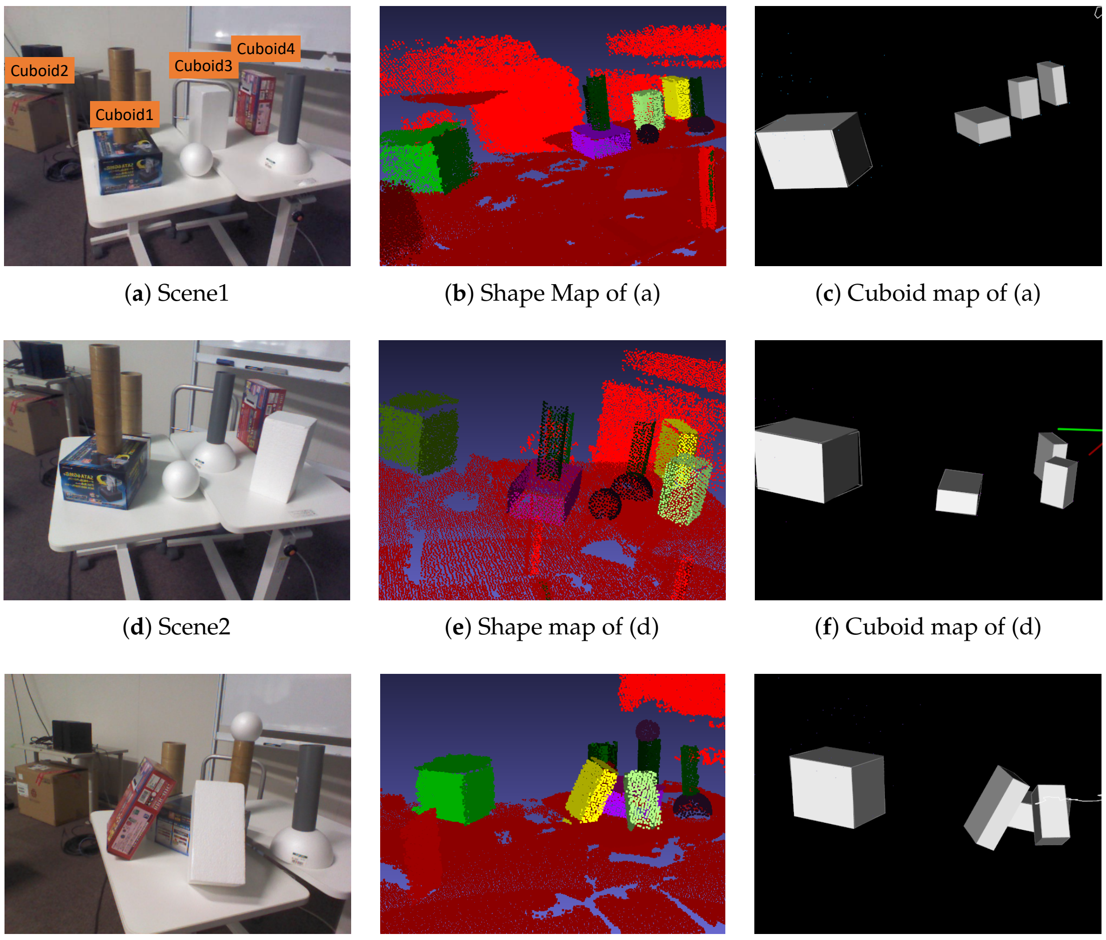
To evaluate the performance of our proposed framework, we first prepared RGB-D image sequences capturing multiple boxes as our dataset because only a dataset for single views was developed in the literature [5] and there is no dataset with RGB-D sequences containing cuboid ground truth annotations. For the camera, a Kinect V1 sensor was used; therefore, the boxes were set up in 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, 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.

8.1. Cuboid-Shape Estimation

As illustrated in Figure 10, three scenes were designed so that four cuboids with different sizes were arranged on a table and other objects were placed as obstacles. In Scene 1, the cuboids were rotated to face the same direction. In Scene 2, the cuboids were rotated to face in different directions except for the top face. In Scene 3, two cuboids were inclined onto a cuboid. In all scenes, one cuboid was placed far from the table. In Figure 10, the first column represents an example image of each scene, the second represents the shape map [21] drawn with different colors per cuboid, and the third represents 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 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 is described in Table 1. The error of Cuboid 2 was larger than the others because this cuboid was located at the farthest position from the table. This resulted from the accuracy degradation of depth images. The shape of Cuboid 2 was not completely measured because an obstacle hid it. In this case, the accuracy was largely decreased. For other cuboids, error variance was relatively small.



(**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)

Figure 10. 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 [21], and its cuboid map, respectively. The size of each cuboid was measured by a ruler as ground truth.

Table 1. Estimated cuboid lengths (cm).

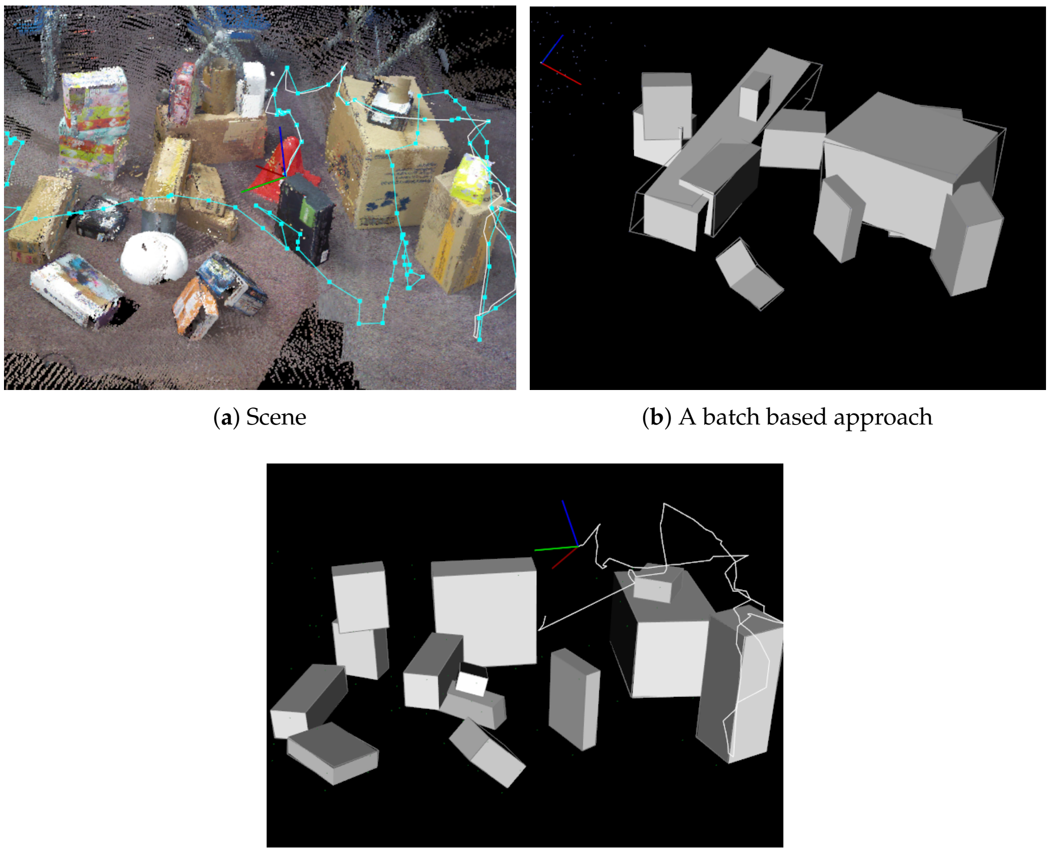
Cuboid 1	**Depth**	**Width**	**Height**	**Cuboid 2**	**Depth**	**Width**	**Height**
Ground truth	16.0	22.5	10.5	*Ground truth*	46.6	49.8	41.0
Scene 1	16.8	23.5	9.8	*Scene1*	46.9	50.3	34.1
Scene 2	16.8	23.1	9.6	*Scene2*	40.9	50.7	36.5
Scene 3	16.3	24.6	9.8	*Scene3*	39.8	47.1	31.2
Cuboid 3	**Depth**	**Width**	**Height**	**Cuboid 4**	**Depth**	**Width**	**Height**
Ground truth	9.8	9.8	19.7	*Ground truth*	7.9	16.0	21.8
Scene 1	10.3	10.7	19.2	*Scene 1*	7.0	17.2	21.3
Scene 2	10.7	11.4	19.4	*Scene 2*	7.4	19.4	21.9
Scene 3	9.9	10.5	19.3	*Scene 3*	7.2	16.7	21.2

8.2. Cuboid Detection in a Cluttered Environment

To show the effectiveness of our incremental approach, a batch-based approach was implemented as follows. An RGB-D SLAM system [29] was applied to an RGB-D image sequence to generate a full point cloud in a scene. Next, a shape-detection method [28] 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 11. In this scene, 19 cuboids were randomly arranged, and many other objects were placed as a cluttered environment. The scene was captured by freely moving around the scene. It should be noted that our visual guidance system was not used to capture the dataset.

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. The results of the cuboid maps are illustrated in Figure 11. 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 the registration error, several false-positive and false-negative planes were detected. The ambiguity of plane normals also 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 Iterative Closest Point (ICP)-based D-SLAM [21]. However, false negatives still occurred in our approach due to the incomplete measurement of the cuboids. For instance, it was still difficult to detect two stacked boxes with similar sizes when they were aligned, as illustrated in Figure 12. We discuss this in detail as a limitation in Section 8.5.

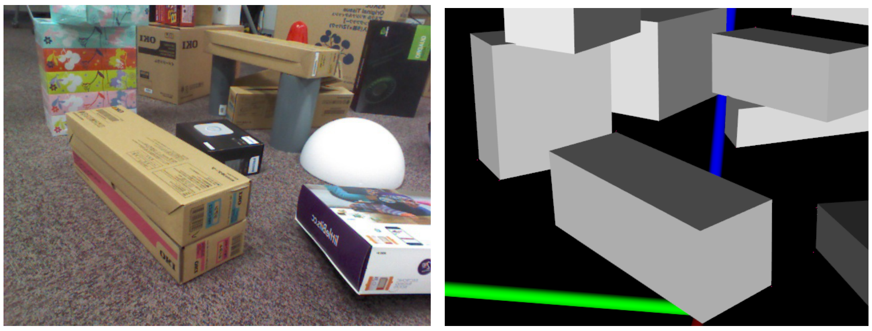

(**a**) Scene (**b**) A batch based approach

(**c**) Our incremental approach

Figure 11. Comparison between a batch-based approach with our incremental one. For (**a**) a scene reconstructed by Reference [29], we applied batch-based cuboid detection to the scene, and had (**b**) as the result. Comparing (**c**) with our result, there were many false positives and false negatives because of noisy point cloud reconstruction. Accuracy details are 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



(**a**) Scene (**b**) False positive

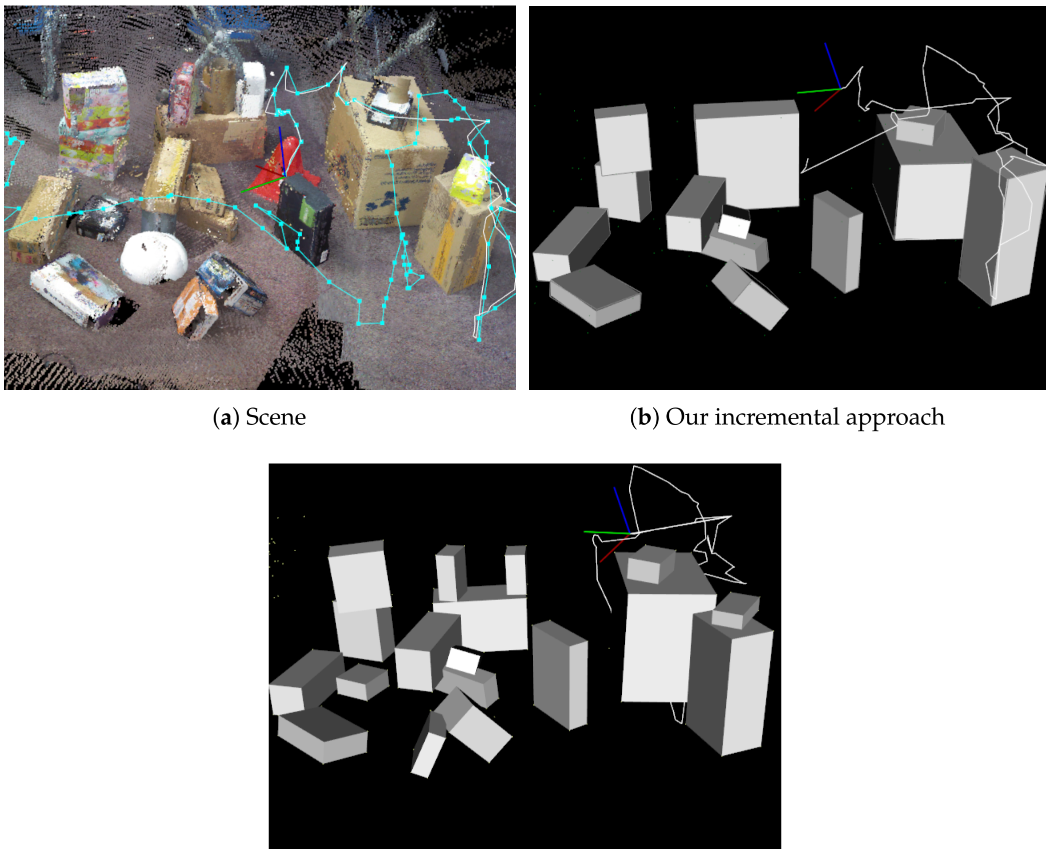
Figure 12. An example of a false-positive result. In (**a**), a box is stacked on another with the same size. It is difficult to distinguish the boxes in (**b**).

8.3. Effectiveness of Drift Compensation

The performance of cuboid detection with drift compensation was evaluated in terms of precision and recall, the same as Section 8.2. The results are described in Table 3. The results of the cuboid maps are also illustrated in Figure 13. In our incremental approach without drift compensation, five cuboids were missing. This is because cuboid faces cannot be properly detected when camera pose estimation error became large and the local map receded from the global one. By incorporating drift compensation, the number of detected cuboids increased owing to camera pose refinement.

Table 3. Performance of cuboid detection.

	Incremental	**Drift Compensation**
Precision	0.92	0.94
Recall	0.63	0.89
Incremental	*Positive*	*Negative*
True	12	-
False	1	7
Drift compensation	*Positive*	*Negative*
True	17	-
False	1	2



(**a**) Scene (**b**) Our incremental approach

(**c**) Drift compensation

Figure 13. Effectiveness of drift compensation. For (**a**) a scene reconstructed by Reference [29], we applied our incremental cuboid detection to the scene, and (**b**) was the result. Comparing (**c**) with drift compensation, accuracy details are presented in Table 2.

8.4. Computational Cost

The computational cost was measured at Scene 1 in Figure 10a with 3.70 GHz of Intel (R) Xeon (R) CPU E 5-1620 v2, as illustrated in Figure 14. In Reference [21], the computational cost required for RGB-D SLAM and plane reconstruction was within 100 ms, on average. Compared to Reference [21], we focused on measuring the cost for 3D cuboid modeling. In cuboid detection, the costs of detecting three planes and computing shape parameters were separately measured. In the figure, orange dots represent the time when a new cuboid is detected.

The shape parameter estimation needed most computational costs, especially in the process of 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 not only from cuboids but also noncuboids such as walls. In this case, the cuboid check was applied to the planes from noncuboids in every frame. Therefore, this process affected the cost increase. Overall, the cost of our framework in a room-scale environment was sufficient for running with RGB-D SLAM.

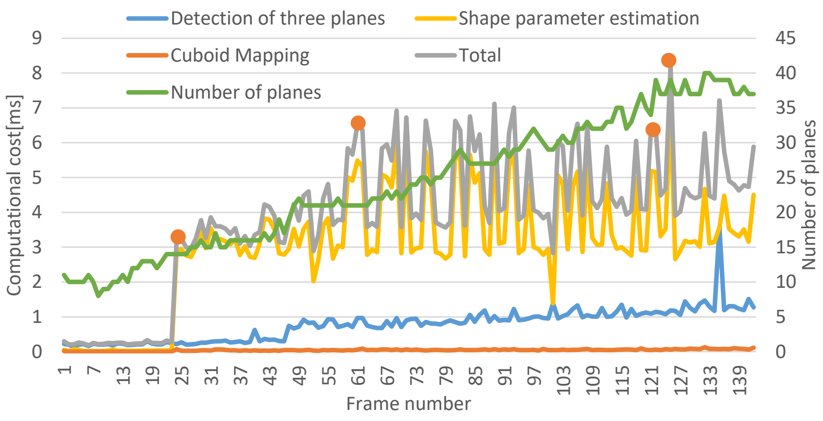


Figure 14. Computational cost. Orange dots represent the time when a new cuboid is detected. The cost of shape-parameter estimation was larger than others. The detection of three planes represents the sum of both second and third plane detections in Section 4. This cost increased as time passed because the number of unassigned planes increased.

8.5. Limitation

As illustrated in Figure 15, cuboid detection using three perpendicular planes sometimes failed when boxes were stacked. Basically, 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 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, detection fails.

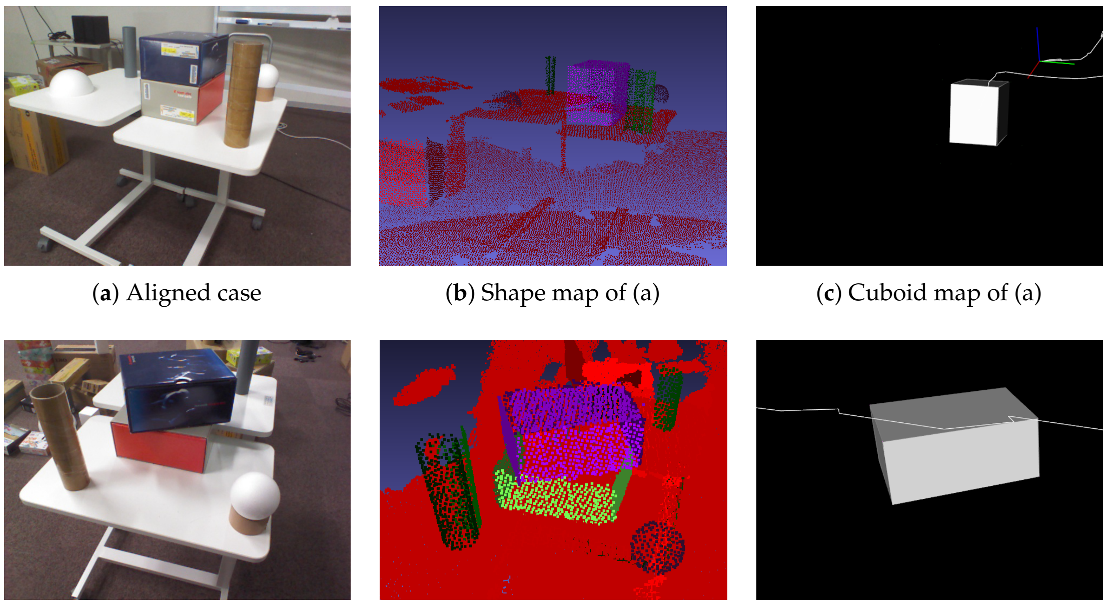

(**a**) Aligned case (**b**) Shape map of (a) (**c**) Cuboid map of (a)
(**d**) Unaligned case (**e**) Shape map of (d) (**f**) Cuboid map of (d)

Figure 15. Limitation. In the first row, two stacked boxes are detected as one cuboid when they are aligned. In 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 cuboid detection can be degraded when boxes are stacked according to the arrangement.

9. 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. In addition, the drift-compensation process is proposed to create the consistent and accurate cuboid map. False detection is further suppressed by using the removed duplicate plane map. An interactive cuboid-modeling system was designed to assist users in reconstructing cuboids.

This evaluation demonstrated that cuboid modeling with our approach is more accurate than a batch-based method. In addition, the evaluation demonstrated that cuboid modeling with the drift-compensation process was more accurate than without drift compensation. Our method also successfully detected cuboids regardless of their arrangements. However, three perpendicular planes are required to be captured to compute cuboid-shape parameters, as our limitation.

In future work, the performance of our framework with various additional scenes will be investigated. Image features from RGB images will also be integrated into our framework to increase the accuracy and robustness of cuboid detection, since the point cloud is obtained by RTAB-MAP [29], which is relatively inaccurate in terms of reconstruction quality compared with a TSDF based fusion method such as KinectFusion [37].

Author Contributions: conceptualization, H.U.; methodology, H.U.; software, M.M.; validation, M.M.; formal analysis, M.M.; investigation, M.M.; data curation, M.M.; writing—original draft preparation, M.M., H.U. and J.P.L.; writing—review and editing, H.U., D.T., R.-i.T., R.R., J.P.L., and V.T.; visualization, M.M.; supervision, H.U., D.T., R.-i.T, R.R. Lima J.P.L. and V.T.; project administration, H.U.; funding acquisition, H.U.

Funding: A part of this research was funded by JSPS KAKENHI grant number JP17H01768.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Saxena, A.; Driemeyer, J.; Ng, A.Y. Robotic grasping of novel objects using vision. *Int. J. Robot. Res.* **2008**, *27*, 157–173. [CrossRef]
2. Hedau, V.; Hoiem, D.; Forsyth, D. Recovering the spatial layout of cluttered rooms. In Proceedings of the IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 1849–1856.
3. 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*; Springer: Berlin, Germany, 2010; pp. 224–237.
4. Del Pero, L.; Guan, J.; Brau, E.; Schlecht, J.; Barnard, K. Sampling bedrooms. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO, USA, 20–25 June 2011; pp. 2009–2016.
5. Xiao, J.; Russell, B.; Torralba, A. Localizing 3D cuboids in single-view images. In *Advances in Neural Information Processing Systems*; Curran Associates: Lake Tahoe, NV, USA, 2012; pp. 746–754.
6. Hejrati, M.; Ramanan, D. Categorizing cubes: Revisiting pose normalization. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016; pp. 1–9.
7. Osada, R.; Funkhouser, T.; Chazelle, B.; Dobkin, D. Shape distributions. *ACM Trans. Graph. (TOG)* **2002**, *21*, 807–832. [CrossRef]
8. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Dolha, M.; Beetz, M. Towards 3D point cloud based object maps for household environments. *Robot. Auton. Syst.* **2008**, *56*, 927–941. [CrossRef]
9. Lin, D.; Fidler, S.; Urtasun, R. Holistic scene understanding for 3d object detection with rgbd cameras. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Lake Tahoe, NV, USA, 1–8 December 2013; pp. 1417–1424.
10. Jiang, H.; Xiao, J. A linear approach to matching cuboids in RGBD images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2013; pp. 2171–2178.

11. Khan, S.H.; He, X.; Bannamoun, M.; Sohel, F.; Togneri, R. Separating objects and clutter in indoor scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 4603–4611.
12. 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, Las Vegas, NV, USA, 27–30 June 2016; pp. 1525–1533.
13. Hashemifar, Z.S.; Lee, K.W.; Napp, N.; Dantu, K. Consistent Cuboid Detection for Semantic Mapping. In Proceedings of the IEEE 11th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, 30 January–1 February 2017; pp. 526–531.
14. Nguyen, T.; Reitmayr, G.; Schmalstieg, D. Structural modeling from depth images. *IEEE Trans. Vis. Comput. Graph.* **2015**, *21*, 1230–1240. [CrossRef] [PubMed]
15. Balsa-Barreiro, J.; Fritsch, D. Generation of 3D/4D photorealistic building models. The testbed area for 4D Cultural Heritage World Project: The historical center of Calw (Germany). In *International Symposium on Visual Computing*; Springer: Berlin, Germany, 2015; pp. 361–372.
16. Balsa-Barreiro, J.; Fritsch, D. Generation of visually aesthetic and detailed 3D models of historical cities by using laser scanning and digital photogrammetry. *Digit. Appl. Archaeol. Cult. Herit.* **2018**, *8*, 57–64. [CrossRef]
17. Reutebuch, S.E.; Andersen, H.E.; McGaughey, R.J. Light detection and ranging (LIDAR): An emerging tool for multiple resource inventory. *J. Forest.* **2005**, *103*, 286–292.
18. Roberto, R.A.; Uchiyama, H.; Lima, J.P.S.; Nagahara, H.; Taniguchi, R.I.; Teichrieb, V. Incremental structural modeling on sparse visual SLAM. *IPSJ Trans. Comput. Vis. Appl.* **2017**, *9*, 5. [CrossRef]
19. Roberto, R.; Lima, J.P.; Uchiyama, H.; Arth, C.; Teichrieb, V.; Taniguchi, R.; Schmalstieg, D. Incremental Structural Modeling Based on Geometric and Statistical Analyses. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 955–963.
20. Pradeep, V.; Rhemann, C.; Izadi, S.; Zach, C.; Bleyer, M.; Bathiche, S. MonoFusion: Real-time 3D reconstruction of small scenes with a single web camera. In Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Adelaide, Australia, 1–4 October 2013; pp. 83–88.
21. Olivier, N.; Uchiyama, H.; Mishima, M.; Thomas, D.; Taniguchi, R.; Roberto, R.; Lima, J.P.; Teichrieb, V. Live Structural Modeling using RGB-D SLAM. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6352–6358.
22. Goldman, R. Intersection of three planes. In *Graphics Gems*; Academic Press Professional, Inc.: Cambridge, MA, USA, 1990; p. 305.
23. Mishima, M.; Uchiyama, H.; Thomas, D.; Taniguchi, R.; Roberto, R.; Lima, J.P.; Teichrieb, V. RGB-D SLAM based Incremental Cuboid Modeling. ECCV Workshop 3D Reconstruction in the Wild, 2018. Available online: http://www.sys.info.hiroshima-cu.ac.jp/3drw2018/procs/W17-08.pdf (accessed on 1 January 2019).
24. Dwibedi, D.; Malisiewicz, T.; Badrinarayanan, V.; Rabinovich, A. Deep cuboid detection: Beyond 2d bounding boxes. *arXiv* **2016**, arXiv:1611.10010.
25. Nguatem, W.; Drauschke, M.; Mayer, H. Finding cuboid-based building models in point clouds. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *XXXIX-B3*, 149–154. [CrossRef]
26. Zhang, C.; Hu, Y. CuFusion: Accurate real-time camera tracking and volumetric scene reconstruction with a cuboid. *Sensors* **2017**, *17*, 2260. [CrossRef] [PubMed]
27. Rodriguez-Garavito, C.; Camacho-Munoz, G.; Álvarez-Martínez, D.; Cardenas, K.V.; Rojas, D.M.; Grimaldos, A. 3D object pose estimation for robotic packing applications. In *WEA 2018: Applied Computer Sciences in Engineering*; Springer: Berlin, Germany, 2018; pp. 453–463.
28. Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2007; Volume 26, pp. 214–226.
29. Labbé, M.; Michaud, F. Online global loop closure detection for large-scale multi-session graph-based slam. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), Chicago, IL, USA, 14–18 September 2014; pp. 2661–2666.
30. Labbe, M.; Michaud, F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Trans. Robot.* **2013**, *29*, 734–745. [CrossRef]

31. Stein, S.C.; Wörgötter, F.; Schoeler, M.; Papon, J.; Kulvicius, T. Convexity based object partitioning for robot applications. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 3213–3220.
32. Sinha, S.N.; Steedly, D.; Szeliski, R.; Agrawala, M.; Pollefeys, M. Interactive 3D architectural modeling from unordered photo collections. *ACM Trans. Graph. (TOG)* **2008**, *27*, 159. [CrossRef]
33. 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 Trans. Graph. (TOG)* **2012**, *31*, 136. [CrossRef]
34. Zhang, Y.; Luo, C.; Liu, J. Walk&sketch: Create floor plans with an rgb-d camera. In *Proceedings of the ACM Conference on Ubiquitous Computing*; ACM: New York, NY, USA, 2012; pp. 461–470.
35. 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, Beijing, China, 17–21 September 2011; pp. 75–84.
36. Kim, Y.M.; Mitra, N.J.; Huang, Q.; Guibas, L. Guided Real-Time Scanning of Indoor Objects. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2013; Volume 32, pp. 177–186.
37. 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 Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Basel, Switzerland, 26–29 October 2011; pp. 127–136.