DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382
Reprinted from
Journal of Imaging Science and Technology® 60(2): 020401-1–020401-24, 2016.

Marker-Less Augmented Reality Framework Using On-Site 3D Line-Segment-based Model Generation

Yusuke Nakayama and Hideo Saito
Graduate School of Science and Technology, Keio University, Yokohama, Japan
E-mail: nakayama@hvrl.ics.keio.ac.jp

Masayoshi Shimizu and Nobuyasu Yamaguchi
Fujitsu Laboratories Ltd, Kawasaki, Japan

Abstract. *The authors propose a line-segment-based marker-less augmented reality (AR) framework that involves an on-site model-generation method and on-line camera tracking. In most conventional model-based marker-less AR frameworks, correspondences between the 3D model and the 2D frame for camera-pose estimation are obtained by feature-point matching. However, 3D models of the target scene are not always available, and feature points are not detected from texture-less objects. The authors' framework is based on a model-generation method with an RGB-D camera and model-based tracking using line segments, which can be detected even with only a few feature points. The camera pose of the input images can be estimated from the 2D–3D line-segment correspondences given by a line-segment feature descriptor. The experimental results show that the proposed framework can achieve AR when other point-based frameworks cannot. The authors also argue that their framework can generate a model and estimate camera pose more accurately than their previous study.* © *2016 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2016.60.2.020401]

INTRODUCTION

Augmented reality (AR) is a technology that places additional information generated by a computer into the real world. It enhances a user's perception of the real world. Camera-pose estimation is a fundamental part of AR. Camera-pose estimation for AR is generally considered as estimating the rotation matrix and the translation vector from the matching between the 2D image and the 3D real world. Conventional methods for camera-pose estimation can be categorized into two groups. One is based on tracking of markers, which is called visual marker AR, and the other is called marker-less AR. With many marker-less AR methods,[1,2] correspondences between 2D and 3D are obtained by feature-point matching, and then the camera pose is computed. Parallel tracking and mapping for small AR workspaces (PTAM),[1] which is a typical marker-less AR method, also uses features from accelerated segment test (FAST)[3] as a feature-point detector and estimates camera pose. However, in a scene where only a few feature points are detected, the 2D–3D point correspondences cannot be obtained, and estimation of the camera pose will fail. In man-made situations that contain texture-less objects, feature points are rarely detected. Therefore, feature-point-based AR methods, such as PTAM, cannot estimate the camera pose in this situation. To achieve marker-less AR in these kinds of situations, another scene feature is needed instead.

One solution to this problem is to use line-segment features instead of point features, because line segments are detected even when only a few feature points are detected. There are several methods[4,5] for marker-less camera tracking based on line segments. These methods involve a model-based approach in which the camera pose is estimated from the line-segment correspondences between the 2D image and the 3D model of the target objects given before camera tracking. Although these methods can be applied to texture-less objects, 3D models of target objects are not always available before camera tracking for AR. The creation of accurate 3D models, such as CAD models, or the creation of large-scale models of the target scene is often not easy. Moreover, in these kinds of texture-less scenes, scene reconstruction methods such as structure-from-motion (SfM)[6] cannot be applied because they often use feature-point matching for recovering the 3D geometrical model. Therefore, on-site model generation for the target scene is needed, especially for target scenes that are not known beforehand and do not provide sufficient feature points.

Therefore, the objective of this research is to generate an accurate 3D model for the target scene on site and estimate the camera pose from line-segment correspondences between the 2D image and the generated 3D model for achieving marker-less AR, even though only a few feature points are detected from the scene.

We propose a line-segment-based marker-less AR framework that involves on-site 3D model generation and on-line camera tracking. This proposed marker-less AR framework is an extension of our previous work.[7] In this previous work,[7] we introduced a marker-less AR framework which estimates the camera pose from 2D–3D line-segment correspondences between the 2D image and the 3D line-segment-based model. This previous framework[7] used the 3D line-segment-based model generated by our previous

Received June 30, 2015; accepted for publication Nov. 10, 2015; published online Dec. 18, 2015. Associate Editor: Henry Y. T. Ngan.
1062-3701/2016/60(2)/020401/24/$25.00

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

model-generation method.[8] However, the generated model was not accurate in some situations. Therefore, the camera pose estimated with the model was also inaccurate. On the other hand, our marker-less AR framework proposed in this article first generates a more accurate 3D line-segment-based model by a new method. This new model-generation method is mainly based on the previous model-generation method,[8] and changes the 3D line-segment-creation method to one using plane clustering and an outlier-elimination algorithm for the 2D–3D line-segment matching. Moreover, we add a bundle adjustment, so that the accuracy of the generated model is improved. With this more accurate 3D line-segment-based model, in our proposed framework, the tracking accuracy is also improved from that of the previous framework.[7]

Our main contributions are as follows.

- We propose a line-segment-based marker-less AR framework, which involves on-site model generation and on-line camera tracking, so that we can achieve AR without knowing the 3D geometry of a target scene that has no texture.
- With plane segmentation and a bundle adjustment, we improve upon our previous work and generate an accurate 3D line-segment-based model of the target scene.
- The accuracy of camera-pose estimation is also improved using the model generated with the new method.

We experimentally demonstrated that our proposed framework can achieve marker-less AR in a texture-less scene. We also demonstrated that the accuracy of the camera-pose estimation in both the model-generation phase and the camera-tracking phase is improved from our previous study.

PROPOSED FRAMEWORK

In this section, we explain our proposed marker-less AR framework, which consists of on-site model-generation and on-line camera-tracking phases. The model-generation phase is for obtaining the 3D geometry of the real-world scene to be augmented. We focus on target scenes mainly containing texture-less objects. After this preliminary model-generation phase, the camera pose of the input frame is computed from the matches between the current 2D frame and the 3D model.

On-site Model Generation

First, we explain how to recover the 3D geometry of a target scene as a 3D model. This model-generation phase takes as input a set of RGB and depth image sequences of the target scene. These image sequences are acquired from multiple viewpoints by using an RGB-D camera, such as that on Kinect. The scene is assumed to be mostly rigid and has no known structures. Moreover, this scene has only a few textures, and only a few feature points are detected. Therefore, we use line segments instead of feature points. The generated model is represented by line segments. Our new model-generation method is an extension of our previous method.[8] For the accurate 3D line-segment-creation method using plane clustering, the target scene should mainly consist of plane structures.

Suppose that we have N RGB images $\{I^N_{rgb}\}$ and N depth images $\{I^N_d\}$ captured with an RGB-D camera as the input sequences. Our method uses these RGB and depth image sequences to estimate the camera poses for each frame and construct a 3D line-segment-based model of the scene. The camera pose at the ith frame is represented as a 3×4 matrix $\boldsymbol{RT}^i_{cw} = [\boldsymbol{R}_i \mid \mathbf{t}_i]$ containing a 3×3 rotation matrix ($\boldsymbol{R}_i$) and a 3D translation vector ($\mathbf{t}_i$). This $\boldsymbol{RT}^i_{cw}$ is a transfer matrix from the ith camera coordinate $O^i_c X^i_c Y^i_c Z^i_c$ to the world coordinate $O_w X_w Y_w Z_w$. With this method, each frame's camera pose $\boldsymbol{RT}^i_{cw}$ is estimated with its previous frame's camera pose $\boldsymbol{RT}^{i-1}_{cw}$. Then, the 2D line segments on the RGB images are back-projected into the 3D world coordinate as a part of the model by the estimated camera pose $\boldsymbol{RT}^i_{cw}$. In the following subsections, we discuss the relative geometry of two consecutive frames, the $(i-1)$th and ith frames. Therefore, we explain how to compute $\boldsymbol{RT}^i_{cw}$ using the known $(i-1)$th frame's camera pose $\boldsymbol{RT}^{i-1}_{cw}$. We set the first frame's camera pose $\boldsymbol{RT}^0_{cw}$ as a 3×4 identity matrix. This is because we assume that the world coordinate $O_w X_w Y_w Z_w$ is defined by the camera coordinate of the first frame $O^0_c X^0_c Y^0_c Z^0_c$. The positional relationship between the target scene and each frame is shown in Figure 1.

2D Line-Segment Detection

We first detect 2D line segments from I^{i-1}_{rgb} and I^i_{rgb}. To do this, we simply use the fast line-segment detector (LSD) algorithm.[9] This algorithm simply returns the two end points of each detected 2D line segment without discriminating which point is the starting point. The two sets of line segments detected from I^{i-1}_{rgb} and I^i_{rgb} with the LSD algorithm are defined as $\mathscr{L}^{i-1} = \{l^j_{i-1} \mid 0 \le j \le M_{i-1}\}$ and $\mathscr{L}^i = \{l^j_i \mid 0 \le j \le M_i\}$, respectively, in which l_i is a 2D line segment detected from I^i_{rgb}, and the number of detected 2D line segments from I^i_{rgb} is $M_i + 1$.

3D Line-Segment Creation with Plane Structure

Next, we generate 3D line segments with the detected 2D line segments ($\mathscr{L}^{i-1}$ and $\mathscr{L}^i$) and the depth values from I^{i-1}_d and I^i_d. With our previous model-generation method,[8] we take points on the 2D line segment and obtain the corresponding depth value from the depth image. Then, the points are back-projected from the 2D image to the 3D camera coordinate. These back-projected points construct a 3D line segment. However, this procedure of 3D line-segment creation results in some false line segments being detected. Figure 2 shows an example of an input RGB image of one shelf in front of a wall. Figure 3 shows an image with detected 2D line segments used for 3D line-segment creation. In Fig. 3, the green 2D line segments are detected and back-projected

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

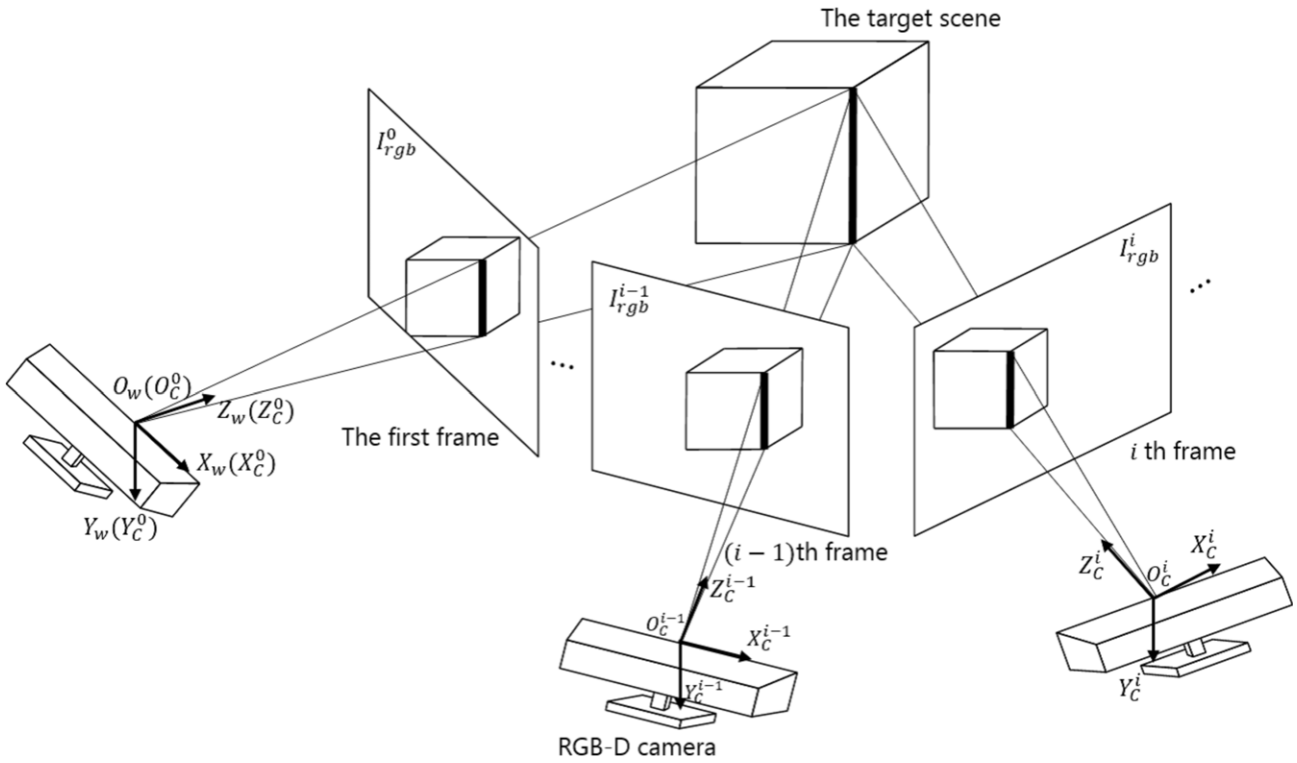


Figure 1. The geometrical relationship between the target scene and every frame.

Figure 2. Example of input RGB image.

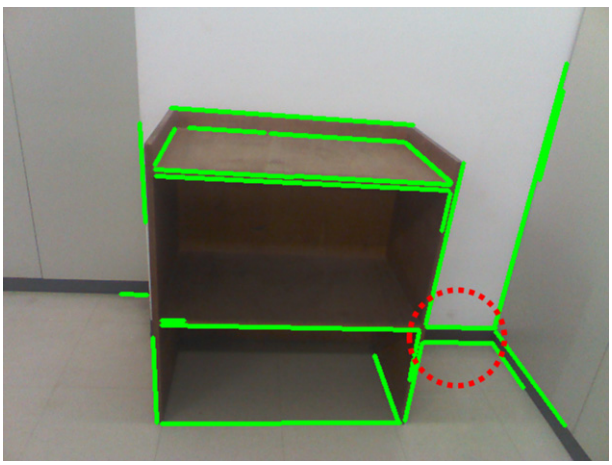


Figure 3. 2D line segments that are back-projected.

into the 3D camera coordinate with our previous method;[8] these line segments are indicated in Figure 4. The red circles in Fig. 4 indicate incorrectly created 3D line segments. These 3D line segments were back-projected from the 2D line segments indicated by the red circle in Fig. 3. The 2D line segments were detected from the wall; however, the created 3D line segments spanned from the front side of the shelf to the wall. This is because Kinect's RGB image and depth image had some misalignment. We simply used the Kinect for Windows software development kit (SDK) to obtain RGB images and depth images. The SDK provides an alignment function for the RGB and depth images, but there are still errors between them. This fact also can be seen in Figure 5, which represents the colored point cloud of the frame. The right image is an enlargement of the area within the red rectangle. As shown in Fig. 5, some parts of the wall are placed in front of the shelf. This is why the false 3D line segments were created.

To avoid this false 3D line-segment creation, we use a 3D line-segment creation method that fits line segments onto planar structures. First, we perform a plane segmentation of the 3D point cloud generated from an input depth image using the Point Cloud Library (PCL) from Willow Garage. With this segmented result of the point cloud, the input depth

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

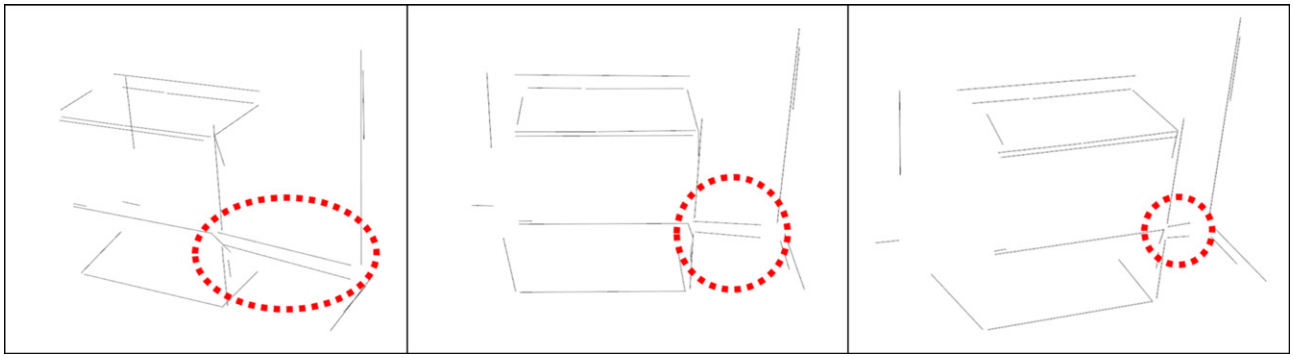


Figure 4. 3D line segments of the example frame with our previous model-generation method.[8]

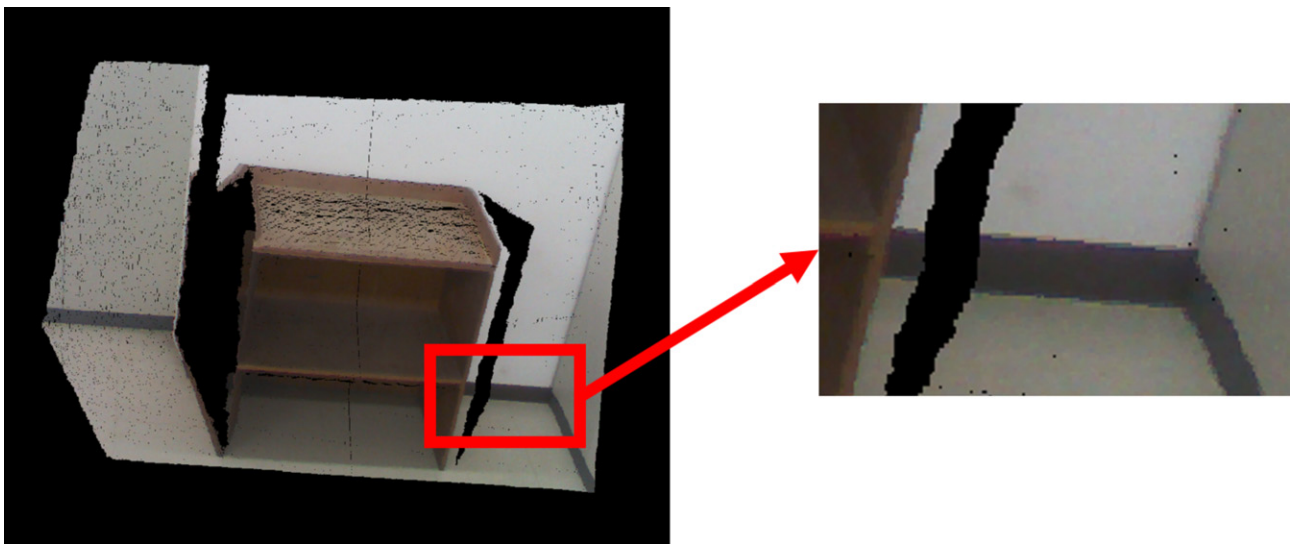


Figure 5. Colored point cloud of frame.

image is also segmented into multiple planar areas. Each pixel in the RGB image is corresponded to the pixel of the depth image, so that the RGB can also be segmented by detected planes, as shown in Figure 6. The same colored area is given the same plane ID in this figure. Next, we put lattice points on a 2D line segment l_i^j. According to the plane-segmentation result of the RGB image, each lattice point has the ID of the plane to which it belongs. Then, the plane ID area is counted for all of the lattice points on l_i^j, and the most common plane ID is regarded as the plane ID of l_i^j. Once the plane ID of l_i^j is obtained, we can choose lattice points that have the same plane ID as l_i^j to use back-projection. These selected 2D lattice points are translated from the image coordinate to the 3D camera coordinate with the corresponding depth value from I_d^i. The two points located at either end of the translated 3D points are assumed as tentative start 3D points and end 3D points. Next, we compute a 3D line that minimizes the lengths of perpendiculars from each 3D point to the 3D line. Finally, we choose the extremities of the perpendiculars from the tentative start and end points as a fixed start 3D point and a fixed end 3D point. Therefore, by connecting these fixed start and end 3D points, we can create a 3D line segment. Figure 7 shows the 3D line segments created with this line-segment creation method from the example frame. Compared with the 3D line segments shown in Fig. 4, incorrectly created line segments are modified. This 3D line-segment creation method is applied to all line segments in $\mathscr{L}^{i-1}$ and $\mathscr{L}^i$. We can then obtain sets of 3D line segments represented as $\mathscr{L}_c^{i-1} = \{L_{c,i-1}^j \mid 0 \le j \le M_{i-1}\}$ and $\mathscr{L}_c^i = \{L_{c,i}^j \mid 0 \le j \le M_i\}$, respectively, where $L_{c,i}$ is the back-projected 3D line segment from l_i into the ith frame's camera coordinate. Moreover, these 3D line segments in $\mathscr{L}_c^{i-1}$ are translated from the camera coordinate to the world coordinate by the known $\boldsymbol{RT}_{cw}^{i-1}$. Then, the translated 3D line segments in the world coordinate are represented as $\mathscr{L}_w^{i-1} = \{L_{w,i-1}^j \mid 0 \le j \le M_{i-1}\}$, where $L_{w,i}$ is a translated 3D line segment from the ith frame's camera coordinate to the world coordinate.

2D Line-Segment Matching by using Directed LEHF

Next, we achieve 2D line-segment matching between two frames by descriptor matching. In this method, we use the directed line-based eight-directional histogram feature

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

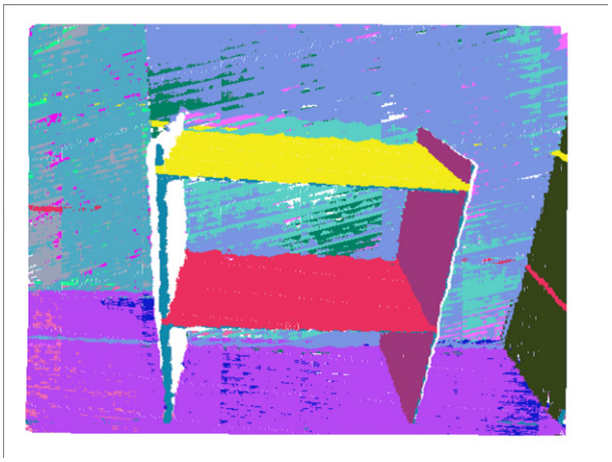


Figure 6. Results of labeling with detected planes.

(LEHF)[10] as a line-segment feature descriptor. The directed LEHF is an improved version of the fast line-segment descriptor LEHF proposed by Hirose et al.[11] In this method, we set the directed-LEHF descriptor as a 112-dimensional vector.

We extract the directed-LEHF descriptor value from each 2D line segment in $\mathscr{L}^{i-1}$ and $\mathscr{L}^{i}$, then obtain a set of matching 2D line segments between them. This set is represented as $\mathscr{LM}^{i} = \{(l_{i-1}^{g(k)}, l_{i}^{f(k)}), k = 0, 1, \ldots, K^{i}\}$, in which $(l_{i-1}^{g(k)}, l_{i}^{f(k)})$ represents a pair of matching 2D line segments, $g(k) \in [0, M_{i-1}]$ and $f(k) \in [0, M_i]$.

2D–3D Line-Segment Correspondences

For estimating the camera pose, we should obtain 2D–3D line-segment correspondences. According to the method described above, we can obtain the matching information of 2D line segments $\mathscr{LM}^{i}$ between the two frames. The matched 2D line segments in the $(i-1)$th frame, in other words, $l_{i-1}^{g(k)}$ from $\mathscr{LM}^{i}$, are back-projected to 3D line segments $L_{c,i-1}^{g(k)}$ in the $(i-1)$th camera coordinate $O_c^{i-1}X_c^{i-1}Y_c^{i-1}Z_c^{i-1}$. These back-projected 3D line segments $L_{c,i-1}^{g(k)}$ should be in $\mathscr{L}_c^{i-1}$. We know the $\boldsymbol{RT}_{cw}^{i-1}$; therefore, the back-projected 3D line segments are translated from the $(i-1)$th camera coordinate to the world coordinate. These translated 3D line segments are represented as $L_{w,i-1}^{g(k)}$, which should be in $\mathscr{L}_w^{i-1}$. With the 2D line-segment matching information $(l_{i-1}^{g(k)}, l_{i}^{f(k)})$, we can correspond the 2D line segments in the ith frame $l_i^{f(k)}$ and the 3D line segments in the world coordinate $L_{w,i-1}^{g(k)}$, which are back-projected from $l_{i-1}^{g(k)}$. This set of 2D–3D line-segment correspondences is represented as $\mathscr{LC}^{i} = \{(L_{w,i-1}^{g(k)}, l_{i}^{f(k)}), k = 0, 1, \ldots, K^{i}\}$, where $(L_{w,i-1}^{g(k)}, l_{i}^{f(k)})$ represents a pair of 2D–3D line-segment correspondences $g(k) \in [0, M_{i-1}]$ and $f(k) \in [0, M_i]$. This procedure for obtaining these correspondences is illustrated in Figure 8.

Camera-Pose Estimation

Given a set of 2D–3D line-segment correspondences, we solve the Perspective-n-Lines (PnL) problem, then estimate the camera pose. The PnL problem is a counterpart of the perspective-n-point (PnP) problem for point correspondences. However, the correspondences $\mathscr{LC}^{i}$ may have some mismatching. Therefore, the camera pose estimated by all 2D–3D line-segment correspondences in $\mathscr{LC}^{i}$ has the potential to be inaccurate. With our previous model-generation method,[8] the camera-pose estimation method used RANSAC[12] for outlier elimination. Its algorithm requires parameter tuning about the threshold for determining inlier correspondences or outlier correspondences. Although we obtain the 2D–3D line-segment correspondences from every two consecutive frames, the threshold is determined to be constant in every frame. Because of the invariable threshold, outlier elimination with the algorithm may fail and the camera pose cannot be obtained according to the circumstances of the two frames. Therefore, to estimate the accurate camera pose in every two consecutive frames, our new model-generation method involves a camera-pose estimation method for the 2D–3D line-segment correspondences using the least-median-of-squares (LMedS) method. This method determines the threshold for the outlier elimination adaptively in every two frames.

We first randomly select four 2D–3D line-segment correspondences from $\mathscr{LC}^{i}$, which is the set of K^{i} 2D–3D

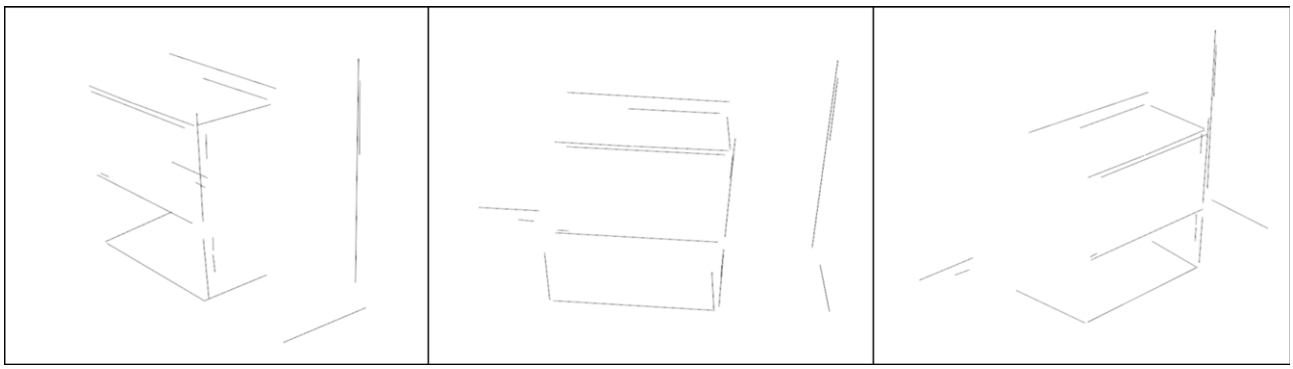


Figure 7. 3D line segments of the example frame with the new model-generation method.

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

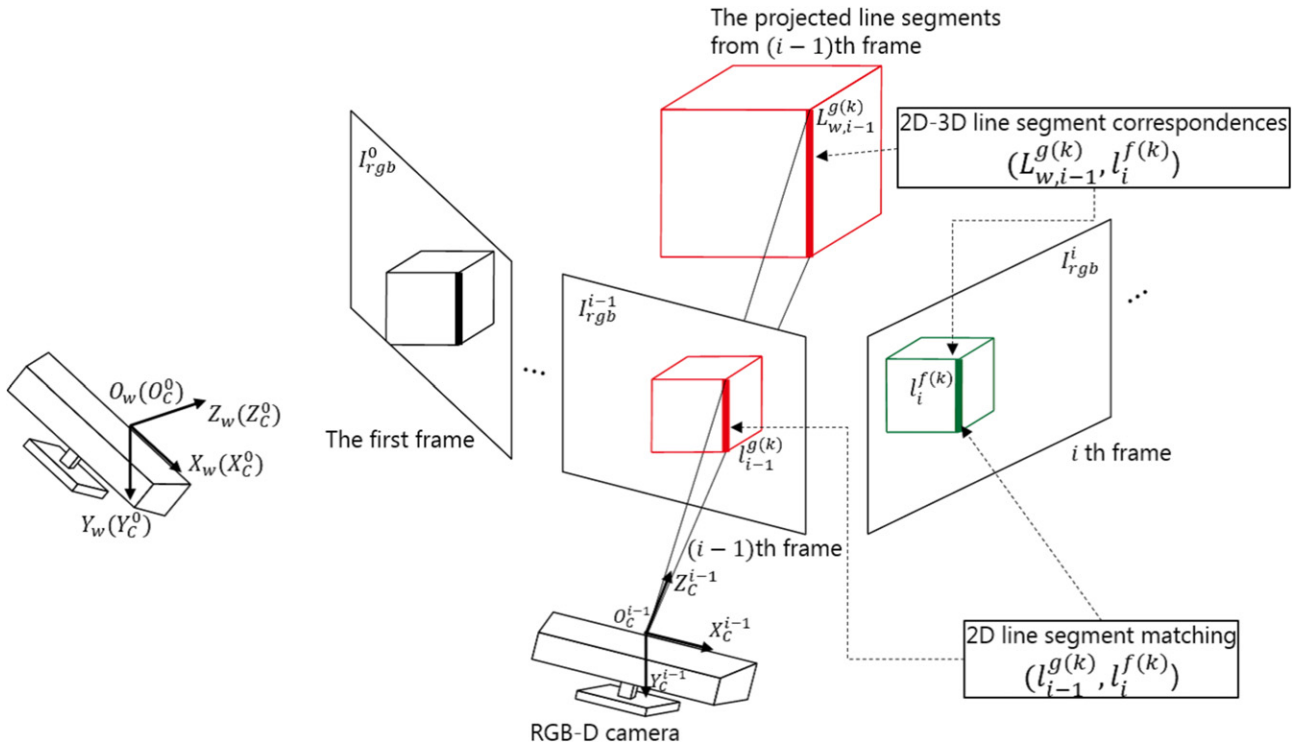


Figure 8. The geometrical relationship between the back-projected 3D line segments and each frame.

line-segment correspondences. Let the set of four 2D–3D line-segment correspondences be represented as $\mathscr{LC}^i_{\text{four}} = \{(L_{w,i-1}^{a(m)}, l_i^{b(m)}), m = 0, 1, 2, 3\}$, where $(L_{w,i-1}^{a(m)}, l_i^{b(m)})$ represents four pairs of 2D–3D line-segment correspondences, $a(m) \in [0, M_{i-1}]$ and $b(m) \in [0, M_i]$. Then, the rest of the $(K^i - 4)$ correspondences are represented as $\mathscr{LC}^i_{\text{rest}} = \{(L_{w,i-1}^{g(k)}, l_i^{f(k)}) \mid 0 \le k \le K^i, g(k) \ne a(m), f(k) \ne b(m), m = 0, 1, 2, 3\}$. With $\mathscr{LC}^i_{\text{four}}$, we solve the PnL problem using a robust perspective-n-line (RPnL) solution,[13] which is a method for camera-pose estimation from 2D–3D line-segment correspondences. It takes $\mathscr{LC}^i_{\text{four}}$ as input then provides the estimated camera pose $fourRT_{cw}$ as output. The 3D line segments in $\mathscr{L}^i_c$, which are back-projected from $l_i^{f(k)}$ in $\mathscr{LC}^i_{\text{rest}}$, are translated from the ith camera coordinate into the world coordinate by $fourRT_{cw}$. Let the 3D line segments in the world coordinate translated by $fourRT_{cw}$ be $L_{w,i}^{'f(k)}$. We calculate the error $e(L_{w,i-1}^{g(k)}, L_{w,i}^{'f(k)})$ between $L_{w,i-1}^{g(k)}$ from $\mathscr{LC}^i_{\text{rest}}$ and $L_{w,i}^{'f(k)}$. The error $e(L^p, L^q)$ between two 3D line segments L^p, L^q is defined as

$$e(L^p, L^q) = S(L^p, L^q)/(\text{length}(L^p) + \text{length}(L^q)), \quad (1)$$

where $S(L^p, L^q)$ is the total area of two triangles obtained by connecting L^p's start and end points and L^q's start point and connecting L^q's start and end points and L^p's end point, respectively. The error $E(fourRT_{cw})$ given by $fourRT_{cw}$ is defined as

$$E(fourRT_{cw}) = \underset{k=0,\ldots,K^i}{\text{median}} \{e^2(L_{w,i-1}^{g(k)}, L_{w,i}^{'f(k)})\}, \quad (2)$$

where $g(k) \ne a(m), f(k) \ne b(m), m = 0, 1, 2, 3$.

We also randomly select another set of $\mathscr{LC}^i_{\text{four}}$ and repeat the steps explained above $N_{\text{iteration}}$ times to estimate $fourRT_{cw}$. We choose the $fourRT_{cw}$ that gives the minimum $E(fourRT_{cw})$ as a tentative camera pose $tentativeRT_{cw}$, and this minimum error $E(fourRT_{cw})$ is defined as ϵ. Next, using $tentativeRT_{cw}$, all of the K^i 3D line segments in $\mathscr{L}^i_c$, which are back-projected from $l_i^{f(k)}$, are translated to the 3D line segments in the world coordinate $L_{w,i}^{f(k)}$. For each translated 3D line segment, we then compute $e(L_{w,i-1}^{g(k)}, L_{w,i}^{f(k)})$. If $e(L_{w,i-1}^{g(k)}, L_{w,i}^{f(k)})$ is less than a threshold $2.5\hat{\sigma}$, we save the 2D–3D line-segment correspondence $(L_{w,i-1}^{g(k)}, l_i^{f(k)})$ as an inlier. The term $\hat{\sigma}$ is the standard deviation of the error and is defined as follows:[14]

$$\hat{\sigma} = 1.4826 \left(1 + \frac{5}{K^i - 4}\right) \sqrt{\epsilon}. \quad (3)$$

Finally, we compute the ith camera pose using another algorithm for the PnL problem proposed by Kumar and Hanson.[15] This algorithm estimates the camera pose iteratively. It requires a set of 2D–3D line-segment correspondences and the initial camera pose as inputs for pose estimation. This algorithm takes the inliers and $tentativeRT_{cw}$

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

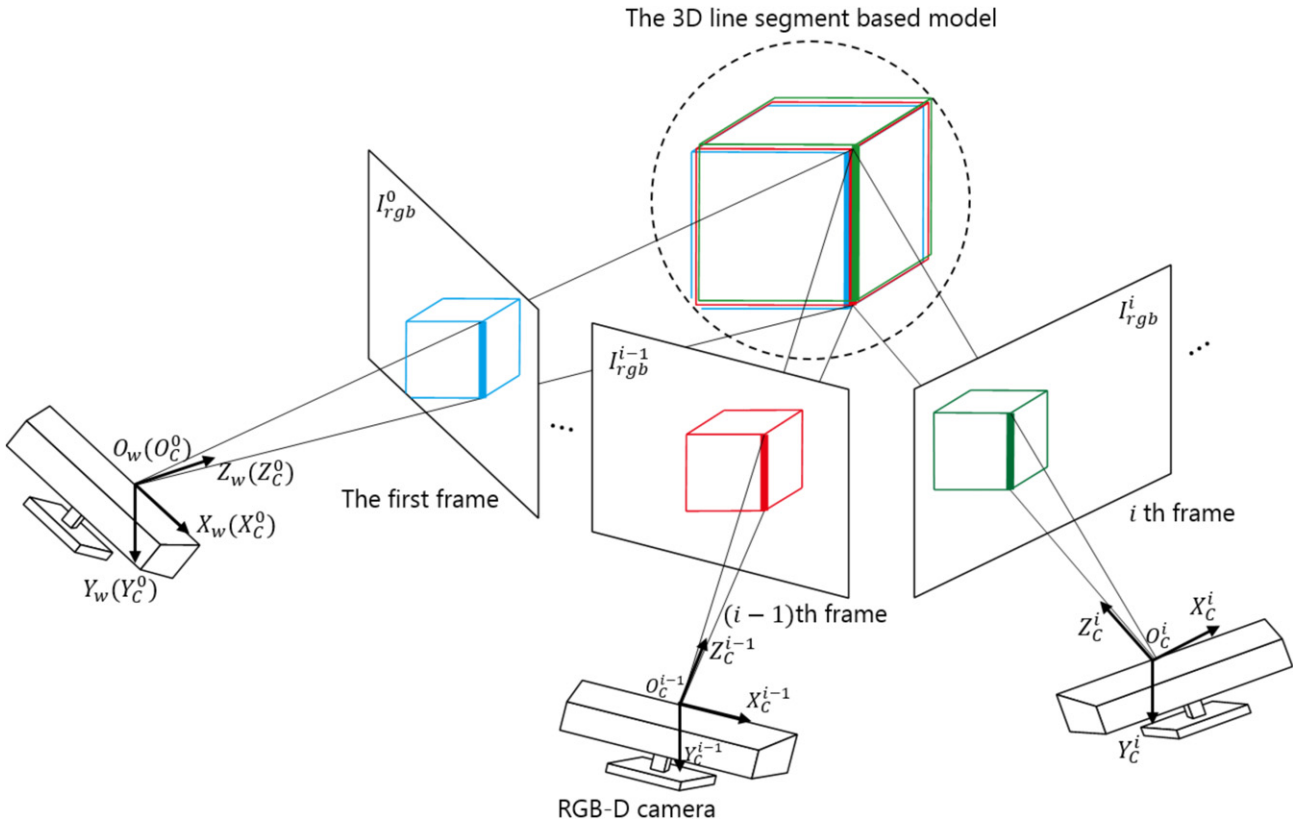


Figure 9. 3D line-segment-based model.

as inputs. We then obtain the estimated camera pose of the ith frame, $\boldsymbol{RT}^i_{cw}$, as the output of the algorithm.

3D Line-Segment-Based Model Generation with Bundle Adjustment

Once $\boldsymbol{RT}^i_{cw}$ is obtained, all of the 3D line segments in the ith camera coordinate $\mathscr{L}^i_c$ are translated into the 3D line segments in the world coordinate $\mathscr{L}^i_w$. The procedures explained above are repeated in every two consecutive frames, so that each frame's camera pose and 3D line segments in the world coordinate can be obtained. All of the 3D line segments from all N frames finally become a 3D line-segment-based model. Figure 9 shows the concept of this 3D line-segment-based model generation.

The procedures explained above are based on an incremental tracking approach of model construction. Therefore, errors are accumulated in the estimated camera pose and 3D line segments as the number of frames is increased. To reduce the accumulated errors, we use a bundle adjustment. Here, we discuss a method to refine the $\boldsymbol{RT}^i_{cw}$ of every N frames. First, we obtain sets of the correspondences of line segments across all frames with reference to the inliers used in the step of solving the PnL problem. From these sets, we create groups of 3D line segments in each frame's $\mathscr{L}^i_c$. Suppose that T groups of 3D line segments are obtained, and the tth group is represented as

$$\mathscr{LG}^t = \{L^{M_i(t)}_{c,i}, i = 0, 1, \ldots, N\}, \tag{4}$$

in which $L^{M_i(t)}_{c,i}$ represents the $M_i(t)$th 3D line segment in the ith camera coordinate, and $M_i(t) \in [0, M_i]$. Each $L^{M_i(t)}_{c,i}$ in the tth group is translated to the world coordinate by $\boldsymbol{RT}^i_{cw}$, and the translated 3D line segments are represented as $L^{M_i(t)}_{w,i}$. This translation is defined as $tl(\boldsymbol{RT}^i_{cw}, L^{M_i(t)}_{c,i})$. If each $\boldsymbol{RT}^i_{cw}$ has no error, every $L^{M_i(t)}_{w,i}$ should overlap in one 3D line in the world coordinate. This 3D line is denoted as L^t_g. $\boldsymbol{RT}^i_{cw}$ is parametrized by using a quaternion as $\boldsymbol{RT}^i_{cw} = (q^i_0, q^i_1, q^i_2, q^i_3, tx^i, ty^i, tz^i)$. L^t_g is parametrized as $L^t_g = (\mathbf{v}^t_g, \mathbf{p}^t_g) = (vx^t_g, vy^t_g, vz^t_g, px^t_g, py^t_g, pz^t_g)$, where $\mathbf{v}^t_g$ is the direction of L^t_g and $\mathbf{p}^t_g$ is one point on L^t_g. This bundle adjustment minimizes the error between each $L^{M_i(t)}_{w,i}$ and L^t_g for all line groups; specifically,

$$\min_{L^t_g, \boldsymbol{RT}^i_{cw}} \sum_t^T \sum_i^N d^2_s(L^{M_i(t)}_{w,i}, L^t_g) + d^2_e(L^{M_i(t)}_{w,i}, L^t_g), \tag{5}$$

in which $L^{M_i(t)}_{w,i} = tl(\boldsymbol{RT}^i_{cw}, L^{M_i(t)}_{c,i})$, and $d_s()$ and $d_e()$ are the distances of the line segment's endpoints to the 3D line. With respect to the line segments that are not seen from RGB images, they are not taken into account in the computation of the objective function. We use the Levenberg–Marquardt algorithm of the MATLAB lsqnonlin function for minimizing the objective function. After this minimization, each $\boldsymbol{RT}^i_{cw}$ of a frame is optimized.

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

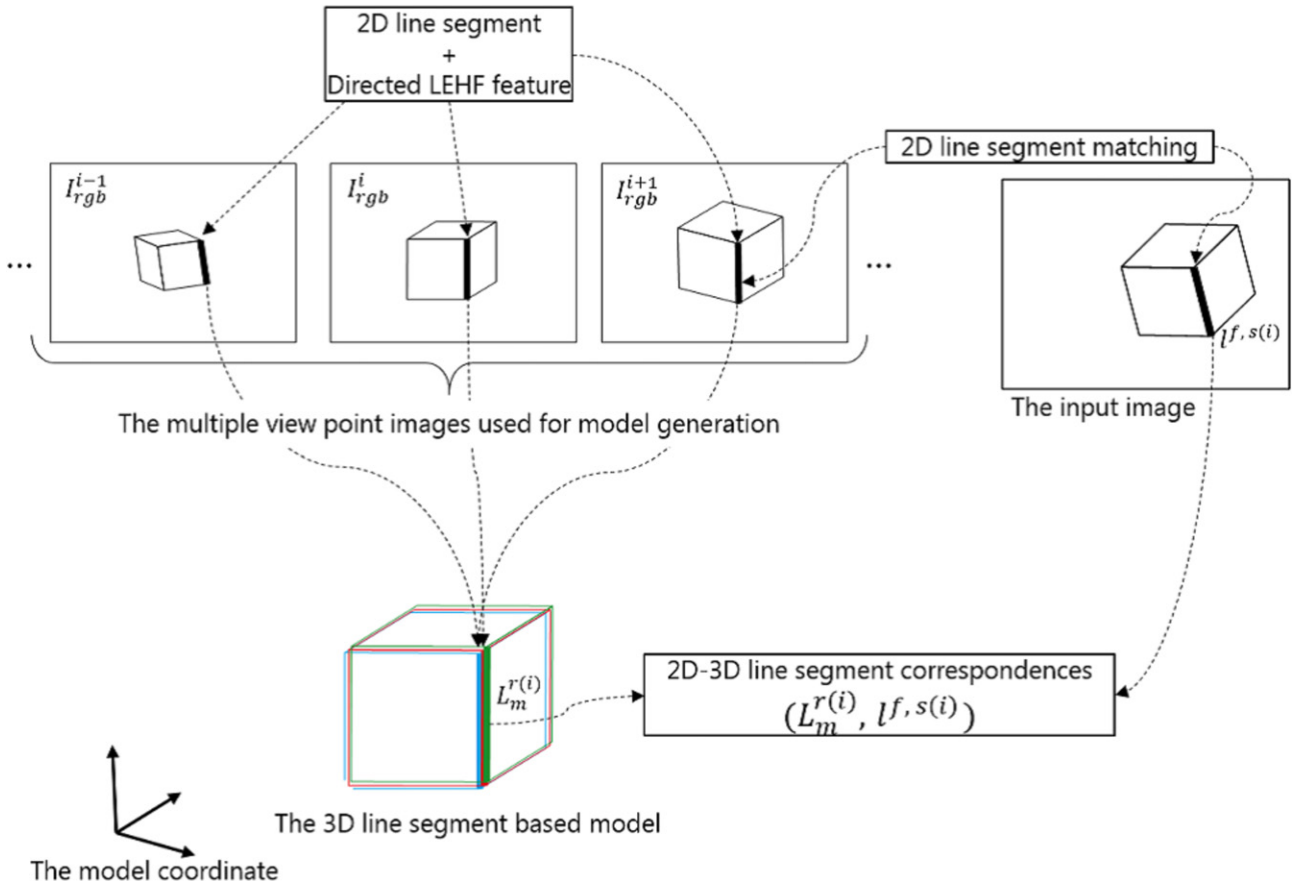


Figure 10. Overview of obtaining the 2D–3D line-segment correspondences.

Therefore, the 3D line-segment-based model generated with the optimized $\boldsymbol{RT}_{cw}^{i}$ is refined.

3D Line-Segment Database Construction

For use in the subsequent on-line camera-tracking phase, we also construct a 3D line-segment database that contains the 3D line segments' positions in the world coordinate and their directed-LEHF descriptor values. The 3D line-segment database has a k-dimensional (k-d) tree of 2D line-segment directed LEHFs and each one has a reference to its 3D back-projected line segment's position. For more details on the construction of this database, please see our previous study.[7]

On-line Camera-Pose Estimation

After generating the 3D line-segment-based model, we use a monocular camera to capture the target scene and achieve marker-less AR. To do this, the input frame's camera pose is estimated for live augmentation with the 3D line-segment database. In this subsection, we explain how to estimate the camera pose $\boldsymbol{RT}_{cm}$, which is the transform matrix from the model coordinate to the input frame's camera coordinate. This camera-pose estimation method is based on our previous study.[7] We assume that the camera's intrinsic parameter is already known.

First, 2D line segments are detected from the input image by using the LSD algorithm. Let these detected 2D line segments from the image be $\mathscr{L}^f = \{l^{f,0}, l^{f,1}, \ldots, l^{f,N}\}$. For each line segment in $\mathscr{L}^f$, the directed LEHF is extracted.

The directed LEHFs from the input image are matched to the features in the 3D line-segment database by a nearest neighbor search. Each feature stored in the database has its link to the 3D line segment, then the 2D line segments from the input image and the 3D line segments from the database are matched. Let the 3D line segments in the database be $\mathscr{L}_m = \{L_m^0, L_m^1, \ldots, L_m^M\}$. The set of 2D–3D line-segment correspondences is represented as $\mathscr{LC}^f = \{(L_m^{r(i)}, l^{f,s(i)}), i = 0, 1, \ldots, K\}$, where $(L_m^{r(i)}, l^{f,s(i)})$ represents a pair of 2D–3D line-segment correspondences, $r(i) \in [0, M]$ and $s(i) \in [0, N]$. The process for obtaining these correspondences is shown in Figure 10.

With a set of 2D–3D line-segment correspondences, we solve the PnL problem to estimate the camera pose. To eliminate the mismatching contained in $\mathscr{LC}^f$, we use a method for solving the PnL problem with an algorithm such as RANSAC.[12] We then estimate $\boldsymbol{RT}_{cm}$. Suppose that we have $\mathscr{LC}^f$, which is a set of K 2D–3D line-segment correspondences, we randomly select four 2D–3D line-segment correspondences from $\mathscr{LC}^f$. Let these four correspondences be represented as $\mathscr{LC}^f_{\text{four}} = \{(L_m^{a(j)}, l^{f,b(j)}), j = 0, 1, 2, 3\}$,

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

(a)

(b)

Figure 11. Input images for model generation of the target scene. (a) Corner of room; (b) window in empty room.

in which $(L_m^{a(j)}, l^{f,b(j)})$ represents the four pairs of 2D–3D line-segment correspondences $a(j) \in [0, M]$ and $b(j) \in [0, N]$. Then, the rest of the $(K-4)$ correspondences are represented as $\mathscr{LC}^f_{\text{rest}} = \{(L_m^{r(i)}, l^{f,s(i)}) \mid 0 \le i \le K, r(i) \ne a(j), s(i) \ne b(j), j = 0, 1, 2, 3\}$. With $\mathscr{LC}^f_{\text{four}}$, we solve the PnL problem using RPnL and estimate the camera pose $fourRT_{cm}$. All of the 3D line segments $L_m^{r(i)}$ in $\mathscr{LC}^f_{\text{rest}}$ are projected to the image frame by using the camera's intrinsic parameter and $fourRT_{cm}$. The projected 2D line segments are defined as $l_m^{'r(i)}$. We then have pairs of the projected 2D line segments $l_m^{'r(i)}$ and the 2D line segments detected from the input image $l^{f,s(i)}$. We calculate the error $e_{2D}(l_m^{'r(i)}, l^{f,s(i)})$ between $l_m^{'r(i)}$ and $l^{f,s(i)}$. We define the error between two 2D line segments $e_{2D}(l^p, l^q)$ as

$$e_{2D}(l^p, l^q) = S(l^p, l^q)/(\text{length}(l^p) + \text{length}(l^q)), \quad (6)$$

where $S(l^p, l^q)$ is the area of a quadrilateral obtained by connecting the four end points of l^p and l^q.

The total error $E_{2D}(fourRT_{cm})$ given by $fourRT_{cm}$ is defined as

$$E_{2D}(fourRT_{cm}) = \sum_{i=0}^{K} e_{2D}(l_m^{'r(i)}, l^{f,s(i)}), \quad (7)$$

where $r(i) \ne a(j), s(i) \ne b(j), j = 0, 1, 2, 3$.

We also randomly select another set of $\mathscr{LC}^f_{\text{four}}$. Then, the steps explained above are repeated $N_{\text{RANSAC\_INPUT}}$ times to estimate $fourRT_{cm}$. We choose the $fourRT_{cm}$ that gives the minimum $E_{2D}(fourRT_{cm})$ as a tentative camera pose $tentativeRT_{cm}$. Using the camera's intrinsic parameter and $tentativeRT_{cm}$, all of the 3D line segments $L_m^{r(i)}$ in $\mathscr{LC}^f$ are then projected to the image frame and their projected 2D line segments $l_m^{r(i)}$ are obtained. We then calculate $e_{2D}(l_m^{r(i)}, l^{f,s(i)})$. If $e_{2D}(l_m^{r(i)}, l^{f,s(i)})$ is less than a threshold (TH_INPUT_e), we save the 2D–3D line-segment correspondences $(L_m^{r(i)}, l^{f,s(i)})$ as inliers.

Finally, we calculate the camera pose of the input frame using the algorithm proposed by Kumar and Hanson. We take the inliers and $tentativeRT_{cm}$ as inputs. We then obtain $\boldsymbol{RT}_{cm}$ of the input frame as an output of the algorithm. Therefore, we can overlay AR content onto the input image from the estimated camera pose.

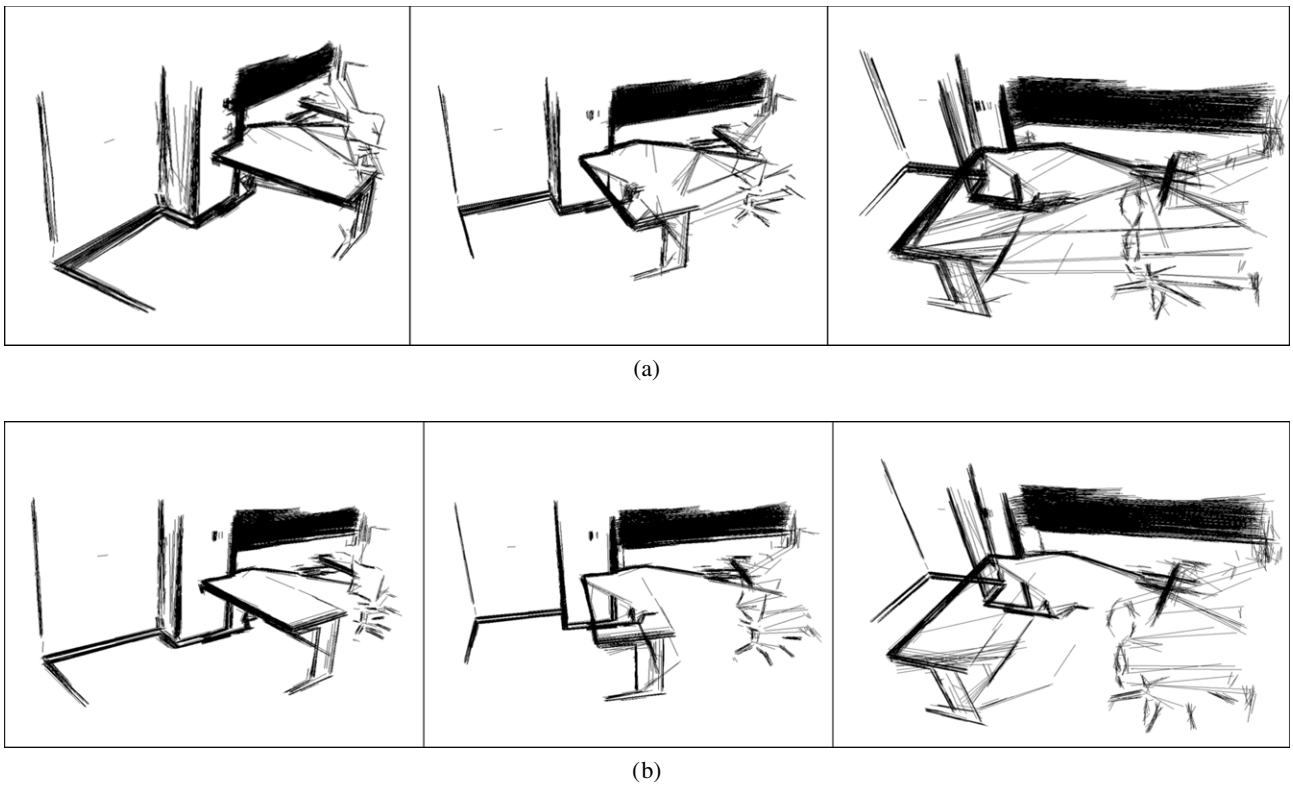
EXPERIMENTS

We conducted experiments to demonstrate that our proposed framework can achieve marker-less AR in a scene containing texture-less objects. We also evaluated the accuracy of the estimated camera pose in both the model-generation and the camera-tracking phases. For comparison, we also tested our previous model-generation method[8] for the model-generation phase, and the previous marker-less AR framework[7] for the camera-tracking phase. In the experiments on the previous framework,[7] we used the 3D line-segment-based model generated by the previous method.[8] In each experiment, we set $N_{\text{iteration}}$ to 1000 for our new model-generation method, and set N_{RANSAC} to 1000 and TH_e to 0.01 for our previous model-generation method.[8] For the camera-tracking phase, we set TH_INPUT_e to 5.

Marker-less AR Result for Texture-Less Object

In this experiment, we argue that our proposed framework, which uses line segments, can achieve AR even when texture-less objects are contained in the scene. We conducted this experiment for the two scenes shown in Figure 11(a) and (b) using Kinect v1, which provided 640 × 480 resolution RGB images and 320 × 240 resolution depth images. Fig. 11(a) shows one corner of the texture-less room and Fig. 11(b) shows the window in the empty room. First, we generated the 3D line-segment-based model in the scenes from 80 input images by using our previous and new model-generation

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382



(a)

(b)

Figure 12. The generated 3D line-segment-based model of the corner scene. (a) Model generated with the previous method;[8] (b) model generated with the new method.

methods. Fig. 11 also shows some of these input images used for model generation. The results of generating the 3D line-segment-based model are shown in Figure 12 for the corner scene and in Figure 13 for the window scene. In each figure, (a) shows the model generated with our previous method[8] and (b) shows the one generated with our new method. As shown in Figs. 12 and 13, our method obtained the 3D geometry of the scene with line segments. Compared with the models generated with our previous method shown in Figs. 12(a) and 13(a), our new method generated more accurate 3D line-segment-based models, which are shown in Figs. 12(b) and 13(b). The bundle adjustment used in our new method refines the estimated camera poses of each frame, so that the 3D line segments can be back-projected from each frame into more accurate positions. Moreover, the plane segmentation removes some incorrectly created 3D line segments. These scenes had few textures; therefore, reconstruction of the 3D geometry of the scene with general feature-point-based SfM methods is difficult.

KinectFusion[16] can be used to reconstruct a texture-less scene because it uses a point-cloud alignment method. However, the reconstructed model with a texture-less surface cannot be applied to a feature-point-based marker-less AR method. Moreover, we have already demonstrated that 3D line-segment-based model generation from object shapes reconstructed using KinectFusion requires further improvement in terms of accuracy.[10] With respect to the scenes shown in Fig. 11, as these scenes consist of plane structures, a point-cloud alignment method may fail. Figure 14 shows object shapes reconstructed with KinectFusion. Misalignments occurred in the object shapes.

With the generated 3D line-segment-based model, we then estimated the camera pose of some input frames and augmented a CG object onto the images. The snapshots from Figures 15 and 16 show the results of marker-less AR in each scene. In each figure, (a) shows the results from our previous marker-less AR framework[7] and (b) shows those from the proposed framework. We put the CG model of a green sofa in both scenes. The green line segments in the images of Figs. 15 and 16 are those used for estimating the camera pose. On comparing (a) and (b) in each figure, the sofa is augmented to a more accurate position in (b), and in some images of (a), the sofa is not augmented to the right place (such as the third, fourth and fifth images of Figs. 15(a) and 16(a)). This fact shows that in (a), the model generated with our previous method[8] contained inaccurate 3D line segments, so that the 2D–3D line-segment correspondences between the model and the input image contained some false correspondences. In the computation of the camera pose with these correspondences, the outliers were eliminated. Therefore, the number of inliers was decreased and the camera pose was estimated incorrectly. As a result, with the inaccurate camera pose, the augmentation of the CG model failed. On the other hand, in (b), the model generated by the new method provided accurate 3D line segments, so that the 2D–3D line-segment correspondences contained many

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

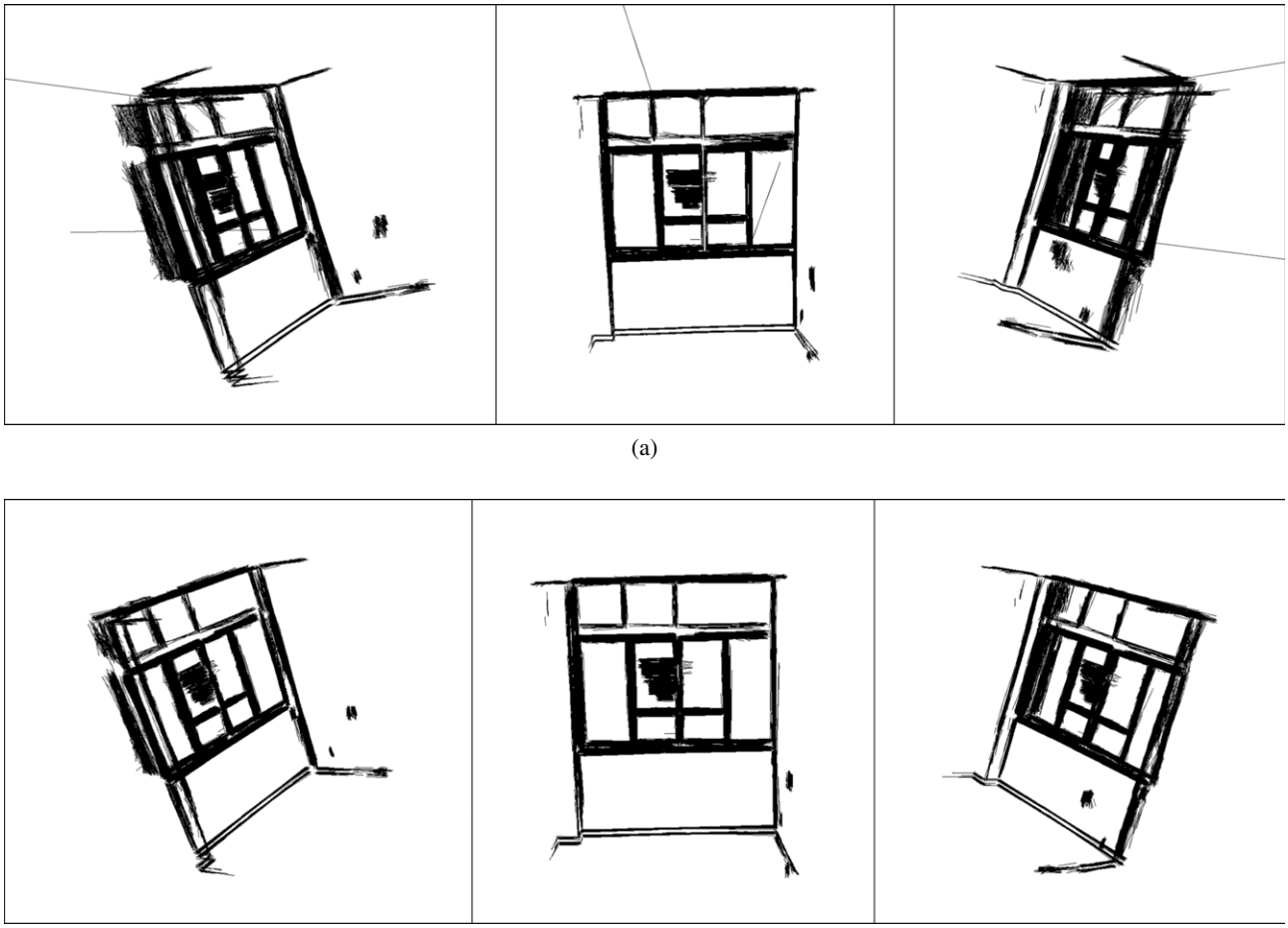


Figure 13. The generated 3D line-segment-based model of the window scene. (a) Model generated with the previous method;[8] (b) model generated with the new method.

inliers. Therefore, from the sufficient 2D–3D line-segment correspondences, the accurate camera pose was estimated and the CG object was augmented to the right position. The difference in the line segments used for the camera-pose estimation can be seen in each resulting image.

We also conducted marker-less AR with the typical feature-point-based AR method, parallel tracking and mapping for small AR workspaces (PTAM),[1] in the same scenes for comparison. The results from PTAM are shown in Figure 17. Only a few feature points were detected from the scene, and PTAM could not estimate an accurate camera pose. In Figs. 15 and 16, however, our new method estimated the camera pose and made marker-less AR possible using the detected line segments.

These results show that our proposed framework can generate the 3D model of the target scene and estimate the camera pose even in a situation with few feature points.

Accuracy Evaluation for Estimated Camera Poses

As mentioned above, our proposed framework consists of model-generation and camera-tracking phases. In the model-generation phase, each input frame's camera pose is estimated, and in the camera-tracking phase, the camera pose of the input frame is also estimated. In this experiment, we evaluated the accuracy of the estimated camera poses for model generation and on-line camera tracking. We used two real image sequences of a texture-less scene used in the previous experiment, another two real image sequences using Kinect v2, which provides 1920 × 1080 resolution RGB images and 512 × 424 resolution depth images, and two sequences of the TUM RGB-D benchmark.[17] The real image sequences do not provide the ground truth of each frame's camera pose; therefore, we measured the re-projection errors of the points shown in Figure 18 using the estimated camera poses. The TUM RGB-D benchmark provides RGB images, depth images and ground truths of their camera poses. The sequences of the TUM RGB-D benchmark we used are "freiburg3_cabinet," and "freiburg3_structure_notexture_far."

Evaluation for the Generated Models

We first evaluated the accuracy of the generated 3D line-segment-based models. For the real image sequences used in the previous experiment, we used the same 80 frames

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

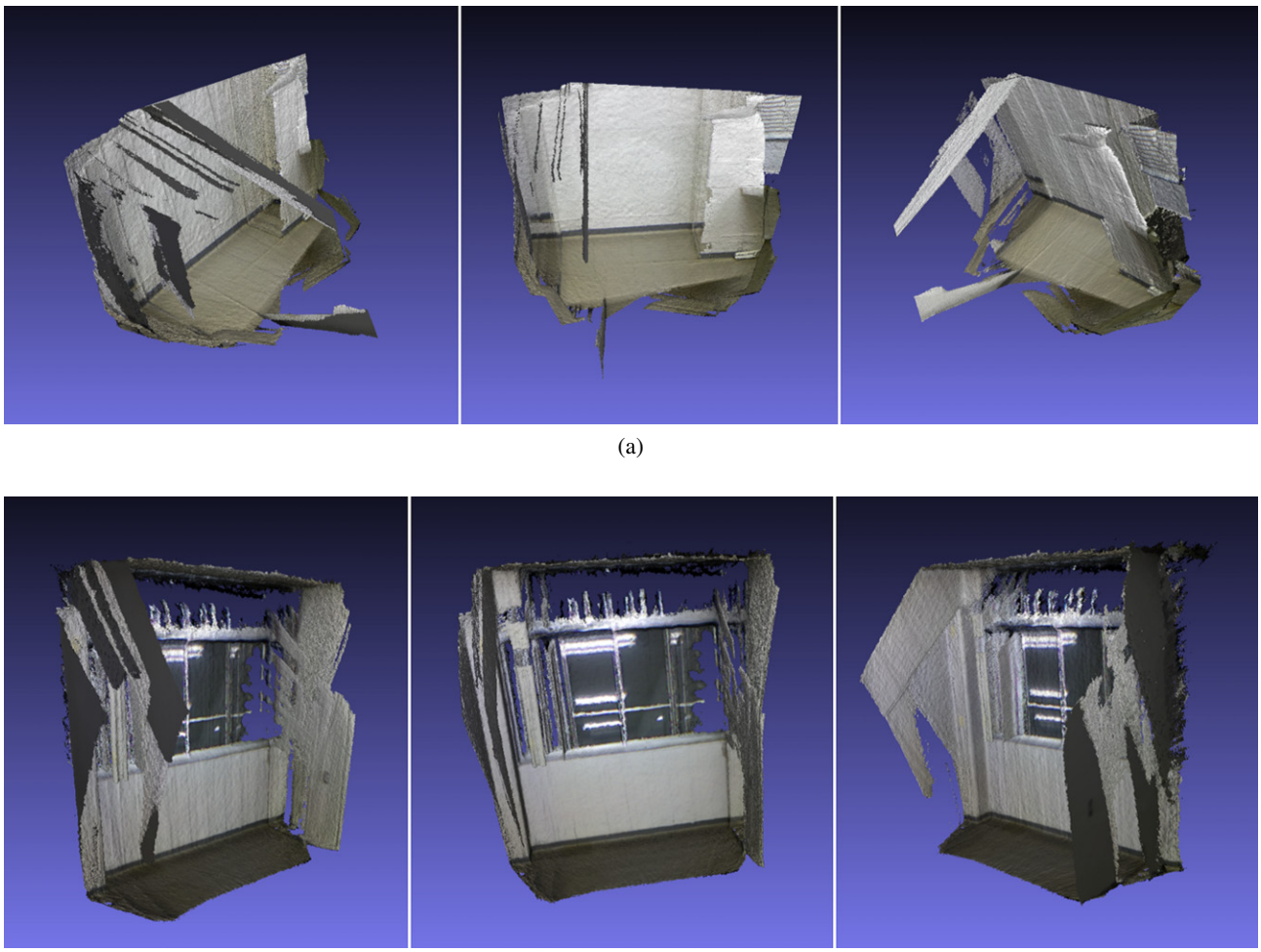


Figure 14. Reconstructed object shapes with KinectFusion. (a) Corner scene; (b) window scene.

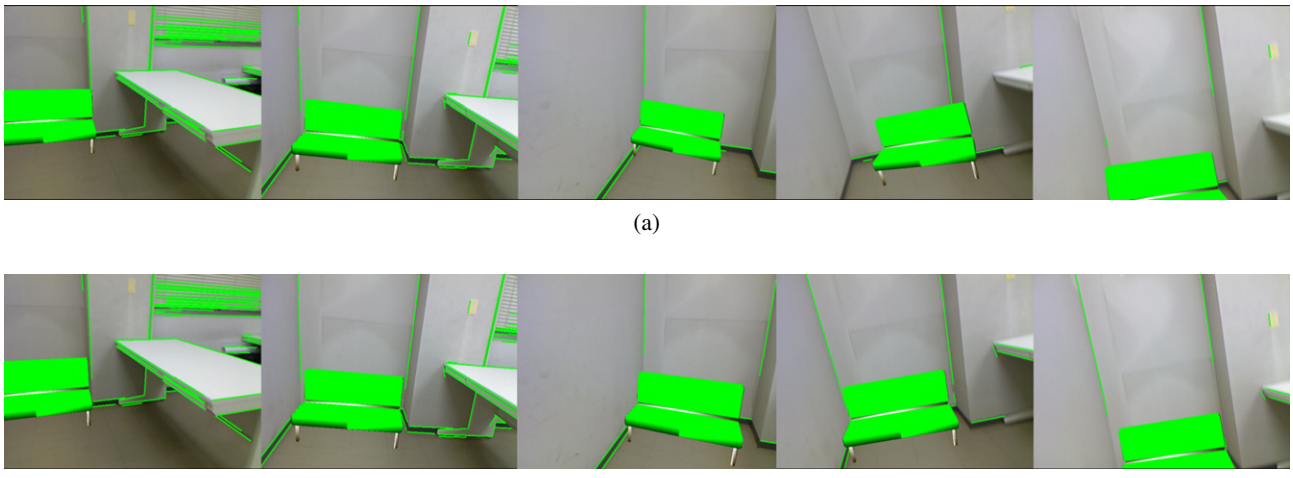
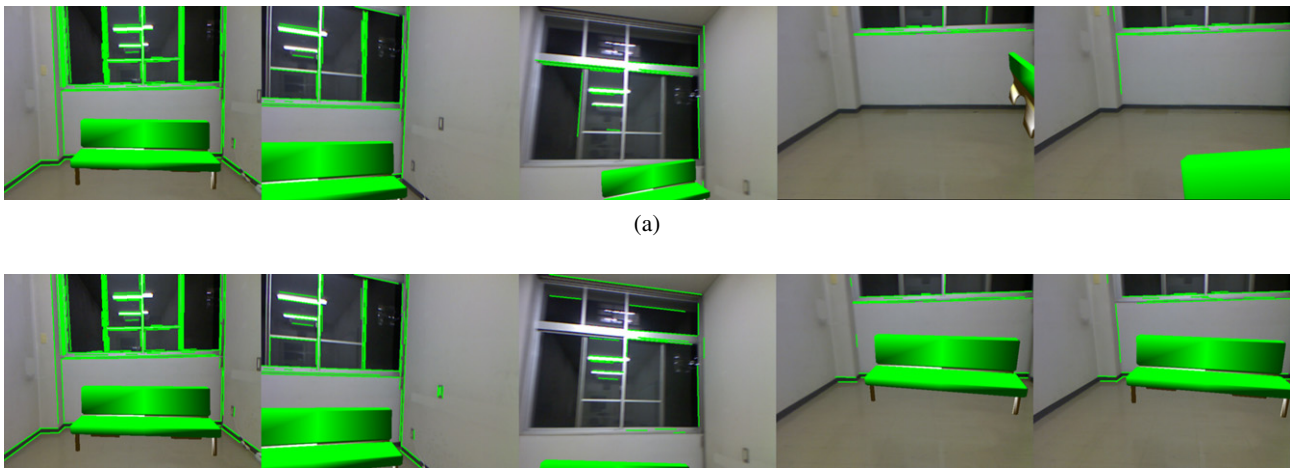


Figure 15. Marker-less AR results from the corner scene. (a) Marker-less AR with our previous framework;[7] (b) marker-less AR with the proposed framework.

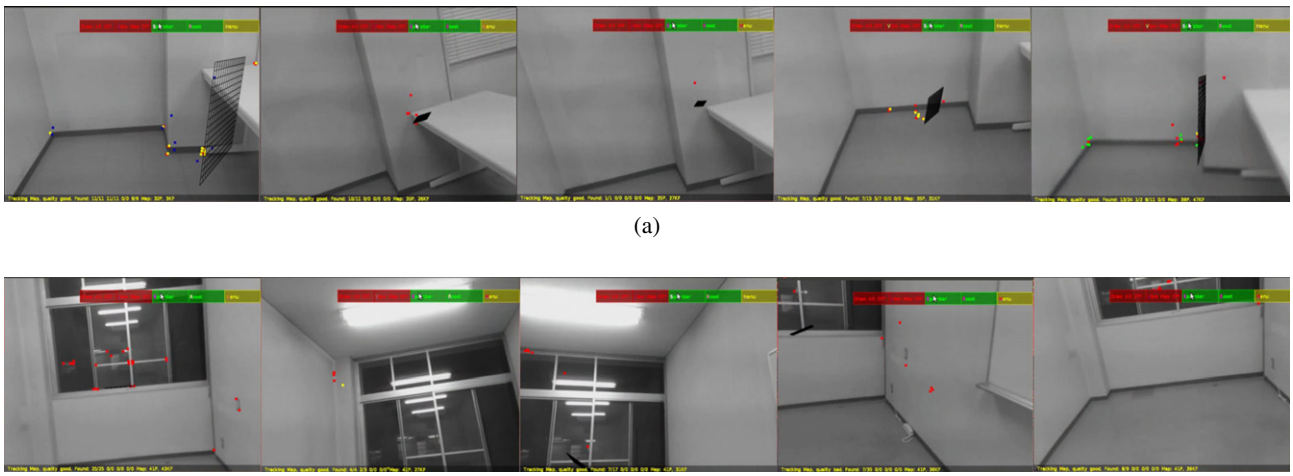
of the two scenes shown in Fig. 11, and 3D line-segment-based models were generated with our previous and new methods. These models are shown in Figs. 12 and 13. We then evaluated the estimated camera poses in the model generation by measuring the re-projection errors. The measured re-projection errors are shown in Figure 19. The

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382



(a) (b)

Figure 16. Marker-less AR results from the window scene. (a) Marker-less AR with our previous framework;[7] (b) marker-less AR with the proposed framework.



(a) (b)

Figure 17. Marker-less AR results with PTAM. (a) Results from the corner scene; (b) results from the window scene.

(a) (b) (c) (d)

Figure 18. Four points used for measuring re-projection errors. (a) Points used in the corner scene; (b) points used in the window scene; (c) points used in the sink scene; (d) points used in the white wall scene.

real image sequences captured by Kinect v2 are shown in Figure 20. There is a sink in one scene, and the other scene has a white wall. To generate 3D line-segment-based models, we used 44 frames for the sink scene and 51 frames for the white wall scene. Fig. 20 also shows some of the input images in each scene used for model generation. The generated models are shown in Figures 21 and 22. In each figure, (a) shows the model generated with the previous model-generation method[8] and (b) shows the model generated with the new method. The re-projection errors from the estimated camera poses in the model generation are also measured and are shown in Figure 23. As shown in the figures, in almost all frames, the errors from the new model-generation method are smaller than those from the previous method.[8]

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

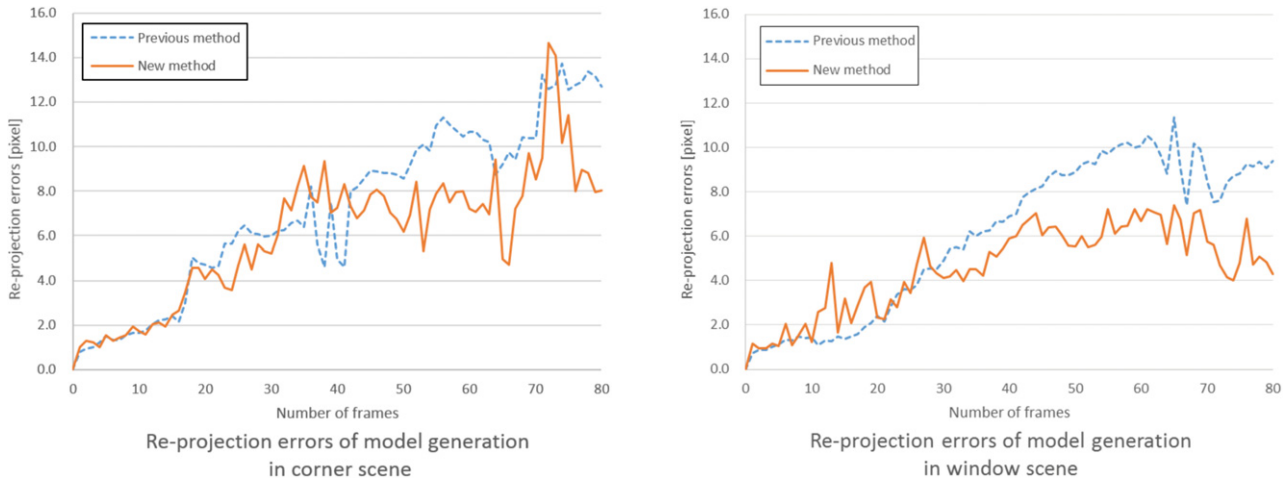


Figure 19. Re-projection errors of model generation. (a) Errors in the corner scene; (b) errors in the window scene.

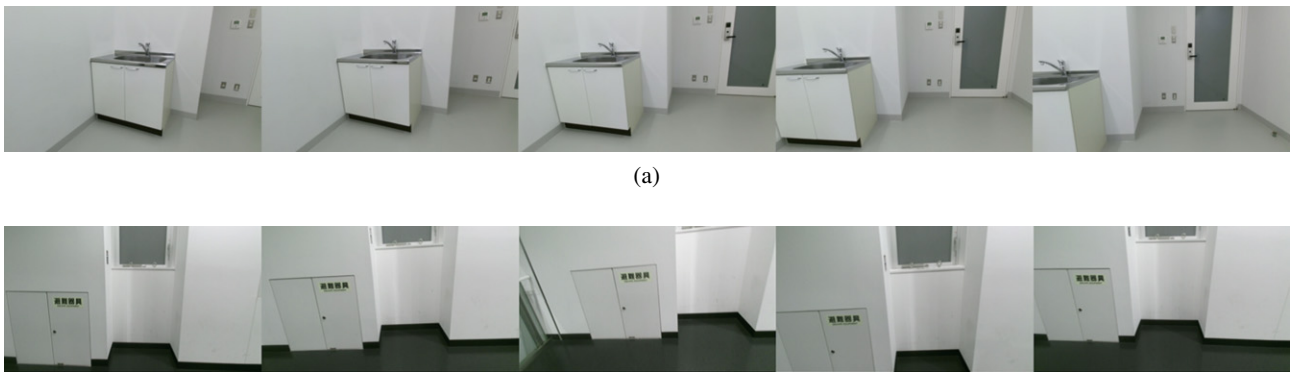


Figure 20. Some input images for model generation of the target scene captured by Kinect v2. (a) Sink in room; (b) white wall.

For the sequences of the TUM RGB-D benchmark, we used RGB images and depth images, then generated 3D line-segment-based models. For generating the models of these sequences, 24 frames were used for freiburg3_cabinet, and for freiburg3_structure_notexture_far, 35 frames were used. Some of the input images are shown in Figures 24(a) and 24(b), respectively. The generated models are shown in Figures 25 and 26. In Figs. 25 and 26, (a) shows the models generated with our previous method[8] and (b) shows those generated with the new method. The new method constructed the scenes more precisely. The estimated camera poses used for model generation were compared with the ground truth, and the errors between them were measured. Figures 27 and 28 show the results of accuracy evaluation for the estimated camera poses during model generation. The upper graphs show the results obtained from our previous method[8] and the others show the results for the camera pose with the new method. In Fig. 27(a) and (c), which shows the translation errors, the errors from the previous model-generation method[8] fit within the range from about −0.03 m to 0.10 m, and the errors from the new method fit within from −0.03 m to 0.03 m. Similarly, for the rotation errors shown in Fig. 27(b) and (d), the errors from the previous model-generation method[8] fit within the range from −0.04 rad to 0.06 rad, and the errors from the new method fit within the range from −0.02 rad to 0.02 rad. Therefore, in many frames, our new model-generation method provides fewer errors than the previous method. As shown in Fig. 28(a) and (b), the model generation of freiburg3_structure_notexture_far with our previous model-generation method estimated the camera poses for only 10 frames, and in subsequent frames, the camera poses could not be estimated. This is mainly because the threshold TH_e used in the outlier-elimination algorithm with RANSAC for the 2D–3D line-segment correspondences is a constant value (in this experiment 0.01) in the previous model-generation method. As we discussed in the Camera-Pose Estimation section, as the threshold depends on the situation of two consecutive frames, a constant threshold is not always appropriate. The experimental result with freiburg3_structure_notexture_far shows this particular situation. In the experiment, after frame 10, the constant value was not sufficient, so that the RANSAC-like algorithm used in the previous model-generation method[8]

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

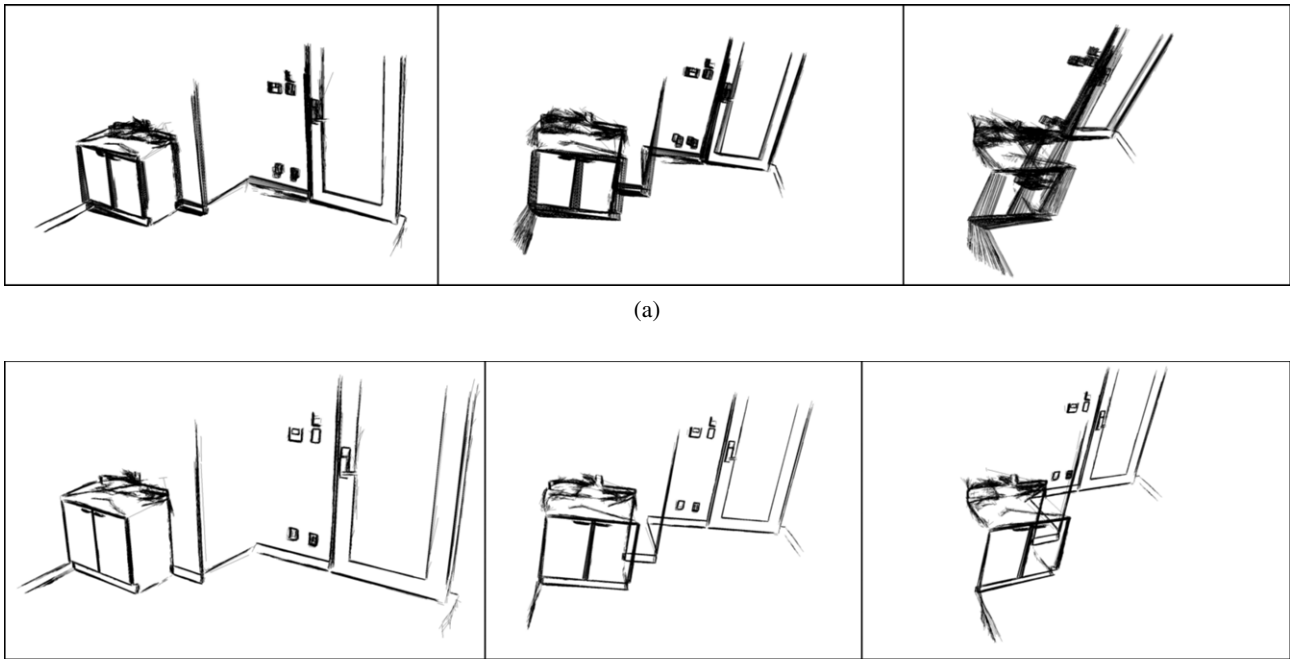


Figure 21. The generated 3D line-segment-based model of the sink scene. (a) Model generated with the previous method;[8] (b) model generated with the new method.

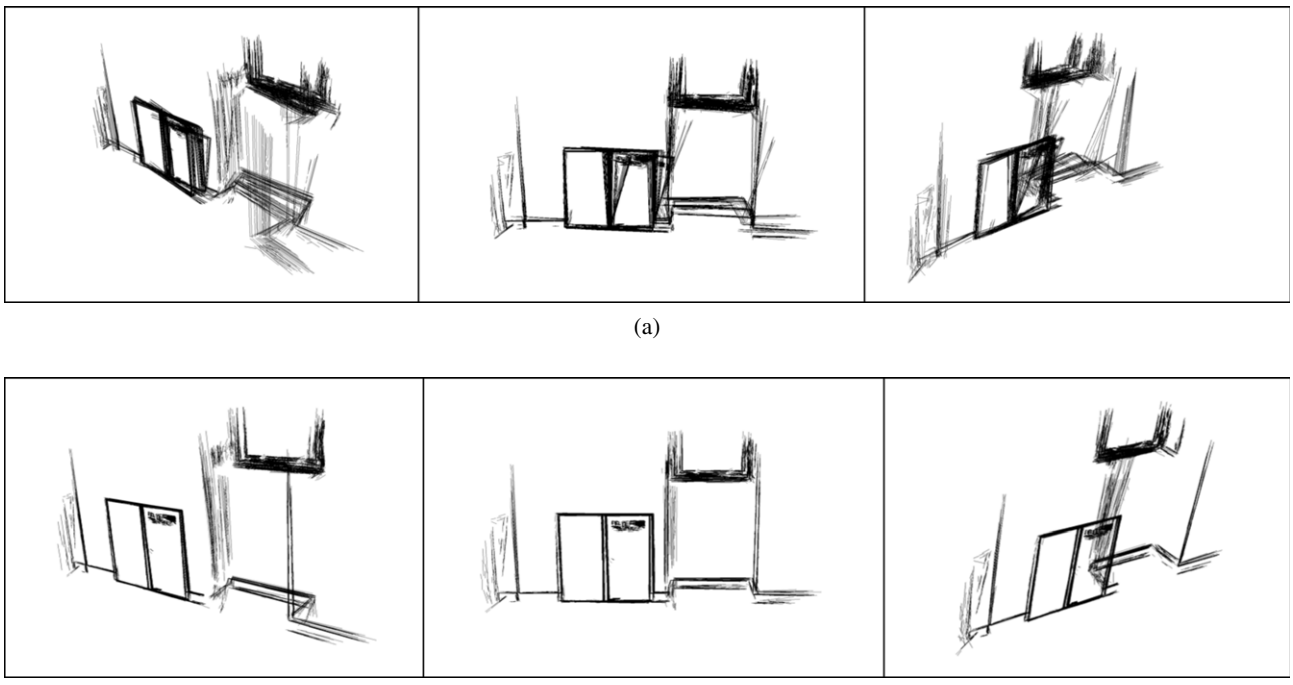


Figure 22. The generated 3D line-segment-based model of the white wall scene. (a) Model generated with the previous method;[8] (b) model generated with the new method.

could not eliminate outliers adequately, and the camera-pose estimation failed. On the other hand, our new model-generation method uses LMedS for outlier elimination to solve this problem. The threshold is decided adaptively for each two frames by LMedS. To demonstrate the efficiency of the LMedS method, we generated a 3D line-segment-based model by a method that replaced the outlier-elimination part using RANSAC in the previous model-generation method[8] with the LMedS method introduced in the Camera-Pose Estimation section. Other parts except for the outlier

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

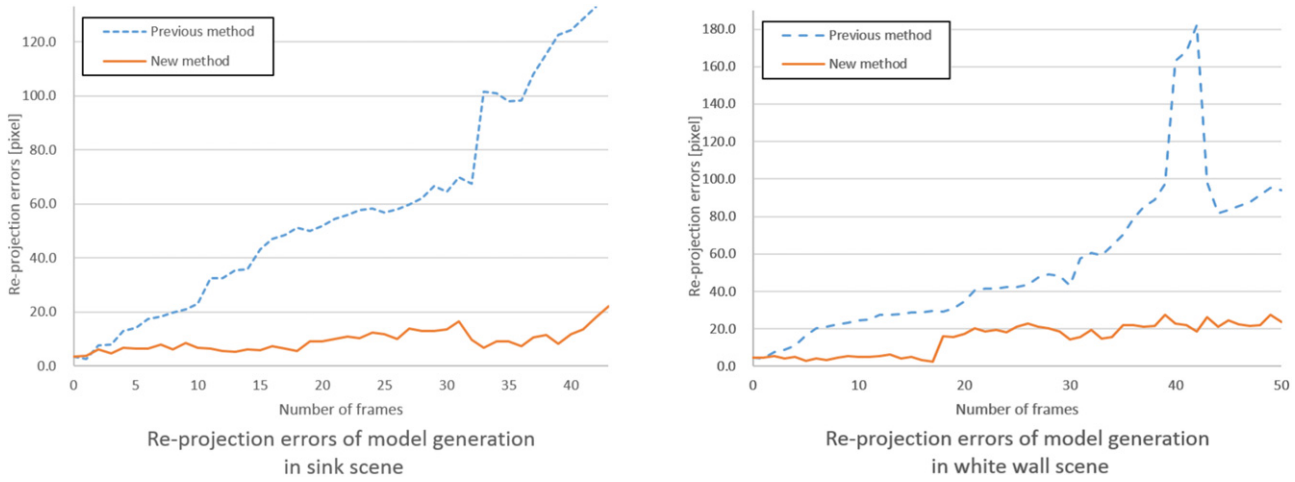


Figure 23. Re-projection errors of model generation. (a) Errors in the sink scene; (b) errors in the white wall scene.

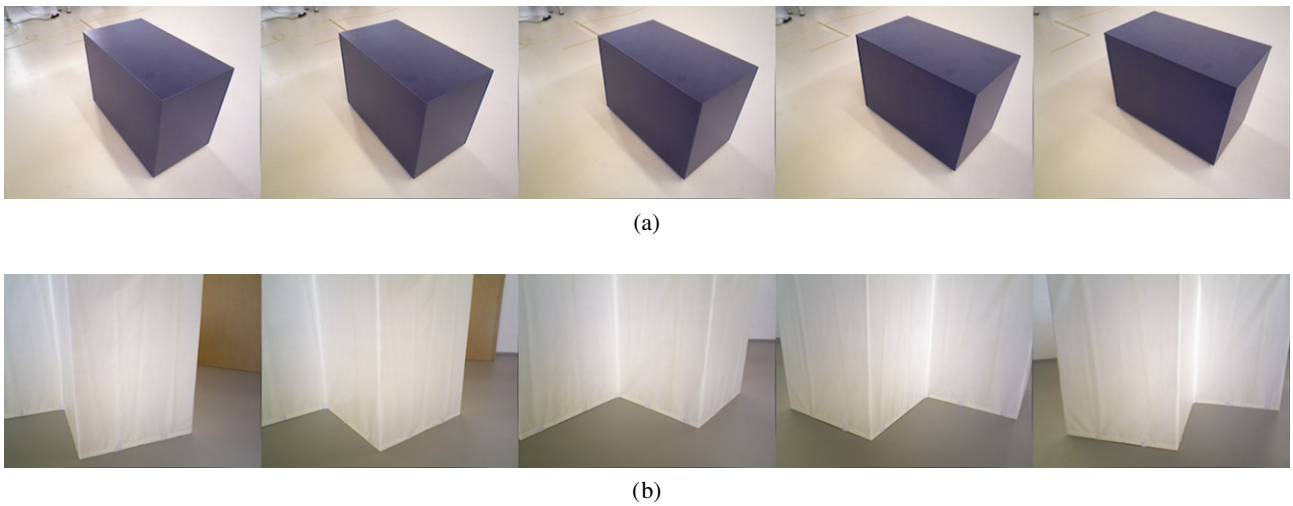


Figure 24. Input images for model generation of the TUM RGB-D benchmark. (a) freiburg3_cabinet; (b) freiburg3_structure_notexture_far.

elimination using LMedS in this method are same as the previous method.[8] Figure 29 shows the generated model and Figure 30 shows the errors of the camera poses estimated during the model generation. As shown in Fig. 30, the method using LMedS estimated the camera pose for every frame. According to this experimental result, the method using LMedS is suitable for this situation. Our new model-generation method also uses LMedS in the outlier elimination and, moreover, performs optimization for every frame. Therefore, the camera poses estimated by the new method were more precise than those of the LMedS method and our previous model-generation method,[8] as shown in Fig. 28(c) and (d).

These experiments demonstrated that the new method is more accurate.

Evaluation for Camera-Pose Estimation

Next, we evaluated the camera-pose estimation accuracy of the camera-tracking phase using the models discussed in the previous experiment. In each experiment, we estimated the camera poses of input images that were different from the images used for the model generation.

For the real image sequences used in the previous experiment, we used the same models as shown in Figs. 12 and 13 and estimated the camera poses of 81 input frames for both scenes. The re-projection errors from the estimated camera poses were measured and are shown in Figure 31.

For the real image sequences captured by Kinect v2, we used the same models as shown in Figs. 21 and 22. The camera poses of 48 input frames for the sink scene and 77 frames for the white wall scene were estimated. The re-projection errors from the result of the camera-pose estimation are shown in Figure 32.

For the TUM RGB-D benchmark sequences, we used only RGB images and estimated the camera pose of 67 frames for freiburg3_cabinet and 35 frames for freiburg3_structure_notexture_far with the same models as shown in Figs. 25 and 26, respectively. The errors between

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

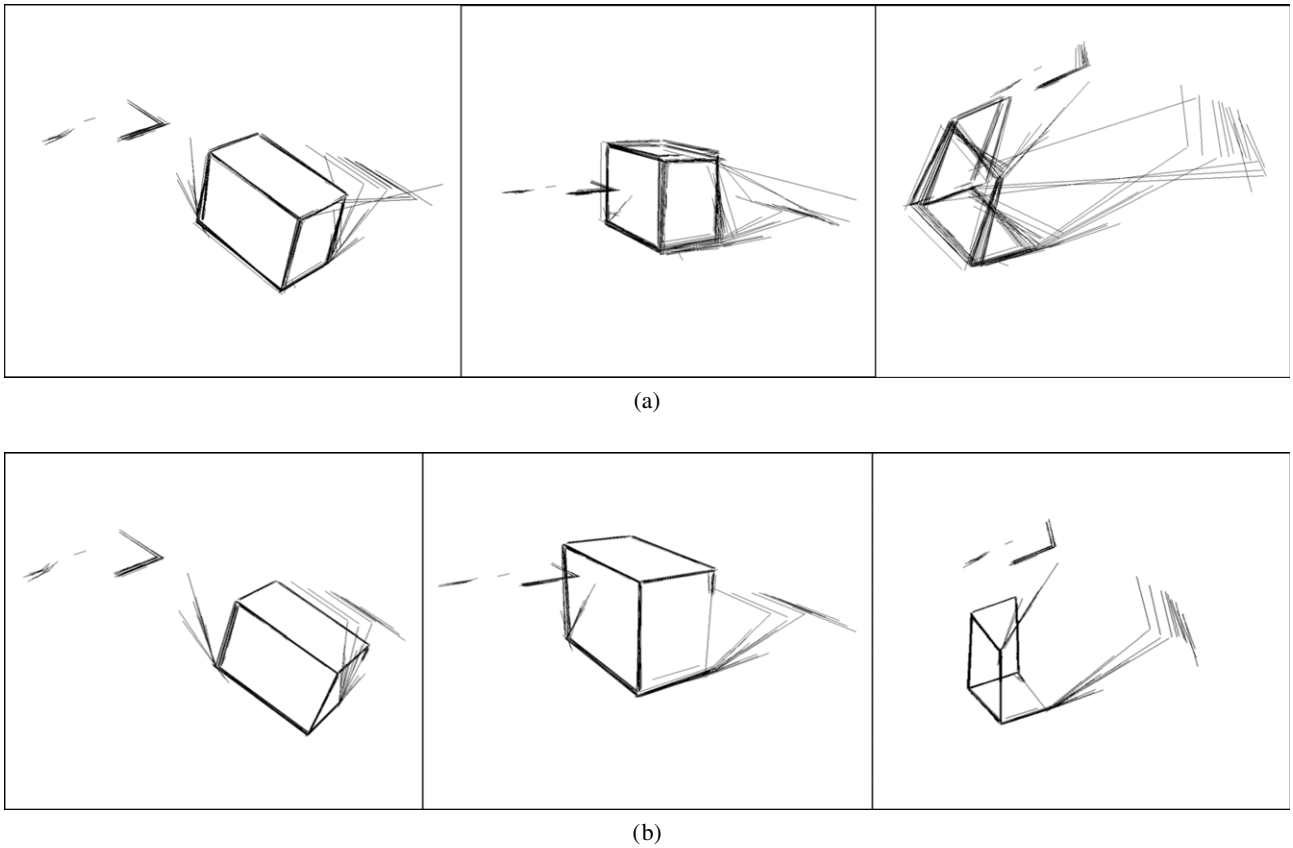


Figure 25. The generated 3D line-segment-based model of freiburg3_cabinet. (a) Model generated with the previous method;[8] (b) model generated with the new method.

the estimated camera poses and the ground truth were also measured. The evaluation results of camera-pose estimation accuracy are shown in Figures 33 and 34. The upper graphs show the estimation errors of input frames with the previous framework,[7] which uses the previous model-generation method[8] in the model-generation phase. The others show the result of camera-pose estimation of input frames with the proposed framework, which uses the new model-generation method proposed in this article for the model-generation phase. In Fig. 33(a) and (c), which shows the translation errors, the errors from the previous framework[7] fit within the range from about −0.15 m to 0.05 m, and the errors from the proposed framework fit within from −0.08 m to 0.05 m. Similarly, for the rotation errors shown in Fig. 33(b) and (d), the errors from the previous framework[7] fit within the range from −0.10 rad to 0.02 rad, and the errors from the proposed framework fit within the range from −0.04 rad to 0.04 rad.

With regard to the sequence of freiburg3_structure_notexture_far, in most frames, the previous framework[7] could not estimate the camera poses, and the errors between the estimated camera poses and the ground truth were large, as shown in Fig. 34(a) and (b). This is because the previous framework used the incomplete 3D line-segment-based model shown in Fig. 26(a), generated with the previous model-generation method,[8] so that the 3D line-segment database constructed by the incomplete model has too few directed LEHFs. Therefore, the number of 2D–3D line-segment correspondences between an input frame and the model is not sufficient for the computation of an accurate camera pose. On the other hand, our proposed framework used the accurate 3D line-segment-based model shown in Fig. 26(b), generated with the new model-generation method, so that in almost all frames, the errors were small, as shown in Fig. 34(c) and (d).

As shown in Figs. 31–34, in most frames, the camera-pose estimation errors in the tracking phase of the proposed framework using the model generated by the new method were smaller than those of the previous framwork[7] using the model generated by the previous model-generation method.[8]

Processing Times

We measured the processing times of the on-site model-generation phase and the camera-tracking phase in our proposed framework with the four datasets used in the Accuracy Evaluation for Estimated Camera Poses section. An analysis of the processing time was carried out on an Intel® Core™ i7-5960X CPU with 3.00 GHz. The average computational costs for each individual step of the model generation by the new method are shown in Table I. In Table I, Model1 is the model shown in Fig. 21(b), Model2 is shown in Fig. 22(b), Model3 is shown in Fig. 25(b) and

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

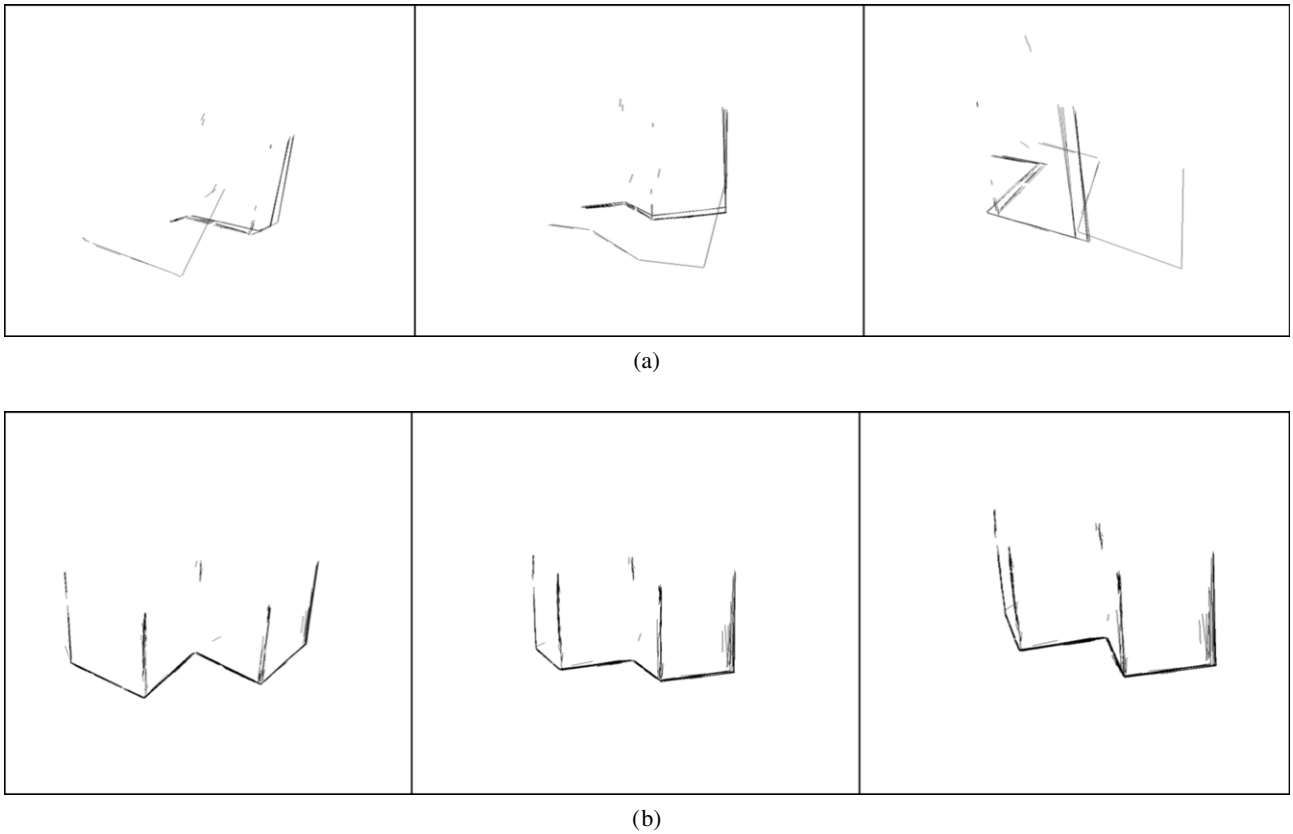


Figure 26. The generated 3D line-segment-based model of freiburg3_structure_notexture_far. (a) Model generated with the previous method;[8] (b) model generated with the new method.

Table I. Average processing times of the individual steps for model generation (Model1, sink scene; Model2, white wall scene; Model3, freiburg3_cabinet; Model4, freiburg3_structure_notexture_far).

	Time in seconds			
Individual step	Model1	Model2	Model3	Model4
Loading RGB image	0.070	0.098	0.001	0.001
Loading depth image	0.014	0.028	0.006	0.006
2D Line-segment detection (LSD)	0.228	0.229	0.029	0.027
Plane segmentation	3.654	1.092	0.344	0.408
3D line-segment creation	1.164	0.819	0.155	0.127
Directed-LEHF extraction	0.456	0.293	0.009	0.008
2D line-segment matching	0.008	0.006	0.000	0.000
2D–3D line-segment correspondences	0.021	0.012	0.000	0.000
Camera-pose estimation from correspondences	0.178	0.200	0.135	0.151
Total of one frame	6.408	3.243	0.751	0.833
Total of all frames without bundle adjustment	275.6	162.1	17.2	28.3

Model4 is shown in Fig. 26(b). The total time shown at the bottom of Table I does not include the computation time of the bundle adjustment because it highly depends on the way in which the optimization algorithms are implemented (in our system, we just used MATLAB). We also measured the average computation time for every individual step of the camera tracking in our proposed framework, which is shown in Table II. This is the time measurement for the four experimental results in the Evaluation for Camera-Pose Estimation section.

Table II. Average processing times of the individual steps for camera tracking (Scene1, sink scene; Scene2, white wall scene; Scene3, freiburg3_cabinet; Scene4, freiburg3_structure_notexture_far).

	Time in seconds			
Individual step	Scene1	Scene2	Scene3	Scene4
Loading frame	0.078	0.079	0.001	0.001
2D line-segment detection (LSD)	0.222	0.227	0.030	0.027
Directed-LEHF extraction	0.176	0.117	0.007	0.003
2D–3D line-segment correspondences	0.443	0.182	0.004	0.003
Camera-pose estimation from correspondences	0.041	0.039	0.013	0.012
Total	0.980	0.690	0.060	0.052

DISCUSSION

In this section, we give an overview of related work on the construction of a model represented by line segments and on marker-less AR methods that use line segments.

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

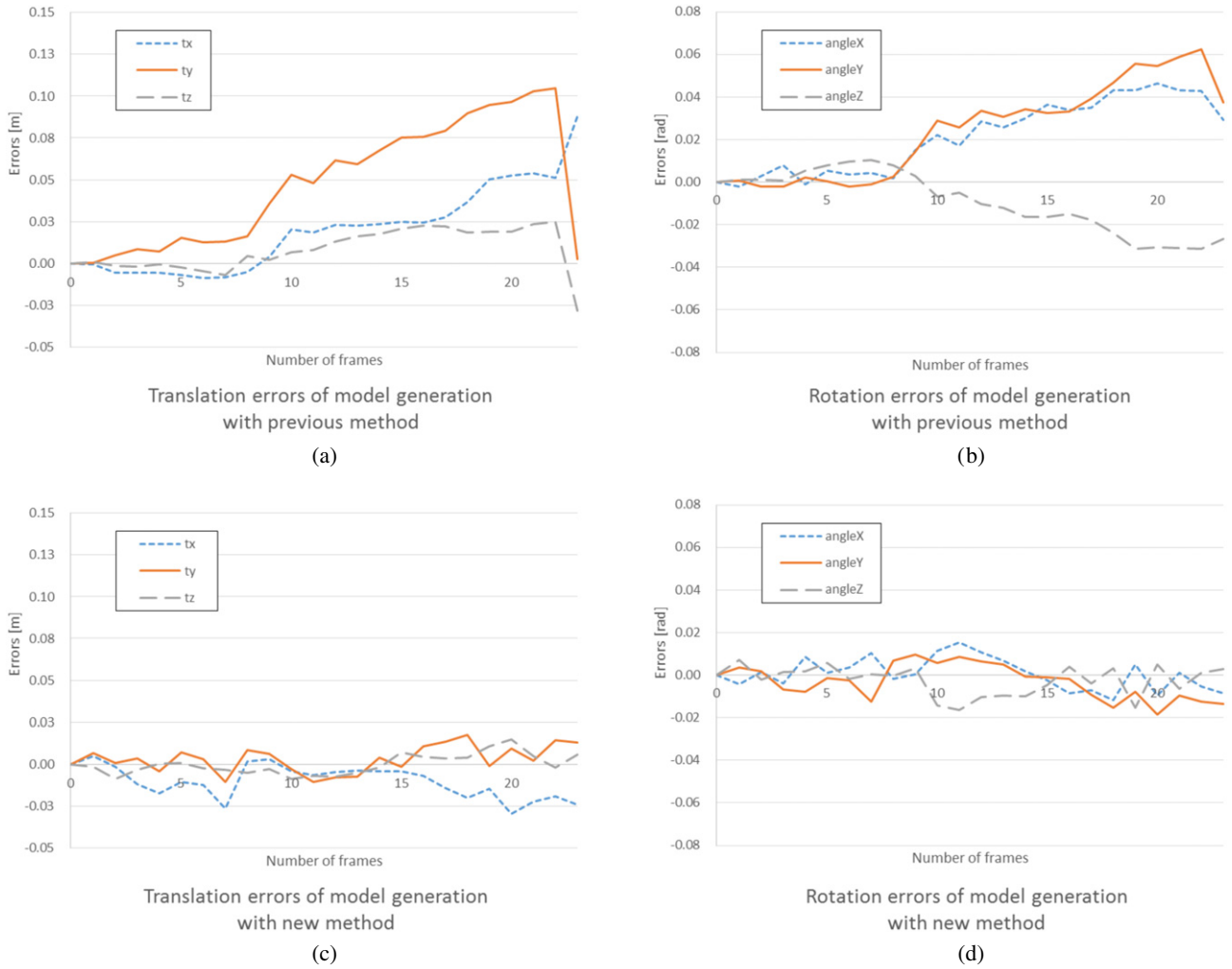


Figure 27. Errors in the estimated camera pose for model generation in freiburg3_cabinet. (a) Translation errors with the previous method;[8] (b) rotation errors with the previous method;[8] (c) translation errors with the new method; (d) rotation errors with the new method.

3D Line-Segment-Based Model Generation

We present the relevant investigation concerning line-segment-based model generation.

Chen et al. proposed a method for detecting 3D line segments on unorganized point clouds from multi-view stereo.[18] The method can be used for generating a 3D line-segment-based model from point clouds. However, it requires the reconstructed shape of an object, which is represented by sparse point clouds obtained from SfM, but in a texture-less scene, feature-point-based SfM methods cannot be used. Therefore, their method cannot be used to reconstruct a 3D line-segment-based model.

Jain et al. proposed a method for reconstructing 3D line segments that represents the 3D objects in a scene from a set of 2D images.[19] Although this method is useful especially in man-made situations in which texture-less objects are contained, there is a limitation in that the camera poses of the input 2D images must be obtained beforehand.

There are other 3D reconstruction methods that use RGB-D cameras, such as Microsoft Kinect. KinectFusion[16] is a method for recovering the object shape of a scene by using Kinect. KinectFusion uses a point-cloud-based alignment method for estimating camera pose, so that it can reconstruct the 3D geometry even when there are few feature points. Although KinectFusion can generate a 3D model of a texture-less object, if the reconstructed shape has no texture, it cannot be used for model-based camera-tracking methods,[20,21] which use feature-point matching between the model and the input frame. We previously proposed a method for improving the accuracy of the object shape reconstructed by KinectFusion using line segments.[10] In this work, we mentioned that the 3D line-segment-based model generated from KinectFusion lacked accuracy.

According to these studies, it has not been easy to generate an accurate 3D line-segment-based model.

Camera Tracking Using Line Segments

We now describe related works concerning line-feature-based camera-tracking methods for marker-less AR, which should work even without a rich texture on object surfaces.

Alvarez et al. proposed a maintenance operation tool based on marker-less AR.[4] This tool provides users with

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

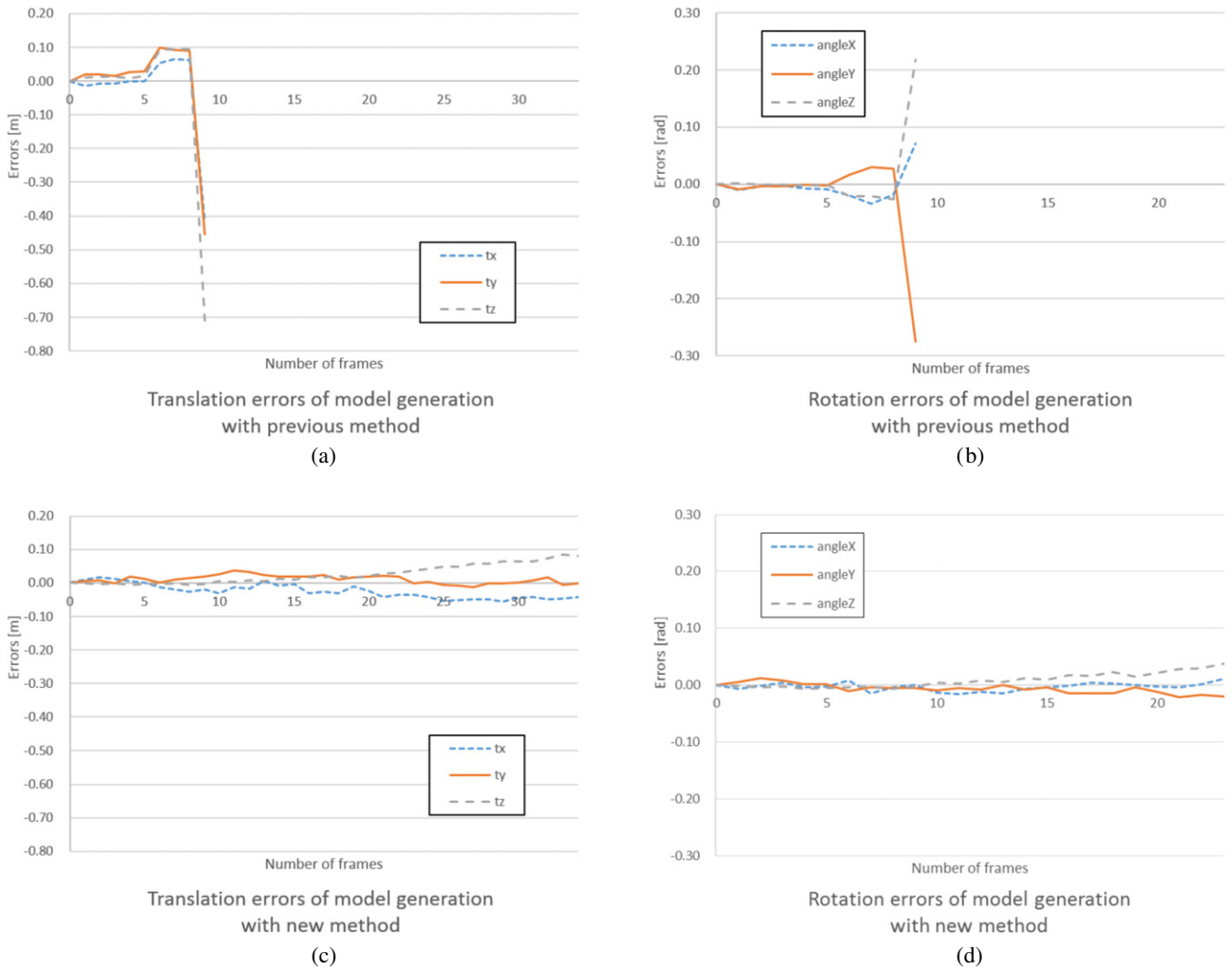


Figure 28. Errors in the estimated camera pose for model generation in freiburg3_structure_notexture_far. (a) Translation errors with the previous method;[8] (b) rotation errors with the previous method;[8] (c) translation errors with the new method; (d) rotation errors with the new method.

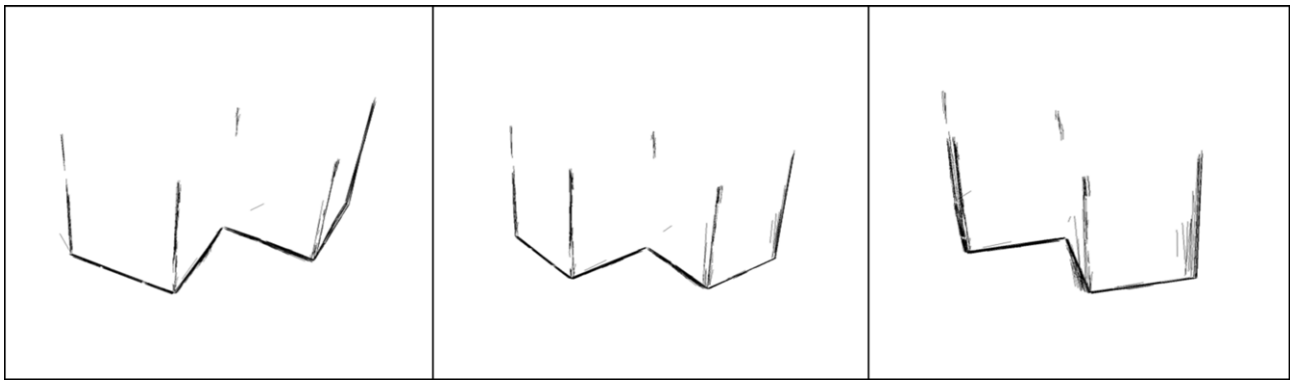


Figure 29. The generated 3D line-segment-based model of freiburg3_structure_notexture_far with the method replacing the RANSAC part in the previous model-generation method[8] with LMedS.

augmented instruction on an assembled object. It can track an object with edge-based marker-less tracking. However, it requires texture-less 3D triangle meshes of the object.

Wuest et al. proposed a model-based line-tracking method with image edge features.[5] This edge-based tracking method also requires a CAD model of the target object. The camera-pose estimation with this method is obtained from the 2D–3D line-segment correspondences. These correspondences are from 3D lines projected from the 3D model and 2D image lines near the 3D lines.

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

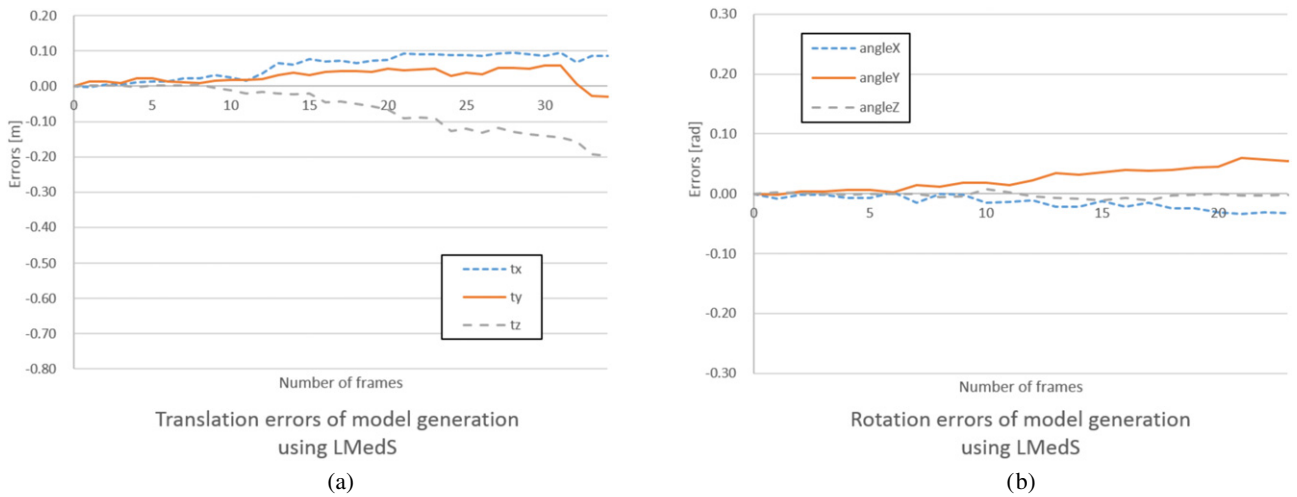


Figure 30. Errors in the estimated camera pose for model generation in freiburg3_structure_notexture_far with the method replacing the RANSAC part in the previous model-generation method[8] with LMedS. (a) Translation errors; (b) rotation errors.

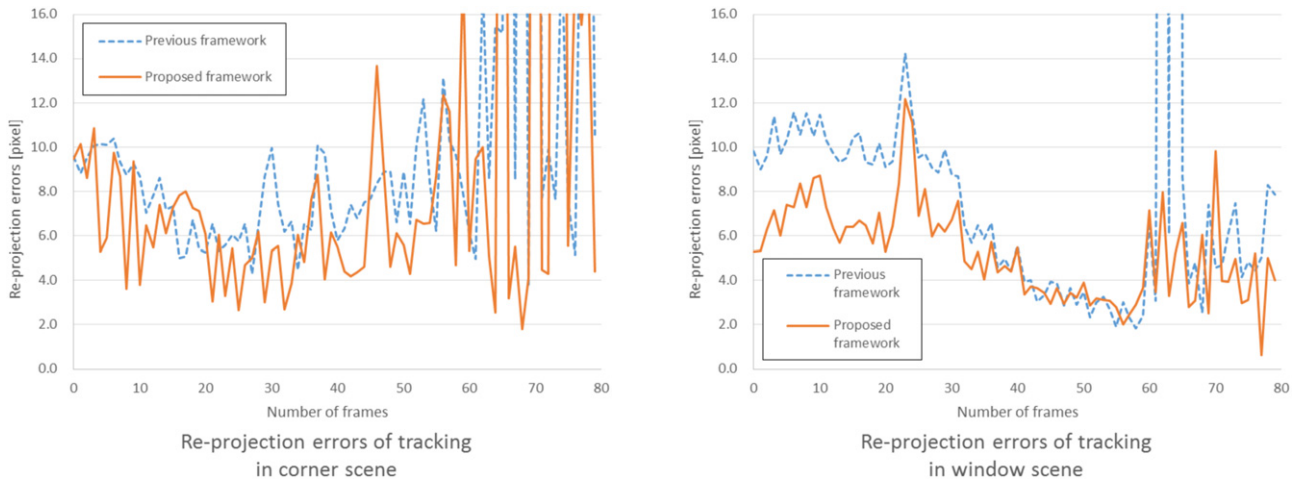


Figure 31. Re-projection errors of tracking. (a) Errors in the corner scene; (b) errors in the window scene.

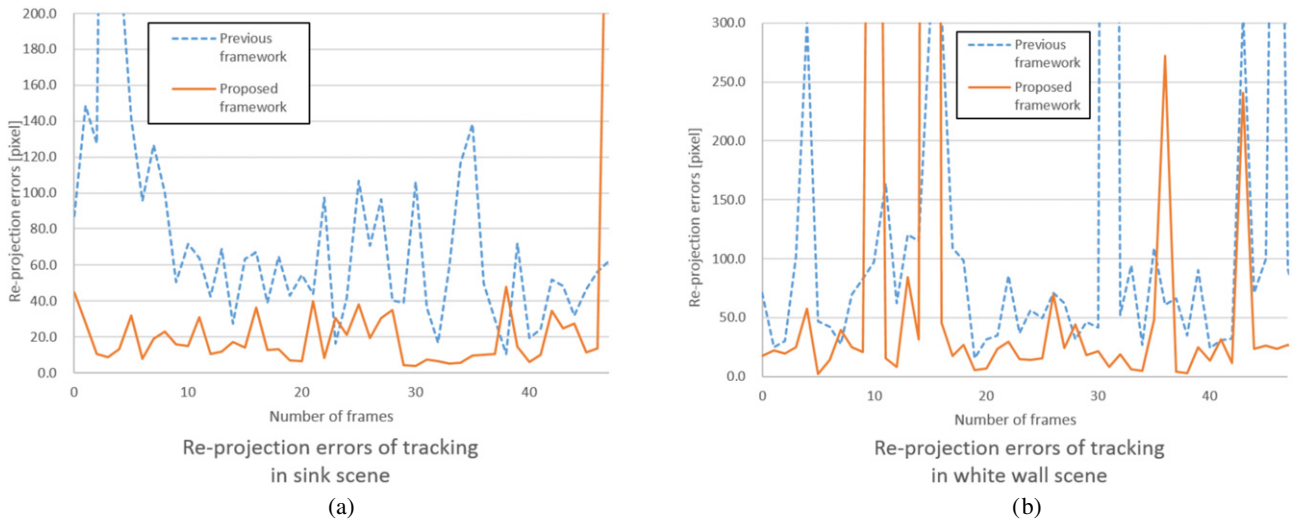


Figure 32. Re-projection errors of tracking. (a) Errors in the sink scene; (b) errors in the white wall scene.

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

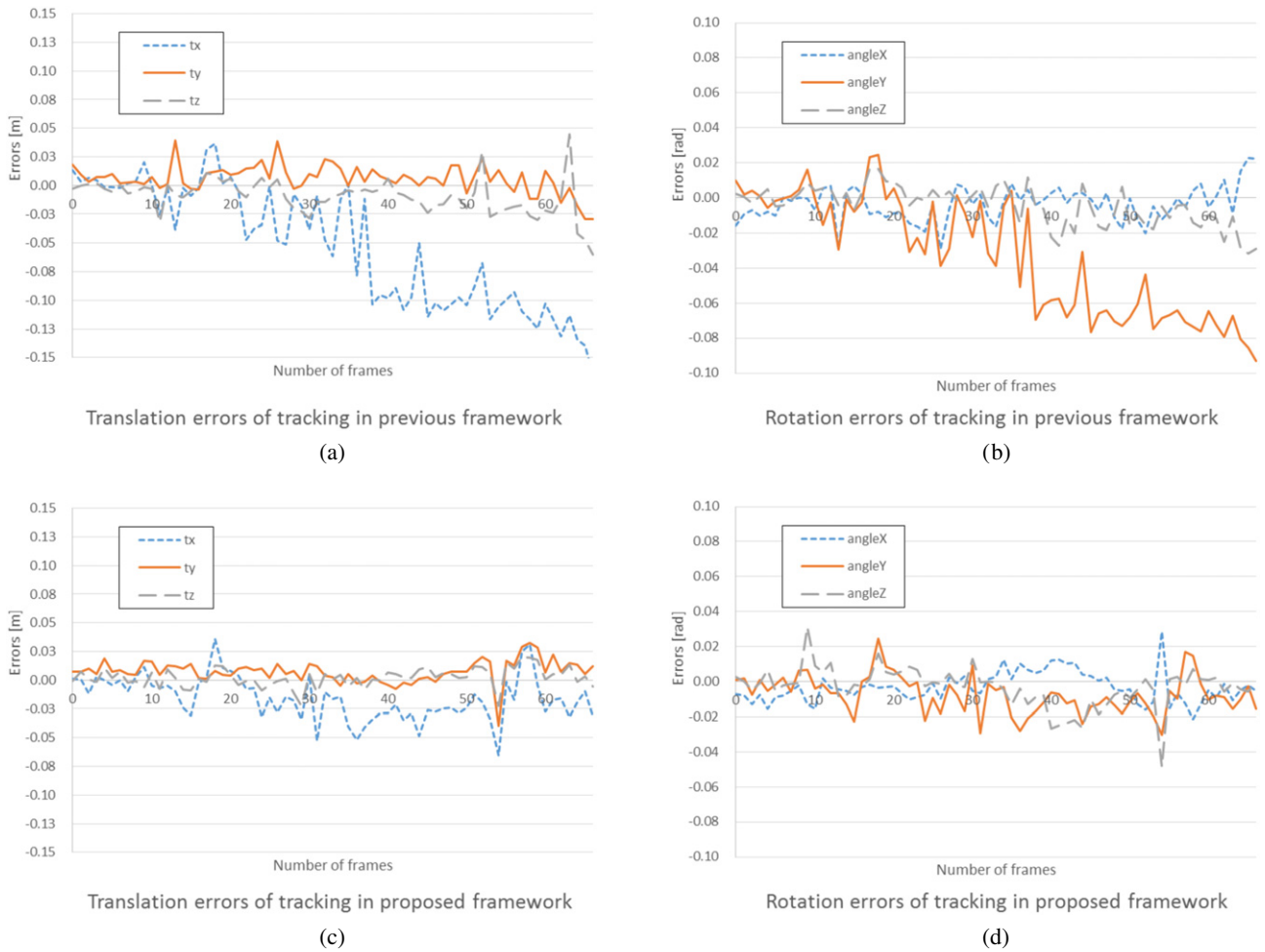


Figure 33. Errors in the estimated camera pose for on-line camera tracking in freiburg3_cabinet. (a) Translation errors of the camera poses estimated with the previous framework;[7] (b) rotation errors of the camera poses estimated with the previous framework;[7] (c) translation errors of the camera poses estimated with the proposed framework; (d) rotation errors of the camera poses estimated with the proposed framework.

On the other hand, our proposed framework involves on-site model generation and on-line camera tracking using line segments. With our framework, 2D–3D line-segment correspondences are obtained by line-segment feature descriptor matching.

CONCLUSION

We proposed a marker-less AR framework, which consists of on-site model-generation and on-line camera-tracking phases. This framework uses line-segment matching and estimates the camera pose from the 2D–3D line-segment correspondences. Therefore, marker-less AR can be achieved even when only a few feature points are detected.

In the model-generation phase, the target scene, which consists of texture-less objects, is captured using an RGB-D camera. The 2D line segments are first detected from these captured frames. Then, 2D line-segment matching between the current frame and the previous frame is obtained from the line-segment feature descriptor. Next, we create accurate 3D line segments using plane-segmentation results detected from the depth image and get 2D–3D line-segment correspondences. These 2D–3D line-segment correspondences provide the camera pose of the current frame by solving the PnL problem. This procedure is repeated for every frame. The estimated camera poses are optimized with a bundle adjustment. Finally, the back-projected 3D line segments from every frame with the refined camera poses construct the 3D line-segment-based model of the scene.

In the on-line camera-tracking phase, we obtain 2D–3D line-segment correspondences between the 2D input image and the 3D line-segment-based model by the line-segment feature descriptor. The camera pose of the input image is estimated from these correspondences. We can then augment computer graphics onto the input image using the estimated camera pose.

The experimental results suggest that our proposed framework can estimate the camera pose even for a target scene without rich texture by matching line segments with a 3D line-segment-based model generated with the new model-generation method of our framework; therefore, we can achieve AR in a scene without rich texture, which feature-point-based methods cannot do for the same scene.

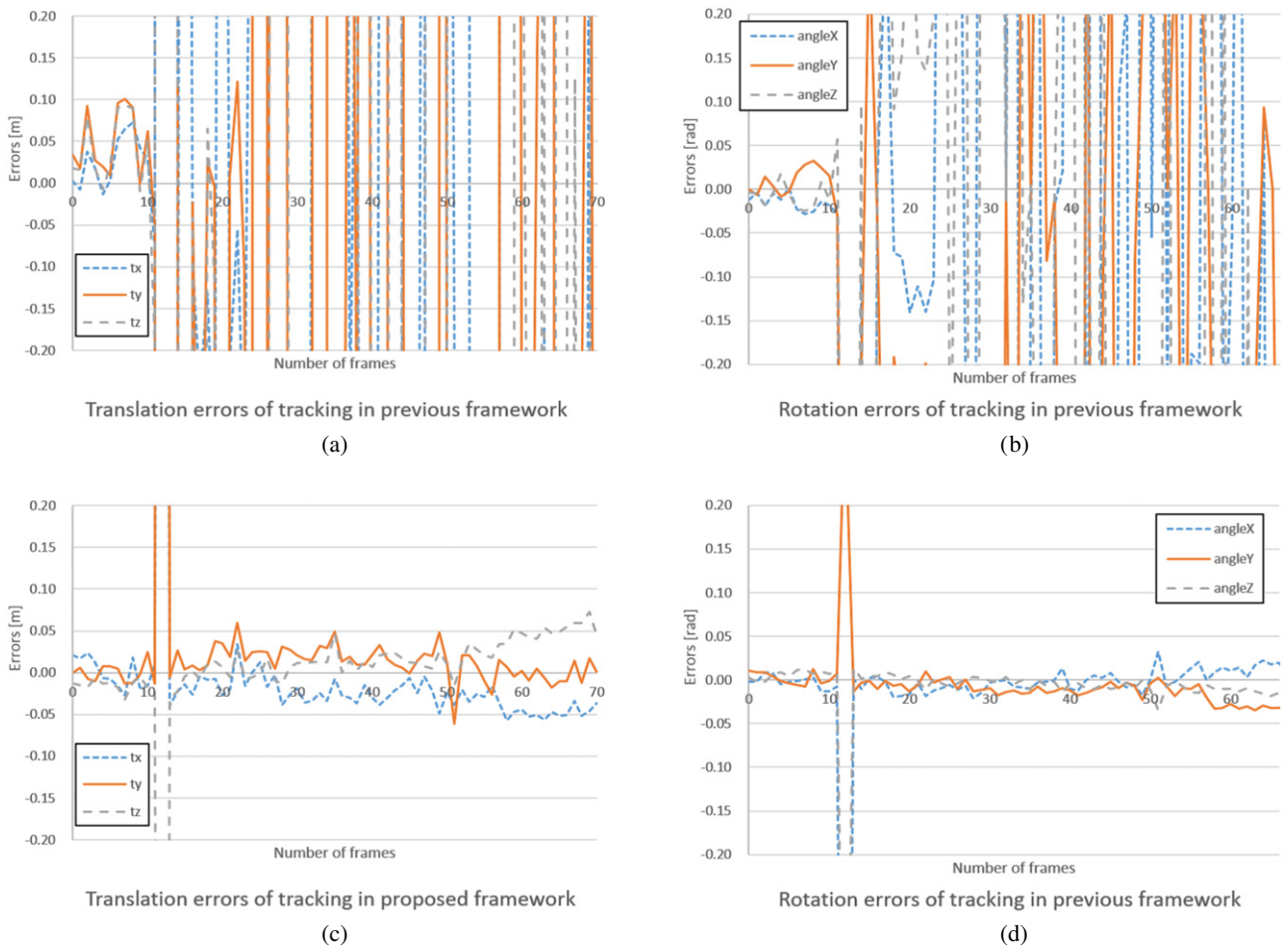


Figure 34. Errors in the estimated camera pose for on-line camera tracking in freiburg3_structure_notexture_far. (a) Translation errors of the camera poses estimated with the previous framework;[7] (b) rotation errors of the camera poses estimated with the previous framework;[7] (c) translation errors of the camera poses estimated with the proposed framework; (d) rotation errors of the camera poses estimated with the proposed framework.

In our new model-generation method in the proposed framework, both RGB and depth images are needed, so that this method cannot construct the model with only RGB images. However, if an accurate model of the target scene is available and 3D line segments with their line-segment features are extracted from the multiple view of the model, the tracking method in our proposed framework can be applied as model-based marker-less AR with only an RGB camera. Therefore, improvement of this aspect is future work.

ACKNOWLEDGMENTS

This work was partially supported by MEXT/JSPS Grant-in-Aid for Scientific Research(S) 24220004, and JST CREST "Intelligent Information Processing Systems Creating Co-Experience Knowledge and Wisdom with Human–Machine Harmonious Collaboration."

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," *Proc. 6th IEEE and ACM Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2007), pp. 225–234.

[2] I. Skrypnyk and D. G. Lowe, "Scene modelling, recognition and tracking with invariant image features," *Proc. 3rd IEEE and ACM Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2004), pp. 110–119.

[3] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision (ECCV 2006)* (Springer, 2006), pp. 430–443.

[4] H. Alvarez, I. Aguinaga, and D. Borro, "Providing guidance for maintenance operations using automatic markerless augmented reality system," *Proc. 10th IEEE Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2011), pp. 181–190.

[5] H. Wuest, F. Vial, and D. Strieker, "Adaptive line tracking with multiple hypotheses for augmented reality," *Proc. 4th IEEE and ACM Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2005), pp. 62–69.

[6] P. Sturm and B. Triggs, "A factorization based algorithm for multi-image projective structure and motion," *Computer Vision (ECCV'96)* (Springer, 1996), pp. 709–720.

[7] Y. Nakayama, H. Saito, M. Shimizu, and N. Yamaguchi, "Marker-less AR system based on line segment feature," Proc. SPIE **9392**, 93920I–93920I (2015).

[8] Y. Nakayama, H. Saito, M. Shimizu, and N. Yamaguchi, "3D line segment based model generation by RGB-D camera for camera pose estimation," *Computer Vision—ACCV 2014 Workshops*, edited by

DOI: 10.2352/ISSN.2470-1173.2016.14.IPMVA-382

C. V. Jawahar and Shiguang Shan, Lecture Notes in Computer Science (Springer International Publishing, 2014), Vol. 9010, pp. 459–472.

[9] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a fast line segment detector with a false detection control," IEEE Trans. Pattern Anal. Mach. Intell. **32**, 722–732 (2010).

[10] Y. Nakayama, T. Honda, H. Saito, M. Shimizu, and N. Yamaguchi, "Accurate camera pose estimation for KinectFusion based on line segment matching by LEHF," *Proc. Int'l Conf. on Pattern Recognition (ICPR)* (2014), pp. 2149–2154.

[11] K. Hirose and H. Saito, "Fast line description for line-based SLAM," *Proc. British Machine Vision Conf. (BMVC)* (2012), pp. 83.1–83.11.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM **24**, 381–395 (1981).

[13] L. Zhang, C. Xu, K.-M. Lee, and R. Koch, "Robust and efficient pose estimation from line correspondences," *Computer Vision (ACCV 2012)* (Springer, 2013), pp. 217–230.

[14] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection* (John Wiley & Sons, 2005), Vol. 589.

[15] R. Kumar and A. R. Hanson, "Robust methods for estimating pose and a sensitivity analysis," CVGIP, Image Underst. **60**, 313–342 (1994).

[16] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: real-time dense surface mapping and tracking," *Proc. 10th IEEE Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2011), pp. 127–136.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *Proc. Int'l Conf. on Intelligent Robot Systems (IROS)* (2012).

[18] T. Chen and Q. Wang, "3D line segment detection for unorganized point clouds from multi-view stereo," *Computer Vision (ACCV 2010)* (Springer, 2011), pp. 400–411.

[19] A. Jain, C. Kurz, T. Thormahlen, and H.-P. Seidel, "Exploiting global connectivity constraints for reconstruction of 3D line segments from images," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2010), pp. 1586–1593.

[20] D. Thachasongtham, T. Yoshida, F. de Sorbier, and H. Saito, "3D object pose estimation using viewpoint generative learning," *Image Anal.* (Springer, 2013), pp. 512–521.

[21] Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab, "Marker-less tracking for AR: a learning-based approach," *Proc. Int'l Symposium on Mixed and Augmented Reality (ISMAR)* (IEEE, Piscataway, NJ, 2002), pp. 295–304.