Robotics and Autonomous Systems 87 (2017) 313–328

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Automatic calibration of camera sensor networks based on 3D texture map information ☆

Yonghoon Ji *, Atsushi Yamashita, Hajime Asama

Graduate School of Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

HIGHLIGHTS

- Automatic calibration of complete 6DOF camera parameters using only 3D texture map information.
- A novel image descriptor based on Quantized Line parameters in the Hough space (QLH) is proposed for 2D–3D matching-based automatic calibration.
- Global estimation of the 6DOF camera parameters without initial conditions can be performed by applying particle filter-based optimization scheme.

ARTICLE INFO

Article history:
Available online 5 October 2016

Keywords:
Camera calibration
Global localization
Sensor network
Intelligent space

ABSTRACT

To construct an intelligent space with a distributed camera sensor network, pre-calibration of all cameras (i.e., determining the absolute poses of each camera) is an essential task that is extremely tedious. This paper considers the automatic calibration method for camera sensor networks based on 3D texture map information of a given environment. In other words, this paper solves a global localization problem for the poses of the camera sensor networks given the 3D texture map information. To manage the complete calibration problem, we propose a novel image descriptor based on quantized line parameters in the Hough space (QLH) to perform a particle filter-based matching process between line features extracted from both a distributed camera image and the 3D texture map information. We evaluate the proposed method in a simulation environment with a virtual camera network and in a real environment with a wireless camera sensor network. The results demonstrate that the proposed system can calibrate complete external camera parameters successfully.

1. Introduction

Distributed sensor networks installed in external environments can recognize various events that occur in the space, so that such space can be of much service in human–robot coexistence environments, as shown in Fig. 1. In recent years, many studies on an intelligent space, have been performed [1,2]. Distributed camera sensor networks with multi-camera systems provide the most general infrastructure for constructing such intelligent space. In order to obtain reliable information from such a system, pre-calibration of all the cameras in the environment (i.e., determining the absolute positions and orientations of each camera) is an essential task that is extremely tedious. In this respect, several studies that provide Bayesian filter-based probabilistic estimates of optimal sensor parameters and target tracks have been conducted. Foxlin proposed the simultaneous localization and auto-calibration (SLAC) concept, which is a very general architectural framework for navigation and tracking systems with environment sensors [3]. Taylor et al. also implemented a simultaneous localization, calibration, and tracking (SLAT) system using radio and ultrasound pulse-based range sensors as environment sensors [4]. However, these methods can only be applied with range and bearing sensors and cannot make use of information from the camera sensor network, which is a popular network system for the intelligent space.

Chen et al. employed an approach that optimizes robot motion to minimize camera calibration errors; however, their approach assumes that the robot motion has no uncertainty, and rough parameters of the camera should be initialized by human observation [5]. Proposals by Rahimi et al. and Funiak et al. recovered the most likely camera poses and the target trajectory given in a sequence of observations from the camera network [6,7]. However, these approaches have limitations in that they can only estimate 3 degree of freedoms (DOF) poses (x, y, ϕ) with a restrictive assumption that requires aligning each camera's ground-plane coordinate system with a global ground-plane coordinate system. The optimization problem of including orientation parameters for all axes (ψ, θ, ϕ)

☆ This work was in part supported by Tough Robotics Challenge, ImPACT Program (2015-PM07-02-01) (Impulsing Paradigm Change through Disruptive Technologies Program).

* Corresponding author.
E-mail address: ji@robot.t.u-tokyo.ac.jp (Y. Ji).

http://dx.doi.org/10.1016/j.robot.2016.09.015
0921-8890/

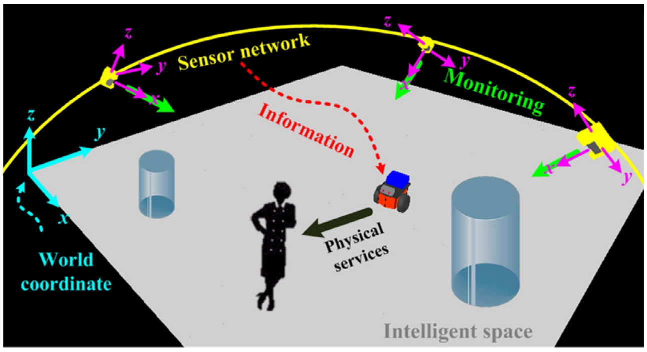

Fig. 1. Concept of intelligent space.

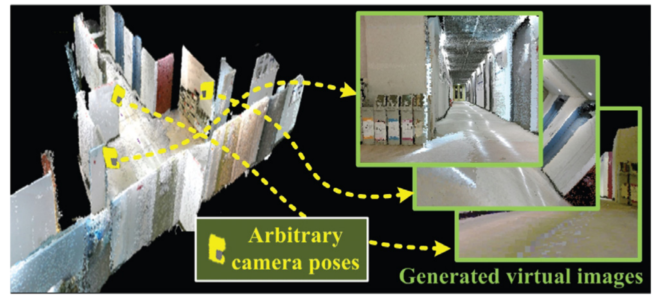

Fig. 2. Arbitrary camera poses and generated virtual images from 3D texture OctoMap information.

could have myriad local minimum solutions without additional constraints because many indistinguishable observations can exist even if the camera poses are different. For the reasons mentioned above, there has been no previous research that tried to estimate complete 6DOF camera parameters.

To overcome this limitation, our research group proposed a novel calibration method that estimates the complete 6DOF poses $(x, y, z, \psi, \theta, \phi)$ for camera sensor networks by applying additional constraints [8]. The additional constraints consist of terms related to grid map information and a two-way observation model based on an assumption that the camera and target can observe each other. However, for this approach, a mobile agent is essential for implementing the calibration methods, so that these methods cannot be applied where the mobile agent cannot be used. In this respect, we propose another approach to achieve a complete calibration scheme for 6DOF external parameters that does not use any mobile agent, but instead, uses only the environmental map information to accommodate situations in which the mobile agent cannot be used. Therefore, the proposed complete 6DOF calibration system is able to construct a camera network system in arbitrary poses in the environment and easily calibrate its parameters under the assumption that there are no large structural alterations to a building (i.e., no wide discrepancies between the real environment and the map information).

In this approach, OctoMap, which is widely used to manage dense 3D environment models with texture information is utilized [9,10]. The OctoMap divides the environment into irregular voxels that are managed in an Octree structure, so that it leads to very efficient memory management compared with a point-based structure. As shown in Fig. 2, the 3D texture map information can be utilized to generate virtual 2D images from arbitrary viewpoints (i.e., arbitrary 6DOF camera poses) using 3D projective geometry when the internal camera parameters are known; thus, the external camera parameters (i.e., 6DOF pose) can be determined by matching the virtual images generated at every viewpoint with real images from the camera sensor networks.

The problem addressed in this study can be considered to be similar to image-based self-localization problems of mobile robots, a dilemma that has been extensively studied over the past few years [11–15]. However, the solutions to these problems cannot be applied to complete 6DOF estimation problems but to only position estimation (or 3DOF estimation) problems, because the motion of mobile robots is expressed by 3DOF in 2D space. Furthermore, these methods can only be applied using omni-directional cameras to efficiently acquire surrounding information of a large environment and cannot make use of information from normal cameras. In particular, the method proposed by Ishizuka et al. achieved self-localization of mobile robots by matching 2D edge points observed from an omni-directional camera to 3D edge points obtained from the 3D environment model [15]. The equivalent method used in this study is called a 2D–3D edge matching scheme and several 6DOF registration techniques for a 3D geometric model (i.e., 3D map information) and 2D image data also have been previously proposed [16–20]. Most of these registration methods are based on the correspondence of 2D photometric edges and projected 3D geometrical edges on the 2D image plane. However, it is difficult to find corresponding edges correctly because robustly extracted edges are limited. Moreover, the initial pose should be manually set close to the correct pose to avoid being stuck in local minimums. To overcome the limitations of setting initial registration, Hara et al. proposed a new registration algorithm that can estimate an optimal pose robustly against initial registration errors [21]. However, it was not completely free from the initial set. The allowed maximum error of the initial registration is 2 m and 20 degrees for each axis. In conclusion, these registration schemes can be applied to matching 3D texture map information with 2D image data; however, initial registration by human observation should be performed and a global search algorithm (i.e., seeking a 6DOF solution in a global space) is yet to be established. Realizing the global search of the 6DOF solution with no strong constraints is impossible because of myriad local minimum solutions; thus, there has been no previous study that attempted this kind of approach. Here, the map information is very useful for reducing the solution space (i.e., the searching space for the 6DOF camera poses) given that the cameras are generally installed on the occupied region, such as interior walls, because of space limitations.

The contributions of this paper are as follows. The limitations of the early approaches are that they can estimate only 3DOF parameters (x, y, ϕ) with restrictive assumptions and a mobile agent is needed for similar calibration patterns, as mentioned above. On the other hand, the proposed complete 6DOF calibration system in this paper only uses the environment map information; therefore, the proposed scheme easily calibrates its parameters without any mobile agent. Moreover, because we apply a novel matching scheme with a line feature-based descriptor that can manage some of the occlusions and clutter, the proposed calibration framework is relatively robust to illumination changes and also manages a few changes in the environment (i.e., discrepancies between the 3D map information and the camera image data) compared with the color information-based approach. In addition, the proposed calibration system requires no initial conditions because the particle filter-based approach, which is adapted for main paradigm for the proposed calibration task, has the ability to solve the global estimation problem, in comparison with previous local estimation scheme [21].

The remainder of this paper is organized as follows. Section 2 presents overview of the proposed calibration process based on 3D texture map information. Section 3 describes the parameterization step which converts 3D texture map information to simpler representation in detail. A novel image descriptor for a fast and accurate line-based matching process is presented in Section 4. Section 5 presents the particle filter-based parameter calibration step. The effectiveness of the proposed calibration scheme is evaluated with the experiment results in Section 6. Finally, Section 7 gives conclusions of this paper.

2. Overview of proposed calibration process

We can take maximum likelihood (ML) or maximum a posterior (MAP) estimation methods into consideration to find the optimal camera pose $\boldsymbol{w}^*$, as follows:

$$\begin{aligned}\boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmax}}\left[p(\boldsymbol{w}|\boldsymbol{I}_R, \mathbb{M}_{oct})\right],\\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\left[p(\boldsymbol{I}_R|\boldsymbol{w}, \mathbb{M}_{oct})p(\boldsymbol{w}|\mathbb{M}_{oct})\right],\\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\left[-\log p(\boldsymbol{I}_R|\boldsymbol{w}, \mathbb{M}_{oct})p(\boldsymbol{w}|\mathbb{M}_{oct})\right],\\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\left[(\boldsymbol{I}_{V(\boldsymbol{w})} - \boldsymbol{I}_R)^\top \Omega_I(\boldsymbol{I}_{V(\boldsymbol{w})} - \boldsymbol{I}_R)\right],\\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\left[\sum_{(u,v)\in \boldsymbol{I}} \|\boldsymbol{I}_{V(\boldsymbol{w})}(u,v) - \boldsymbol{I}_R(u,v)\|^2\right],\end{aligned} \tag{1}$$

where $\boldsymbol{w} = [x_c\ y_c\ z_c\ \psi_c\ \theta_c\ \phi_c]^\top$ denotes the 6DOF camera pose and $\boldsymbol{I}_{V(\boldsymbol{w})}$ represents the virtual image generated from the arbitrary camera pose $\boldsymbol{w}$. $\boldsymbol{I}_R$ is the real image from the camera sensor network. $\mathbb{M}_{oct}$ denotes pre-given 3D texture OctoMap information. Ω_I is an information matrix with regard to the noise of the image data.

In ML estimation method, prior probability distribution $p(\boldsymbol{w}|\mathbb{M}_{oct})$ is neglected, and thus this problem can be very simplest and intuitive method that determines the optimal camera pose $\boldsymbol{w}^*$ by calculating the differences of all pixel intensities between the generated virtual images $\boldsymbol{I}_V$ and the real camera image $\boldsymbol{I}_R$. However, it is impossible to generate innumerable virtual images from the entire solution space because it has a huge number of cases (i.e., in countless numbers of $\boldsymbol{I}_{V(\boldsymbol{w})}$) and the calculation of all projective transformations from a large-scale 3D map information to the 2D image plane demands excessive computational time. Furthermore, if the real image is corrupted by noise (e.g., illumination changes or moving objects), the matching results are significantly affected.

On the other hand, our proposed method to find complete 6DOF external parameters in the global solution space stands in contrast to the ML estimation method mentioned above, given that we exploit the prior information of camera pose with 3D texture OctoMap information $p(\boldsymbol{w}|\mathbb{M}_{oct})$, which is neglected in Eq. (1), as much as possible. We extract line features from both the 3D texture OctoMap information and real image data in order to make strong constraints from the given map information. The line features occupy only small parts in an overall environment, and thus the computational cost to manage this type of light features can be significantly reduced. As shown in the Fig. 3, line segments appear to be very efficient features because they are noticeable as common segments between the 3D texture map information and 2D image data, and are relatively unaffected by illumination changes compared to color information; however, lengths, angles, and parallelism of the line features are not conserved in 3D projective geometry. In this respect, this study proposes a novel image descriptor based on Quantized Line parameters in the Hough space (QLH) in order to determine 6DOF camera poses using a particle filter-based approach, which is one of the popular implementations of Bayesian filters that can track the distribution of probability using a set of particles.

Fig. 4 shows a flowchart of the overall proposed 6DOF calibration process in this study. The process is divided into two steps: "parameterization of 3D geometric lines" to generate parameters of the 3D geometric line segments that correspond to the entire environment, and "particle filter-based calibration" to perform camera pose estimation. During the parameterization step, the 3D geometric line parameters of the environment are generated automatically from dense 3D texture OctoMap information. The calibration step determines the 6DOF camera parameters by matching the generated 3D geometric line parameters with the 2D photometric line parameters extracted from the real image data from the camera sensor network. Here, the sequential importance resampling (SIR) particle filter algorithm is used for the matching process [22]. This study focuses on considering the expression of the line features and design of their new measurement model in order to apply the SIR particle filter algorithm.

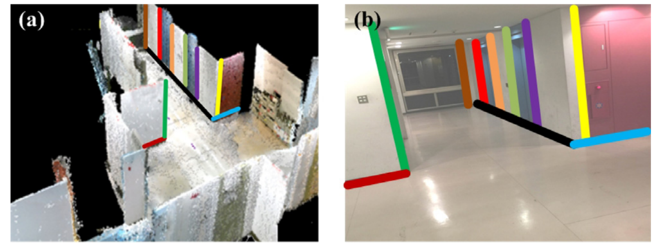

Fig. 3. Line features (bold lines in same color refer to features located on same area): (a) line features in 3D texture map information (i.e., 3D geometric lines) and (b) line features in 2D image (i.e., 2D photometric lines). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

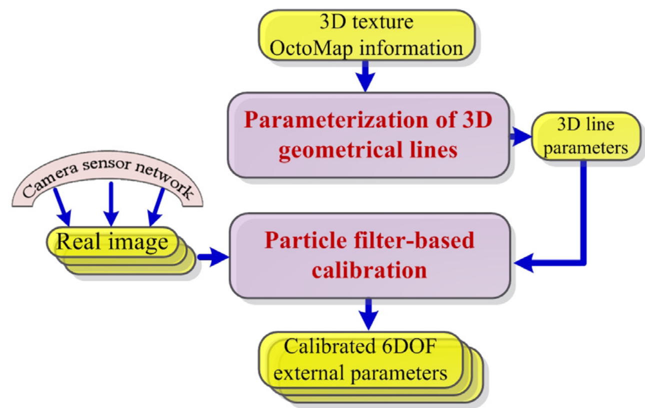

Fig. 4. Overview of proposed 6DOF calibration process.

3. Parameterization of 3D geometric lines based on 3D texture OctoMap

We can take direct parameterization of 3D geometric line segments into consideration from the 3D map information as shown in Fig. 5(a) because the 3D map may appear clear and valid for extracting the major line segments by estimating the intersections of the planes. However, the map structure consists not of planes, but instead of many voxels (i.e., Octree structure as mentioned in Section 1). Thus, additional processing is required to estimate geometric plane information from the voxels in advance [23]. In this study, the parameterization process of the 3D geometric line segments for the entire environment consists of three major steps: extraction of the point cloud on the line segments, outlier elimination, and learning robust line parameters, as illustrated in Fig. 5(b), (c), and (d). These steps make use of the 3D texture OctoMap information as training data. Each step functions as follows:

1. The "extraction of the point cloud on line segments" step involves generating candidate 3D geometric line segments

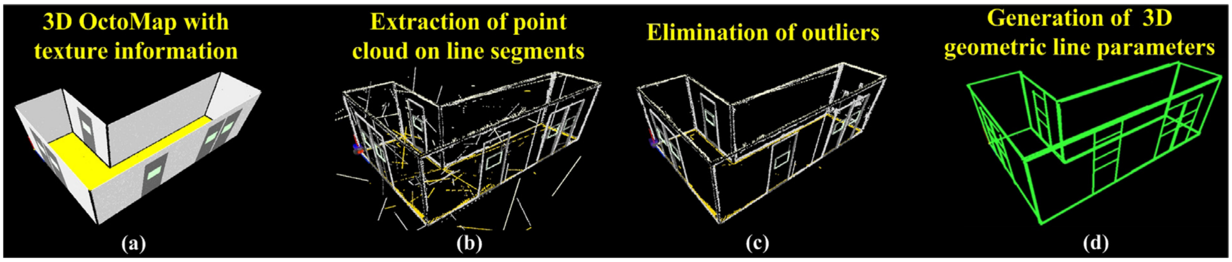

Fig. 5. Parameterization of 3D geometric lines: (a) 3D texture OctoMap information $\mathbb{M}_{oct}$ (for clarity, occupied nodes that correspond to ceiling area are removed), (b) extracted point cloud $\mathbb{p}_{lines}$ on 3D geometric line segments, (c) point cloud $\mathbb{p}_{lines}$ where outliers are removed, and (d) learned robust 3D geometric line parameters $\mathbb{l}$.

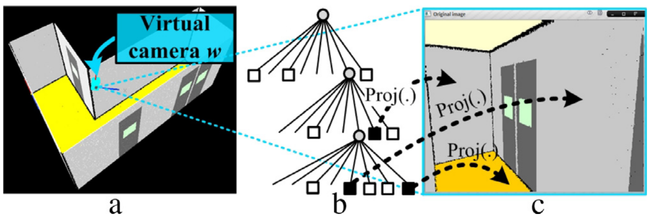

Fig. 6. Generation of virtual image: (a) 3D texture OctoMap information $\mathbb{M}_{oct}$, (b) Octree structure that constitutes 3D OctoMap (free and occupied nodes are represented by white and black squares, respectively), and (c) generated virtual image from virtual camera $\boldsymbol{w}$.

by searching occupied nodes of an Octree structure that constitutes the 3D textured OctoMap. As shown in Fig. 6(b), the Octree structure is composed of tree-based node information, and thus high-speed searches can be performed. Searching the occupied nodes of the 3D texture OctoMap information is reasonable because the occupied nodes represent the occupied spaces where the camera sensor networks can be installed.

2. During the "outlier elimination" step, the principle component analysis (PCA), Bayes' rule, and clustering in a direction vector space are adopted to keep only frequently observed 3D geometric line segments (i.e., robustly extracted ones at any viewpoint) and remove rarely observed ones, as shown in Fig. 5(c).
3. The "learning robust line parameters" step learns coefficients of the 3D geometric line segments, as shown in Fig. 5(d). Here, the random sample consensus (RANSAC) algorithm is recursively performed.

3.1. Extraction of point cloud on line segments

Table 1 describes an algorithm for generating the point cloud $\mathbb{p}_{lines}$ on the 3D geometric line segments from the 3D texture OctoMap information $\mathbb{M}_{oct}$. The 3D geometric line segments are composed of point cloud data $\mathbb{p}_{lines}$ in this process. This process performs iteration for every occupied node $\boldsymbol{n}_{\text{occ}}$. First, a position vector (x, y, z) that corresponds to an occupied node $\boldsymbol{n}_{\text{occ}}$ and a random orientation vector (ψ, θ, ϕ) are produced to generate an arbitrary viewpoint (i.e., an arbitrary 6DOF camera pose $\boldsymbol{w}$) in lines 2–4. In line 5, a virtual image $\boldsymbol{I}_{V(\boldsymbol{w})}$ that corresponds to the arbitrary camera pose $\boldsymbol{w}$ is generated as described in the following paragraphs.

Fig. 6 shows that the color information of every occupied node that constitutes the 3D OctoMap is projected onto a virtual 2D image plane (i.e., the color information assigns to each corresponding pixel value $\boldsymbol{I}_{V(\boldsymbol{w})}(u, v)$) using the camera parameters. The position $\check{\boldsymbol{u}} = [u\ v\ s]^{\top}$ and its corresponding pixel area size A_{pixel} are calculated by:

$$\begin{aligned}\check{\boldsymbol{u}} &= proj(\check{\boldsymbol{n}}_{\text{occ}}) \\ &= \boldsymbol{E}\boldsymbol{T}\check{\boldsymbol{n}}_{\text{occ}}\end{aligned}$$

$$\begin{bmatrix} u \\ v \\ s \end{bmatrix} = \begin{bmatrix} f_u & f_{\text{skew}} & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_{\text{occ}} \\ y_{\text{occ}} \\ z_{\text{occ}} \\ 1 \end{bmatrix}, \tag{2}$$

$$A_{\text{pixel}} = \frac{(l_{\text{node}})^2 f_u f_v}{(d_{\text{optical}})^2}, \tag{3}$$

$$d_{\text{optical}} = \begin{bmatrix} r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \check{\boldsymbol{n}}_{\text{occ}}, \tag{4}$$

where $\check{\boldsymbol{n}}_{\text{occ}} = [x_{\text{occ}}\ y_{\text{occ}}\ z_{\text{occ}}\ 1]^{\top}$ is the position vector of an occupied node. s is a scale factor. $\boldsymbol{E}$ denotes the internal parameter matrix that includes focal length (f_u, f_v), skew coefficient f_{skew}, and principal point (c_u, c_v). $\boldsymbol{T}$ is the external parameter matrix transformed from the 6DOF camera pose $\boldsymbol{w}$. l_{node} and d_{optical} respectively represent the cube of edge length that corresponds to the occupied node $\boldsymbol{n}_{\text{occ}}$ for the 3D OctoMap and the distance on the camera's optical axis (i.e., x-axis) between the occupied node and camera position. Here, only the nodes of the shortest distance d_{optical} are projected onto virtual image $\boldsymbol{I}_{V(\boldsymbol{w})}$ on the optical camera characteristic when multiple nodes are projected onto the same pixel location. Note that the coordinate system adopted in this study (i.e., the relationship between the camera coordinate system and the global coordinate system in the 3D space) is appeared in Fig. 1.

After generating virtual image $\boldsymbol{I}_{V(\boldsymbol{w})}$, 2D photometric line segments $\mathbb{u}_{lines}$ are extracted (line 6). Fig. 7 shows the procedure for generating the 2D photometric line segments. First, a binary image is generated from the virtual image using the Canny edge detection process, as shown in Fig. 7(a) and (b). The straight line segments are then extracted through the probabilistic Hough

Table 1
Algorithm to generate point cloud on 3D geometric line segments.

Algorithm Extraction of point cloud in line segments
input : 3D texture OctoMap $\mathbb{M}_{\text{oct}}$, intrinsic parameter matrix $\boldsymbol{E}$
output : point cloud on line segments $\mathbb{p}_{\text{lines}}$
1: for every occupied node $\boldsymbol{n}_{\text{occ}} \in \mathbb{M}_{\text{oct}}$
2: get position (x, y, z) that corresponds to occupied node $\boldsymbol{n}_{\text{occ}}$
3: generate random orientation $(\psi.\theta, \phi)$
4: $\boldsymbol{w} = (x, y, z, \psi.\theta, \phi)$
5: generate virtual image $\boldsymbol{I}_{V(\boldsymbol{w})}$ where pose of virtual camera is $\boldsymbol{w}$ with its intrinsic parameter matrix $\boldsymbol{E}$
6: extract pixels on 2D photometric line segments $\mathbb{u}_{\text{lines}}$ from virtual image $\boldsymbol{I}_{V(\boldsymbol{w})}$
7: generate point cloud $\acute{\mathbb{p}}_{\text{lines}}$ that corresponds to 2D photometric line segments by back-projecting $\mathbb{u}_{\text{lines}}$ onto $\mathbb{M}_{\text{oct}}$
8: add $\acute{\mathbb{p}}_{\text{lines}}$ to accumulated $\mathbb{p}_{\text{lines}}$
9: end for
10: return $\mathbb{p}_{\text{lines}}$

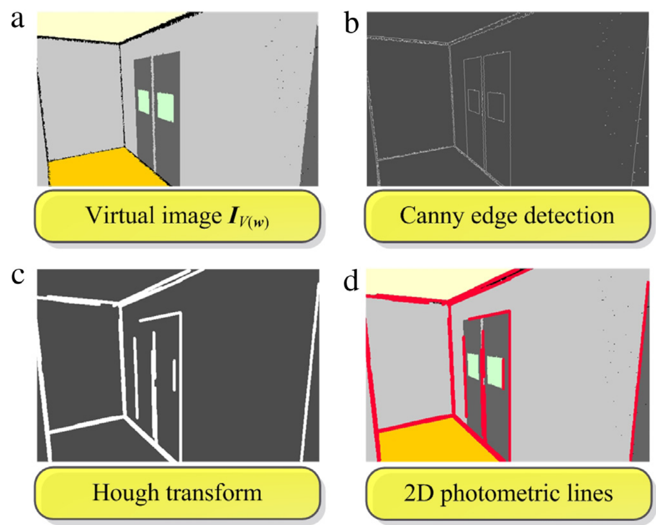

Fig. 7. Procedure for generating 2D photometric line segments: (a) virtual image generated from arbitrary camera pose $\boldsymbol{w}$, (b) result of binary image after applying Canny edge detection, (c) extracted straight line segments (red lines) through Hough transform, and (d) 2D photometric lines $\mathbb{u}_{\text{lines}}$ which are represented by white pixels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

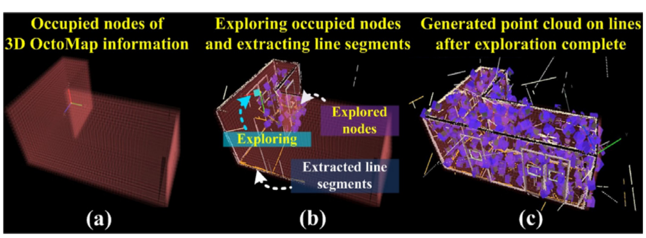

Fig. 8. Overall process for generating point cloud $\mathbb{p}_{lines}$ on 3D geometric line segments: (a) all occupied nodes $\boldsymbol{n}_{\text{occ}}$ of 3D OctoMap $\mathbb{M}_{oct}$, (b) exploration for occupied nodes to extract 3D geometric line segments, and (c) generated point cloud $\mathbb{p}_{lines}$ on 3D geometric line segments after exploration complete.

transform (Fig. 7(c)) [24]. Next, a set of 2D pixel coordinates $\mathbb{u}_{lines}$ that correspond to the 2D photometric lines are generated from the line image, as shown in Fig. 7(d).

$$\mathbb{u}_{lines} = \left[\mathbb{u}_{\text{line}}^{(k)} \quad | \quad 1 \le k \le N_{\text{line}}\right], \tag{5}$$

$$\mathbb{u}_{\text{line}}^{(k)} = \left[\boldsymbol{u}_i^{(k)} \quad | \quad 1 \le i \le N^{(k)}\right], \tag{6}$$

$$\boldsymbol{u}_i^{(k)} = \left[u_i^{(k)} \quad v_i^{(k)}\right]^{\top}. \tag{7}$$

Here, $\mathbb{u}_{\text{line}}^{(k)}$ and $N^{(k)}$ denote the pixel points on the kth 2D photometric line segment and the number of pixels points, respectively. N_{line} represents the number of 2D photometric line segments in virtual image plane $\boldsymbol{I}_{V(\boldsymbol{w})}$.

In line 7, the pixel points on the 2D photometric lines $\mathbb{u}_{lines}$ are back-projected onto the 3D environment model (i.e., 3D OctoMap), so that 3D point cloud $\acute{\mathbb{p}}_{lines}$ composed of the candidates of the 3D geometric lines are generated, as follows:

$$\acute{\mathbb{p}}_{lines} = \left[\acute{\mathbb{p}}_{\text{line}}^{(k)} \quad | \quad 1 \le k \le N_{\text{line}}\right], \tag{8}$$

$$\acute{\mathbb{p}}_{\text{line}}^{(k)} = \left[\acute{\boldsymbol{p}}_i^{(k)} \quad | \quad 1 \le i \le N^{(k)}\right], \tag{9}$$

$$\acute{\boldsymbol{p}}_i^{(k)} = \left[x_i^{(k)} \quad y_i^{(k)} \quad z_i^{(k)}\right]^{\top}. \tag{10}$$

Here, $\acute{\mathbb{p}}_{\text{line}}^{(k)}$ denotes the points on the kth 3D geometric line segment and the 3D geometric point.

Table 2
Relationship between shape of point cloud and each eigenvalue.

Shape	Linear (1D)	Planar (2D)	Volumetric (3D)
λ_1 value	large	large	large
λ_2 value	small	large	large
λ_3 value	small	small	large
Relationship	$\lambda_1 \gg \lambda_2 \cong \lambda_3$	$\lambda_1 \cong \lambda_2 \gg \lambda_3$	$\lambda_1 \cong \lambda_2 \cong \lambda_3$

Finally, the point set of the 3D geometric lines $\acute{\mathbb{p}}_{lines}$ extracted from every node are integrated into the overall point cloud $\mathbb{p}_{lines}$ (line 8). The overall process described above is shown in Fig. 8.

3.2. Outlier elimination

As illustrated in Figs. 5(b) and 8(c), the generated point cloud $\mathbb{p}_{lines}$ contains outliers, thus leading to inaccurate matching results for a posterior calibration process. To eliminate these outliers produced during the back-projection step, three types of noise removal schemes can be applied: PCA, Bayes' rule, and direction vectors of 3D geometric lines.

3.2.1. Outlier removal using PCA evaluation

The candidates for 3D geometric lines are evaluated through PCA to determine whether the shapes are straight lines even on a 3D space, so that only those points that are included in straight lines in both the 2D and 3D spaces are kept. The PCA evaluation scheme for the 3D geometric line candidates is represented by:

$$\boldsymbol{Cov}\left(\acute{\mathbb{p}}_{\text{line}}^{(k)}\right) = \frac{1}{N^{(k)}} \sum_{i \in \text{Line } k} \left(\acute{\boldsymbol{p}}_i^{(k)} - \bar{\acute{\boldsymbol{p}}}_i^{(k)}\right)\left(\acute{\boldsymbol{p}}_i^{(k)} - \bar{\acute{\boldsymbol{p}}}_i^{(k)}\right)^{\top}, \tag{11}$$

$$\bar{\acute{\boldsymbol{p}}}_i^{(k)} = \frac{1}{N^{(k)}} \sum_{i \in \text{Line } k} \acute{\boldsymbol{p}}_i^{(k)}. \tag{12}$$

Here, $\acute{\boldsymbol{p}}_i^{(k)}$ and $\bar{\acute{\boldsymbol{p}}}_i^{(k)}$ are 3D point coordinates and the center value that constitutes the kth line candidate, respectively. $N^{(k)}$ represents the number of points that constitute the kth line candidate. By performing eigenvalue analysis of covariance matrix $\boldsymbol{Cov}(\acute{\mathbb{p}}_{\text{line}}^{(k)})$, eigenvalues $\lambda_1^{(k)} > \lambda_2^{(k)} > \lambda_3^{(k)}$ and eigenvectors $\boldsymbol{e}_1^{(k)}$, $\boldsymbol{e}_2^{(k)}$, and $\boldsymbol{e}_3^{(k)}$ can be calculated, respectively. Dimensionality labeling of the point cloud (i.e., whether the point cloud is spread into one, two, or three dimensions) can be derived from these eigenvalues. Given the relationship between the value of each eigenvalue and the shape of the point cloud (Table 2), the shape of the point cloud can be classified by calculating the difference between each eigenvalue, as follows:

$$d^{(k)} = \text{argmax}_{i=1,2,3}\left[s_i^{(k)}\right], \tag{13}$$

$$s_1^{(k)} = \lambda_1^{(k)} - \lambda_2^{(k)}, \tag{14}$$

$$s_2^{(k)} = \lambda_2^{(k)} - \lambda_3^{(k)}, \tag{15}$$

$$s_3^{(k)} = \lambda_3^{(k)}. \tag{16}$$

In Eq. (13), $d^{(k)}$ represents the index i of s with maximum value. Therefore, line segments where the evaluation result is $d^{(k)} = 1$ can be accepted as a 1D straight-line feature in both the 2D and 3D space, and others (i.e., $d^{(k)} = 2$ or 3) are not. More details on the shape classification for the point cloud can be found in [25]. Fig. 9 illustrates examples of 2D photometric lines and their back-projection results. The unexplainable 2D photometric line is occasionally extracted and it should be eliminated because it is not a 3D geometric line.

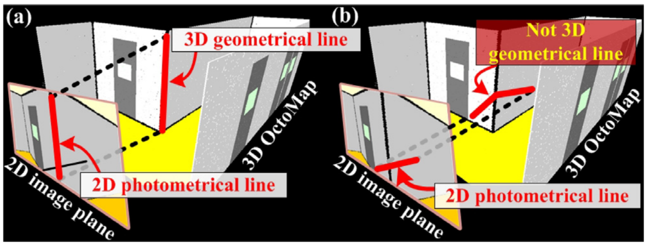

Fig. 9. 2D photometric lines and their back-projected examples onto 3D OctoMap information: (a) 3D geometric line ($d^{(k)} = 1$) and (b) not 3D geometric line ($d^{(k)} = 3$).

3.2.2. Outlier removal using Bayes' rule

In order to maintain only frequently extracted lines and remove rarely detected ones, Bayes' rule can also be adopted while the algorithm for generating the point cloud on the 3D geometric line segment (Table 1) is performed. A probability is assigned to each line segments and is updated according to the Bayesian update formula, as follows:

$$p(S(\acute{\boldsymbol{p}}) = T_{\text{line}} | O(\acute{\mathbb{p}}_{lines})_{1:k}) = \frac{p(O(\acute{\mathbb{p}}_{lines})_k | S(\acute{\boldsymbol{p}}) = T_{\text{line}}) p(S(\acute{\boldsymbol{p}}) = T_{\text{line}} | O(\acute{\mathbb{p}}_{lines})_{k-1})}{\sum_{S(\acute{\boldsymbol{p}})} p(O(\acute{\mathbb{p}}_{lines})_k | S(\acute{\boldsymbol{p}})) p(S(\acute{\boldsymbol{p}}) = | O(\acute{\mathbb{p}}_{lines})_{k-1})}, \quad (17)$$

where $S(\acute{\boldsymbol{p}})$ denotes whether point $\acute{\boldsymbol{p}}$ belongs to a 3D geometric line ($S(\acute{\boldsymbol{p}}) = T_{\text{line}}$) or not ($S(\acute{\boldsymbol{p}}) = F_{\text{line}}$). $O(\acute{\mathbb{p}}_{lines})_{1:k} = [O(\acute{\mathbb{p}}_{lines})_1, O(\acute{\mathbb{p}}_{lines})_2, \ldots, O(\acute{\mathbb{p}}_{lines})_k]^\top$ represents the observations of point $\acute{\boldsymbol{p}}$ from 1st to kth iteration of the exploration. The initial probability $p(S(\acute{\boldsymbol{p}}) = T_{\text{line}})$ is set to 0.5. The likelihood $p(O(\acute{\mathbb{p}}_{lines})_k | S(\acute{\boldsymbol{p}}))$ can be defined as 0.8 when the line segment is observed ($S(\acute{\boldsymbol{p}}) = T_{\text{line}}$), and 0.1 when the line segment is not observed ($S(\acute{\boldsymbol{p}}) = F_{\text{line}}$). By performing sequential updates based on Eq. (17), the probabilities of frequently observed points that belong to line segments gradually increase; thus, the points with low probability (e.g., less than 0.9) can be eliminated. Note that Eq. (17) is the same expression as the update rule of occupancy probabilities in the occupancy grid mapping algorithm [26].

3.2.3. Outlier removal using direction vectors

The removal of outliers can be performed by clustering direction vector information $\boldsymbol{a}^{(k)}$ corresponding to each 3D geometric line $\mathbb{p}^{(k)}_{\text{line}}$.

$$\mathbb{a}_{\text{lines}} = \left[\boldsymbol{a}^{(k)} \quad | \quad 1 \le k \le N_{\text{line}}\right], \quad (18)$$

$$\boldsymbol{a}^{(k)} = \left[a_x^{(k)} \quad a_y^{(k)} \quad a_z^{(k)}\right]^\top. \quad (19)$$

Here, the normalized (i.e., $\|\boldsymbol{a}^{(k)}\| = 1$) direction vector $\boldsymbol{a}^{(k)}$ can be easily calculated using two points on the 3D geometric line segment. Each direction vector can be plotted in the 3D direction vector space, as shown in Fig. 10(c). Here, because we can intuitively recognize that areas of dense points mean principal directions among all 3D geometric line segments, the outliers are simply eliminated through a radius outlier removal filter, as shown Fig. 10(d). Fig. 11 helps visualize the action of the radius outlier removal filter. The user specifies a number of neighbors where every index must be within a specified radius r to remain in the original points. For example, if one neighbor is specified, only the green point is removed from the original points. If two neighbors are specified, both the green and yellow points are removed from the original points. The elimination result shown in Fig. 11(b) illustrates a case where two neighbors are specified.

Codebooks for principle directions of 3D geometric line segments are then generated through clustering for the remaining

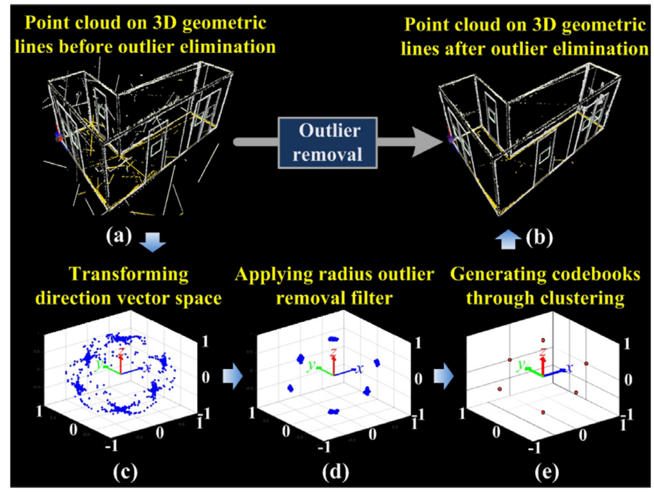

Fig. 10. Outlier elimination by clustering in direction vector space: (a) point cloud $\mathbb{p}_{lines}$ on 3D geometric line segments before outlier elimination, (b) point cloud $\mathbb{p}_{lines}$ after outlier elimination, (c) each direction vector $\mathbb{a}_{\text{lines}}$ plotted in direction vector space before applying radius outlier removal filter, (d) each direction vector $\mathbb{a}_{\text{lines}}$ after applying radius outlier removal filter, and (e) generated codebooks for principle directions of 3D geometric line segments.

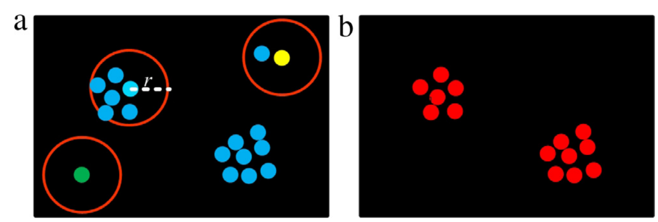

Fig. 11. Radius outlier removal filter: (a) before applying and (b) after applying. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

direction vectors, as shown Fig. 10(e). The codebooks can be obtained by calculating the center for each classified cluster. Here, the Euclidean clustering method with Kd-tree representation [27] is used to extract clusters that correspond to the direction vectors.

In a typical indoor environment, as shown in Fig. 10(e), six codebooks are generated (cf. without considering the sign, three codebooks) for the direction vectors. Finally, the line segments (strictly speaking, point cloud data) for which there is no codebook within a threshold distance can be eliminated by calculating Euclidean distances between each direction vector of the line segments and the codebooks.

3.3. Generation of 3D geometric line parameters

$\mathbb{p}_{lines}$ is composed of still point cloud data; thus, a simpler representation for defining 3D geometric line segments should be considered because the computational burden would be very high to process it. For example, a considerable number of variables is necessary for representing the point cloud data, whereas only six parameters are sufficient for representing the equation of a line in 3D Space. During this process, therefore, parameterization of 3D geometric line segments through the RANSAC algorithm [28] is performed, as shown in Fig. 5(d). The parametric model is represented through a set of coefficients. The 3D geometric line segments can be parameterized using only six coefficients, as follows:

$$\frac{x - x_s}{x_e - x_s} = \frac{y - y_s}{y_e - y_s} = \frac{z - z_s}{z_e - z_s}. \quad (20)$$

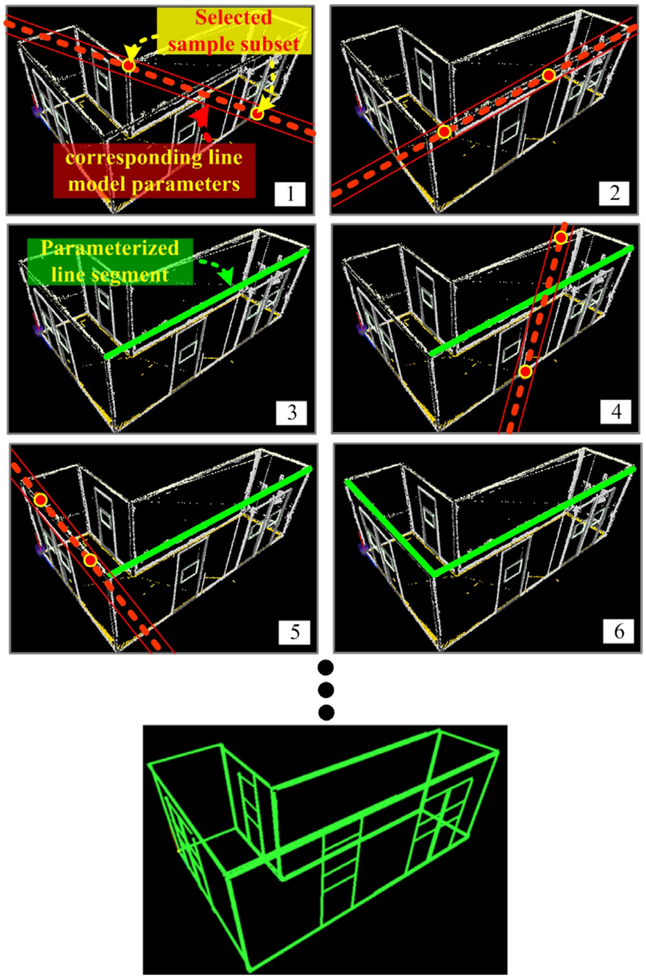

Fig. 12. Overall RANSAC process for generating 3D geometric line parameters.

Here, (x_s, y_s, z_s) and (x_e, y_e, z_e) denote the start and end point of a 3D geometric line segment, respectively. They also represent the parameters of a 3D geometric line segment. Essentially, this step produces a list of these six line parameters from point cloud data $\mathbb{p}_{lines}$ composed of 3D geometric line segments.

The RANSAC algorithm is an iterative learning method for estimating the parameters $(x_s, y_s, z_s, x_e, y_e, z_e)$ of a mathematical model (i.e., the 3D line) from a set of observed data (i.e., point cloud data $\mathbb{p}_{lines}$). It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increases as more iterations are allowed [28]. RANSAC uses a voting scheme to find the optimal fitting result. The RANSAC algorithm for parameterizing 3D geometric lines that consist of $\mathbb{p}_{lines}$ is divided into two steps repeated iteratively. In the first step, a sample subset that contains minimal data points (e.g., two points in the case of the line model) is randomly selected from input point cloud $\mathbb{p}_{lines}$. A fitting model and the corresponding line model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest that is sufficient for determining the model parameters. In the second step, the algorithm checks which elements of the entire point cloud are consistent with the line model instantiated by the estimated model parameters obtained from the first step. A data element is considered an outlier if it does not fit the line model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the noise effect. The set of inliers obtained for the fitting model is called a consensus set. The RANSAC algorithm iteratively repeats the above two steps until the consensus set obtained in certain iteration has sufficient inliers. The overall process for RANSAC mentioned above is illustrated in Fig. 12. Through repetition of these steps, every 3D geometric line segment in $\mathbb{p}_{lines}$ can be parameterized as follows:

$$\mathbb{l} = \left[\boldsymbol{l}^{(k)} \quad | \quad 1 \le k \le N_{\text{line}} \right], \tag{21}$$

$$\boldsymbol{l}^{(k)} = \left[x_s^{(k)} \quad y_s^{(k)} \quad z_s^{(k)} \quad x_e^{(k)} \quad y_e^{(k)} \quad z_e^{(k)} \quad \right]^{\top}. \tag{22}$$

In conclusion, the number of variables necessary for representation of the parameterized line segments is very low (i.e., $N_{\text{line}} \times 6$); thus, the computational burden for the posterior processes can be reduced significantly.

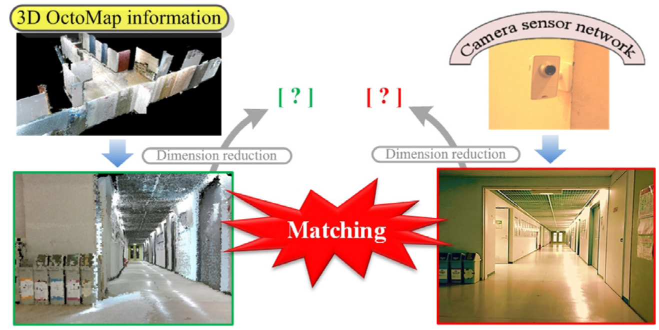

Fig. 13. Compressed-dimensional image descriptor for comparing image data from real camera with virtual image data from arbitrary camera poses in 3D texture map information.

4. Novel image descriptor based on QLH

4.1. Concept of QLH descriptor

In order to estimate camera parameters based on 3D texture map information, a novel feature comparison model is required for a fast and accurate matching process. In other words, as shown in Fig. 13, an image descriptor (i.e., the compressed-dimensional signature) should be defined to compare image data from a real camera with virtual image data from arbitrary camera poses in the map information.

To this end, the concept of histograms is widely exploited in image matching processes (e.g., object recognition or similar image search) because it can be used to represent diverse items, such as an intensity of the color distribution of an image. Fig. 14(b) shows examples of the intensity histograms that correspond to two image data. To compare two histograms (i.e., similarity comparison between the two images), many matching criteria have been proposed, such as correlation, Chi-square, intersection, and Bhattacharyya distance [29]. Among them, Chi-square or Bhattacharyya distance produce good matching results for color intensity histogram-based image matching when the environmental conditions are the same. However, intensity-based comparison schemes depend on the environment's lighting conditions, and thus they are not robust to illumination changes. Furthermore, moving objects also significantly impact the distribution of the color intensity information in the image data. Using local features in images (e.g., SIFT [30] or SURF [31]) can be also considered because these are relatively robust to illumination changes. Furthermore, it is possible to exploit bag of visual words (BoVW) techniques to perform these types of keypoints matching [32–35]. Recently, BoVW has been applied to the localization problem [36–38]. These schemes, however, utilize corner-based feature

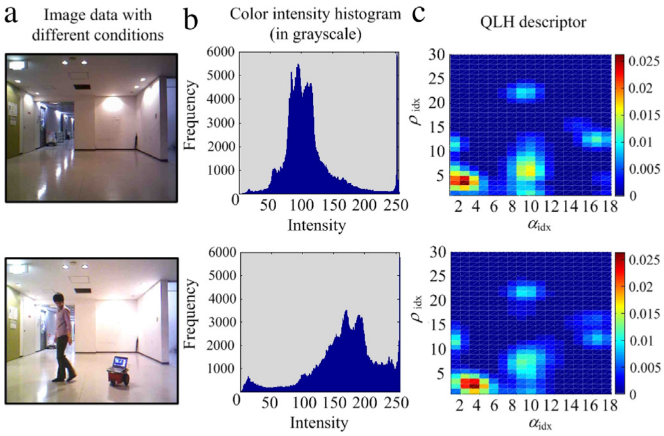

Fig. 14. Examples of image matching criteria: (a) original images, (b) color intensity histograms, and (b) QLH descriptors.

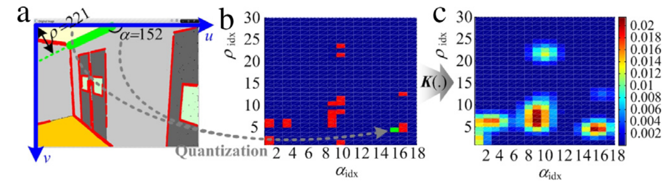

Fig. 15. Example of QLH descriptor: (a) extracted 2D photometric lines from image data, (b) quantization and registration at Hough space, (c) after applying Gaussian Kernel function $\boldsymbol{K}(\cdot)$ to manage noise.

points, and thus they do not apply to non-textured environments where robust feature points cannot be reliably extracted. FAB-MAP proposed by M. Cummins et al. also uses the BoVW descriptor to solve the problem of recognizing places based on their appearance [39–41]. However, this appearance-based approach using Bayesian pattern recognition is only able to recognize similar scenes, instead of precise position and orientation estimation, given that these types of local keypoints are invariant to uniform scaling and orientation. Tomono also exploited a BoVW-based image-retrieval scheme using edge points in order to estimate 6DOF camera poses. In this study, localization is performed based on a landmark image database; thus, the performance might decrease when the target scene does not include any similar visual vocabulary generated from the image database [42]. Higher order local autocorrelation (HLAC) [43], generalized local correlation (GLC) [44], and global Gaussian (GG) [45], which represent an image as a single descriptor, are also not suitable solutions for position or orientation estimation. The central problem of image matching in this study is that the generated virtual images (Fig. 13 left) provide very poor resolution compared to real camera images (Fig. 13 right).

To this end, a novel image descriptor based on QLH that represents the distribution of the slope and distance from the origin of the 2D image plane for the 2D photometric lines is proposed in this study, as shown in Fig. 14(c), because the 2D photometric line features are robustly extracted without reference to the image resolution, unlike the keypoint-based features. Furthermore, the proposed QLH descriptor-based matching scheme utilized in this study does not need to generate low-resolution virtual images because the 3D texture map information $\mathbb{M}_{oct}$ is converted into 3D geometric line segments $\mathbb{l}$ beforehand, as described in Section 3. Unlike the image descriptors previously mentioned, the QLH descriptor is sensitive to translation and rotation changes of the camera scenes since it is generated based on line parameters. Thus, it is highly suitable for pose estimation, including the position and orientation. Tomono also suggested a similar concept of an Euclidean invariant signature which is defined by an index table in the Hough space [46]. It was utilized for scan matching between 2D measurements from a laser range finder (LRF). On the other hand, the proposed QLH descriptor is generated from image information captured by cameras, instead of 2D range data. Therefore, the QLH descriptor is relatively robust to illumination changes because it does not include any intensity information and it is always available, provided that the edge information is detected. For example, as shown in Fig. 14(b), the color intensity histogram changes significantly when it is affected by illumination conditions and moving objects; therefore, similarity matching cannot be performed correctly regardless of whether the scenes are from the same environment or not. On the other hand, the proposed QLH descriptor is relatively robust to changes in the environment, as illustrated in Fig. 14(c). The generation method for the QLH descriptor is described in detail in next subsection.

4.2. Generation of QLH descriptor

2D photometric line information in the image plane is uniquely determined by two properties: slope α and the distance from the origin ρ given by:

$$\rho = u\cos\alpha + v\sin\alpha, \tag{23}$$

where (u, v) represents the image coordinates. ρ is the distance from the origin to the closest point on the straight line, and α is the angle between the u-axis and the line that connects the origin with the closest point. Therefore, it is possible to associate each line of the image with a pair (ρ, α). The (ρ, α) plane is referred to as the Hough space for the set of straight lines in the 2D space [47]. Hence, it can be combined to design a new image descriptor for line segment-based matching. The descriptor is generated by quantizing the ρ and α information of 2D photometric lines in the Hough space with a Gaussian Kernel function.

The QLH descriptor for a real camera image is generated simply from the image that includes 2D photometric line segments, as illustrated in Fig. 7(d). On the other hand, the predicted QLH image descriptors from the arbitrary camera poses in the 3D map information can be obtained from 2D photometric line segments $\mathbb{l}_{2D}$ generated by projecting the parameterized 3D geometric lines segments $\mathbb{l}$ onto the virtual 2D image plane. Table 2 lists the algorithm for generating the QLH descriptor from 2D photometric line segments.

Lines 1–4 quantize the slope and the distance values of each 2D photometric line $\boldsymbol{l}_{2D}^{(j)} = (\rho^{(j)}, \alpha^{(j)})$ to transform into the index $\boldsymbol{l}_{idx} = (\rho_{idx}, \alpha_{idx})$ on the Hough space, and $round(\cdot)$ represents the rounding function. Here, $b_\rho \times b_\alpha$ is the quantum size on the Hough space. After quantizing, line 5 calculates each element of QLH descriptor $\boldsymbol{Q}$ using the typical Gaussian Kernel function $\boldsymbol{K}(\cdot)$ (i.e., $\boldsymbol{O}$ mean and $\boldsymbol{I}$ covariance), which is given by

$$\boldsymbol{K}(\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x}\right), \tag{24}$$

where N_j, h, and D denote the number of 2D photometric lines, smoothing parameter, and dimensions (two in this case), respectively. A matrix norm of $\boldsymbol{Q}$ depends on the number of registered lines; thus, it affects accuracy in the matching process. To eliminate this effect, the result is normalized in line 7. An intuitive example for generating a QLH descriptor is shown in Fig. 15. Each 2D photometric line segment is quantized $(\rho_{idx}, \alpha_{idx})$ and registered at the corresponding cell in the Hough space (Fig. 15(b)). Then, Gaussian Kernel function $\boldsymbol{K}(\cdot)$ is applied to manage unexpected noise (Fig. 15(c)). Although the environment is fixed, the QLH image descriptor changes depending on the camera pose, and thus the QLH descriptor can be used as an available signature for the calibration process, as described in next subsection.

4.3. Evaluation of QLH descriptor based on earth mover's distance

The similarity between two sets of QLH descriptors is computed with regard to their earth mover's distance (EMD). EMD is a measure of the distance between two multi-dimensional distributions in some feature space [48]. In this study, the QLH descriptor can be managed as a 2D distribution, which is defined in the feature space $(\rho_{\mathrm{idx}}, \alpha_{\mathrm{idx}})$. In order to compute EMD between two QLH descriptors, each QLH descriptor is converted to a set of clusters $\boldsymbol{Q}_{\mathrm{sig}}$.

$$\boldsymbol{Q}_{\mathrm{sig}}^{(1)} = \left[(\boldsymbol{q}_i^{(1)}, \varpi_i^{(1)}) \quad | \quad 1 \le i \le N_c\right], \tag{25}$$

$$\boldsymbol{Q}_{\mathrm{sig}}^{(2)} = \left[(\boldsymbol{q}_j^{(2)}, \varpi_j^{(2)}) \quad | \quad 1 \le j \le N_c\right], \tag{26}$$

where $\boldsymbol{q}$ and ϖ indicate the cluster representative (i.e., the coordinates $(\rho_{\mathrm{idx}}, \alpha_{\mathrm{idx}})$ of the QLH descriptor) and the weight value that belongs to that cluster (i.e., $\boldsymbol{Q}(\rho_{\mathrm{idx}}, \alpha_{\mathrm{idx}})$), respectively. The size of cluster N_c is equal to the size of the QLH descriptor (e.g., $30 \times 18 = 530$ in the case of Fig. 15). Computing EMD is based on a solution to the well-known transportation problem. In other words, EMD measures the minimum amount of work required to change one signature to another. Here, the notion of work is based on the user-defined ground distance d_{ij}. Therefore, a flow matrix $\boldsymbol{F}$ with flow elements f_{ij} between $\boldsymbol{q}_i^{(1)}$ and $\boldsymbol{q}_j^{(2)}$ that minimizes the following overall cost is calculated.

$$\begin{aligned}
&WORK(\boldsymbol{Q}_{\mathrm{sig}}^{(1)}, \boldsymbol{Q}_{\mathrm{sig}}^{(2)}, \boldsymbol{F}) = \sum_{i=1}^{N_c}\sum_{j=1}^{N_c} f_{ij} d_{ij}\\
&\text{s.t.1} \ f_{ij} \ge 0 \ \ 1 \le i \le N_c, 1 \le j \le N_c\\
&\text{s.t.2} \ \sum_{j=1}^{N_c} f_{ij} \le \varpi_i^{(1)} \ \ 1 \le i \le N_c\\
&\text{s.t.3} \ \sum_{i=1}^{N_c} f_{ij} \le \varpi_j^{(2)} \ \ 1 \le j \le N_c\\
&\text{s.t.4} \ \sum_{i=1}^{N_c}\sum_{j=1}^{N_c} f_{ij} = \min\left(\sum_{i=1}^{N_c} \varpi_i^{(1)}, \sum_{j=1}^{N_c} \varpi_j^{(2)}\right),
\end{aligned} \tag{27}$$

where d_{ij} represents the user-defined ground distance, which is the distance between clusters $\boldsymbol{q}_i^{(1)}$ and $\boldsymbol{q}_j^{(2)}$, as follows:

$$d_{ij} = \sqrt{\tau_\rho(\Delta\rho_{\mathrm{idx}})^2 + \tau_\alpha(\Delta\alpha_{\mathrm{idx}})^2}, \tag{28}$$

$$\Delta\rho_{\mathrm{idx}} = \rho_j^{(2)} - \rho_i^{(1)}, \tag{29}$$

$$\Delta\alpha_{\mathrm{idx}} = \arctan\left(\frac{\tan\alpha_j^{(2)} - \tan\alpha_i^{(1)}}{1 + \tan\alpha_j^{(2)}\tan\alpha_i^{(1)}}\right). \tag{30}$$

Here, τ_ρ and τ_α are the weight parameters that adjust the importance of the distance and slope distributions. s.t.1 allows moving supplies from $\boldsymbol{Q}_{\mathrm{sig}}^{(1)}$ to $\boldsymbol{Q}_{\mathrm{sig}}^{(2)}$ and not vice versa. s.t.2 and s.t.3 limit the amount of supplies that can be sent by the clusters in $\boldsymbol{Q}_{\mathrm{sig}}^{(1)}$ to their weights, and the clusters in $\boldsymbol{Q}_{\mathrm{sig}}^{(2)}$ to receive no more supplies than their weights. s.t.4 forces moving the maximum amount of supplies possible. This amount is called the total flow. By solving the transportation problem, optimal flow matrix $\boldsymbol{F}$ can be obtained. Hence, EMD is defined as the work normalized by the total flow, as follows:

$$\boldsymbol{EMD}(\boldsymbol{Q}_{\mathrm{sig}}^{(1)}, \boldsymbol{Q}_{\mathrm{sig}}^{(2)}) = \frac{\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} f_{ij} d_{ij}}{\sum_{i=1}^{N_c}\sum_{j=1}^{N_c} f_{ij}}. \tag{31}$$

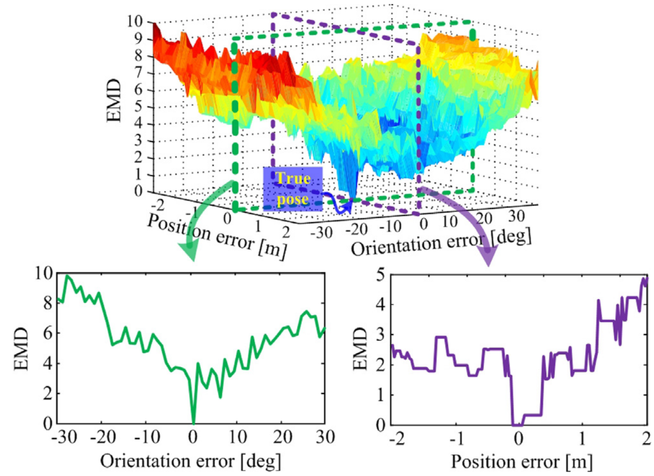

Fig. 16. EMD calculation results around correct camera pose.

Additional details on EMD can be found in [48]. In this study, an important advantage of EMD is that it allows partial matches in a very natural way. In other words, it can manage the occlusions and clutter that can occur frequently in line-based matching.

Fig. 16 shows an example of the EMD values around the correct camera pose. Here, the EMD calculation results that correspond to only one translation and orientation are plotted given that the results that correspond to the complete 6DOF state values cannot be illustrated. A robust descriptor should generate the lowest distance (i.e., highest similarity) near the true pose and gradually increases values according to the distance from the true pose. Fig. 16 illustrates this characteristic well, and thus the EMD calculation between the proposed QLH descriptors can be used as reasonable criteria for parameter calibration. Even if this function has some local minimum (i.e., not a convex function), as shown in Fig. 16, a global solution can be found using a particle filter-based estimation which is described in detail in the next section.

5. Particle filter-based parameter calibration

A particle filter is used as the main paradigm for the calibration task in this study. It is one of the popular methods for implementing a Bayesian filter that can estimate the probability distribution using a set of random particles. The state (i.e., 6DOF camera pose in this case) is represented by the weighted sum of all particles. The particle filter has the ability to solve not only a local estimation, but also a global estimation problem with high accuracy because it can represent non-parametric multi-modal probability distributions (i.e., not bound to a unimodal distribution). This section describes the Bayesian filter briefly and the particle filter-based parameter estimation method to obtain the most likely camera pose $\boldsymbol{w}$.

5.1. Bayesian filter

The Bayesian filter in this study determines the posterior probability of the 6DOF camera pose $\boldsymbol{w}$ based on measurement data (i.e., the QLH descriptor in this case) from the camera image and pre-given map information. The posterior probability distribution of the camera parameter $\boldsymbol{w}$, given the all conditions (i.e., measurement $\boldsymbol{Q}$ and map information $\mathbb{M}_{oct}$) is calculated as follows:

$$p(\boldsymbol{w}|\mathbb{M}_{oct}) = \int p(\boldsymbol{w}|\boldsymbol{w}_{k-1}, \mathbb{M}_{oct})p(\boldsymbol{w}_{k-1}|\boldsymbol{Q}, \mathbb{M}_{oct})d\boldsymbol{w}_{k-1}, \tag{32}$$

$$\begin{aligned}
p(\boldsymbol{w}|\boldsymbol{Q}, \mathbb{M}_{oct}) &= \eta \cdot p(\boldsymbol{Q}|\boldsymbol{w}, \mathbb{M}_{oct})p(\boldsymbol{w}|\mathbb{M}_{oct})\\
&= \eta \cdot p(\boldsymbol{Q}|\boldsymbol{w}, \mathbb{I})p(\boldsymbol{w}|\mathbb{M}_{oct}),
\end{aligned} \tag{33}$$

where $\boldsymbol{w} = [x_c\ y_c\ z_c\ \psi_c\ \theta_c\ \phi_c]^\top$ denotes the 6DOF camera pose at kth iteration. Here, the subscript k that represents the state of the current iteration is omitted. $\boldsymbol{Q}$ denotes the QLH descriptor from the real camera image (i.e., the measurement data), and $\mathbb{M}_{oct}$ is the 3D texture OctoMap information. $p(\boldsymbol{w}|\mathbb{M}_{oct})$, represented in Eq. (32), is the prior distribution on the camera parameters, and Markov assumption is applied on the right side. This means the prediction phase for predicting the distribution of state $\boldsymbol{w}$ by applying the prediction model $p(\boldsymbol{w}|\boldsymbol{w}_{k-1}, \mathbb{M}_{oct})$ to the posterior distribution at previous iteration $p(\boldsymbol{w}_{k-1}|\boldsymbol{Q}, \mathbb{M}_{oct})$. Eq. (33) represents the update phase to update the probability distribution of state $\boldsymbol{w}$ by applying the measurement model $p(\boldsymbol{Q}|\boldsymbol{w}, \mathbb{l})$ to prior distribution $p(\boldsymbol{w}|\mathbb{M}_{oct})$. Measurement model $p(\boldsymbol{Q}|\boldsymbol{w}, \mathbb{l})$ is defined by the likelihood function that should be designed for the QLH descriptor in this study. Here, η is the normalizing constant . Note that the prior information $\mathbb{M}_{oct}$ is converted into the 3D geometric line parameters $\mathbb{l}$ at the update phase as described in Section 3. In conclusion, the Bayesian filter recursively performs the prediction phase (Eq. (32)) and update phase (Eq. (33)) to estimation state vector $\boldsymbol{w}$.

5.2. Particle filter

In this study, the particle filter is an implementation of the Bayesian filter to calibrate complete external camera parameters. To represent the probability distribution on camera pose $\boldsymbol{w}$, the particle filter uses a set of random weighted particles represented by:

$$\mathbb{S} = \left[\langle \boldsymbol{w}_i, \varpi_i \rangle \quad | \quad 1 \le i \le N_p\right], \tag{34}$$

where $\boldsymbol{w}_i$ is the pose of the ith particle with associated importance weight ϖ_i at kth iteration and N_p is the number of particles in one set. At each iteration, the probabilities (weights) of the particles are updated using prediction and measurement models, and then the particles are resampled. A set of particles describes posterior distribution $p(\boldsymbol{w}|\boldsymbol{Q}, \mathbb{M}_{oct})$, which represents camera pose $\boldsymbol{w}$ conditioned on the measurement data (i.e., QLH descriptor $\boldsymbol{Q}$ generated from the real camera image) and map information $\mathbb{M}_{oct}$. The posterior distribution on camera pose $\boldsymbol{w}$ is approximated from a set of weighted particles, as follows:

$$p(\boldsymbol{w}|\boldsymbol{Q}, \mathbb{M}_{oct}) \approx \sum_{i=1}^{N_p} \varpi_i \delta(\boldsymbol{w} - \boldsymbol{w}_i), \tag{35}$$

where $\delta(\cdot)$ is the Dirac delta function. This subsection reviews the particle filter briefly. More details on the particle filter can be found in [49,50]. Among several variants of particle filters, the SIR particle filter algorithm is adopted in this study [22]. This algorithm is composed of the following steps: sampling, importance weighting, and resampling.

5.2.1. Sampling step

In the sampling step, a new particle set $\mathbb{S}^-$ is generated from past particle set $\mathbb{S}_{k-1}$ based on state transition probability $p(\boldsymbol{w}|\boldsymbol{w}_{k-1}, \mathbb{M}_{oct})$, as follows:

$$\boldsymbol{w}_i^- \sim p(\boldsymbol{w}|\boldsymbol{w}_{i,k-1}, \mathbb{M}_{oct}). \tag{36}$$

Because cameras are generally installed on the interior wall because of space limitations, the prediction model in the sampling step uses the map information that contains such information as constraints. Dense 3D map information can provide two types of constraints for the prediction model: a constraint on camera position (x_c, y_c, z_c) and a constraint on camera orientation $(\psi_c, \theta_c, \phi_c)$.

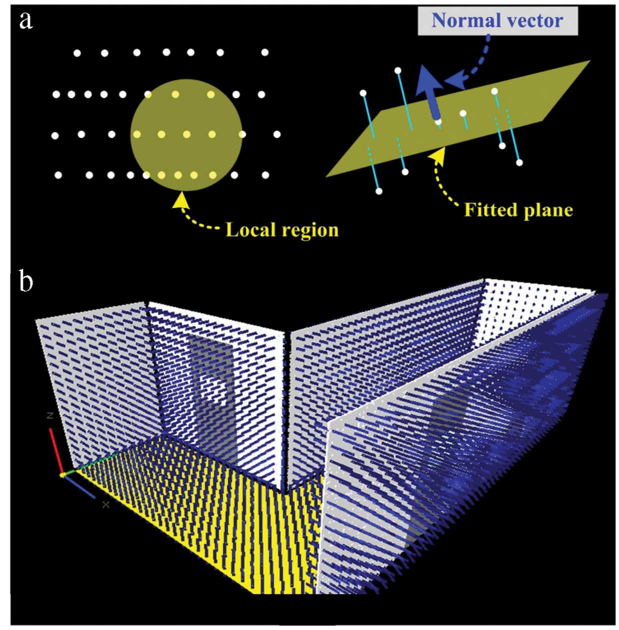

Fig. 17. Extraction of normal vectors to make camera orientation constraints: (a) calculation of normal vector by fitting plane and (b) extracted normal vectors $\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$ (blue lines) that correspond to all points in 3D OctoMap. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Thus, the sampling step that considers these two constraints can be implemented as follows:

$$\begin{aligned}
&\boldsymbol{w}_i^- = \boldsymbol{w}_{i,k-1} + \epsilon \\
&\text{s.t.1}\ \exists (x_i^-, y_i^-, z_i^-) \in \boldsymbol{n}_{\text{occ}} \\
&\text{s.t.2}\ \ \arctan 2(\hat{n}_y, \hat{n}_x) - \frac{\pi}{2} > \arctan 2(\cos\theta_i^- \sin\phi_i^-, \\
&\qquad \cos\theta_i^- \cos\phi_i^-) > \arctan 2(\hat{n}_y, \hat{n}_x) + \frac{\pi}{2} \\
&\text{s.t.3}\ \ \arccos(\hat{n}_z) - \frac{\pi}{2} \\
&\qquad > \arccos\left(-\sin\theta_i^-\right) > \arccos(\hat{n}_z) + \frac{\pi}{2} \\
&\text{s.t.4}\ \epsilon \sim \mathcal{N}(0, \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\psi^2, \sigma_\theta^2, \sigma_\phi^2)),
\end{aligned} \tag{37}$$

where noise variable ϵ follows a $\boldsymbol{O}$ mean Gaussian distribution with small variances (s.t.4). The superscript - indicates that the predicted state before the measurement data at the current iteration is not updated. s.t.1 represents the constraint on the position where the predicted particle's position (x_i^-, y_i^-, z_i^-) should be located at the occupied node $\boldsymbol{n}_{\text{occ}}$ of 3D OctoMap. s.t.2 and s.t.3 are the constraints on the orientation where predicted pitch θ_i^- and yaw angle ϕ_i^- cannot be more than 90 degrees from a normal vector of the corresponding wall plane. Here, $\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$ refers to the normal vector. Note that roll angle ψ_c does not have any constraints because it represents the rotation on the optical axis.

In order to calculate the normal vector $\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$ that corresponds to all the occupied nodes of the 3D OctoMap information, the plane-fitting algorithm presented in [51] is performed. The corresponding normal vectors are extracted using the all point cloud data that consists of the all center points of the all occupied nodes $\boldsymbol{n}_{\text{occ}}$ of the 3D OctoMap information. As shown in Fig. 17 (a), it is assumed that the points in a small local region are located on a local plane, which is expressed in the form of the following equation:

$$z = ax + by + c. \tag{38}$$

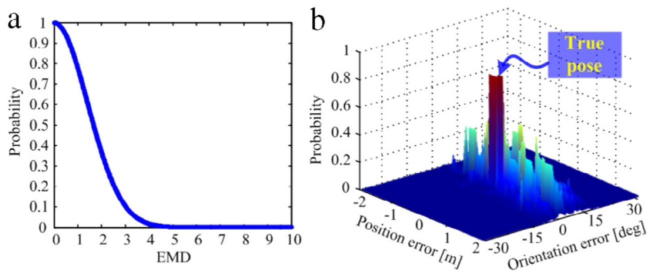

Fig. 18. Measurement model for QLH descriptor: (a) measurement model for transforming EMD into a probability (i.e., particle's weight) and (b) probability distribution around correct camera pose.

This optimization problem can be solved by the following calculations:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i,j} (x_i)^2 & \sum_{i,j} x_i y_j & \sum_{i,j} x_i \\ \sum_{i,j} x_i y_j & \sum_{i,j} (y_j)^2 & \sum_{i,j} y_j \\ \sum_{i,j} x_i & \sum_{i,j} y_j & N_{\mathrm{area}} \end{bmatrix} \begin{bmatrix} \sum_{i,j} x_i z_{ij} \\ \sum_{i,j} y_j z_{ij} \\ \sum_{i,j} z_{ij} \end{bmatrix}, \quad (39)$$

$$\begin{bmatrix} \hat{n}_x \\ \hat{n}_y \\ \hat{n}_z \end{bmatrix} = \frac{1}{\sqrt{a^2+b^2+1}} \begin{bmatrix} -a \\ -b \\ 1 \end{bmatrix}. \quad (40)$$

Here, (x_i, y_j, z_{ij}) are the values of each point in the local area. N_{area} represents the number of points in the local area. Through these calculations, the normal vector $\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$ that corresponds to each node $\boldsymbol{n}_{\mathrm{occ}}$ can be acquired; therefore, the constraints for the camera orientations represented in Eq. (37) can be applied for the prediction model. The extraction result of the normal vectors for the entire 3D OctoMap is shown in Fig. 17(b).

5.2.2. Importance weighting step

In a particle filter, the probability of a particle is updated using a measurement model in the importance-weighting step. In other words, importance factor ϖ is evaluated using measurement model $p(\mathbf{Q}|\boldsymbol{w}, \mathbb{M}_{oct})$, as follows:

$$\varpi_i = \eta \cdot p(\mathbf{Q}|\boldsymbol{w}_i^-, \mathbb{M}_{\mathrm{oct}}), \quad (41)$$

where η is a normalization constant. Eq. (41) is the same as the measurement model that represents the likelihood function. Thus, $p(\mathbf{Q}|\boldsymbol{w}_i^-, \mathbb{M}_{oct})$ is computed by the function for similarity comparison that assigns a low weight when the difference between the QLH descriptor predicted from the arbitrary particle pose and that extracted from the real camera image is large, or assigns a high weight when the difference is small. Here, the proposed QLH descriptor can be exploited for criteria of similarity comparison to determine particle weight, and thus a new measurement model is required.

Fig. 18(a) shows a measurement model for the QLH descriptor. The uncertainty can be modeled as a Gaussian distribution. Because the difference between two sets of QLH image descriptors are computed by EMD, the EMD distribution (Fig. 16) can be converted into a probability distribution around the correct pose (Fig. 18(b)) through the following measurement model:

$$p(\mathbf{Q}|\boldsymbol{w}_i^-, \mathbb{M}_{\mathrm{oct}}) = \frac{1}{\sigma_{\mathrm{EMD}}\sqrt{2\pi}} \exp\left(-\frac{\mathrm{EMD}(\mathbf{Q}, \boldsymbol{h}(\boldsymbol{w}_i^-))^2}{2\sigma_{\mathrm{EMD}}^2}\right), \quad (42)$$

where $\mathbf{Q}$ and $\boldsymbol{h}(\boldsymbol{w}_i^-)$ refer to the QLH descriptor extracted from a real camera image and that predicted from the pose of ith particle $\boldsymbol{w}_i^-$, respectively. σ_{EMD} represents the standard deviation associated with the uncertainty of the QLH descriptor (e.g. $\sigma_{\mathrm{EMD}} = 3.0$ in the case of Fig. 18). $\boldsymbol{EMD}(\cdot)$ denotes the earth mover's distance between the two sets of QLH descriptors. EMD is the appropriate criterion for the similarity comparison of this problem based on the discussion in Section 4.3. The result of Eq. (42) is transformed into the weighting of the particle, and the calculated weights are used for the resampling in the next step.

5.2.3. Resampling step

In the resampling step, a new particle set $\mathbb{S}$ is randomly chosen from $\mathbb{S}^-$ according to the distribution defined by the importance factor ϖ_i.

$$\mathbb{S} = \left[\langle \boldsymbol{w}_j, 1/N_p \rangle \quad | \quad 1 \le j \le N_p\right] \sim \left[\boldsymbol{w}_i^-, \varpi_i\right]. \quad (43)$$

Particles with high weight generate many particles. Otherwise, particles with low weight generate few particles or none. The prior probability of each particle of the new particle set $\mathbb{S}$ at the current iteration k is initialized to $1/N_p$.

Through the three recursive steps, the particles converge on the pose with highest probability. The estimated camera state $\boldsymbol{w}^*$ is calculated by the weighted average, as follows:

$$\boldsymbol{w}^* = \frac{1}{N_p} \sum_{i=1}^{N_p} \varpi_i \boldsymbol{w}_i. \quad (44)$$

5.3. Time complexity of algorithm

The time complexity of algorithms is most widely expressed using the big-$\mathcal{O}$ notation. The time complexity of the proposed calibration process can be divided into two computations: computational burden to generate the QLH descriptor from the 2D photometric line segments extracted by the Hough transform, which corresponds to a real camera image, and iterative calculations required for the particle filter procedure. Here, the former involves negligible amount of calculation because this process is executed only once. The SIR particle filter procedure used in this study is divided into three steps: sampling, importance weighting, and resampling. Each complexity can be written as follows:

$$\mathcal{O}_{\mathrm{sample}} = \mathcal{O}(N_p), \quad (45)$$

$$\begin{aligned} \mathcal{O}_{\mathrm{weight}} &= N_p(\mathcal{O}_{\mathrm{proj}} + \mathcal{O}_{\mathrm{QLH}} + \mathcal{O}_{\mathrm{EMD}}) \\ &= N_p(\mathcal{O}(N_{\mathrm{line}}) + \mathcal{O}(N_j^2) + \mathcal{O}(N_c^2)) \\ &= O(N_p(N_{\mathrm{line}} + N_j^2 + N_c^2)), \end{aligned} \quad (46)$$

$$\mathcal{O}_{\mathrm{resample}} = \mathcal{O}(N_p), \quad (47)$$

where N_{line}, N_j, and N_c denote the number of entire 3D photometric lines, projected 2D photometric lines, and bins constituting the QLH descriptor, respectively. N_p is the number of particles and it spends a large part of the computation time. In the importance weighting step, the computation time for one particle is composed of three parts: the complexity $\mathcal{O}_{\mathrm{proj}} = \mathcal{O}(N_{\mathrm{line}})$ required for projecting 3D geometric line segments $\mathbb{I}$ into a 2D image plane, complexity $\mathcal{O}_{\mathrm{QLH}} = \mathcal{O}(N_j^2)$ required for generating the QLH descriptor $\mathbf{Q}$ from projected 2D photometric lines $\mathbb{I}_{2D}$ (Table 3), and complexity $\mathcal{O}_{\mathrm{EMD}} = \mathcal{O}(N_c^2)$ required for calculation of the EMD between the QLH descriptors (Eq. (31)). In conclusion, the time complexity $\mathcal{O}_{\mathrm{iter}}$ required for one iteration in the proposed calibration framework can be represented as follows:

$$\begin{aligned} \mathcal{O}_{\mathrm{iter}} &= \mathcal{O}_{\mathrm{sample}} + \mathcal{O}_{\mathrm{weight}} + \mathcal{O}_{\mathrm{resample}} \\ &= \mathcal{O}(N_p + N_p(N_{\mathrm{line}} + N_j^2 + N_c^2) + N_p) \\ &= \mathcal{O}(N_p(2 + N_{\mathrm{line}} + N_j^2 + N_c^2)) \\ &\approx \mathcal{O}(N_p(N_j^2 + N_c^2)). \end{aligned} \quad (48)$$

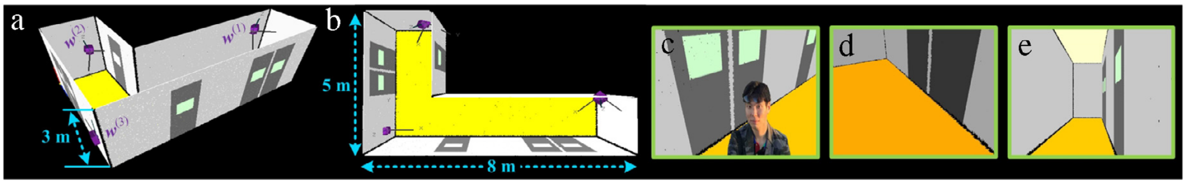

Fig. 19. Simulation environment with virtual camera network of up to three cameras: (a) 3D texture OctoMap information of simulation environment, (b) top view, (c) simulated image data that includes moving object from camera $\boldsymbol{w}^{(1)}$, (d) simulated image data by changing illumination condition from camera $\boldsymbol{w}^{(2)}$, (e) simulated image data from camera $\boldsymbol{w}^{(3)}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

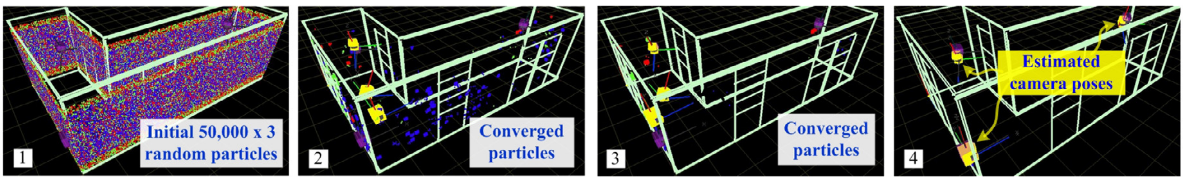

Fig. 20. Simulation results for several stages of particle filter. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Algorithm to generate QLH descriptor Q from 2D photometric line segments $\mathbb{l}_{2D}$.

Algorithm Generation of QLH descriptor
input : 2D photometric line segments $\mathbb{l}_{2D} = \left[\boldsymbol{l}_{2D}^{(j)} \quad | \quad 1 \le j \le N_j\right]$
output : QLH descriptor $\boldsymbol{Q}$
for every 2D photometric line segments $\boldsymbol{l}_{2D}^{(j)} = (\rho^{(j)}, \alpha^{(j)})$
  get quantized distance index $\rho_{\text{idx}} = round(\rho^{(j)}/b_\rho)$
  get quantized distance index $\alpha_{\text{idx}} = round(\alpha^{(j)}/b_\alpha)$
  $\boldsymbol{l}_{\text{idx}} = (\rho_{\text{idx}}, \alpha_{\text{idx}})$
  $\boldsymbol{Q}(\rho_{\text{idx}}, \alpha_{\text{idx}}) = \frac{1}{N_j h^D} \sum_{j=1}^{N_j} \boldsymbol{K}\left(\frac{\boldsymbol{l}_{\text{idx}} - \boldsymbol{l}_{2D}^{(j)}}{h}\right)$
end for
normalize $\boldsymbol{Q}$
return $\boldsymbol{Q}$

Here, the computations that correspond to a constant and N_{line} (at most 100) are negligible compared to quadratic terms.

6. Experiment results

6.1. Simulation

A simulation was conducted with a virtual camera network of up to three cameras. Fig. 19(a) and (b) show the 3D texture OctoMap information of the simulation environment. The OctoMap information in this simulation has 623,613 nodes of which the minimum voxel size (i.e., the cube of edge length) is 20 mm. The size of the simulation environment is 5 m × 8 m × 3 m, including various line features located on the sides of the walls, doors, and windows. Here, the real poses of the virtual cameras that should be estimated in this study are represented in purple with black axes, and each corresponding simulated image is as illustrated in Fig. 19(c), (d), and (e). Note that we treated these simulated images as real image data from camera sensors in the simulation experiments. Changes in the environment caused by a moving object and illumination condition are considered as shown in Fig. 19(c) and (d) in order to validate robustness. In this simulation, internal parameters $f_v, f_u, f_{\text{skew}}$, and (c_u, c_v) for all virtual cameras are set to 930, 930, 0, and (640, 400), respectively. The image dimensions are 1280×800.

Global 6DOF pose estimations (i.e., complete external parameter calibrations) of the camera sensor network with no initial conditions are performed in the simulation environment. As shown in Fig. 20, the particles are initialized globally based not on the initial conditions, but on prior information (i.e., the constraints from the map information as described in Eq. (37)). We can assume that pose estimations are finished when all particles converge with very small variances. In this simulation, a maximum of 50,000 particles are used for each camera pose and the number of particles is adjusted according to the variance of the particles' distribution. The QLH descriptor-based similarities between the 2D photometric line segments from the camera sensor network and those from all particles are computed. The results are then used in the probability update of each particle.

The several stages of the particle filter iterations and convergence process for all camera poses are illustrated in Figs. 20 and 21, respectively. After approximately 15 iterations, most camera poses (colored axes) accurately converge on the real poses (black axes). Note that the roll angle (i.e., the rotation on the optical axis) has slight effect on the scope of the camera observation, and thus it is fixed at 0 degrees and an estimation is not performed. The simulation results show that the complete 6DOF external parameters are estimated accurately even if environmental conditions change because of illumination or moving objects. In addition, the matching process of the particle filter algorithm is very fast in that it utilizes not the heavy 3D volumetric information, but only 3D geometric line parameters. Note that the computation time required for one iteration was about 6 s in the case of $N_p = 50{,}000$, average $N_j = 12$, and $N_c = 120 \times 72$ (in a 4.0 GHz quad core CPU).

6.2. Real data

Experiments were conducted in a real environment by implementing a camera network system using two wireless IP cameras (AXIS M1004-W). Camera calibration was performed before the experiment in order to define an internal camera matrix, and distortion parameters were used for the coordinate transformation and to obtain the undistorted image. Calibrated internal parameters $f_u, f_v, f_{\text{skew}}$, and (c_u, c_v) for the wireless IP cameras were 930.261, 926.302, 0, and (663.912, 456.287), respectively. Fig. 22 illustrates the real environment for the experiments and the real image data captured from the installed wireless IP camera network. The image dimensions are $1,280 \times 800$. Even if line information is relatively not much affected by illumination changes compared to color information, 2D photometric lines in real image data might not be extracted robustly when the line features are obscured by high-intensity light. Namely, the image processing illustrated in Fig. 7 cannot be performed correctly under the image conditions. In this case, manual adjustment of the thresholds in the Canny edge detector or Hough transform might be required; furthermore, it is possible for operator instructions

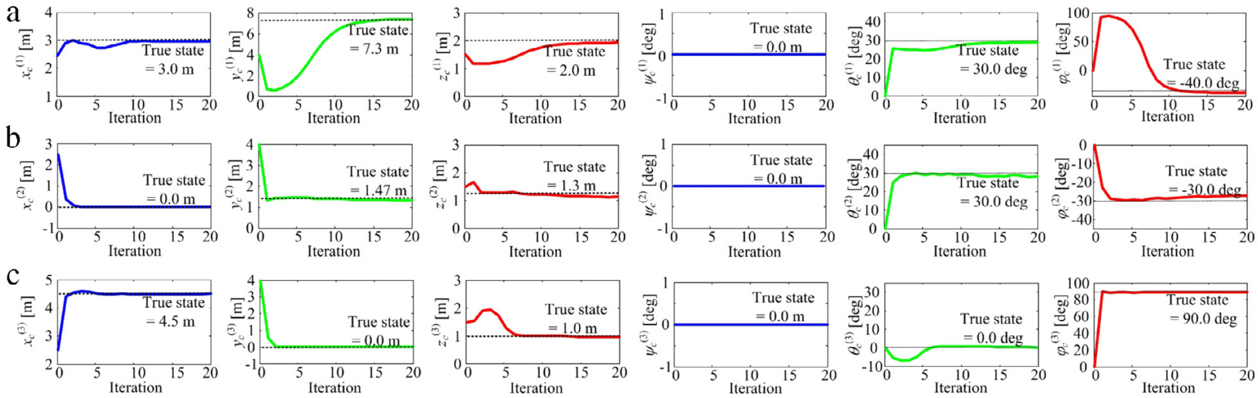

Fig. 21. Convergence process for camera state variables $\boldsymbol{w} = [x_c\ y_c\ z_c\ \psi_c\ \theta_c\ \phi_c]^\top$ for camera sensor network system in simulation experiment: (a) for camera $\boldsymbol{w}^{(1)}$, (b) for camera $\boldsymbol{w}^{(2)}$, and (c) for camera $\boldsymbol{w}^{(3)}$.

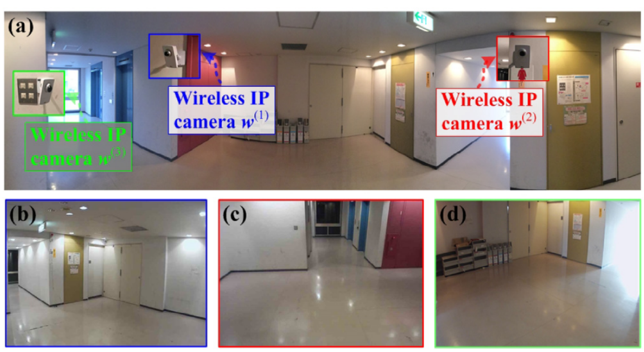

Fig. 22. Experimental environment with wireless IP camera network: (a) real environment, (b) real image data from wireless IP camera $\boldsymbol{w}^{(1)}$, (c) real image data from wireless IP camera $\boldsymbol{w}^{(2)}$, and (d) real image data from wireless IP camera $\boldsymbol{w}^{(3)}$.

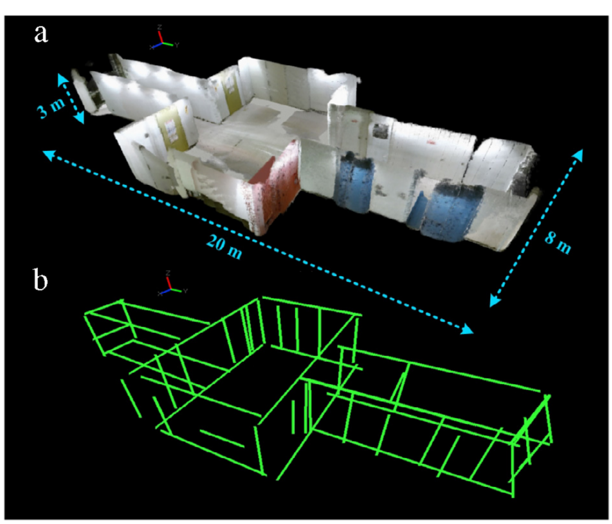

Fig. 23. Environment model: (a) 3D texture OctoMap information $\mathbb{M}_{oct}$, (b) learned 3D geometric line parameters $\mathbb{L}$.

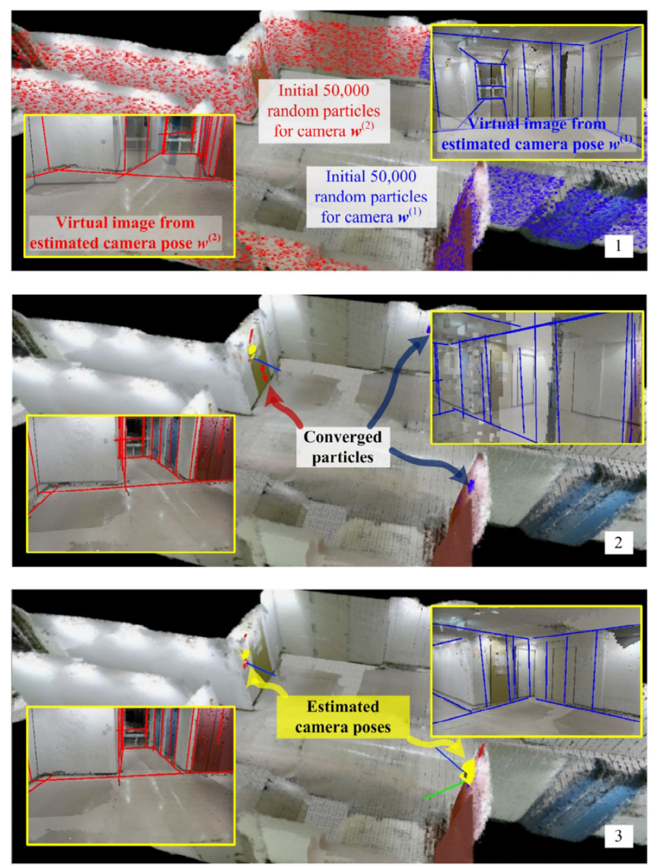

Fig. 24. Experimental results for several stages of particle filter in real environment for camera $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ (global estimation). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to be required in some cases. Fig. 23 shows the corresponding 3D texture OctoMap information and the learned 3D geometric line segments. The size of the entire map information is approximately 20 m × 8 m × 3 m. The OctoMap information in real experiments has 7,446,590 nodes of which the minimum voxel size (i.e., the cube of edge length) is 20 mm. The cameras were mounted on the wall in the experimental environment, and these roll angles were installed at 0 degrees for the reason mentioned in Section 6.1.

6.2.1. Global estimation

To begin with, the particles are widely spread with no initial condition in the same manner as the simulation in order to verify the performance of global estimation for the camera parameters. The stages for the calibration using real data ($\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$) are shown in Fig. 24. As shown in the results, the proposed calibration method can estimate the complete 6DOF external parameters that are close to the correct poses on the ground where virtual images with back-projected 2D photometric lines (bold red and blue lines)

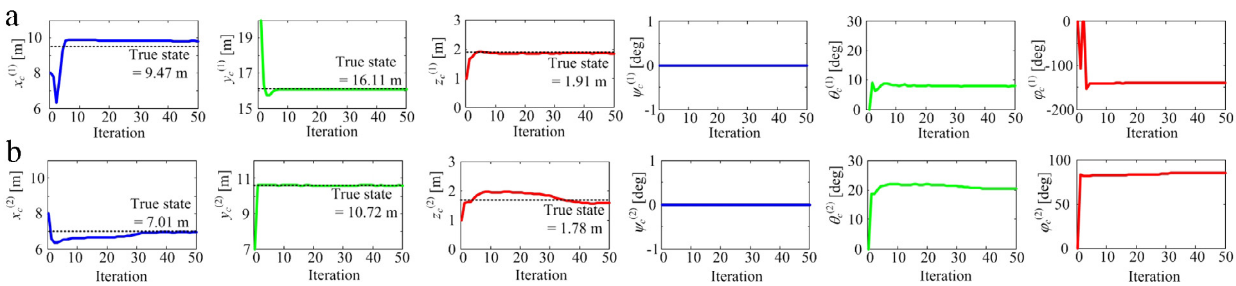

Fig. 25. Convergence process for camera state variables $\boldsymbol{w} = [x_c\ y_c\ z_c\ \psi_c\ \theta_c\ \phi_c]^\top$ for wireless camera sensor network system in real experiment: (a) for camera $\boldsymbol{w}^{(1)}$ and (b) for camera $\boldsymbol{w}^{(2)}$.

generated from estimated camera poses are almost the same as the contour of the real images. Here, the images illustrated in Fig. 24 are alpha blended images that include both the real images and the generated virtual images. Thus, we can initially easily recognize large differences between the real images and generated virtual images, but after convergence, these images are almost identically matched with small errors. The cause for these errors can be considered as a result of the error of the 3D OctoMap information itself, which is the basis for the position information.

The convergence processes for the camera poses ($\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$) in the real experiment are depicted in Fig. 25. After approximately 40 iterations, most camera poses converge similarly to the simulation experiments. Fig. 26 shows the convergence of particles to the correct camera position as a function of the resampling step. The convergence rate is defined as the number of particles closer than 0.3 m to the correct position divided by the number of total particles. In the case of camera $\boldsymbol{w}^{(2)}$, more iterations were required for the particles to converge on the correct pose in places where similar scenes were observed to exist.

On the other hand, an example of global estimation failure for camera $\boldsymbol{w}^{(3)}$ is shown in Figs. 27(a) and 28(a). As shown in Fig. 22(d), the image captured by camera $\boldsymbol{w}^{(3)}$ has a jumble of waste baskets at the left side, whereas the images from cameras $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ are structurally clean and stable, as shown in Fig. 22(b) and (c). Therefore, in case of the image data captured from camera $\boldsymbol{w}^{(3)}$, the robust and intuitive 2D photometric line segments cannot be extracted relatively. As a result, the particles for camera $\boldsymbol{w}^{(3)}$ converged into an incorrect pose and wrong camera parameters were estimated. Furthermore, as illustrated in the alpha blended image in the final stage of Fig. 27(a), back-projected 2D photometric lines (bold green lines) are almost same as the contours of the real captured image from camera $\boldsymbol{w}^{(3)}$, despite the different place. This result proves clearly that the proposed calibration framework still has limitation for global estimation in the case of environments where a lot of similar line structures exist.

6.2.2. Local estimation

The discussions mentioned above are the limitations in the case of the global estimate only. The proposed calibration framework makes it possible to estimate the accurate 6DOF camera parameters when using the roughly measured state by human eyes as initial information (i.e., the local estimation problem) in spite of structurally unclean environments. An experiment for camera $\boldsymbol{w}^{(3)}$ was conducted to verify the local estimation performance. As shown in the first stage of Fig. 27(b), particles are initialized within around 2 m based on human observation. The several stages of particle filter iterations and the convergence process for the camera parameters are illustrated in Figs. 27(b) and 28(b), respectively. The results show that the complete 6DOF external parameters are estimated very accurately in case of the local estimation problem even if the image captured from camera $\boldsymbol{w}^{(3)}$ is not structurally stable. Note that the complete 6DOF parameter calibration including the roll angle ψ_c, which were excluded in the global estimation, could be carried out successfully.

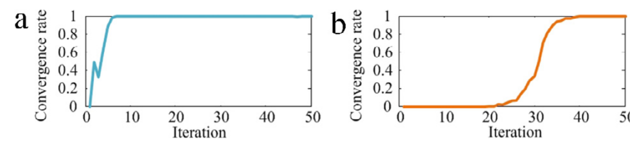

Fig. 26. Particle convergence on correct camera position versus iteration of particle filter: (a) for camera $\boldsymbol{w}^{(1)}$ and (b) for camera $\boldsymbol{w}^{(2)}$.

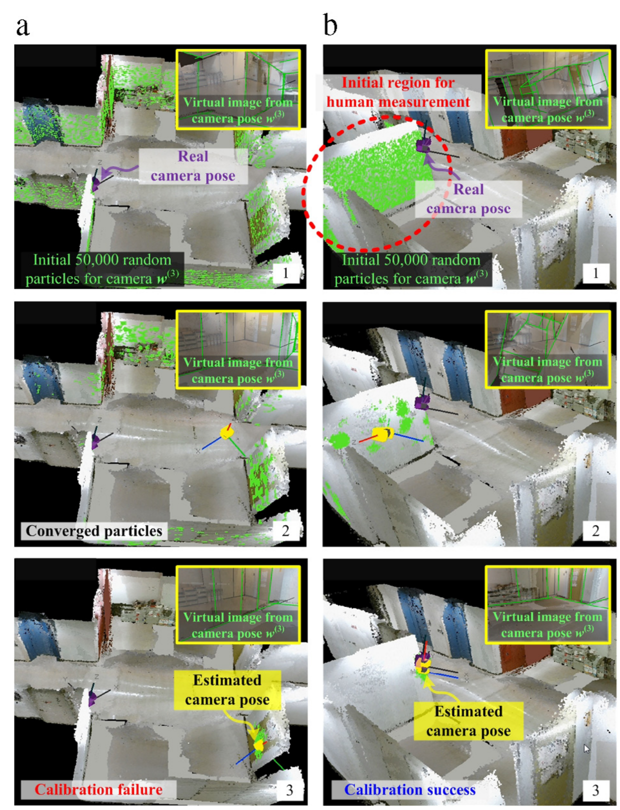

Fig. 27. Experimental results for several stages of particle filter in real environment for camera $\boldsymbol{w}^{(3)}$: (a) global estimation and (b) local estimation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusion

It has been impossible to achieve a global estimation of complete 6DOF camera parameters with no strong constraints because of myriad local minimum solutions. To overcome this difficulty, a novel approach for an automatic and complete parameter calibration system that uses 3D texture map information for camera sensor networks was proposed in this study. The particle filter-based approach that is an implementation of the Bayesian filter

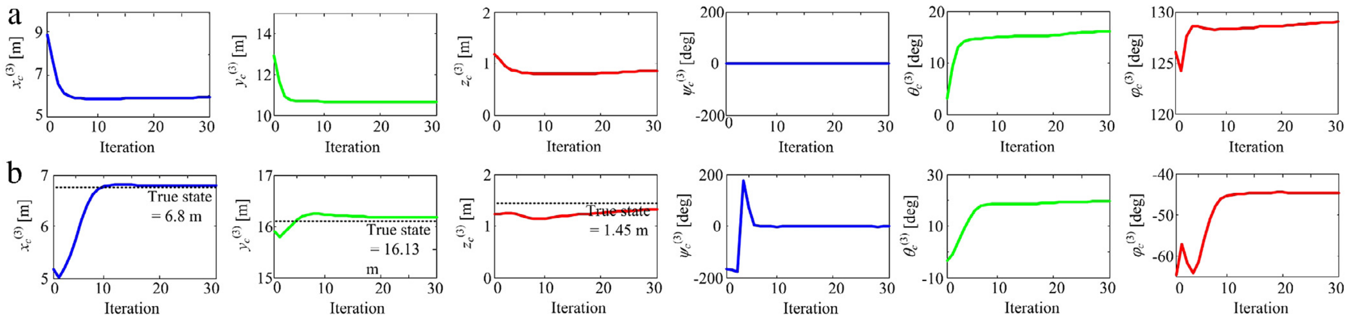

Fig. 28. Convergence process for camera state variables $\boldsymbol{w}^{(3)} = [x_c^{(3)}\ y_c^{(3)}\ z_c^{(3)}\ \psi_c^{(3)}\ \theta_c^{(3)}\ \phi_c^{(3)}]^\top$ for wireless camera sensor network system in real experiment: (a) global estimation and (b) local estimation.

was used to estimate the complete camera poses. The validity of the proposed automatic calibration method was investigated through both simulation and real experiments, and the following conclusions were drawn:

- The proposed learning method can automatically generate 3D geometric line segments from 3D texture map information by applying point cloud processing and RANSAC algorithm. The generated 3D geometric line segments can be used to represent the entire environment in a small number of parameters for a fast matching process, whereas the heavy 3D volumetric map information cannot manage this problem.
- The QLH descriptor with EMD-based novel similarity comparison criteria can serve as an efficient signature for vision-based automatic calibration systems. Using the proposed criteria, the complete external parameters of the camera network system can be estimated automatically.
- By applying the particle filter-based approach, global estimation without initial conditions can be performed for a large indoor environment, whereas the 2D–3D matching schemes introduced in Section 1 can only be applied to the local estimation problem which needs initial registration close to a correct pose. In addition, the map information that contains occupied areas can be incorporated into the prior distribution over the camera states and be considered as strong constraints.
- The proposed QLH descriptor-based matching scheme still has limitations for global estimation when the captured camera image is not structurally stable and where the environments have a lot of similar line structures since these environments are virtually indistinguishable using the QLH descriptor alone. Local estimation was, however, performed successfully with small errors in most cases. This shows that the proposed calibration framework produces reliable performance in case of the local estimation, even when compared to the previous 2D–3D matching schemes.

Finally, the future works related to this paper are as follows:

- We will take more robust descriptors (e.g., additional consideration of lengths of 2D line segments) into consideration owing to following reasons. First, we have to overcome the limitations in global estimation because the current approach struggles to distinguish the environments with similar line configurations. Second, the precision of the 3D OctoMap information should also be considered given that the accuracy of the calibration results depends on it.
- As of now, heuristic calibration of parameters is sometimes applied in the extraction of both 3D geometric (e.g., choosing the number of neighbors in the clustering) and 2D photometric lines (e.g., Canny edge detection and Hough transform). In the case of the 3D geometric lines, it is necessary to simplify the parameterization task by exploiting a more compact data structure (e.g., a set of 2D planes) for the 3D environment model, instead of an Octree structure. The task for 2D line extraction should also be improved to address the environmental conditions.
- As the number of particles increases, the result is a linear increase in the computational load as shown in Eq. (48). In other words, a factor that has the greatest effect on the computational load is the size of the environment that influences the required number of particles. Thus, an improved optimization method that can manage a large environment should be considered.

References

[1] J.-H. Lee, H. Hashimoto, Intelligent space—concept and contents, Adv. Robot. 16 (3) (2002) 265–280.

[2] T. Sato, Y. Nishida, H. Mizoguchi, Robotic room: Symbiosis with human through behavior media, Robot. Auton. Syst. 18 (1) (1996) 185–194.

[3] E.M. Foxlin, Generalized architecture for simultaneous localization, auto-calibration, and map-building, in: Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS2002, 2002, pp. 527–533.

[4] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, A. Grue, Simultaneous localization, calibration, and tracking in an ad-hoc sensor network, in: Proceedings of the 5th IEEE/ACM International Conference on Information Processing in Sensor Networks, IPSN2002, 2006, pp. 27–33.

[5] H. Chen, K. Matsumoto, J. Ota, T. Arai, Self-calibration of environmental camera for mobile robot navigation, Robot. Auton. Syst. 55 (3) (2007) 177–190.

[6] S. Funiak, C. Guestrin, M. Paskin, R. Sukthankar, Distributed localization of networked cameras, in: Proceedings of the 5th IEEE/ACM International Conference on Information Processing in Sensor Networks, IPSN2002, 2006, pp. 34–42.

[7] A. Rahimi, B. Dunagan, T. Darrell, Simultaneous calibration and tracking with a network of non-overlapping sensors, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR2004, vol. 1, 2004, pp. I–187.

[8] Y. Ji, A. Yamashita, H. Asama, Automatic calibration and trajectory reconstruction of mobile robot in camera sensor network, in: Proceedings of the 2015 IEEE International Conference on Automation Science and Engineering, CASE2015, IEEE, 2015, pp. 206–211.

[9] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Octomap: An efficient probabilistic 3d mapping framework based on octrees, Auton. Robots 34 (3) (2013) 189–206.

[10] OctoMap, 2016. [Online]. Available: http://octomap.github.io/. (Accessed12 February 2016).

[11] A.C. Murillo, J.J. Guerrero, C. Sagüés, Surf features for efficient robot localization with omnidirectional images, in: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, ICRA2007, 2007, pp. 3901–3907.

[12] S. Ramalingam, S. Bouaziz, P. Sturm, M. Brand, Geolocalization using skylines from omni-images, in: Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009, pp. 23–30.

[13] G.L. Mariottini, D. Prattichizzo, Uncalibrated video compass for mobile robots from paracatadioptric line images, in: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS2007, 2007, pp. 226–231.

[14] H. Iwasa, N. Aihara, N. Yokoya, H. Takemura, Memory-based self-localization using omnidirectional images, Syst. Comput. Japan 34 (5) (2003) 56–68.

[15] D. Ishizuka, A. Yamashita, R. Kawanishi, T. Kaneko, H. Asama, Self-localization of mobile robot equipped with omnidirectional camera using image matching and 3d–2d edge matching, in: Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops, 2011, pp. 272–279.
[16] M.D. Elstrom, P.W. Smith, Stereo-based registration of multi-sensor imagery for enhanced visualization of remote environments, in: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, ICRA1999, vol. 3, 1999, pp. 1948–1953.
[17] I. Stamos, P. Alien, Automatic registration of 2-d with 3-d imagery in urban environments, in: Proceedings of the 8th IEEE International Conference on Computer Vision, ICCV2001, vol. 2, 2001, pp. 731–736.
[18] R. Kurazume, K. Nishino, Z. Zhang, K. Ikeuchi, Simultaneous 2d images and 3d geometric model registration for texture mapping utilizing reflectance attribute, in: Proceedings of the 5th Asian Conference on Computer Vision, ACCV2002, 2002, pp. 99–106.
[19] L. Liu, I. Stamos, Automatic 3d to 2d registration for the photorealistic rendering of urban scenes, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR2005, vol. 2, 2005, pp. 137–143.
[20] Y. Iwashita, R. Kurazume, K. Konishi, M. Nakamoto, M. Hashizume, T. Hasegawa, Fast alignment of 3d geometrical models and 2d grayscale images using 2d distance maps, Syst. Comput. Japan 38 (14) (2007) 52–62.
[21] K. Hara, Y. Kabashima, Y. Iwashita, R. Kurazume, T. Hasegawa, Robust 2d–3d alignment based on geometrical consistency, in: Proceedings of the 6th International Conference on 3-D Digital Imaging and Modeling, 3DIM2007, 2007, pp. 273–280.
[22] A. Smith, A. Doucet, N. de Freitas, N. Gordon, Sequential Monte Carlo Methods in Practice, Springer Science & Business Media, 2013.
[23] S. Yeon, C. Jun, H. Choi, J. Kang, Y. Yun, N. Lett Doh, Robust-PCA-based hierarchical plane extraction for application to geometric 3D indoor mapping, Ind. Robot, Int. J. 41 (2) (2014) 203–212.
[24] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic hough transform, Pattern Recognition 24 (4) (1991) 303–316.
[25] J. Demantke, C. Mallet, N. David, B. Vallet, Dimensionality based scale selection in 3D lidar point clouds, Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 38 (Part 5) (2011) W12.
[26] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer 22 (6) (1989) 46–57.
[27] R.B. Rusu, Semantic 3d object maps for everyday manipulation in human living environments (Ph.D. thesis), Technische Universität München, 2009.
[28] M.A. Fischler, R.C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.
[29] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 2008.
[30] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[31] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[32] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, in: Proceedings of the 9th IEEE International Conference on Computer Vision, ICCV2003, 2003, pp. 1470–1477.
[33] F.-F. Li, R. Fergus, A. Torralba, Recognizing and learning object categories, Tutorial at ICCV, 2007.
[34] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, in: Workshop on Statistical Learning in Computer Vision, ECCV, vol. 1–22, 2004, pp. 1–2.
[35] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR2006, vol. 2, 2006, pp. 2169–2178.
[36] D. Filliat, A visual bag of words method for interactive qualitative localization and mapping, in: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, ICRA2007, 2007, pp. 3921–3926.
[37] J. Wang, R. Cipolla, H. Zha, Vision-based global localization using a visual vocabulary, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA2005, 2005, pp. 4230–4235.
[38] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, H.S. Sawhney, Ten-fold improvement in visual odometry using landmark matching, in: Proceedings of the 11th IEEE International Conference on Computer Vision, ICCV2007, 2007, pp. 1–8.
[39] M. Cummins, P. Newman, Fab-map: Probabilistic localization and mapping in the space of appearance, Int. J. Robot. Res. 27 (6) (2008) 647–665.
[40] M. Cummins, P. Newman, Appearance-only slam at large scale with fab-map 2.0, Int. J. Robot. Res. 30 (9) (2011) 1100–1123.
[41] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, G. Wyeth, Openfabmap: An open source toolbox for appearance-based loop closure detection, in: Proceedings of the 2012 IEEE International Conference on Robotics and Automation, ICRA2012, vol. 3, 2012, pp. 4730–4735.
[42] M. Tomono, 3d localization based on visual odometry and landmark recognition using image edge points, in: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS2010, 2010, pp. 5953–5959.
[43] T. Kurita, N. Otsu, T. Sato, A face recognition method using higher order local autocorrelation and multivariate analysis, in: Proceedings of the 11th International Conference on Pattern Recognition, ICPR1992, 1992, pp. 213–216.
[44] H. Nakayama, T. Harada, Y. Kuniyoshi, Dense sampling low-level statistics of local features, IEICE Trans. Inf. Syst. 93 (7) (2010) 1727–1736.
[45] H. Nakayama, T. Harada, Y. Kuniyoshi, Global Gaussian approach for scene categorization using information geometry, in: Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR2010, 2010, pp. 2336–2343.
[46] M. Tomono, A scan matching method using Euclidean invariant signature for global localization and map building, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA2004, vol. 1, 2004, pp. 866–871.
[47] R.O. Duda, P.E. Hart, Use of the hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[48] Y. Rubner, C. Tomasi, L.J. Guibas, A metric for distributions with applications to image databases, in: Proceedings of the 6th IEEE International Conference on Computer Vision, ICCV1998, 1998, pp. 59–66.
[49] B. Ristic, S. Arulampalam, N.J. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, 2004.
[50] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174–188.
[51] J.W. Weingarten, G. Gruener, R. Siegwart, Probabilistic plane fitting in 3d and an application to robotic mapping, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA2004, vol. 1, 2004, pp. 927–932.

Yonghoon Ji received his B.S. degrees from the Department of Mechanical Engineering and the Department of Computer Engineering, Kyunghee University, Korea, in 2010. He received his M.S. degree from the Department of Mechatronics, Kore University, Korea, in 2012. His Ph.D. degree was from the Department of Precision Engineering, the University of Tokyo, Japan, in 2016. He is an Overseas researcher under Postdoctoral Fellowship of Japan Society for the Promotion of Science (JSPS). His research interests cover mobile robotics, unmanned vehicle technologies, and intelligent space. He is a member of IEEE, RSJ, JSME and SICE.

Atsushi Yamashita received his B.E., M.E., and Ph.D. degrees from the Department of Precision Engineering, the University of Tokyo, Japan, in 1996, 1998, and 2001, respectively. From 1998 to 2001, he was a Junior Research Associate in the RIKEN (Institute of Physical and Chemical Research). From 2001 to 2008, he was an Assistant Professor of Shizuoka University. From 2006 to 2007, he was a Visiting Associate of California Institute of Technology. From 2008 to 2011, he was an Associate Professor of Shizuoka University. From 2011, he is an Associate Professor in the Department of Precision Engineering, the University of Tokyo. His research interests include robot vision, image processing, multiple mobile robot system, and motion planning. He is a member of ACM, IEEE, JSPE, RSJ, IEICE, JSME, IEEJ, IPSJ, ITE and SICE.

Hajime Asama received his B.S., M.S., and Dr. Eng degrees from the University of Tokyo, Japan, in 1982, 1984, and 1989, respectively. He was Research Associate, Research Scientist, and Senior Research Scientist in RIKEN (The Institute of Physical and Chemical Research, Japan) from 1986 to 2002. He became a Professor of RACE (Research into Artifacts, Center for Engineering), the University of Tokyo in 2002, and a Professor of School of Engineering, the University of Tokyo since 2009. Currently, he is the chairman of the Task Force for Remote Control Technology of the Council for the Decommissioning of TEPCO's Fukushima Daiichi NPS, the leader of Project on Disaster Response Robots and their Operation System of Council on Competitiveness-Japan, and the chairman of Robotics Task Force for Anti-Disaster (ROBOTAD). His main research interests are distributed autonomous robotic systems, smart spaces, service engineering, and Mobiligence, and service robotics.