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.

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.

© 2016 Elsevier B.V. All rights reserved.

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 Funia 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 degrees of freedom (DOF) poses \((x, y, \phi)\) 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 \((\psi, \theta, \phi)\)
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 \((x, y, \phi)\) 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 posteriori (MAP) estimation methods into consideration to find the optimal camera pose \( \omega^* \), as follows:

\[
\begin{align*}
\omega^* &= \arg\max_{\omega} \{ p(\omega | R, \Omega) \}, \\
&= \arg\min_{\omega} \{ p(I_0 | \omega, \Omega) p(\omega | \Omega) \}, \\
&= \arg\min_{\omega} \left\{ -\log p(I_0 | \omega, \Omega) p(\omega | \Omega) \right\}, \\
&= \arg\min_{\omega} \left\{ \left[ I_{V(w)} - I_R \right]^T \Omega_0 \left[ I_{V(w)} - I_R \right] \right\}, \\
&= \arg\min_{\omega} \left\{ \sum_{(u,v) \in I} \| I_{V(w)}(u, v) - I_0(u, v) \|^2 \right\}, \tag{1}
\end{align*}
\]

where \( \omega = [x, y, z, \phi, \theta, \psi]^T \) denotes the 6DOF camera pose and \( I_{V(w)} \) represents the virtual image generated from the arbitrary camera pose \( \omega \). \( I_0 \) is the real image from the camera sensor network. \( \Omega_0 \) denotes pre-generated 3D texture OctoMap information. \( \Omega_0 \) is an information matrix with regard to the noise of the image data.

In ML estimation method, prior probability distribution \( p(\omega | \Omega) \) is neglected, and thus this problem can be very simplest and intuitive method that determines the optimal camera pose \( \omega^* \) by calculating the differences of all pixel intensities between the generated virtual images \( I_0 \) and the real camera image \( I_0 \). 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 \( I_{V(w)} \)) 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 \( p(\omega | \Omega) \), 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.

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...
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 \(P_{\text{lines}}\) on the 3D geometric line segments from the 3D texture OctoMap information \(M_{\text{oct}}\). The 3D geometric line segments are composed of point cloud data \(P_{\text{lines}}\) in this process. This process performs iteration for every occupied node \(n_{\text{occ}}\). First, a position vector \((x, y, z)\) that corresponds to an occupied node \(n_{\text{occ}}\) and a random orientation vector \((\psi, \theta, \phi)\) are produced to generate an arbitrary viewpoint (i.e., an arbitrary 6DOF camera pose \(w\)) in lines 2–4. In line 5, a virtual image \(I_{(w)}\) that corresponds to the arbitrary camera pose \(w\) 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 \(I_{(w)}(u, v)\)) using the camera parameters. The position \(\hat{u} = [u \ v \ s]^T\) and its corresponding pixel area size \(A_{\text{pixel}}\) are calculated by:

\[
\hat{u} = \text{proj}(\hat{n}_{\text{occ}}) = ET\hat{n}_{\text{occ}}
\]

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

(2)

\[
A_{\text{pixel}} = \frac{(I_{\text{node}})^2 f_u}{(d_{\text{optical}})^2},
\]

(3)

\[
d_{\text{optical}} = \begin{bmatrix} f_31 & f_32 & f_33 & t_3 \end{bmatrix} \hat{n}_{\text{occ}},
\]

(4)
transform (Fig. 7(c)) [24]. Next, a set of 2D pixel coordinates \( u_{\text{line}} \) that correspond to the 2D photometric lines are generated from the line image, as shown in Fig. 7(d).

\[
\begin{align*}
  u_{\text{line}} &= \left\{ u_{(k)} \mid 1 \leq k \leq N_{\text{line}} \right\}, \\
  v_{\text{line}} &= \left\{ v_{(k)} \mid 1 \leq k \leq N_{(k)} \right\}, \\
  u_{(k)} &= \left[ u_{(i)}^{(k)} \mid 1 \leq i \leq N^{(k)} \right].
\end{align*}
\]

Here, \( u_{(k)} \) and \( v_{(k)} \) denote the pixel points on the \( k \)th 2D photometric line segment and the number of points respectively, \( N_{\text{line}} \) represents the number of 2D photometric line segments in virtual image plane \( I_{\text{t}}(w) \).

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

\[
\begin{align*}
  P_{\text{lines}} &= \left\{ p_{\text{line}} \mid 1 \leq k \leq N_{\text{line}} \right\}, \\
  p_{\text{line}}^{(k)} &= \left[ p_{(i)}^{(k)} \mid 1 \leq i \leq N^{(k)} \right], \\
  p_{(i)}^{(k)} &= \left[ u_{(i)}^{(k)} \ v_{(i)}^{(k)} \ z_{(i)}^{(k)} \right]^T.
\end{align*}
\]

Finally, the point set of the 3D geometric lines \( P_{\text{lines}} \) extracted from every node are integrated into the overall point cloud \( P_{\text{lines}} \) (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 \( P_{\text{lines}} \) 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:

\[
\text{Cov} \left( p_{\text{line}}^{(k)} \right) = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} \left( p_{(i)}^{(k)} - \bar{p}_{(k)}^{(k)} \right) \left( p_{(i)}^{(k)} - \bar{p}_{(k)}^{(k)} \right)^T, \tag{11}
\]

\[
\hat{p}_{(k)}^{(k)} = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} p_{(i)}^{(k)}. \tag{12}
\]

Here, \( p_{(i)}^{(k)} \) and \( \bar{p}_{(k)}^{(k)} \) are 3D point coordinates and the center value that constitutes the \( k \)th line candidate, respectively. \( N^{(k)} \) represents the number of points that constitute the \( k \)th line candidate. By performing eigenvalue analysis of covariance matrix \( \text{Cov} \left( p_{\text{line}}^{(k)} \right) \) eigenvalues \( \lambda_1 > \lambda_2 > \lambda_3 \) and eigenvectors \( e_1^{(k)}, e_2^{(k)} \) and \( e_3^{(k)} \) 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:

\[
d^{(k)} = \arg\max_{i=1,2,3} s(i), \tag{13}
\]

\[
s(i) = \lambda_i - \lambda_{i-1} \tag{14}
\]

\[
s(i) = \lambda_{i-1} - \lambda_{i-2} \tag{15}
\]

\[
s(i) = \lambda_3 \tag{16}
\]

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.
ments are generated through clustering for the remaining
illustrates a casewhere twoneighbors are specified.
the original points. The elimination result shown in Fig. 11(b)
areaspecified, both the green and yellow points are removed from
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.

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:

\[
p(S(\mathbf{p})) = T_{line}(O(P_{lines}), k) = \frac{p(O(P_{lines})|S(\mathbf{p})) = T_{line})p(S(\mathbf{p}) = T_{line}|O(P_{lines}))}{\sum_{S(\mathbf{p})} p(O(P_{lines})|S(\mathbf{p}))p(S(\mathbf{p}) = T_{line}|O(P_{lines}))},
\]

where \( S(\mathbf{p}) \) denotes whether point \( \mathbf{p} \) belongs to a 3D geometric
line \( S(\mathbf{p}) = T_{line} \) or not \( S(\mathbf{p}) = F_{line} \). \( O(P_{lines}), k \) \( = [O(P_{lines})_1, O(P_{lines})_2, \ldots, O(P_{lines})_k] \) represents the observations
of point \( \mathbf{p} \) from 1st to \( k \)th iteration of the exploration. The initial
probability \( p(S(\mathbf{p}) = T_{line}) = 0.5 \). The likelihood
potential \( p(O(P_{lines})|S(\mathbf{p})) = F_{line} \) can be defined as 0.8 when the line segment is
observed \( S(\mathbf{p}) = T_{line} \), and 0.1 when the line segment is not
observed \( S(\mathbf{p}) = F_{line} \). 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 [28].

3.2.3. Outlier removal using direction vectors
The removal of outliers can be performed by clustering direction
vector information \( a^{(k)} \) corresponding to each 3D geometric
line \( P_{lines}^{(k)} \).

\[
\alpha_{lines} = [a^{(k)} | 1 \leq k \leq N_{line}],
\]

\[
a^{(k)} = [a_x^{(k)} \ a_y^{(k)} \ a_z^{(k)}]^{\top}.
\]

Here, the normalized (i.e., \( \|a^{(k)}\| = 1 \)) direction vector \( a^{(k)} \) 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
direction vectors, as shown Fig. 10(e). The codebooks can be ob-
tained 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 code-
books are generated (cf. without considering the sign, three code-
books) 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

\( P_{lines} \) 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 rep-
resented through a set of coefficients. The 3D geometric line seg-
ments can be parameterized using only six coefficients, as follows:

\[
\begin{align*}
x - x_s &= \frac{y - y_s}{y_e - y_s} = \frac{z - z_s}{z_e - z_s}, \\
x_e - x_s &= \frac{y_e - y_s}{y_e - y_s} = \frac{z_e - z_s}{z_e - z_s}.
\end{align*}
\]
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 \(p_{\text{lines}}\) 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 \(p_{\text{lines}}\)). 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 \(p_{\text{lines}}\) 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 \(p_{\text{lines}}\). 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 \(p_{\text{lines}}\) can be parameterized as follows:

\[
\ell = \left[ \begin{array}{c} l(k) \\ 1 \leq k \leq N_{\text{line}} \end{array} \right],
\]

\[
l(k) = \left[ \begin{array}{cccccc} x_s^{(k)} & y_s^{(k)} & z_s^{(k)} & x_e^{(k)} & y_e^{(k)} & z_e^{(k)} \end{array} \right]^T.
\]

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.

### 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 3D texture 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 keypoint matching [32–35]. Recently, BoVW has been applied to the localization problem [36–38]. These schemes, however, utilize corner-based feature
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 \( M_{3D} \) is converted into 3D geometric line segments \( \mathbf{L} \) 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 \( \alpha \) and the distance from the origin \( \rho \) given by:

\[
\rho = u \cos \alpha + v \sin \alpha, \quad (23)
\]

where \((u, v)\) represents the image coordinates. \( \rho \) is the distance from the origin to the closest point on the straight line, and \( \alpha \) 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 \((\rho, \alpha)\). The \((\rho, \alpha)\) 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 \( \rho \) and \( \alpha \) 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 \( \mathbf{L}_{2D} \) generated by projecting the parameterized 3D geometric lines segments \( \mathbf{L} \) 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 \( \mathbf{L}_{2D} = (\rho(1), \alpha(1)) \) to transform into the index \( \mathbf{L}_{2D} = (\rho(2), \alpha(2)) \) on the Hough space, and \( \text{round}(\cdot) \) represents the rounding function. Here, \( b_{\rho}, b_{\alpha} \) is the quantum size on the Hough space. After quantizing, line 5 calculates each element of QLH descriptor \( \mathbf{Q} \) using the typical Gaussian Kernel function \( K(\cdot) \) (i.e., \( \mathbf{O} \) mean and \( \mathbf{I} \) covariance), which is given by

\[
K(x) = \frac{1}{(2\pi)^{D/2}} \exp \left( -\frac{1}{2} x^T \mathbf{x} \right), \quad (24)
\]

where \( N_y, N_h, \) and \( D \) denote the number of 2D photometric lines, smoothing parameter, and dimensions (two in this case), respectively. A matrix norm of \( \mathbf{Q} \) 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 \((\rho(2), \alpha(2))\) and registered at the corresponding cell in the Hough space (Fig. 15(b)). Then, Gaussian Kernel function \( K(\cdot) \) is applied to manage unexpected

\[Fig. 14.\] 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 \( K(\cdot) \) to manage noise.
noise (Fig. 15(c)). Although the environment is fixed, the Q LH image descriptor changes depending on the camera pose, and thus the Q LH descriptor can be used as an available signature for the calibration process, as described in next subsection.

4.3. Evaluation of Q LH descriptor based on earth mover’s distance

The similarity between two sets of Q LH 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 Q LH descriptor can be managed as a 2D distribution, which is defined in the feature space \((\rho_{\text{QH}}, \alpha_{\text{QH}})\). In order to compute EMD between two Q LH descriptors, each Q LH descriptor is converted to a set of clusters \(Q_{\text{sig}}\).

\[
Q_{\text{sig}}^{(1)} = \left\{ q_i^{(1)}, \sigma_i^{(1)} \mid 1 \leq i \leq N_{q} \right\},
\]

\[
Q_{\text{sig}}^{(2)} = \left\{ q_j^{(2)}, \sigma_j^{(2)} \mid 1 \leq j \leq N_{q} \right\},
\]

where \(q\) and \(\sigma\) indicate the cluster representative (i.e., the coordinates \((\rho_{\text{QH}}, \alpha_{\text{QH}})\) of the Q LH descriptor) and the weight value that belongs to that cluster (i.e., \(Q_{\rho_{\text{QH}}, \alpha_{\text{QH}}}\)), respectively. The size of cluster \(N_{q}\) is equal to the size of the Q LH 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 \(F\) with flow elements \(f_{ij}\) between \(q_i^{(1)}\) and \(q_j^{(2)}\) that minimizes the following overall cost is calculated.

\[
\text{WORK}(Q_{\text{sig}}^{(1)}, Q_{\text{sig}}^{(2)}, F) = \sum_{i=1}^{N_{q}} \sum_{j=1}^{N_{q}} f_{ij} d_{ij}
\]

s.t. \(f_{ij} \geq 0, 1 \leq i \leq N_{q}, 1 \leq j \leq N_{q}\)

\(\sum_{j=1}^{N_{q}} f_{ij} \leq \sigma_i^{(1)}, 1 \leq i \leq N_{q}\)

\(\sum_{i=1}^{N_{q}} f_{ij} \leq \sigma_j^{(2)}, 1 \leq j \leq N_{q}\)

\(\sum_{i=1}^{N_{q}} \sum_{j=1}^{N_{q}} f_{ij} = \min \left( \sum_{i=1}^{N_{q}} \sigma_i^{(1)}, \sum_{j=1}^{N_{q}} \sigma_j^{(2)} \right)\),

where \(d_{ij}\) represents the user-defined ground distance, which is the distance between clusters \(q_i^{(1)}\) and \(q_j^{(2)}\), as follows:

\[
d_{ij} = \sqrt{\tau_{\rho} (\Delta \rho_{\text{QH}})^2 + \tau_{\alpha} (\Delta \alpha_{\text{QH}})^2},
\]

\[
\Delta \rho_{\text{QH}} = |\rho_j^{(2)} - \rho_i^{(1)}|,
\]

\[
\Delta \alpha_{\text{QH}} = \arctan \frac{\tan \alpha_j^{(2)} - \tan \alpha_i^{(1)}}{1 + \tan \alpha_j^{(2)} \tan \alpha_i^{(1)}}.
\]

Here, \(\tau_{\rho}\) and \(\tau_{\alpha}\) are the weight parameters that adjust the importance of the distance and slope distributions. s.t.1 allows moving supplies from \(Q_{\text{sig}}^{(1)}\) to \(Q_{\text{sig}}^{(2)}\) and not vice versa. s.t.2 and s.t.3 limit the amount of supplies that can be sent by the clusters in \(Q_{\text{sig}}^{(1)}\) to their weights, and the clusters in \(Q_{\text{sig}}^{(2)}\) 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 \(F\) can be obtained. Hence, EMD is defined as the work normalized by the total flow, as follows:

\[
\text{EMD}(Q_{\text{sig}}^{(1)}, Q_{\text{sig}}^{(2)}) = \sum_{i=1}^{N_{q}} \sum_{j=1}^{N_{q}} f_{ij} d_{ij} \sum_{i=1}^{N_{q}} \sum_{j=1}^{N_{q}} f_{ij}.
\]
where \( \mathbf{w} = [x_c, y_c, z_c, \psi_c, \theta_c, \phi_c]^T \) denotes the 6DOF camera pose at \( k \)th iteration. Here, the subscript \( k \) that represents the state of the current iteration is omitted. \( \mathbf{Q} \) denotes the QLH descriptor from the real camera image (i.e., the measurement data), and \( \mathcal{M}_{\text{oct}} \) is the 3D texture OctoMap information. \( p(\mathbf{w}|\mathcal{M}_{\text{oct}}) \), 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 \( \mathbf{w} \) by applying the prediction model \( p(\mathbf{w}|\mathbf{w}_{k-1}, \mathcal{M}_{\text{oct}}) \) to the posterior distribution at previous iteration \( p(\mathbf{w}_{k-1}|\mathbf{Q}, \mathcal{M}_{\text{oct}}) \). Eq. (33) represents the update phase to update the probability distribution of state \( \mathbf{w} \) by applying the measurement model \( p(\mathbf{Q}|\mathbf{w}, i) \) to prior distribution \( p(\mathbf{w}|\mathcal{M}_{\text{oct}}) \). Measurement model \( p(\mathbf{Q}|\mathbf{w}, i) \) is defined by the likelihood function that should be designed for the QLH descriptor in this study. Here, \( \eta \) is the normalizing constant. Note that the prior information \( \mathcal{M}_{\text{oct}} \) is converted into the 3D geometric line parameters \( i \) 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 \( \mathbf{w} \).

### 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 \( \mathbf{w} \), the particle filter uses a set of random weighted particles represented by:

\[
S = \{ (\mathbf{w}_i, w_i) \mid 1 \leq i \leq N_p \},
\]

where \( \mathbf{w}_i \) is the pose of the \( i \)th particle with associated importance weight \( w_i \) at \( k \)th 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 \( p(\mathbf{w}|\mathbf{Q}, \mathcal{M}_{\text{oct}}) \), which represents camera pose \( \mathbf{w} \) conditioned on the measurement data (i.e., QLH descriptor \( \mathbf{Q} \) generated from the real camera image) and map information \( \mathcal{M}_{\text{oct}} \). The posterior distribution on camera pose \( \mathbf{w} \) is approximated from a set of weighted particles, as follows:

\[
p(\mathbf{w}|\mathbf{Q}, \mathcal{M}_{\text{oct}}) \approx \sum_{i=1}^{N_p} w_i \delta(\mathbf{w} - \mathbf{w}_i),
\]

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 \( S^- \) is generated from past particle set \( S_{k-1} \) based on state transition probability \( p(\mathbf{w}|\mathbf{w}_{k-1}, \mathcal{M}_{\text{oct}}) \), as follows:

\[
\mathbf{w}_i^* \sim p(\mathbf{w}|\mathbf{w}_{k-1}, \mathcal{M}_{\text{oct}}).
\]

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 \( (\psi_c, \theta_c, \phi_c) \).

### 5.3. Importance weighting

To calculate the normal vector to make camera orientation constraints:

(a) calculation of normal vector by fitting plane and (b) extracted normal vectors \( \hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z) \) (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:

\[
\mathbf{w}_i^* = \mathbf{w}_{i,k-1} + \epsilon
\]

\[\text{s.t.1 } \exists (x^*, y^*, z^*) \in \mathbf{n}_{\text{occ}} \]

\[\text{s.t.2 } \arctan(\hat{\mathbf{n}}_y, \hat{\mathbf{n}}_x) - \frac{\pi}{2} > \arctan(\cos \theta^*_i \sin \phi^*_i) \]

\[\text{s.t.3 } \arccos(\hat{\mathbf{n}}_z) - \frac{\pi}{2} > \arccos(-\sin \theta^*_i) \]

\[\text{s.t.4 } \epsilon \sim \mathcal{N}(0, \text{diag}(\sigma^2_\psi, \sigma^2_\theta, \sigma^2_\phi)), \]

where noise variable \( \epsilon \) follows a 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 \( \mathbf{n}_{\text{occ}} \) of 3D OctoMap. s.t.2 and s.t.3 are the constraints on the orientation where predicted pitch \( \theta^*_i \) and yaw angle \( \phi^*_i \) cannot be more than 90 degrees from a normal vector of the corresponding wall plane. Here, \( \hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z) \) refers to the normal vector. Note that roll angle \( \psi^*_i \) does not have any constraints because it represents the rotation on the optical axis.

In order to calculate the normal vector \( \hat{\mathbf{n}} = (\hat{n}_x, \hat{n}_y, \hat{n}_z) \) 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 \( \mathbf{n}_{\text{occ}} \) 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:

\[
z = ax + by + c.
\]
model for the entire 3D OctoMap is shown in Fig. 17(b).

determination of the camera orientations represented in Eq. (37) can be applied for each node  \( \hat{x} \) calculations, then the normal vector through the following measurement model:

\[
\begin{align*}
&\mathbf{a} = \begin{bmatrix}
\sum_{i,j} x_i y_j \\
\sum_{i,j} y_i \\
\sum_{i,j} z_i
\end{bmatrix}, \\
&\mathbf{b} = \begin{bmatrix}
\sum_{i,j} x_i y_j \\
\sum_{i,j} y_i \\
\sum_{i,j} z_i
\end{bmatrix}, \\
&\mathbf{c} = \begin{bmatrix}
\sum_{i,j} x_i y_j \\
\sum_{i,j} y_i \\
\sum_{i,j} z_i
\end{bmatrix}, \\
&\mathbf{d} = \begin{bmatrix}
\sum_{i,j} y_i z_j \\
\sum_{i,j} y_i \\
\sum_{i,j} z_i
\end{bmatrix}.
\end{align*}
\]

This optimization problem can be solved by the following calculations:

\[
\begin{align*}
&\hat{n}_x = \frac{1}{\sqrt{a^2 + b^2 + 1}} \begin{bmatrix} -a \\ -b \\ 1 \end{bmatrix}, \\
&\hat{n}_y = \frac{1}{\sqrt{a^2 + b^2 + 1}} \begin{bmatrix} a \\ -b \\ 1 \end{bmatrix}, \\
&\hat{n}_z = \frac{1}{\sqrt{a^2 + b^2 + 1}} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}.
\end{align*}
\]

Here, \((x_i, y_i, z_i)\) 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 \(\hat{n} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)\) that corresponds to each node \(n_{loc}\) 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 \(\sigma_i\) is evaluated using measurement model \(p(Q|w, M_{oct})\), as follows:

\[
\sigma_i = \eta \cdot p(Q|w_i^*, M_{oct}),
\]

where \(\eta\) is a normalization constant. Eq. (41) is the same as the measurement model that represents the likelihood function. Thus, \(p(Q|w_i^*, M_{oct})\) 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:

\[
p(Q|w_i^*, M_{oct}) = \frac{1}{\sqrt{2\pi \sigma_{EMD}^2}} \exp\left(\frac{-EMD(Q, h(w_i^*))^2}{2\sigma_{EMD}^2}\right),
\]

where \(Q\) and \(h(w_i^*)\) refer to the QLH descriptor extracted from a real camera image and that predicted from the pose of \(i\)th particle \(w_i^*\), respectively. \(\sigma_{EMD}\) represents the standard deviation associated with the uncertainty of the QLH descriptor (e.g. \(\sigma_{EMD} = 3.0\) in the case of Fig. 18). \(EMD(\cdot, \cdot)\) 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 \(S\) is randomly chosen from \(S^*\) according to the distribution defined by the importance factor \(\sigma_i\):

\[
S = \left\{ [w_j, 1/N_p] \mid 1 \leq j \leq N_p \right\} \sim \left[ w_i^*, \sigma_i \right].
\]

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 \(S\) 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 \(w^*\) is calculated by the weighted average, as follows:

\[
w^* = \frac{1}{N_p} \sum_{i=1}^{N_p} \sigma_i w_i.
\]

5.3. Time complexity of algorithm

The time complexity of algorithms is most widely expressed using the big-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:

\[
\begin{align*}
O_{sample} &= O(N_p), \\
O_{weight} &= N_p(C_{proj} + C_{QLH} + C_{EMD}) \\
&= N_p(O(N_{line}) + O(N_j^2) + O(N_j^2)) \\
&= O(N_p(N_{line} + N_j^2 + N_j^2)), \\
O_{resample} &= O(N_p), \\
\end{align*}
\]

where \(N_{line}, N_p,\) and \(N_j\) 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 \(O_{proj} = O(N_{line})\) required for projecting 3D geometric line segments \(i\) into a 2D image plane, complexity \(O_{QLH} = O(N_j^2)\) required for generating the QLH descriptor \(Q\) from projected 2D photometric lines \(2D\) (Table 3), and complexity \(O_{EMD} = O(N_j^2)\) required for calculation of the EMD between the QLH descriptors (Eq. (31)). In conclusion, the time complexity \(O_{iter}\) required for one iteration in the proposed calibration framework can be represented as follows:

\[
\begin{align*}
O_{iter} &= O_{sample} + O_{weight} + O_{resample} \\
&= O(N_p + N_p(N_{line} + N_j^2 + N_j^2) + N_p) \\
&= \approx O(N_p(N_j^2 + N_j^2)).
\end{align*}
\]
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 $w^{(1)}$, (d) simulated image data by changing illumination condition from camera $w^{(2)}$, (e) simulated image data from camera $w^{(3)}$. (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

<table>
<thead>
<tr>
<th>Algorithm to generateQLH descriptor Q from 2D photometric line segments $l_{2D}$.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Generation of QLH descriptor Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>2D photometric line segments $l_{2D} = [l_{ij}^{(2)}</td>
</tr>
<tr>
<td>output</td>
<td>QLH descriptor Q</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>for every 2D photometric line segments $l_{ij}^{(2)} = (p_j^{(2)}, a_j^{(2)})$</td>
</tr>
<tr>
<td>2</td>
<td>get quantized distance index $\rho_{a_{ij}} = \text{round}(\alpha_{j}^{(2)}/b_{\text{max}})$</td>
</tr>
<tr>
<td>3</td>
<td>get quantized distance index $a_{i_{ij}} = \text{round}(\alpha_{j}^{(2)}/b_{\text{min}})$</td>
</tr>
<tr>
<td>4</td>
<td>$l_{k_{ij}} = (\rho_{a_{ij}}, a_{i_{ij}})$</td>
</tr>
<tr>
<td>5</td>
<td>$Q = \frac{1}{N_j \times K} \sum_{j=1}^{N_j} K \left( \frac{l_{k_{ij}} - \bar{F}<em>{l}}{\bar{J} - \bar{F}</em>{l}} \right)$</td>
</tr>
<tr>
<td>6</td>
<td>end for</td>
</tr>
<tr>
<td>7</td>
<td>normalize Q</td>
</tr>
<tr>
<td>8</td>
<td>return Q</td>
</tr>
</tbody>
</table>

Here, the computations that correspond to a constant and $N_{\text{line}}$ (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 $\times$ 8 m $\times$ 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 $f_u, f_v, f_{\text{skew}}$, and $(c_u, c_v)$ for all virtual cameras are set to 930.261, 926.302, 0, and (663.912, 456.287), respectively. The image dimensions are 1280 $\times$ 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 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 $N_f = 12$, and $N_t = 120 \times 72$ (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...
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 \((\mathbf{w}^{(1)})\) and \((\mathbf{w}^{(2)})\) 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)
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 \((w^{(1)}_c)\) and \((w^{(2)}_c)\) 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 \(w^{(3)}_c\), 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 \(w^{(3)}_c\) is shown in Figs. 27(a) and 28(a). As shown in Fig. 22(d), the image captured by camera \(w^{(3)}_c\) has a jumble of waste baskets at the left side, whereas the images from cameras \(w^{(1)}_c\) and \(w^{(2)}_c\) are structurally clean and stable, as shown in Fig. 22(b) and (c). Therefore, in case of the image data captured from camera \(w^{(3)}_c\), the robust and intuitive 2D photometric line segments cannot be extracted relatively. As a result, the particles for camera \(w^{(3)}_c\) 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 \(w^{(3)}_c\), 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 \(w^{(3)}_c\) 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 \(w^{(3)}_c\) is not structurally stable. Note that the complete 6DOF parameter calibration including the roll angle \(\psi_c\), which were excluded in the global estimation, could be carried out successfully.

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
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


