Received March 3, 2020, accepted April 10, 2020, date of publication April 17, 2020, date of current version May 4, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2988464*

# New Monte Carlo Localization Using Deep Initialization: A Three-Dimensional LiDAR and a Camera Fusion Approach

**HYUNGGI JO AND EUNTAI KIM, (Member, IEEE)**
School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea

Corresponding author: Euntai Kim (etkim@yonsei.ac.kr)

This work was supported in part by the Industrial Convergence Core Technology Development Program (Development of Robot Intelligence Technology for Mobility with Learning Capability Toward Robust and Seamless Indoor and Outdoor Autonomous Navigation) funded by the Ministry of Trade, Industry and Energy (MOTIE), South Korea, under Grant 10063172.

**ABSTRACT** Fast and accurate global localization of autonomous ground vehicles is often required in indoor environments and GPS-shaded areas. Typically, with regard to global localization problem, the entire environment should be observed for a long time to converge. To overcome this limitation, a new initialization method called deep initialization is proposed and it is applied to Monte Carlo localization (MCL). The proposed method is based on the combination of a three-dimensional (3D) light detection and ranging (LiDAR) and a camera. Using a camera, pose regression based on a deep convolutional neural network (CNN) is conducted to initialize particles of MCL. Particles are sampled from the tangent space to a manifold structure of the group of rigid motion. Using a 3D LiDAR as a sensor, a particle filter is applied to estimate the sensor pose. Furthermore, we propose a re-localization method for performing initialization whenever a localization failure or the situation of robot kidnapping is detected. Either the localization failure or the kidnapping is detected by combining the outputs from a camera and 3D LiDAR. Finally, the proposed method is applied to a mobile robot platform to prove the method's effectiveness in terms of both the localization accuracy and time consumed for estimating the pose correctly.

**INDEX TERMS** 3D LiDAR, camera, fusion, particle filter, deep learning, sensor pose regression.

## I. INTRODUCTION

Global localization is a fundamental problem in autonomous navigation of mobile robots [1], [2]. As commercial robots run into various challenging situations such as an unknown initial pose or the kidnapped robot problem, fast and accurate yet reliable global localization is highly required. The three-dimensional (3D) light detection and ranging (LiDAR) sensor has been widely used as a primary sensor for global localization. The 3D LiDAR provides accurate distance and intensity information about the surrounding environment, and it has received a lot of attention with the development of consumer-grade products. Studies on global localization using the 3D LiDAR can be divided into the following two approaches:

The first approach is based on the registration using point clouds align methods such as the iterative closest point (ICP) algorithm [3], [4] and normal distribution transform (NDT) [5], [6]. These methods determine the sensor pose by registering the difference between the map and the currently observed points. Such registration methods have the shortcoming of falling into local minima, since they are based on the minimization of the cost function.

The second approach is based on the Monte Carlo framework [7]–[11]. The Monte Carlo localization (MCL) method allows localizing a sensor pose using the distribution of state represented by particles. In MCL, particles are sampled to estimate the robot pose, and they are updated based on the comparison of the sensor measurements with a given map. MCL-based methods converge into a global pose more consistently compared to registration methods [12], [13]. However, MCL-based methods have the shortcoming of taking a long time to converge, since measurements should be observed continuously for a certain period of time. Therefore, the judicious choice of the initial estimate about the sensor pose is of great importance in MCL since it can significantly narrow down the search space and shorten the runtime.

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

Several methods for MCL initialization have been proposed to date. The majority of these methods use other sensors in addition to the 3D LiDAR sensor [14]–[18]. The most common method for MCL initialization uses the global positioning system (GPS). However, the use of the GPS is not attractive in urban environments with many GPS-shaded areas such as inside buildings or under piers [19], [20].

Other methods for MCL initialization use a camera and employ visual features or artificial landmarks to estimate the initial pose [18], [21], [22]. However, a large database, in which key-frames and visual feature information are stored, should be used to estimate the initial pose. Furthermore, an efficient method for retrieving the pose from the database, such as perspective-n-point (PnP) with RANSAC, is also required [21], [23].

In this paper, a new initialization method called deep initialization is proposed and it is applied to MCL. To make promising candidates for MCL, each particle is sampled from the tangent space to the manifold structure of the *Special Euclidean* group, SE(n). Concerning the deep initialization, pose regression based on a convolutional neural network (CNN) [24], [25] is employed in the proposed deep initialization method to estimate the initial sensor pose. An initial set of particles is generated by adding centered Gaussian noise [26] around the initial pose. Concerning the MCL, a localization failure detection algorithm is further developed to recognize the kidnapped situation during MCL. Deep initialization is performed again whenever a localization failure is detected. To prove the effectiveness of the proposed method, it is applied to a mobile robot platform.

The contribution of this paper can be summarized as follows:

1) An effective initialization method for MCL is proposed and the method is named deep initialization. CNN-based pose regression is employed to estimate the initial pose of the robot. Then, particles are sampled from a tangent space to the manifold structure of SE(3) using Lie algebra to generate an initial set of particles.
2) A localization failure detection algorithm is developed to recognize the occasions of kidnapping or when the robot fails to estimate its location. The occasions are recognized by combining the outputs from a 3D LiDAR and a camera. Upon recognizing such situations, particles are re-initialized.

The rest of this paper is organized as follows. Section II provides an overview of the related work. Section III describes the system framework employed in this study. Section IV presents the proposed MCL method using deep initialization and a particle filter with localization failure detection. Section V presents experimental results, while Section VI concludes the paper.

## II. RELATED WORKS

The 3D LiDAR technology has received a lot of attention within the robotics community. When the 3D LiDAR is used for global localization, MCL tends to perform better than the registration method.

In the MCL using a 3D LiDAR, effective initialization is of great importance and some researches have been conducted regarding the initialization. As an early study on LiDAR-based MCL, Levinson and Thrun [8] created a 2D map using the reflectivity of the ground and performed localization using a particle filter. Since 2D maps are not enough to contain 3D information, Kim *et al.* [13] proposed an entropy-weighted particle filter with 2.5D grid map, while Saarinen *et al.* [12] updated a particle filter by storing the entire map as a normal distribution (ND) map and calculating the likelihood using the NDT with the current sensor input. MCL-based methods guarantee reliable global optimality; however, it takes a long time to converge. This is because particles are sampled uniformly throughout the entire space, and each particle is evaluated against the prior map over a period of time. Therefore, an initial estimate of MCL is important to narrow the search space.

Reading from additional sensors are often used in localization algorithms to get a reasonable initial estimate. The easiest and most common way of obtaining the initial estimate is to use GPS signal. Blanco-Claraco *et al.* [14] and Suhr *et al.* [15] proposed a global localization method using particle filters based on poor GPS signals. However, GPS signals are not available in GPS-shaded areas such as indoors.

Another way of obtaining the initial estimate is to utilize a camera sensor, which can provide intuitive information about the environment such as visual features [21], [27], [28] and artificial landmarks [18] that cannot be obtained from the LiDAR sensor. Zhang and Singh [27] and Huang and Stachniss [28] improved the localization accuracy by using the initial pose calculated with correspondences of visual features; however, the initialization problem of global localization was not considered in these studies. Su *et al.* [21] performed the initialization of MCL by retrieving the best-matched key frame compared with a current frame. However, this approach has the disadvantage of relying on large database containing visual feature information, and thus requiring a memory size that is linearly proportional to the size of the environment. Furthermore, PnP optimization is required for estimating the metric pose.

More recently, deep learning has been used for estimating absolute 6-DOF poses in an end-to-end manner [25], [29]–[32]. Methods based on deep learning exploit the advantages of metric localization [23] and place recognition [33] resulting in fast estimation of sensor pose using only one image. Our work takes the advantages of pose regression based on deep learning and accomplishes fast and robust MCL in urban environments as described next.

## III. SYSTEM OVERVIEW

### A. SENSOR SETUP

The sensor system used in this study comprised a stereo camera mounted under a 3D LiDAR sensor as illustrated in Fig. 1. Velodyne VLP-16 was used as a 3D LiDAR sensor

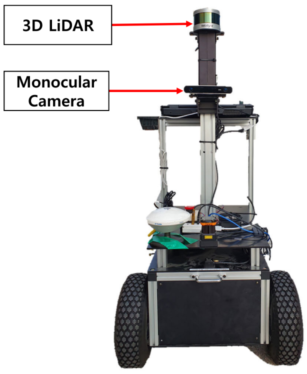


**FIGURE 1. Experimental sensor system equipped with a laptop, a 3D LiDAR sensor, and a monocular camera.**

centered at the top. A ZED stereo camera was installed facing forward under VLP-16. The right-side camera was turned off and the ZED camera was used as a monocular camera.

The extrinsic calibration between the 3D LiDAR and the camera was conducted using the automatic calibration method developed by Vel'as *et al.* [34]. By comparing the artificial landmarks detected by the LiDAR and camera, the transformation matrix $\mathbf{T}_{CL} \in SE(3)$ between the two sensors can be found as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \cdot {\mathbf{T}_{CL}}^{-1} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{1}$$

where SE(3) denotes a *Special Euclidean* group in 3D; $\begin{bmatrix} x & y & z \end{bmatrix}^T \in \mathbb{R}^3$ denotes a point detected by the 3D LiDAR, and $\begin{bmatrix} u & v \end{bmatrix}^T \in \mathbb{R}^2$ denotes its corresponding point detected by the camera; $s$ is a scale factor and $P$ is the projection matrix that can be obtained from the intrinsic parameters of the camera. Using the transformation $\mathbf{T}_{CL}$, when the global pose of the camera is given by $\mathbf{x}^{Camera} \in SE(3)$, the pose of the LiDAR $\mathbf{x}^{Lidar} \in SE(3)$ is obtained as

$$\mathbf{x}^{Lidar} = \mathbf{x}^{Camera} \oplus \mathbf{T}_{CL}, \tag{2}$$

where $\oplus$ denotes the pose composition operator [35]. Two versions of the pose composition operators were used in this study defined as $\oplus_1 : SE(3) \times SE(3) \rightarrow SE(3)$ and $\oplus_2 : SE(3) \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$. Both can be found in [36]. For simplicity, these two operators are represented in this paper simply as $\oplus$ without specification. Since one pose can be computed from another pose using (2), we focus only on $\mathbf{x}^{Lidar}$, which we denote as un-superscripted $\mathbf{x}$ to simplify the notation in the subsequent developments.

## B. OVERVIEW

The proposed method includes three stages: deep initialization, a particle filter (PF), and localization failure detection. During the deep initialization stage, deep learning is employed to generate an initial set of particles. Particles are sampled from twist coordinates around the deep learning based visual localization result to generate the most probable particles. Then, a set of particles is applied to the PF to conduct MCL. In this study, deep initialization was conducted using the monocular camera, whereas particle filtering was performed using the LiDAR. To predict whether localization would fail, the deep learning result from the camera is compared to the PF result from the LiDAR. If localization is very likely to fail, deep initialization is conducted again to re-localize the sensor system. The overview of the proposed MCL is provided in Fig. 2.

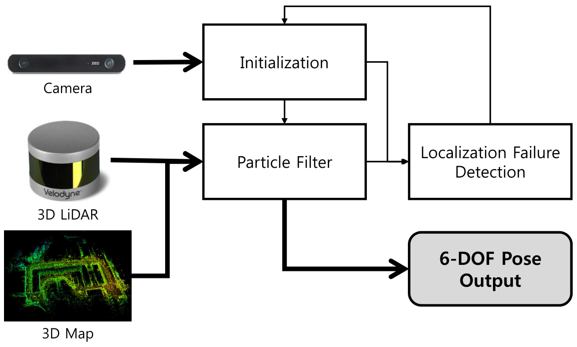


**FIGURE 2. Overview of the proposed Monte Carlo localization method.**

Particle filtering is performed using the LiDAR, wheel odometry, and a 3D map. Thus, we can estimate the sensor pose accurately. The 3D map is a point-cloud metric map defined as

$$\mathbf{m} = \left\{ m_n \in \mathbb{R}^3 | n = 1, \cdots, N_{map} \right\}, \tag{3}$$

where $N_{map}$ denotes the number of point clouds in the map $\mathbf{m}$. The map can be built in advance using either the (1) RTK-grade localization solution or (2) SLAM algorithm based on pose graph optimization [37] using LiDAR odometry [38] and visual SLAM [23]. In this study, the map $\mathbf{m}$ was built by using the second method.

## C. NOTATION

There are four ways to represent the 6-DOF pose of a sensor: (1) 3D translation + yaw-pitch-roll (RPY), (2) 3D translation + quaternion, (3) 4 x 4 transformation matrix in SE(3), and (4) twist coordinates. Since the RPY notation has a critical shortcoming known as the gimbal lock problem, the first notation is not used in this study. In the subsequent development, we use the following three notations and assume that one notation can be systematically converted to another on a one-to-one basis according to [36]:

- $\bar{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\upsilon} & \boldsymbol{\omega} \end{bmatrix}^T \in \mathbb{R}^6$ *: twist coordinates of Lie algebra.* $\boldsymbol{\upsilon} \in \mathbb{R}^3$ *and* $\boldsymbol{\omega} \in \mathbb{R}^3$ *;* $\boldsymbol{\upsilon}$ *and* $\boldsymbol{\omega}$ *denote the rotated versions of the translation and axis-angle representation of* $\bar{\mathbf{x}}$*, respectively.*
- $\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{p} & \mathbf{q} \end{bmatrix}^T \in \mathbb{R}^7$ *: 3D position + quaternion,* $\mathbf{p} \in \mathbb{R}^3$ *and* $\mathbf{q} \in \mathbb{R}^4$*;* $\mathbf{p}$ *and* $\mathbf{q}$ *denote the corresponding position and quaternion of* $\tilde{\mathbf{x}}$*, respectively.*

- $\mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \subset \mathbb{R}^{4\times 4}$ *: a* $4 \times 4$ *transformation matrix in SE(3),* $\mathbf{R} \in SO(3)$ *and* $\mathbf{p} \in \mathbb{R}^3$*;* $\mathbf{R}$ *and* $\mathbf{p}$ *denote the corresponding rotation matrix and translation vector of* $\mathbf{x}$*, respectively.*

That is, the vanilla, tilded and barred characters denote the 6-DOF sensor pose in the notations of the twist coordinates, 3D translation + quaternion, and a $4 \times 4$ matrix in SE(3), respectively. Obviously, $\mathbf{p}$ in $\tilde{\mathbf{x}}$ is the same as $\mathbf{p}$ in $\mathbf{x}$. Furthermore, the twist coordinates $\bar{\mathbf{x}}$ and matrix $\mathbf{x}$ in SE(3) are related as

$$\bar{\mathbf{x}} = \left(\hat{\mathbf{x}}\right)^{\vee} = \{\log(\mathbf{x})\}^{\vee}, \tag{4}$$

where $\hat{\mathbf{x}} = \log(\mathbf{x}) = \begin{bmatrix} (\boldsymbol{\omega})_{\times} & \boldsymbol{\upsilon} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{se}(3)$ is a *twist* belonging to the tangent space of SE(3) at the identity [39]. In (4), the *vee* operator ($\vee$) is a mapping from a twist $\hat{\mathbf{x}} = \begin{bmatrix} (\boldsymbol{\omega})_{\times} & \boldsymbol{\upsilon} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ to twist coordinates $\bar{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\upsilon} & \boldsymbol{\omega} \end{bmatrix}^T$ and it is defined by

$$\begin{bmatrix} (\boldsymbol{\omega})_{\times} & \boldsymbol{\upsilon} \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{bmatrix} \in \mathbb{R}^6. \tag{5}$$

The *hat* operator ($\wedge$) is its inverse operator and it is defined by

$$\begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{bmatrix}^{\wedge} = \begin{bmatrix} (\boldsymbol{\omega})_{\times} & \boldsymbol{\upsilon} \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(3), \tag{6}$$

where $(\cdot)_{\times}$ is an operator which maps a vector in $\mathbb{R}^3$ to its skew-symmetric matrix as

$$(\boldsymbol{\omega})_{\times} = \left(\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}\right)_{\times} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{7}$$

Conversely, the $4 \times 4$ matrix $\mathbf{x}$ in SE(3) can be represented in terms of a twist $\hat{\mathbf{x}}$ as

$$\begin{aligned} \mathbf{x} = \exp\left(\hat{\mathbf{x}}\right) &= \begin{bmatrix} \exp(\boldsymbol{\omega}) & \mathbf{V}\boldsymbol{\upsilon} \\ \mathbf{0} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \text{if } \boldsymbol{\omega} \neq 0 \end{aligned} \tag{8}$$

where $\exp(\boldsymbol{\omega}) = \mathbf{I}_3 + \frac{\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|}(\boldsymbol{\omega})_{\times} + \frac{(1-\cos(\|\boldsymbol{\omega}\|))}{\|\boldsymbol{\omega}\|^2}(\boldsymbol{\omega})_{\times}{}^2$ by *Rodrigues' formula*, $\mathbf{V} = \mathbf{I}_3 + \frac{(1-\cos(\|\boldsymbol{\omega}\|))}{\|\boldsymbol{\omega}\|^2}(\boldsymbol{\omega})_{\times} + \frac{\|\boldsymbol{\omega}\|-\sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^3}(\boldsymbol{\omega})_{\times}{}^2$ [40].

In summary, the LiDAR pose is represented by one of the three notations $\bar{\mathbf{x}}$, $\tilde{\mathbf{x}}$ and $\mathbf{x}$, and the pose of the camera is also be represented by one of the three notations $\mathbf{x}^{Camera}$, $\tilde{\mathbf{x}}^{Camera}$ and $\bar{\mathbf{x}}^{Camera}$. As stated before, the two sensor poses are related as $\mathbf{x} = \mathbf{x}^{Camera} \oplus \mathbf{T}_{CL}$.

## IV. MCL WITH DEEP INITIALIZATION

To run a PF in MCL, an initial set of particles is required. The most common way to generate an initial set of particles is to sample from a uniform distribution over the entire search space; however, this is not efficient. Instead, we propose a deep initialization method for generating an initial set of particles. Re-localization is conducted when MCL is considered to have failed due to some reasons such as kidnapping or fast rotation during MCL.

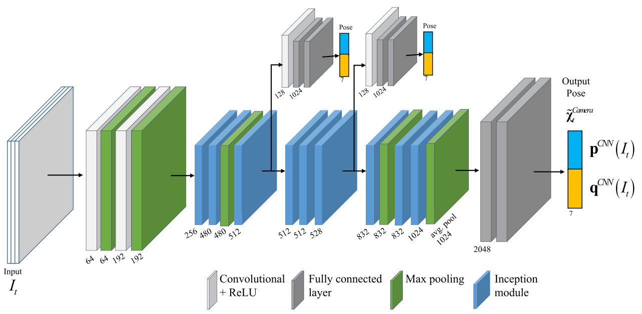


**FIGURE 3.** **A network architecture of pose regression.**

### A. DEEP INITIALIZATION

#### 1) POSE REGRESSION USING MONOCULAR CAMERA

A CNN is applied to the camera image to estimate the absolute 6-DOF sensor pose. To train the pose regression network, an off-line SLAM algorithm based on the work presented in [38] and [37] is applied to build a set of training samples $\mathcal{T} = \left\{\left(I, \tilde{\mathbf{x}}_I\right) | I \in \mathcal{I},\ \tilde{\mathbf{x}}_I = [\mathbf{p}_I,\ \mathbf{q}_I]^T\right\}$, where $\mathcal{I}$ denotes a set of training images. Each training sample has a 6-DOF pose defined as $\tilde{\mathbf{x}}_I = [\mathbf{p}_I,\ \mathbf{q}_I]^T \in \mathbb{R}^7$, where $\mathbf{p}_I \in \mathbb{R}^3$ denotes the position of the sensor in the 3D Euclidean space, and $\mathbf{q}_I \in \mathbb{R}^4$ denotes the orientation of the sensor represented using a quaternion. Note that subscription $I$ indicates that an image $I$ is taken at $\tilde{\mathbf{x}}_I$. In this study, PoseNet [24] and its updated version [25] was employed to implement the mapping from an image $I$ to the corresponding sensor pose $\tilde{\mathbf{x}}_I = [\mathbf{p}_I,\ \mathbf{q}_I]^T$. Similar to the study presented in [24], [25], GoogLeNet [41] was used as a backbone; PoseNet is modified version of GoogLeNet replacing the two auxiliary classifiers taken from the middle of the network and a top output classifier with affine regressors, as shown in Fig. 3. In the figure, the blue cubes denote the inception modules in GoogLeNet. In PoseNet, CNN was trained to minimize the Euclidean loss function defined as

$$\mathcal{L}_I = \left\|\mathbf{p}^{CNN}(I) - \mathbf{p}_I\right\|_2 + \beta\left\|\mathbf{q}^{CNN}(I) - \frac{\mathbf{q}_I}{\|\mathbf{q}_I\|}\right\|_2, \tag{9}$$

where $\beta$ is a scale factor, $\mathbf{p}^{CNN}(I)$ and $\mathbf{q}^{CNN}(I)$ are the outputs from the CNN when an image $I$ is applied. They are actually the estimated position and orientation of the camera, at which $I$ is taken, respectively. After completing the training of CNN, we can estimate the absolute pose of the camera $\tilde{\mathbf{x}}_t^{Camera}$ at which a new test image $I_t$ is taken by applying the test image $I_{test}$ to the pose regression network and obtaining

$$\tilde{\chi}_t^{Camera} = CNN(I_t) = \begin{bmatrix} \mathbf{p}^{CNN}(I_t) \\ \mathbf{q}^{CNN}(I_t) \end{bmatrix}, \tag{10}$$

where $t$ denotes the time index. Throughout this paper, the character $\chi$ denotes the estimate of the pose $\mathbf{x}$, with the vanilla, tilded, and barred characters also have the same meaning as described in Section III. In particular, $\boldsymbol{\chi}$, $\tilde{\boldsymbol{\chi}}$ and $\bar{\boldsymbol{\chi}}$ are the estimates of $\mathbf{x}$, $\tilde{\mathbf{x}}$ and $\bar{\mathbf{x}}$, respectively.

### 2) PARTICLE INITIALIZATION

To apply the MCL, a set of initial particles defined as

$$S_0 = \left\{ \left(\boldsymbol{\chi}_0^{(j)}, w_0^{(j)}\right) \mid \boldsymbol{\chi}_0^{(j)} \in \mathrm{SE}(3), \right. \\ \left. w_0^{(j)} \in \mathbb{R}^+, j = 1, \cdots, N_p \right\} \quad (11)$$

should be generated, where $N_P$ denotes the number of particles; the superscript $(j)$ denotes a $j$–th particle, and $w_0^{(j)}$ denotes the associated weight of $\boldsymbol{\chi}_0^{(j)}$. To use the result from the pose regression in (10), a set of particles is generated around the output $\tilde{\boldsymbol{\chi}}_0^{Camera} \in \mathbb{R}^7$ from the pose regression network. Let us suppose that the initial estimate of the camera pose $\tilde{\boldsymbol{\chi}}_0^{Camera}$ is given and it can be converted into $\boldsymbol{\chi}_0^{Camera} \in \mathrm{SE}(3)$ as

$$\boldsymbol{\chi}_0^{Camera} = M_{quat\rightarrow \mathrm{SE}(3)}\left(\tilde{\boldsymbol{\chi}}_0^{Camera}\right), \quad (12)$$

where the definition of the mapping $M_{quat\rightarrow \mathrm{SE}(3)}(\cdot)$ can be found in [36]. Then, the initial estimate of the LiDAR pose can be expressed in SE(3) as $\boldsymbol{\chi}_0 = \boldsymbol{\chi}_0^{Camera} \oplus \mathbf{T}_{CL}$.

To generate initial particles around $\boldsymbol{\chi}_0$, some noise samples $\boldsymbol{\xi}_0^{(j)}$ should be drawn from the Gaussian distribution and a new particle set should be generated as $\boldsymbol{\chi}_0^{(j)} = \boldsymbol{\chi}_0 + \boldsymbol{\xi}_0^{(j)}$. Unfortunately, however, it is hard to sample $\boldsymbol{\xi}_0^{(j)}$ from the Gaussian distribution such that $\boldsymbol{\xi}_0^{(j)}$ belongs to SE(3). Even when $\boldsymbol{\xi}_0^{(j)} \in \mathrm{SE}(3)$, $\boldsymbol{\chi}_0^{(j)} = \boldsymbol{\chi}_0 + \boldsymbol{\xi}_0^{(j)} \notin \mathrm{SE}(3)$, since SE(3) is not a vector space but belongs to the smooth manifold. The key idea of this paper is to sample noise not from a physical space SE(3) but from the space of twist coordinates, which is a simple vector space. First, the initial estimate $\boldsymbol{\chi}_0$ is converted into the twist coordinates $\bar{\boldsymbol{\chi}}_0$ using (4) as

$$\bar{\boldsymbol{\chi}}_0 = \left\{\log\left(\boldsymbol{\chi}_0\right)\right\}^{\vee} \quad (13)$$

and particles are generated by sampling $\bar{\boldsymbol{\xi}}_0^{(j)}$ from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\delta)$ and added to the initial guess $\bar{\boldsymbol{\chi}}_0$ as $\bar{\boldsymbol{\chi}}_0^{(j)} = \bar{\boldsymbol{\chi}}_0 + \bar{\boldsymbol{\xi}}_0^{(j)}$, where $\boldsymbol{\Sigma}_\delta$ is a $6 \times 6$ covariance matrix, and $\bar{\boldsymbol{\chi}}_0^{(j)} \in \mathbb{R}^6$, obviously. Then, the initial particle set can be generated by mapping back to SE(3) using (5),(6) as

$$\begin{aligned} \boldsymbol{\chi}_0^{(j)} &= \exp\left\{\left(\bar{\boldsymbol{\chi}}_0 + \bar{\boldsymbol{\xi}}_0^{(j)}\right)^{\wedge}\right\} = \exp\left(\hat{\boldsymbol{\chi}}_0 + \hat{\boldsymbol{\xi}}_0^{(j)}\right) \\ &= \exp\left(\hat{\boldsymbol{\chi}}_0\right)\exp\left(\hat{\boldsymbol{\xi}}_0^{(j)}\right) \\ &= \boldsymbol{\chi}_0 \exp\left(\hat{\boldsymbol{\xi}}_0^{(j)}\right), \quad j = 1, \cdots, N_P \end{aligned} \quad (14)$$

Now, $S_0$ can be obtained, where the associated weights are set to $w_0^{(j)} = 1/N_p$. This noise sampling procedure is illustrated in Fig. 4. $\bar{\boldsymbol{\xi}}_0^{(j)}$ is sampled from the blue Gaussian distribution in the twist coordinates and converted to the corresponding a red twist $\hat{\boldsymbol{\xi}}_0^{(j)}$ in $\mathfrak{se}(3)$. The initial particles are generated by multiplying $\exp\left(\hat{\boldsymbol{\xi}}_0^{(j)}\right)$ to the initial estimate $\boldsymbol{\chi}_0$ on the manifold structure of SE(3). The relationship between $\boldsymbol{\chi}_0$, $\boldsymbol{\chi}_0^{Camera}$, and $\boldsymbol{\chi}_0^{(j)}$ is summarized in Fig. 5(a).

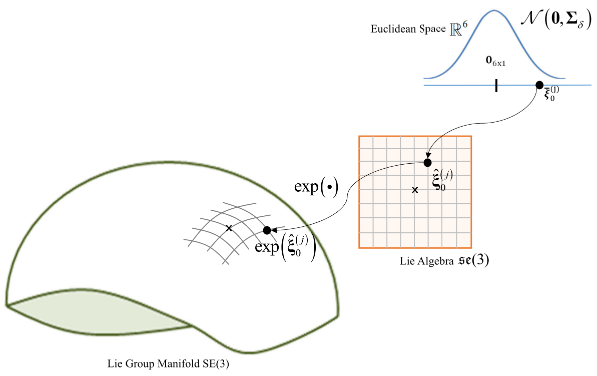
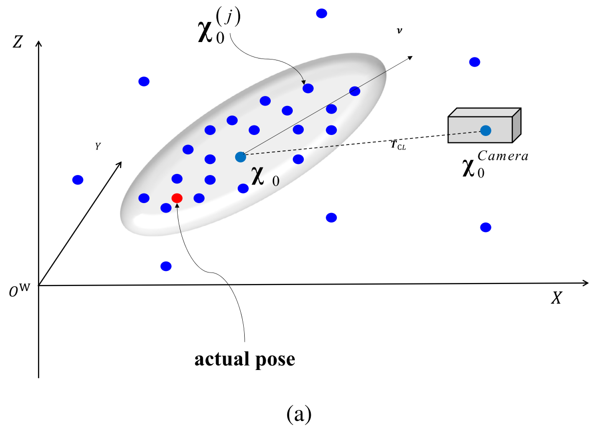

Euclidean Space $\mathbb{R}^6$
$\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\delta)$
$\mathbf{0}_{6\times1}$
$\bar{\xi}_0^{(j)}$
$\exp(\bullet)$
$\hat{\xi}_0^{(j)}$
$\exp(\hat{\xi}_0^{(j)})$
Lie Algebra $\mathfrak{se}(3)$
Lie Group Manifold SE(3)

**FIGURE 4.** Illustration of sampling noise from the Euclidean space to the manifold structure of SE(3).



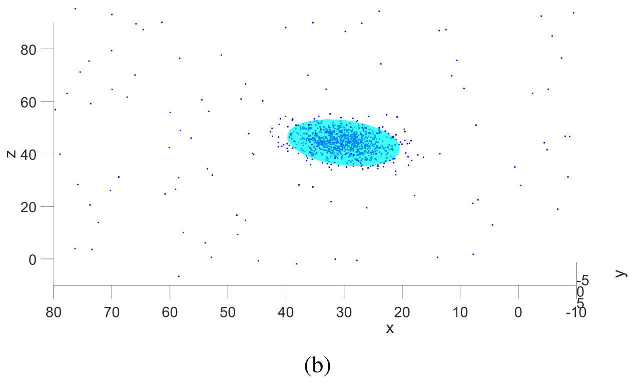


**FIGURE 5.** (a) An illustration of the proposed particle spreading method. The blue dots represent sampled particles and a red dot represents the actual pose. (b) An example of initial particles.

To determine the covariance $\boldsymbol{\Sigma}_\delta$, from a set of training samples $\mathcal{T} = \left\{\left(I, \tilde{\mathbf{x}}_I\right) \mid I \in \mathcal{I},\ \tilde{\mathbf{x}}_I = [\mathbf{p}_I,\ \mathbf{q}_I]^T\right\}$, we obtain the CNN output $\tilde{\boldsymbol{\chi}}^{CNN}(I) = \left[\mathbf{p}^{CNN}(I),\ \mathbf{q}^{CNN}(I)\right]^T$

and compute the *training* sample covariance in the twist coordinates by

$$\begin{aligned}\boldsymbol{\Sigma}_{\upsilon}^{training} &= \frac{1}{|\mathcal{I}|}\sum_{I\in\mathcal{I}}\left(\boldsymbol{\upsilon}^{CNN}(I)-\boldsymbol{\upsilon}_I\right)\left(\boldsymbol{\upsilon}^{CNN}(I)-\boldsymbol{\upsilon}_I\right)^T\\ \boldsymbol{\Sigma}_{\omega}^{training} &= \frac{1}{|\mathcal{I}|}\sum_{I\in\mathcal{I}}\left(\boldsymbol{\omega}^{CNN}(I)-\boldsymbol{\omega}_I\right)\left(\boldsymbol{\omega}^{CNN}(I)-\boldsymbol{\omega}_I\right)^T\end{aligned} \tag{15}$$

where

$$\bar{\boldsymbol{\chi}}^{CNN}(I) = \begin{bmatrix}\boldsymbol{\upsilon}^{CNN}(I)\\ \boldsymbol{\omega}^{CNN}(I)\end{bmatrix} \tag{16}$$

$$= M_{quat\to twist}\left(\begin{bmatrix}\mathbf{p}^{CNN}(I)\\ \mathbf{q}^{CNN}(I)\end{bmatrix}\right),$$

$$\bar{\mathbf{x}}_I = \begin{bmatrix}\boldsymbol{\upsilon}_I\\ \boldsymbol{\omega}_I\end{bmatrix} = M_{quat\to twist}\left(\begin{bmatrix}\mathbf{p}_I\\ \mathbf{q}_I\end{bmatrix}\right). \tag{17}$$

As stated in Section IV.A1., $\bar{\boldsymbol{\chi}}$ is the estimate of $\bar{\mathbf{x}}$. The mapping $M_{quat\to twist}(\cdot)$ can be found in [36]. Since the uncertainty in the test phase is larger than that of the training phase, we increase $\boldsymbol{\Sigma}_{\upsilon}$ and $\boldsymbol{\Sigma}_{\omega}$ from $\boldsymbol{\Sigma}_{\upsilon}^{training}$ and $\boldsymbol{\Sigma}_{\omega}^{training}$ by $k$ times, respectively. That is, $\boldsymbol{\Sigma}_{\upsilon} = k \times \boldsymbol{\Sigma}_{\upsilon}^{training}$ and $\boldsymbol{\Sigma}_{\omega} = k \times \boldsymbol{\Sigma}_{\omega}^{training}$. Through the experiments, we set $k$ to 2.

The proposed MCL combined with deep initialization is not a vanilla MCL but a Random Particle MCL in [7]. Also, the proposed MCL generates particles not by sampling from a Gaussian prior centered on the regression network output but by sampling around 10% of the particles from a uniform distribution over the whole environment, as shown in Fig. 5. Thus, the propose MCL is always ready for coping with the ambiguity from the regression network. An example of initial particles is presented in Fig. 5(b).

## B. PARTICLE FILTER

### 1) PARTICLE FILTER USING 3D LiDAR

Using the initial set of particles from deep initialization, a particle filter using LiDAR measurements is applied to estimate the sensor pose. Let us suppose that a set of particles at time $t-1$

$$S_{t-1} = \left\{\left(\boldsymbol{\chi}_{t-1}^{(j)}, w_{t-1}^{(j)}\right) | \boldsymbol{\chi}_{t-1}^{(j)} \in SE(3),\; w_{t-1}^{(j)} \in \mathbb{R}^+, j=1,\cdots,N_p\right\} \tag{18}$$

are given, where $\mathbb{R}^+$ denotes a set of non-negative reals.

First, $j$-th particle at time $t$ is sampled from the motion model as

$$\boldsymbol{\chi}_t^{(j)} = \left(\boldsymbol{\chi}_{t-1}^{(j)} \oplus \mathbf{u}_t\right) \oplus \exp\left(\hat{\boldsymbol{\delta}}\right), \tag{19}$$

where $\mathbf{u}_t \in SE(3)$ denotes the pose difference obtained using wheel odometry and $\boldsymbol{\delta} \in \mathbb{R}^6$ [39]. Second, a set of particles $S_t$ are weighted using the measurements from the LiDAR. Let us suppose that a set of measurements $\mathbf{Y}_{L,t} = \left\{\mathbf{y}_t^k \in \mathbb{R}^3 | k=1,\cdots,N_{scan}\right\}$ are received from the LiDAR at time $t$ and a point cloud map $\mathbf{m}$ is given as (3). For as $j$–th particle $\boldsymbol{\chi}_t^{(j)}$, the measurement likelihood is defined as

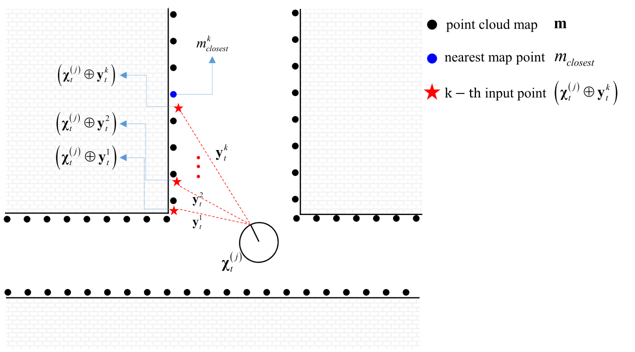

$m_{closest}^k$
$\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k\right)$
$\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^2\right)$
$\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^1\right)$
$\mathbf{y}_t^k$
$\mathbf{y}_t^2$
$\mathbf{y}_t^1$
$\boldsymbol{\chi}_t^{(j)}$
● point cloud map $\mathbf{m}$
● nearest map point $m_{closest}$
★ k – th input point $\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k\right)$

**FIGURE 6. Illustration of the observation likelihood. The k-th input LiDAR point denoted as a red star is transformed to the global coordinates with respect to the j-th particle pose. The measurement likelihood is calculated using the distances between the LiDAR point and the closest map point represented with the blue dot.**

$$p\left(\mathbf{Y}_{L,t} \middle| \boldsymbol{\chi}_t^{(j)}, \mathbf{m}\right) = \prod_{k=1}^{N_{scan}} \exp\left\{-\frac{\left\|\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k\right) - m_{closest}^k\right\|}{2\sigma^2}\right\} \tag{20}$$

where $\sigma$ denotes the standard deviation of the LiDAR sensor noise. The likelihood $p\left(\mathbf{Y}_{L,t} \middle| \boldsymbol{\chi}_t^{(j)}, \mathbf{m}\right)$ is illustrated in Fig. 6, where $m_{closest}^k = \arg\min_{m\in\mathbf{m}} \left\|\left(\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k\right) - m\right\|$. That is, $\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k$ marked with the red stars indicate the positions of scanned points $\mathbf{y}_t^k$ with respect to the pose of a $j$–th particle. Furthermore, $m_{closest}^k$ marked with the blue dot denotes a map point closest to the point $\boldsymbol{\chi}_t^{(j)} \oplus \mathbf{y}_t^k$. The likelihood in (20) implies how much a $j$–th particle $\boldsymbol{\chi}_t^{(j)}$ respects the current LiDAR measurement $\mathbf{Y}_{L,t}$, and each particle is weighted using the likelihood as

$$w_t^{(j)} \propto p\left(\mathbf{Y}_{L,t} \middle| \boldsymbol{\chi}_t^{(j)}, \mathbf{m}\right) \cdot w_{t-1}^{(j)}. \tag{21}$$

Then, we can obtain a new set of particles $S_t = \left\{\left(\boldsymbol{\chi}_t^{(j)}, w_t^{(j)}\right) | j=1,\cdots,N_p\right\}$, where $\boldsymbol{\chi}_t^{(j)}$ and $w_t^{(j)}$ are given in (19) and (21), respectively. When particle degeneracy occurs and the effective number of particles $N_{eff} = \left[\sum_{j=1}^{N_P}\left(w_t^{(j)}\right)^2\right]^{-1}$ becomes less than $N_p/2$, resampling is performed to replace low weight particles with high weight particles. When a set of particles $S_t$ at time $t$ is given and $\boldsymbol{\chi}_t^{(j)} = \begin{bmatrix}\mathbf{R}_{\chi,t}^{(j)} & \mathbf{p}_{\chi,t}^{(j)}\\ \mathbf{0} & 1\end{bmatrix}$, the estimate of the LiDAR pose can be expressed as

$$\boldsymbol{\chi}_t = \begin{bmatrix}\mathbf{R}_{\chi,t} & \mathbf{p}_{\chi,t}\\ \mathbf{0} & 1\end{bmatrix}, \tag{22}$$

where

$$\mathbf{p}_{\chi,t} = \frac{\sum_j w_t^{(j)} \mathbf{p}_{\chi,t}^{(j)}}{\sum_j w_t^{(j)}}$$

$$\mathbf{R}_{\chi,t} = \underset{\mathbf{R}\in\mathrm{SO}(3)}{\arg\min} \sum_{j=1}^{N_p} w_t^{(j)} \left\| \log\left(\mathbf{R}^T \mathbf{R}_{\chi,t}^{(j)}\right)\right\|_F^2. \tag{23}$$

$\|\cdot\|_F$ denotes the *Frobenius* norm and $\mathbf{R}_{\chi,t}$ denotes *weighted geometric mean* of the set of rotation matrices, which is one of the rotation averaging method presented in [42].

### 2) RE-LOCALIZATION

When the LiDAR pose is tracked by applying the PF, the MCL method sometimes loses the track of the LiDAR pose due to the kidnapped situation or localization failure. In this case, re-localization is performed to reset the tracking of the LiDAR. Re-localization consists of two steps: (1) detecting localization failure, and (2) re-initializing particles as described in Section IV-A2. Since our system is equipped with a camera, we can determine in the first step whether re-localization should be performed at time $t$ by comparing $\boldsymbol{\chi}_t \in \mathrm{SE}(3)$ and $\boldsymbol{\chi}_t^{Net} \in \mathrm{SE}(3)$, where $\boldsymbol{\chi}_t$ is given in (22) and represents the estimate of the LiDAR pose using MCL, while

$$\boldsymbol{\chi}_t^{Net} = M_{quat\rightarrow \mathrm{SE}(3)}\left(\tilde{\chi}_t^{Camera}\right) \oplus \mathbf{T}_{CL} \tag{24}$$

is the estimate of the LiDAR pose using only pose regression network, where $\tilde{\chi}_t^{Camera}$ is given in (10). If the difference between $\boldsymbol{\chi}_t$ and $\boldsymbol{\chi}_t^{Net}$ continues to be larger than a threshold for a period of time, the estimate $\boldsymbol{\chi}_t$ is assumed to be incorrect and re-initialization is performed. That is, when $\boldsymbol{\chi}_t = \begin{bmatrix} \mathbf{R}_{\chi,t} & \mathbf{p}_{\chi,t} \\ \mathbf{0} & 1 \end{bmatrix}$ and $\boldsymbol{\chi}_t^{Net} = \begin{bmatrix} \mathbf{R}_t^{Net} & \mathbf{p}_t^{Net} \\ \mathbf{0} & 1 \end{bmatrix}$ are given at time $t$, the distances of position and orientation between two poses can be defined as

$$\begin{aligned} d_p &= \left\|\mathbf{p}_{\chi,t} - \mathbf{p}_t^{Net}\right\| \\ d_R &= \frac{1}{\sqrt{2}} \left\| \log\left(\mathbf{R}_{\chi,t}{}^T \mathbf{R}_t^{Net}\right)\right\|_F^2, \end{aligned} \tag{25}$$

respectively. The position distance is the *Euclidean* norm of the position vector's difference, and the distance between the two orientations represented by the rotation matrix uses the geodesic distance used in [43]. Finally, the instantaneous localization failure is determined as follows:

$$O_t = \begin{cases} 1, & \mathit{if}\ \left(d_p > \eta_p\right)\ \mathit{and}\ \left(d_R > \eta_R\right) \\ 0, & \mathit{otherwise} \end{cases} \tag{26}$$

When localization fails, the failure indicator $O_t$ keeps occurring for consecutive $\tau$ steps. Then, particle initialization described in Section IV-A2. is applied, which is actually the combination of deep initialization and MCL. The overall process of re-localization is summarized in Algorithm 1.

**Algorithm 1** Re-Localization

```
Require: $\tilde{\chi}_t^{Camera}$, $S_t$
1:  $\boldsymbol{\chi}_t^{Net}$ ← LiDAR pose using $\tilde{\chi}_t^{Camera}$
2:  $\boldsymbol{\chi}_t$ ← Weighted mean of $S_t$
3:  $d_p, d_R$ ← Distance between two poses $\boldsymbol{\chi}_t^{Net}, \boldsymbol{\chi}_t$
4:  $O_t$ ← Calculate localization failure indicator
5:  if $\sum_{i=t-\tau}^{t} O_i \geq \tau$ then
6:      $\breve{S}_t$ ← Particle initialization :
7:      $\left\{\boldsymbol{\chi}_t^{(j)}\right\}_{j=1:N_P} = \boldsymbol{\chi}_t^{Net} \exp\left(\hat{\xi}_0^{(j)}\right)$
8:      $\left\{w_t^{(j)}\right\}_{j=1:N_P} = 1/N_P$
9:  end if
10: return $\breve{S}_t = \left\{\left(\boldsymbol{\chi}_t^{(j)}, w_t^{(j)}\right) \mid j = 1, \cdots, N_p\right\}$
```

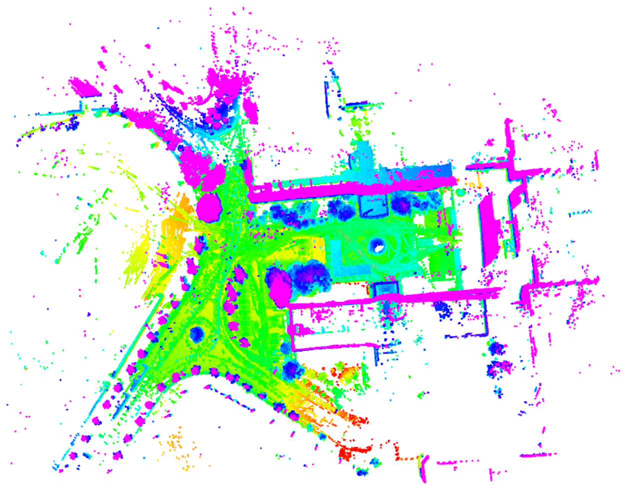
## V. EXPERIMENTAL RESULTS

### A. EXPERIMENTAL SETUP

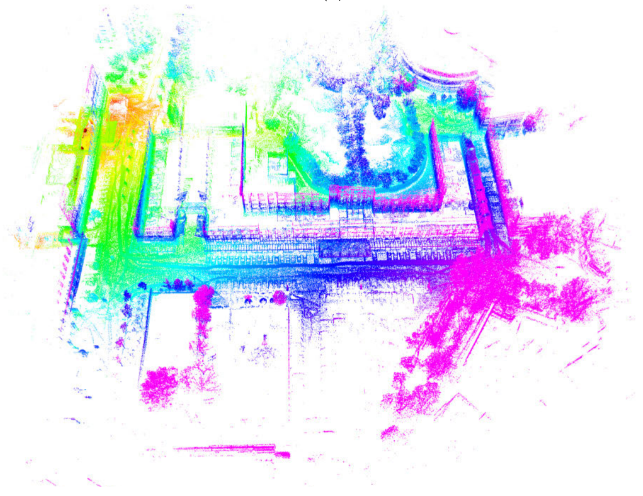
We applied the proposed method to global localization in two urban locations: the Engineering Building (*YU_ENGR*) at Yonsei University and the Innovation Building (*KU_INNO*) at Korea University, Korea. The two buildings are very populated. The experiment was conducted using the presented mobile robot platform (Fig. 1), and the robot navigated back and forth between the indoor and outdoor environments of the two buildings. The sizes of *YU_ENGR* and *KU_INNO* are $70m \times 80m$ and $200m \times 100m$, respectively, and the total lengths of the trajectory navigated by the robot in the two locations are 732.47 m and 624.75 m, respectively. Average speed of the platform is 0.75 m/s and 0.58 m/s for *YU_ENGR* and *KU_INNO*, respectively. Thus, the duration time of each tested trajectory is 968 seconds and 1081 seconds. The Velodyne VLP-16 sensor output approximately 300,000 points per second and measured up to 100 m. The ZED camera in front of the robot has the resolution of $1280 \times 720$ at 30 fps. All experiments were conducted on a laptop computer with 32GB of RAM, an Intel i7-7700HQ processor, and a NVIDIA GeForce GTX 1070 GPU. The system operated under the full robot operating system (ROS) package interface (version Kinetic). Using the ROS interface, we can easily publish the sensor pose into the ROS interface.

To evaluate the performance of the proposed global localization method, the following four metrics were used as defined in [1]: the correct localization rate (CLR), false localization rate (FLR), localization failure rate (LFR), and first correct localization time (FCLT). The CLR is the ratio of the number of time steps of successful robot localization over the total number of time steps. The value of CLR equal to 100% indicates perfect localization over the entire operation time. The FLR is the ratio of the number of time steps that a robot has localized at a wrong location over the total number of time steps, while the LFR is the ratio of the number of time steps of the algorithm failing to localize the robot over the total number of time steps. The CLR, FLR, and LFR should sum to 100%. Finally, FCLT indicates the time step when the robot is first localized at the correct location.

As stated before, a 3D map of the environment is required for the proposed method to perform global localization. A 3D map was built in advance using off-line graph optimization, and pose regression network was also trained off-line together with map building. The sensor poses were obtained by optimizing the pose graph using the visual SLAM [23] and LiDAR odometry [38]. After registering the LiDAR points with respect to the poses obtained using graph optimization, a voxel grid filter was applied to the LiDAR points to reduce the total number of map points. The leaf size of the voxel grid filter was set to 0.1 m $\times$ 0.1 m $\times$ 0.1 m. The resulting map $\mathbf{m}$ (3) consisted of 3D point clouds. The maps for *YU_ENGR* and *KU_INNO* are shown in Figs. 7(a) and 7(b), respectively.



(a)



(b)

**FIGURE 7. Pre-built 3D maps of (a) *YU_ENGR* (b) *KU_INNO* environments.**

## B. CNN-BASED POSE REGRESSION

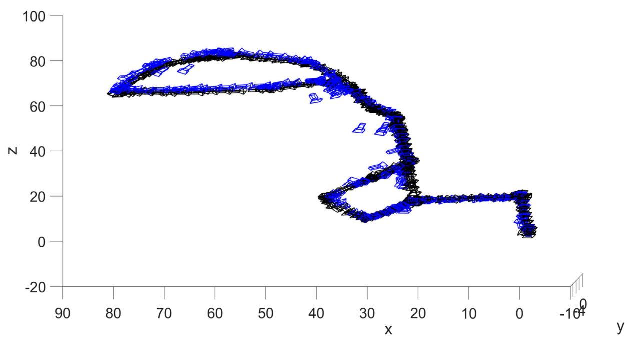
To implement the proposed pose regression in the ROS environment, the pose regression network was wrapped into a

(a)

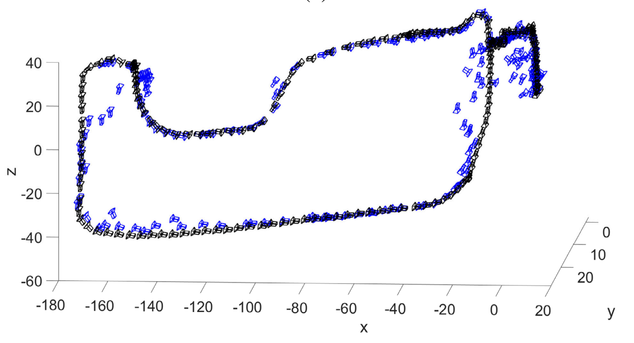
(b)

**FIGURE 8. Example images for training pose regression network. (a) *YU_ENGR* (b) *KU_INNO* environments.**

ROS package. The pose regression network was trained using 9,279 images of *YU_ENGR* and 10,328 images of *KU_INNO*. Example images for training are shown in Fig 8. Since the experiment environments encompassed both indoor and outdoor scenes, the $\beta$ in the loss function (9) was set to 500, as recommended in [24]. The evaluation of pose regression took about 5.9 msec for each test image. The root-mean-squared errors (RMSE) for translation and orientation in *YU_ENGR* were 2.2352 m and 4.6775 deg, respectively, while the RMSE for translation and orientation in *KU_INNO* were 4.6727 m and 3.7919 deg, respectively.



(a)



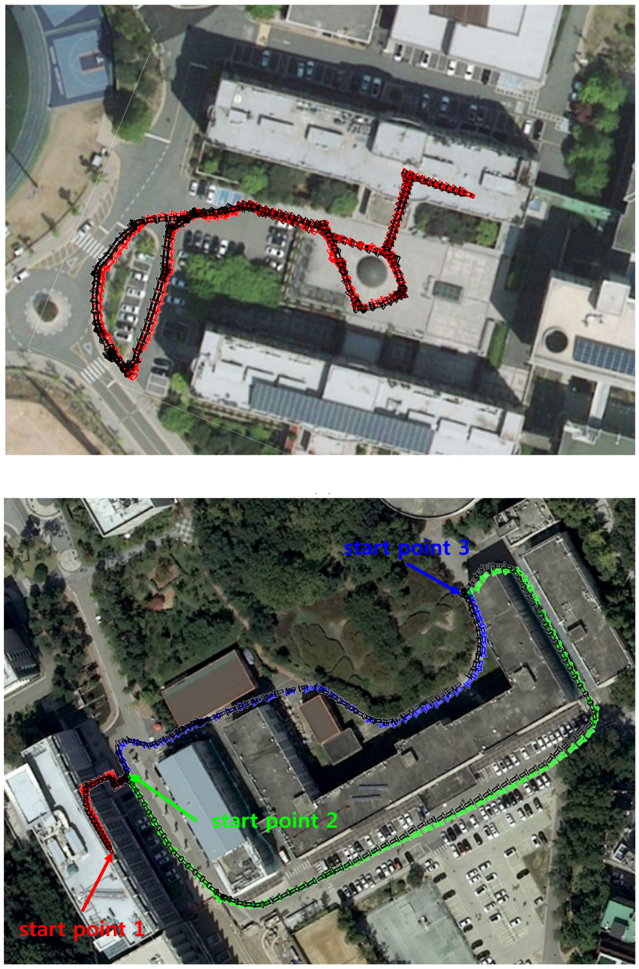
(b)

**FIGURE 9. Image-based pose regression results of (a) *YU_ENGR* (b) *KU_INNO*. The black and blue camera symbols indicate ground truth poses of test images and estimated poses, respectively.**

The pose regression results are shown in Fig. 9, where the black and blue arrows indicate the true and estimated poses of the LiDAR sensor, respectively. In both experiments, the robot platform began to move inside the respective building and went out of it. Global localization in *KU_INNO* was more difficult than that in *YU_ENGR* since *KU_INNO* was more spacious and had more repeating patterns such as buildings compared to *YU_ENGR*. Thus, the accuracy was higher for *KU_INNO* than that for *YU_ENGR*. It can be noticed from Fig. 9 that the pose regression results were not good enough and they were used as an initial estimate for MCL.

### C. MCL WITH DEEP INITIALIZATION

The global localization results of the proposed method are superimposed on the aerial image of the experimental environments in Fig. 10. Figs. 10(a) and 10(b) show the experimental results for *YU_ENGR* and *KU_INNO*, respectively. In the *YU_ENGR* experiment, the mobile robot platform started to move inside the building. In Fig. 10(a), the black and red arrows indicate the true and estimated poses, respectively. In the *YU_ENGR*, start points were randomly selected and the results are average value of 20 repeated experiments. In the *KU_INNO*, the trajectories with three different fixed start points were tested and indicated in red, green, and blue. For clear visualization, Fig. 10(a) depicts only one trajectory of *YU_ENGR*, whereas Fig. 10(b) depicts three different trajectories of *KU_INNO*.



(b)

**FIGURE 10. Final experimental results of the proposed method for (a) *YU_ENGR* (b) *KU_INNO*. The black and red camera symbols indicate the trajectory of the ground truth and that of our method, respectively. In *KU_INNO* experiments, the entire path was divided into three sections as represented by the red, green and blue symbols.**

The proposed method was applied to the two environments and the results were compared to that achieved by previously proposed methods, namely, 2D-MCL [44], 3D-MCL [14] and camera-only PF method [31], in terms of the RMSE, CLR, FLR, LFR, and FCLT. In this experiment, the thresholds $\eta_p$ and $\eta_R$ defined by (26) were set to 3 m and 1.5 deg, respectively, while the time threshold $\tau$ was set to 10 seconds (the latter was used in the re-localization indicator as described in Section IV-B2. This implies that if the translational and rotational differences between $\boldsymbol{\chi}_t^{Net}$ and $\boldsymbol{\chi}_t$ continued to be larger than 3 m and 1.5 deg, respectively, for more than 10 seconds, the current localization is deemed as failed.

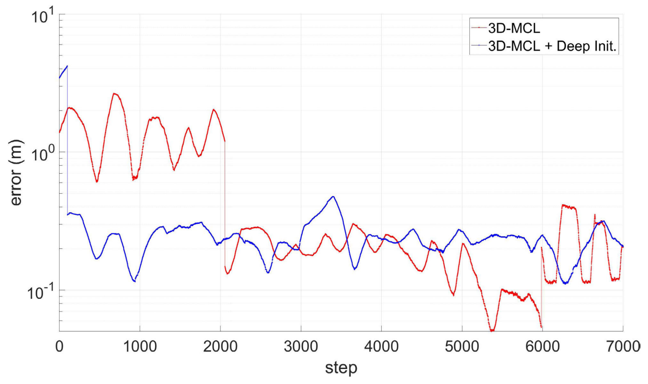
The experimental results for the two environments are summarized in Tables 1 and 2, respectively. In Table 1, the localization error is defined in terms of the RMSE. When measuring the localization error, the poses until the particles converged were not included in the error calculation for fair comparisons. In Table 1, we present localization accuracies achieved by the proposed and other considered methods over a series of datasets. The density of particles in the conventional 3D-MCL was set to 1.5 (particles/$m^2$), which is a minimum density recommended by [14] to ensure the convergence of particles. Thus, the number of particles was 8,400 for *YU_ENGR* and 30,000 for *KU_INNO*. In both experiments, the proposed method outperforms the previously reported global localization methods in terms of the RMSE.

**TABLE 1. Localization errors of the compared methods.**

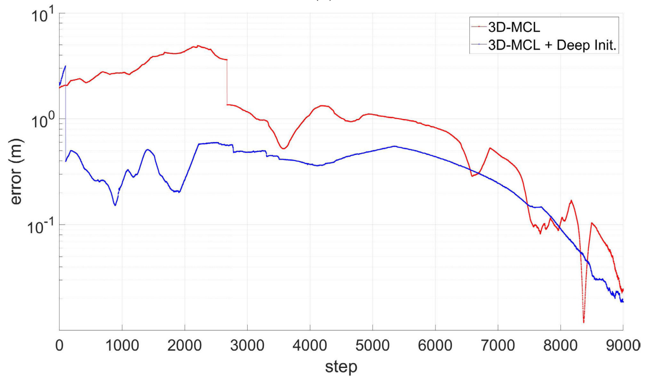
| Method | YU_ENGR | | KU_INNO | |
|---|---|---|---|---|
| | # particles | Error (m) | # particles | Error (m) |
| Camera-PF | 2,000 | 1.89 | 2,000 | 3.25 |
| 2D-MCL | 2,000 | 7.45 | 2,000 | 10.01 |
| 2D-MCL + Deep Init. | 500 | 1.54 | 500 | 2.64 |
| 3D-MCL | 8,400 | 1.02 | 30,000 | 2.15 |
| 3D-MCL + Deep Init. | 500 | **0.56** | 500 | **0.88** |

**TABLE 2. Performance results of the compared methods.**

| Map | Method | CLR (%) | FLR (%) | LFR (%) | FCLT (step) |
|---|---|---|---|---|---|
| YU_ENGR | 2D-MCL | 3.87 | 91.01 | 5.12 | 1585 |
| | 3D-MCL | 24.48 | 57.25 | 18.27 | 2057 |
| | Proposed ( [24]+MCL) | 90.60 | 3.22 | 6.18 | 105 |
| | Proposed ( [25]+MCL) | 91.10 | 3.12 | 5.78 | 101 |
| KU_INNO | 2D-MCL | 2.94 | 85.51 | 11.55 | 1606 |
| | 3D-MCL | 13.54 | 65.57 | 20.89 | 2677 |
| | Proposed ( [24]+MCL) | 87.54 | 4.59 | 7.87 | 154 |
| | Proposed ( [25]+MCL) | 88.51 | 4.62 | 6.87 | 138 |

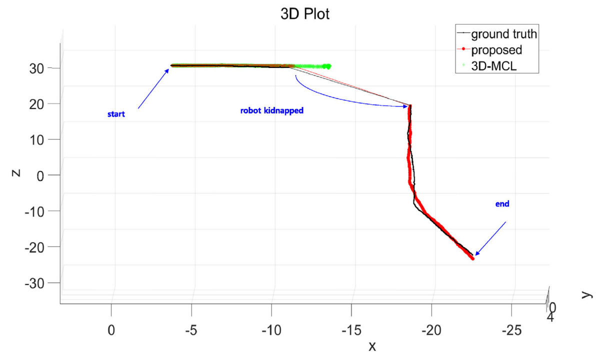


(a)

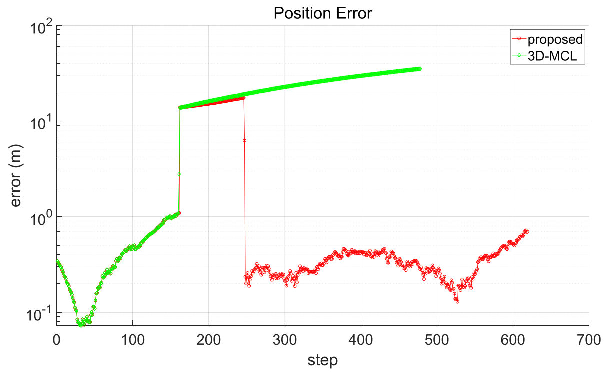


(b)

**FIGURE 11. Localization errors of 3D-MCL (red) and 3D-MCL with deep initialization (blue) (a) *YU_ENGR*, (b) *KU_INNO*.**



(a)



(b)

**FIGURE 12. Experimental results of re-localization algorithm. (a) 3D trajectory of the kidnapped robot scenario. The black, green, and red lines indicate the estimates of ground truth, 3D-MCL, and proposed method, respectively. (b) position errors.**

Obviously, the previous 2D and 3D MCL should explore the whole space and thus the localization error remains high during the initial time of the operation. The proposed MCL, however, immediately focuses on the region around the correct location and the increase in the localization error is very limited during the initial time of the operation. To show this, the error graph of each experiment is given in Fig. 11. As can be seen from the figure, the proposed method shows that the error decreases very quickly after the beginning of the operation. It should also be noted that 8,400 and 30,000 particles are used in 3D-MCL, whereas 500 particles are used in the proposed MCL.

The results achieved by the compared methods in terms of CLR, FLR, LFR, and FCLT are listed in Table 2. It can be noticed from the table that the FLR values are high for the previously reported methods indicating their poor performance, whereas the FLR of the proposed method is almost zero. The compared methods estimated wrong poses in the most of steps. The excellent performance of the proposed method can be explained by its reliable convergence to the true pose regardless of the initial pose. The FLR of 2D-MCL is very high, since this method cannot estimate the pose correctly and converges to the wrong pose. 3D-MCL has a relatively high FCLT, which means that this method takes a long time to find a correct pose for the first time.

The main contribution of our method is to initialize 6-DOF pose using a Gaussian prior around the visual localization result in twist coordinates. Therefore, our method can be applied to (or be combined with) any other visual localization methods. The more accurate visual localization method is used, the better performance is obtained as shown in Table 2.

When there is high ambiguity in the pose from the regression network, it is possible that the MCL with deep initialization works worse than the one with the simple initialization by uniform prior. But it is rarely the case when it is applied to real-time applications with huge environment. Let us consider the MCL applied to *YU_ENGR* with the area size of 70 m × 80 m. It is admitted that it is difficult to run more than 2000 particles for real-time applications. When a simple initialization by uniform prior is adopted and 2000 particles is used, the MCL easily falls into particle impoverishment and fail in the localization task. In 3D-MCL, 2000 particles are not enough and 8,400 and 30,000 particles are used based on the guideline in [14]. On the other hand, when the MCL with deep initialization is adopted, 500 particles are enough to converge to the correct pose. As shown in Tables 1 and 2, the proposed deep initialization with 500 particles shows better performance than initialization by uniform prior in terms of both convergence time and localization accuracy. The vanilla

MCLs such as 2D-MCL and 3D-MCL do not work in the environment with the size of 70 m × 80 m (*YU_ENGR*), since the environment is too spacious to explore using a simple uniform prior. However, when the proposed deep initialization is used, the MCL starts with a good initial guess and it explains the large variation in Table 2.

The FCLT includes the time of pose regression by PoseNet and the operating time of particle filter. PoseNet is trained off-line together with map building. When 500 particles are used, the runtime for particle filter is 95 milliseconds (i.e. 10.53 update/sec) on average. Thus, if the LiDAR speed is less than 10 Hz, the localization filter is fast enough to handle all the sensor measurements.

### D. RE-LOCALIZATION

To show the effectiveness of the proposed localization failure detection and re-localization, the kidnapped robot problem is considered in this subsection. The problem is illustrated in Fig. 12(a). First, a mobile robot starts and travels approximately 10 m along a straight line. At the 160th step, the robot is forced ("kidnapped") to move to another location. Then, the robot moves again. The starting, ending, and kidnapped points are marked in Fig. 12(a). The proposed method and 3D-MCL [14] are applied to the scenario and the performance of recovering the pose is evaluated. In Fig. 12(a), the black, red, and green lines indicate the trajectory of the ground truth, proposed method, and 3D-MCL, respectively. As shown in the figure, the proposed method copes with the kidnapping well, whereas the 3D-MCL loses robot's trajectory and fails in recovering the localization. The position errors during the experiment are shown in Fig. 12(b).

## VI. CONCLUSION

In this paper, a new particle initialization method named deep initialization has been proposed and been applied to Monte Carlo localization (MCL). The proposed method is based on the combination of a 3D LiDAR and a camera. Pose regression was used to estimate the absolute 6-DOF pose in global coordinates from a single image using a pretrained CNN. To initialize a set of particles for MCL, particles were sampled from the tangent space of the manifold structure around the estimated pose using the CNN. A novel localization failure detection algorithm was further developed to overcome the kidnapped robot problem. Our experimental results demonstrated that the proposed method outperforms existing methods in terms of both the localization accuracy and time consumed to estimate pose correctly.

## REFERENCES

[1] H. Choi, K. Woong Yang, and E. Kim, "Simultaneous global localization and mapping," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 4, pp. 1160–1170, Aug. 2014.

[2] M. Sualeh and G.-W. Kim, "Simultaneous localization and mapping in the epoch of semantics: A survey," *Int. J. Control, Autom. Syst.*, vol. 17, no. 3, pp. 729–742, Mar. 2019.

[3] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. Robot., Sci. Syst.*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[4] H. Kim, S. Song, and H. Myung, "GP-ICP: Ground plane ICP for mobile robots," *IEEE Access*, vol. 7, pp. 76599–76610, 2019.

[5] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Oct. 2003, pp. 2743–2748.

[6] E. Javanmardi, M. Javanmardi, Y. Gu, and S. Kamijo, "Factors to evaluate capability of map for vehicle localization," *IEEE Access*, vol. 6, pp. 49850–49867, 2018.

[7] W. Burgard, D. Fox, and S. Thrun, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005, ch. 8.

[8] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 4372–4378.

[9] T. Rofer and M. Jungel, "Vision-based fast and reactive monte-carlo localization," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Sep. 2003, pp. 856–861.

[10] A. Y. Hata and D. F. Wolf, "Feature detection for vehicle localization in urban environments using a multilayer LIDAR," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 420–429, Feb. 2016.

[11] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte Carlo localization in outdoor terrains using multilevel surface maps," *J. Field Robot.*, vol. 25, nos. 6–7, pp. 346–359, Jun. 2008.

[12] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal distributions transform monte-carlo localization (NDT-MCL)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 382–389.

[13] H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung, "Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1518–1524, Jul. 2017.

[14] J. L. Blanco-Claraco, F. Mañas-Alvarez, J. L. Torres-Moreno, F. Rodriguez, and A. Gimenez-Fernandez, "Benchmarking particle filter algorithms for efficient velodyne-based vehicle localization," *Sensors*, vol. 19, no. 14, p. 3155, 2019.

[15] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor fusion-based low-cost vehicle localization system for complex urban environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1078–1086, May 2017.

[16] J. Castorena and S. Agarwal, "Ground-edge-based LIDAR localization without a reflectivity calibration for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 344–351, Jan. 2018.

[17] H. Lee, S. Kim, S. Park, Y. Jeong, H. Lee, and K. Yi, "AVM/LiDAR sensor based lane marking detection method for automated driving on complex urban roads," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1434–1439.

[18] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit, "Instant outdoor localization and SLAM initialization from 2.5D maps," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 11, pp. 1309–1318, Nov. 2015.

[19] G. Chowdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, "GPS-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *J. Field Robot.*, vol. 30, no. 3, pp. 415–438, May 2013.

[20] Z. Fang and S. Scherer, "Real-time onboard 6DoF localization of an indoor MAV in degraded visual environments using a RGB-D camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5253–5259.

[21] Z. Su, X. Zhou, T. Cheng, H. Zhang, B. Xu, and W. Chen, "Global localization of a mobile robot using lidar and visual features," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2017, pp. 2377–2383.

[22] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard, "Monocular camera localization in 3D LiDAR maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1926–1931.

[23] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[24] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2938–2946.

[25] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 3, Jul. 2017, p. 8

[26] A. Barrau and S. Bonnabel, "Stochastic observers on lie groups: A tutorial," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 1264–1269.

[27] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2174–2181.

[28] K. Huang and C. Stachniss, "Joint ego-motion estimation using a laser scanner and a monocular camera through relative orientation estimation and 1-DoF ICP," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 671–677.

[29] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 4762–4769.

[30] T. Naseer and W. Burgard, "Deep regression for monocular camera-based 6-DoF global localization in outdoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1525–1530.

[31] H. Jo, H. M. Cho, S. Lee, and E. Kim, "A deep convolutional neural network based 6-DOF relocalization with sensor fusion system," (in Korean), *J. Korea Robot. Soc.*, vol. 14, no. 2, pp. 87–93, May 2019.

[32] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo, "CamNet: Coarse-to-fine retrieval for camera re-localization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2871–2880.

[33] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, Jun. 2008.

[34] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of RGB camera with velodyne LiDAR," in *Proc. WSCG Commun. Papers*. Union Agency, vol. 2014, 2014, pp. 135–144. [Online]. Available: https://otik.uk.zcu.cz/handle/11025/26408

[35] P. Cheeseman, R. Smith, and M. Self, "A stochastic map for uncertain spatial relationships," in *Proc. 4th Int. Symp. Robotic Res.*, 1987, pp. 467–474.

[36] J.-L. Blanco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization," Univ. Malaga, Málaga, Spain, Tech. Rep., Sep. 2010, vol. 3.

[37] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "$G^2o$: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.

[38] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot., Sci. Syst.*, vol. 2, 2014, p. 9.

[39] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

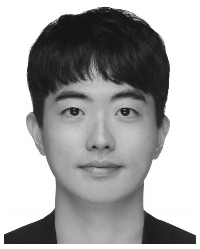
[40] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2012, vol. 26.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[42] M. Moakher, "Means and averaging in the group of rotations," *SIAM J. Matrix Anal. Appl.*, vol. 24, no. 1, pp. 1–16, Jan. 2002.

[43] S. Mahendran, H. Ali, and R. Vidal, "3D pose regression using convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 2174–2182.

[44] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.



**HYUNGGI JO** received the B.S. degree in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2012, where he is currently pursuing the Ph.D. degree in electrical and electronic engineering. His research interests include mobile robotics, sensor fusion, and three-dimensional simultaneous localization and mapping.

**EUNTAI KIM** (Member, IEEE) was born in Seoul, South Korea, in 1970. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from Yonsei University, Seoul, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a Full-Time Lecturer with the Department of Control and Instrumentation Engineering, Hankyong National University, Gyeonggi, South Korea. He was a Visiting Researcher with the Berkeley Initiative in Soft Computing, University of California, Berkeley, CA, USA, in 2008. Since 2002, he has been a Faculty Member with the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. His current research interests include computational intelligence, statistical machine learning, deep learning and their application to intelligent robotics, autonomous vehicles, and robot vision.