Highlighted Map for Mobile Robot Localization and Its Generation Based on Reinforcement Learning

RYOTA YOSHIMURA, ICHIRO MARUTA, KENJI FUJIMOTO, KEN SATO, AND YUSUKE KOBAYASHI

ABSTRACT This article proposes a new kind of map for mobile robot localization and its generation method. We call the map a highlighted map, on which uniquely shaped objects (landmarks) in monotonous environments are highlighted. By using this map, robots can use such landmarks as clues for localization, and thus, their localization performance can be improved without having to update their sensors or online computation. Furthermore, this map can be easily combined with many other existing localization algorithms. We formulate the problem of making a highlighted map and propose a numerical optimization method based on reinforcement learning. This optimization method automatically identifies and emphasizes the important landmarks on the map. The generated highlighted map is adapted to situations such as the sensor characteristics and robot dynamics because this method uses the actual sensor measurement data. It is proven that the optimization converges under certain technical assumptions. We performed a numerical simulation and real-world experiment showing that the highlighted map provides better localization accuracy than a conventional map.

INDEX TERMS Highlighted map, mobile robots, Monte Carlo localization, particle filters, reinforcement learning.

I. INTRODUCTION

Autonomous navigation technology has been applied to various robots, including automated guided vehicles (AGVs) and cars. For robots, estimation of their pose, namely localization, is essential.

Monte Carlo localization (MCL) is one of the most popular localization methods [1]–[3]. It is a particle-filter-based algorithm, which scatters a large number of particles (candidates for the pose estimate) onto the environment map and calculates the likelihood of each particle (the likelihood of the corresponding pose estimate) to estimate a probability distribution of the robot’s pose. Since MCL can handle complex multimodal probability distributions, it works robustly in various scenes. Thus, it has been widely studied and used, e.g., [4] improved MCL by adding the process of injecting random particles, and [5] proposed Kullback-Leibler divergence (KLD) sampling which dynamically adapts the number of particles. In [6], a pre-caching technique was applied to MCL for reduction of the computational burden and efficient sampling of particles. The method in [7] combined MCL with a laser scan matching algorithm to inherit the stability of MCL and the high accuracy of the scan matching. A particle initialization method by a convolutional neural network was proposed in [8]. The major target of MCL is a wheeled robot, but it has also been applied to an underwater robot [9], an unmanned aerial vehicle [10], and the field of wireless sensor networks [11].

As described above, many robots utilize MCL, however, MCL often fails in monotonous environments, e.g., a long straight corridor, a large vacant room, and a warehouse where the same shelves are equally spaced.

The details of MCL are described here in order to explain the problem. MCL includes a measurement model, which
gives the likelihood of a pose estimate by matching the map and measurements of the robot’s sensor, such as a laser range finder (LRF), ultrasonic sensor, or camera. The measurement model is often built based on a likelihood field (see, for instance [12]–[17]). A likelihood field is a map that considers the sensor’s measurement noise. Fig. 1 shows an example. Fig. 1(a) is the original map, and Fig. 1(b) is the corresponding likelihood field, which is generated by blurring the original map by the magnitude of the measurement noise. The map of Fig. 1(b) denotes that the darker a location is, the more likely it is to be observed by the sensor. Fig. 2 is an example demonstrating how the likelihood of a pose estimate is calculated from a likelihood field in the case of using an LRF sensor. Fig. 2(a) shows the LRF measurement data, and Fig. 2(b) and (c) show the projection of the measurement data onto the likelihood field at each pose estimate. The measurement data in Fig. 2(c), which are the red points, overlap the dark locations more than in Fig. 2(b); thus, the pose estimate of Fig. 2(c) is judged to be more likely.

The most common conventional method for localization in such environments is to use additional sensors having different characteristics from those of an LRF (e.g., use a GPS [18], [19] or a camera [20], [21]) and fuse these data. The sensor fusion method in [22] adopted the fault detection and isolation techniques to exclude the effects of faulty sensors. Another approach is localization based on a moving horizon estimation for the environments [23]. Methods that prevent a robot from entering such monotonous areas were presented in [24], [25].

In this article, we consider how to make full use of a few uniquely shaped objects in a monotonous environment (hereinafter referred to as landmarks) to increase the localization accuracy. Our approach is to generate a map on which the landmarks are emphasized and use the map for localization instead of a likelihood field. For example, if the landmarks are darkened, the likelihood of the pose estimate increases when the measurement overlaps the landmarks. As a result, the localization accuracy improves. Fig. 4 demonstrates the idea. The measurement in Fig. 4(a) overlaps only the light locations, whereas the measurement in Fig. 4(b) also overlaps the dark locations. Thus, the pose estimate of Fig. 4(b) is judged to be more likely than that of Fig. 4(a).
We call the map whose shade is modified for making a more accurate localization a highlighted map. The advantage of the highlighted map approach is that it requires no additional sensor or online computation for localization (only an offline pre-computation to generate a highlighted map is needed). Therefore, the performance of many robots, especially low-cost robots, can be improved at no additional cost. In addition, the highlighted map approach can be easily combined with other existing algorithms based on MCL.

Now let us consider how to generate a highlighted map. The simplest method is to find landmarks directly based on the shape of the map and modify their shade. However, it is difficult to automate the process of finding them, because there is no rule regarding the shapes of the landmarks. For example, there are several protrusions of the same shape in Fig. 1, but, as indicated in Fig. 5,

- The protrusion at the center of the upper corridor (in a long straight corridor) is a landmark.
- The protrusion at the upper right (in a corner) is not an important landmark compared to the above landmark, because it is easier to estimate pose in a corner than in a long straight corridor.
- The protrusions in the lower corridor are not landmarks whereas the place without protrusions is a landmark, because the most unique shape in that corridor is the flat area, not the protrusions.

Moreover, in such an approach, it is also difficult to make a map that reflects the sensor characteristics and robot dynamics, and to guarantee the optimality of localization.

This article presents the reverse approach, where landmarks are searched for from the localization results instead of searching for them directly according to the shape of the map. In this approach, the following two steps are repeated.

1) Estimate the pose with the map and evaluate the accuracy.
2) Modify the shade of the map based on the result of Step 1).

Here, we use reinforcement learning (RL), which has attracted much attention in various fields including robotics [26], [27], to optimize the shade of the map. Our idea is to regard the randomness of MCL (particle filter) as the randomness required for RL and to modify MCL to fit the RL framework. The features of this approach are as follows:

- Landmarks useful for localization can be automatically identified and emphasized.
- The map is optimized using the actual robot data (e.g., control data and measurement data). Nevertheless, unlike many other RL applications, we only need to drive the actual robot once.
- Since the optimization uses actual robot data, it is possible to obtain a highlighted map specialized for situations, such as the type and specifications of the sensor, the robot’s velocity, the robot’s route, and the MCL setting.

Note that our study is different from simultaneous localization and mapping (SLAM). SLAM is the problem of associating the measurement data at each position to construct a map of the environment [1]. The constructed map is generally used for path planning as well as localization. Various SLAM methods, such as EKF-SLAM [28], [29], particle-filter-based SLAM [30], [31], and graph-based SLAM [32], [33], have been studied, and their main topic has been loop closure, which is the technology used to create a consistent map by adjusting the positional relationships of these measurement data when the robot revisits a known area. On the other hand, this article considers modifying the shade of each location in the map based on the existing map (e.g., a map created by SLAM) for the purpose of improving localization. Therefore, the construction of a highlighted map is a process performed after SLAM.

To summarize, the main contributions of this article are as follows:

- Highlighted maps, on which uniquely shaped objects are highlighted, are newly proposed. The map can improve the accuracy of the MCL algorithm without additional sensor or online computation. Moreover, the idea can be combined with various versions of MCL.
- The method of generating a highlighted map based on RL is proposed.

This article is organized as follows. After explaining the MCL algorithm with the likelihood field, Section II formulates the problem of generating a highlighted map, and Section III proposes the solution for it. The numerical simulation described in Section IV and experiment described in
Section V show the effectiveness of using a highlighted map. Section VI is the conclusion.

Notation: Let \( \mathbb{R}, \mathbb{R}_+, \mathbb{R}_{0+}, \mathbb{N}, \mathbb{N}_0, \) and \( \mathbb{P}^m \) denote the real number field, the set of positive real numbers, the set of nonnegative real numbers, the set of natural numbers, the set of nonnegative integers, and the set of the \( n \times n \) positive semidefinite matrices, respectively. We use \( \emptyset \) to represent the empty set. For \( N \) data \( x(1), x(2), \ldots, x(N) \), the notation \( \{x(i)\}_{i=1}^{N} \) denotes the set of them, i.e., \( \{x(i)\}_{i=1}^{N} := \{x(1), x(2), \ldots, x(N)\} \). For time-varying data \( x_t, x_{t+1}, \ldots, x_{t+L} \) denotes sequences of data from time \( t_1 \) to time \( t_2 \), i.e., \( x_{t_1:t_2} := \{x_{t_1}, x_{t_1+1}, x_{t_1+2}, \ldots, x_{t_2}\} \). For a vector \( v \in \mathbb{R}^n \), \( \|v\| \) and \( \nabla_v f \) as follows: \( \|v\| := (v_1^2 + v_2^2 + \cdots + v_n^2)^{1/2} \), \( \nabla_v f := (\partial f/\partial v_1, \partial f/\partial v_2, \ldots, \partial f/\partial v_n)^T \).

II. PROBLEM FORMULATION

Although the highlighted map can be applied to various localization algorithms related to MCL, we consider the combination with the most basic MCL for simplicity. This section describes the MCL algorithm and formulates the problem of generating a highlighted map.

A. GRID MAP

A grid map is a grid array that discretizes the environment with a certain resolution. Each grid cell has a value. Let the map be denoted by \( m := (m(1), m(2), \ldots, m(M)^T) \), where \( M \in \mathbb{N} \) and \( m(\mu) \in \mathbb{R} \) are the number of grid cells and the value of the \( \mu \)-th cell, respectively. Fig. 6 is an example of the grid map.

Several kinds of maps can be represented as such a grid by setting the cell values \( m(\mu) \) in an appropriate manner. In the case of Fig. 1(a), which is called an occupancy grid map [1], \( m(\mu) \) indicates the probability that the \( \mu \)-th cell is occupied by an object. In the case of Fig. 1(b), which is called a likelihood field, \( m(\mu) \) denotes the probability that the \( \mu \)-th cell is observed by a sensor with measurement noise. Although Fig. 1 is the case of a 2D map, a 3D map can also be expressed in the same form by regarding \( m(\mu) \) as a voxel value.

B. MONTE CARLO LOCALIZATION

Let \( x_t \in \mathbb{R}^n \) denote the true pose of the robot at time \( t \in \mathbb{N}_0 \). In the case of a robot moving on a plane, \( x_t \) is given by \( x_t := (x_t, y_t, \theta_t)^T \), where \( (x_t, y_t) \in \mathbb{R}^2 \) is the position and \( \theta_t \in \mathbb{R} \) is the orientation.

Algorithm 1 illustrates MCL for estimating \( x_t \), which corresponds to Table 8.2 in [1]. Algorithm 1 is based on a particle filter, where \( x(t) \in \mathbb{R}^2 \) is the state of the \( i \)-th particle, \( N \in \mathbb{N} \) is the number of particles, and \( X_t := \{x(i)\}_{i=1}^{N} \) is the particle set. Each particle \( x(i) \) represents a candidate for the pose estimate, and the set \( X_t \) approximates the probability distribution of the pose. The purpose of Algorithm 1 is to update \( X_t \) at each time.

Algorithm 1 has inputs \( u_t, z_t, \) and \( m \) in addition to \( x_{t-1} \), where \( u_t \in \mathbb{R}^n \) are control data such as velocity commands or wheel odometry, \( z_t := \{z(i)\}_{i=1}^{L} \) are data measured by a sensor (e.g., LRF), and \( m \in \mathbb{R}^M \) is a map. The measurement data \( z_t \) is a point cloud consisting of \( L \in \mathbb{N} \) points \( z(i) \in \mathbb{R}^n \), which is an \( n \)-dimensional vector of an object’s position measured by the sensor in the sensor coordinates.

In Algorithm 1, \( f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a function that applies control data \( u_t \) to the state of a particle in accordance with the robot motion model. The output \( \hat{x}_{t|t-1} \in \mathbb{R}^n \) is the state of the particle after the update.

The function \( g : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^M \rightarrow \mathbb{R}_{0+} \) corresponds to a measurement model, which provides the likelihood of a pose estimate by matching the measurement data \( z_t \) to the map \( m \). The output \( w(i) \in \mathbb{R}_{0+} \) is the likelihood of the \( i \)-th particle. Note that \( g \) is different from an output equation in control theory which outputs measurement data.

The process of Algorithm 1 is outlined below. After calculating the set \( \tilde{X}_t := \{x(i)\}_{i=1}^{N} \) (lines 3–7), the particles are randomly sampled based on the ratio of the likelihood \( w(i) \) (lines 8–11). This sampling process is called

![Algorithm 1 MCL](image-url)
resampling. Finally, $X_t$ and $\hat{X}_t$ are returned. Note that the algorithm in [1] returns $X_t$ only, whereas Algorithm 1 also returns $\hat{X}_t$. This is because we use $\hat{X}_t$ for generating the highlighted map, as described later.

C. MEASUREMENT MODEL AND LIKELIHOOD FIELD

The measurement model simulates the probability distribution $p(z_t \mid x_t, m)$. The function $g$ in Algorithm 1 calculates the likelihood of a pose estimate based on the measurement model.

There are several kinds of measurement models, e.g., the beam model and the likelihood field model [1]. The likelihood field model is often used, not only for 2D maps [12]–[15] but also for 3D maps [16], [17], because it tends to produce smoother distributions even in cluttered areas and it is computationally efficient.

Here, we describe the details of the likelihood field model. An example of the calculation based on it is presented in Table 6.3 in [1] as algorithm likelihood_field_range_finder_model. This algorithm includes a costly operation of searching for the nearest object on the map for each measurement data. Therefore, it is useful to pre-compute a lookup table that implements a likelihood field, so that the most part of the likelihood calculation can be skipped by referring it [1], [17].

In the case of a 2D map, the likelihood field $m_{\text{lf}} \in \mathbb{R}^M$ is generated from an occupancy grid map $m_{\text{occ}} \in \mathbb{R}^M$ by

$$
m_{\text{lf}}^{(\mu)}(x, y) = \kappa_1 N\left(d\left(x^\mu, y^\mu, m_{\text{occ}}\right), \sigma\right),$$

$$d(x, y, m) := \min_{x', y'} \left\{ \sqrt{(x - x')^2 + (y - y')^2} \mid (x', y') \text{ occupied in } m \right\}$$

where $\kappa_1 \in \mathbb{R}_+$ is a constant, $(x^\mu, y^\mu) \in \mathbb{R}^2$ is the map coordinates of the $\mu$-th cell, and $N: \mathbb{R}_0^+ \times \mathbb{R}_+ \to \mathbb{R}_+$ is a function. The value $N(d, \sigma)$ represents the probability density at $d \in \mathbb{R}_0^+$ under a zero-centered Gaussian distribution with standard deviation $\sigma \in \mathbb{R}_+$. Note that the likelihood field model assumes that the sensor measurement noise is Gaussian.

The likelihood field implemented by the lookup table simplifies the calculation in the likelihood field model. In the case of using algorithm likelihood_field_range_finder_model in [1] as the function $g$; $g$ includes only the following operation:

$$g(z_t, x_t, m) := \prod_{\ell=1}^L q(z_t^{(\ell)}, x_t, m),$$

$$q(z_t^{(\ell)}, x_t, m) := \begin{cases} 1 & \text{if } \ell - \text{th sensor does not find object} \\ \kappa_2 \lambda(z_t^{(\ell)}, x_t, m) + \kappa_3 & \text{otherwise} \end{cases}$$

where $m$ is the likelihood field and $\kappa_2, \kappa_3 \in \mathbb{R}_+$ are the model parameters. Also, $\lambda: \mathbb{R}_n^+ \times \mathbb{R}^n \times \mathbb{R}^M \to \mathbb{R}$ is a function and $\lambda(z_t^{(\ell)}, x_t, m)$ is the cell value $m^{(\mu)}$ of the likelihood field containing the measurement data $z_t^{(\ell)}$, where $z_t^{(\ell)}$ is the position obtained by converting $z_t^{(\ell)}$ (the measurement data in the sensor coordinates) to map coordinates based on the robot pose $x_t$. For example, in Fig. 7, the measurement data are located on the 14th, 20th, and 23rd cells; thus, the likelihood of the pose estimate is

$$w_t^{(i)} = \left(\kappa_2 m^{(14)} + \kappa_3\right) \left(\kappa_2 m^{(20)} + \kappa_3\right)^2 \left(\kappa_2 m^{(23)} + \kappa_3\right).$$

D. HIGHLIGHTED MAP GENERATION PROBLEM

Here, we consider modifying $m$ consisting of $M$ values $m^{(1)}, m^{(2)}, \ldots, m^{(M)}$. If we obtain a suitably modified map $m$, using it in the function $g$ may improve the performance of Algorithm 1. We call the modified map a highlighted map.

We define the objective function for constructing the highlighted map $m$ as follows:

$$J(m) := \sum_{t=1}^T \hat{\rho}(x_t, \chi_t)$$

where $T \in \mathbb{N}$ is the operating time, and $\rho: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ computes the estimation accuracy at each time from the true pose $x_t$ and the particle set (probability distribution of the pose estimate) $\chi_t$. For example, $\rho$ can be defined as the error between $x_t$ and the average of $\chi_t$

$$\rho(x_t, \chi_t) := \exp \left\{ - (x_t - h(\chi_t))^\top A (x_t - h(\chi_t)) \right\}$$

or the variance of $\chi_t$

$$\rho(x_t, \chi_t) := \exp \left\{ - \sum_{i=1}^N (\hat{\chi}_t^{(i)} - h(\chi_t))^\top A (\hat{\chi}_t^{(i)} - h(\chi_t)) \right\},$$

where

$$h(\chi_t) := \frac{1}{N} \sum_{i=1}^N \hat{\chi}_t^{(i)}$$

FIGURE 7. Example of likelihood calculation with likelihood field. The 14th, 20th, and 23rd cells are measured, and the likelihood is calculated with $m^{(14)}$, $m^{(20)}$, and $m^{(23)}$. 

VOLUME 8, 2020
and $A \in \mathbb{P}_{0+}^n$ is a weight coefficient matrix. Note that (6) does not use the true pose $x_t$.

Our problem is formulated as follows.

**Problem 1:** For Algorithm 1 and given $f$, $g$, and $\rho$, assume that an initial particle set $\chi_0$, control data $u_{1:T}$, measurement data $z_{1:T}$, and true poses $x_{1:T}$ are given. Then, find the optimal highlighted map

$$\arg \max_m \overline{J}(m)$$

where $\overline{J} : \mathbb{R}^M \rightarrow \mathbb{R}$ is the expected value of (4),

$$\overline{J}(m) := E \left( J(m) \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T} \right).$$

Note that the solution (8) is optimal just for the past data $u_{1:T}$, $z_{1:T}$, $x_{1:T}$. Optimality is not guaranteed for the future data, but it is assumed that the map provides the same performance when the robot moves on a trajectory similar to the one the robot followed when the data $u_{1:T}$, $z_{1:T}$, $x_{1:T}$ were measured. In fact, we verified that it is effective for future data by conducting a simulation and experiment (see Sections IV and V).

### III. HIGHLIGHTED MAP GENERATION

This section presents the solution to Problem 1 based on RL, which is an approach for finding the optimal policy through trial and error [34].

#### A. REDUCTION OF HIGHLIGHTED MAP GENERATION PROBLEM TO REINFORCEMENT LEARNING

RL deals with the system in Fig. 8, composed of an agent and an environment. The behavior is as follows:

1) The agent observes a state of the environment $s_t$.
2) The agent selects and executes an action $a_t$ according to $s_t$ and the policy $\pi$.
3) The environment outputs a reward $r_t$ according to $s_t$ and $a_t$.
4) The environment $s_t$ transitions to $s_{t+1}$ by referring to $a_t$, and return to Step 1).

For this system, the purpose of RL is to determine the optimal policy $\pi$ that gives the highest cumulative reward.

Let us associate RL with MCL. RL is characterized by a mechanism to search for the optimal policy by using a probabilistic policy that selects an action at random. MCL is the algorithm that estimates the pose by resampling particles at random. Therefore, by regarding the resampling process of MCL as a policy $\pi$ and the resampling result as an action $a_t$, the highlighted map, which is the design parameter used in MCL, can be optimized in the framework of RL.

There are various methods of RL; We use the method in [35], a kind of policy gradient method, which directly optimizes the parameters of the policy [36], [37].

![Frame of RL](image)

First, we convert MCL into the framework of RL by the following correspondence:

- **State $s_t$:** States of all particles of MCL $x_{t|t-1}^{(1)}$, $x_{t|t-1}^{(2)}$, ..., $x_{t|t-1}^{(N)}$ and current time $t$.
- **Action $a_t$:** Resampling result of MCL $a_t^{(1)}$, $a_t^{(2)}$, ..., $a_t^{(N)}$, where $a_t^{(i)} \in \mathbb{N}_0$ is the number of particles resampled to the state $x_{t|t-1}^{(i)}$ on lines 8–11 of Algorithm 1.
- **Policy $\pi$:** Function $g$ and resampling process of MCL. The highlighted map $m$ is a parameter that determines the behavior of these processes.
- **Reward $r_t$:** Output of the function $\rho$ in (4).

Fig. 9 shows the relationship between MCL and RL in this case. In Fig. 9, $D$ is equivalent to the system

$$D : \begin{cases} (1, 0, 0, \ldots, 0) \top & \text{with probability } \frac{w_t^{(1)}}{\sum_{i=1}^{N} w_t^{(i)}} \\ (0, 1, 0, \ldots, 0) \top & \text{with probability } \frac{w_t^{(2)}}{\sum_{i=1}^{N} w_t^{(i)}} \\ \vdots & \vdots \\ (0, 0, 0, \ldots, 1) \top & \text{with probability } \frac{w_t^{(N)}}{\sum_{i=1}^{N} w_t^{(i)}} \end{cases}$$

$$\left( a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(N)} \right) \top = \sum_{j=1}^{N} y_t^{(j)}.$$  

which calculates the frequency distribution of the $N$ samples from the probability distribution represented by $w_t^{(1)}, w_t^{(2)}, \ldots, w_t^{(N)}$. Also, $A$ in Fig. 9 is the process to compose $\chi_t$, where $a_t^{(i)}$ particles in $\chi_t$ are set to be identical to $x_{t|t-1}^{(i)}$ ($i = 1, 2, \ldots, N$). Then,

$$p \left( s_{t+1} \mid s_t, a_{t|\tau}, \pi \right) = p \left( s_{t+1} \mid s_t, a_t \right),$$

$$p \left( a_t \mid s_{t|\tau}, a_{t|\tau}, \pi \right) = p \left( a_t \mid s_t, m \right),$$

$$E \left( r_t \mid s_{t|\tau}, a_{t|\tau}, \pi \right) = E \left( r_t \mid s_t, a_t \right)$$

are satisfied, and the whole system can be regarded as a Markov decision process, which is the target system for many RL algorithms [34], [35].

Next, we derive the characteristic eligibility [38] because many policy gradient methods including [35] need
it [36, 37]. The characteristic eligibility is given by

$$e_t := \nabla_m \ln \{ \bar{\pi}(a_t, s_t, m) \}$$  \hfill (14)

where $\bar{\pi} : \mathbb{N}_0^N \times (\mathbb{R}^{N \times N} \times \mathbb{N}_0) \times \mathbb{R}^M \rightarrow \mathbb{R}_{+}$ is the probability density function corresponding to $\pi$

$$\bar{\pi}(a_t, s_t, m) := p(a_t | s_t, m).$$  \hfill (15)

The characteristic eligibility $e_t$ represents the value of the executed action evaluated from information theory, and if the action is less frequent and is more sensitive to $m$, $e_t$ becomes larger. First, consider $\bar{\pi}$. It is derived as follows:

$$\bar{\pi}(a_t, s_t, m) = \left( \frac{N!}{\prod_{l=1}^{N} (a_t^{(l)l})} \right) \prod_{j=1}^{N} \left( \frac{w_t^{(j)}}{\sum_{j=1}^{N} w_t^{(j)}} \right) \delta^{(t)}_t$$  \hfill (16)

where $w_t^{(j)} / \left( \sum_{j=1}^{N} w_t^{(j)} \right)$ expresses the probability that a particle is resampled to the state $z^{(t)}_{i_t=1}$. Also, $N! / \left( \prod_{l=1}^{N} (a_t^{(l)l}) \right)$ is the number of patterns in which to distribute the $N$ (distinguishable) particles into $N$ groups with the capacity of $a_t^{(1)}$, $a_t^{(2)}$, ..., $a_t^{(N)}$, and $\prod_{j=1}^{N} \left( w_t^{(j)} / \sum_{j=1}^{N} w_t^{(j)} \right) \delta^{(t)}_t$ is the probability that the resampling process results in one of the patterns. That is, (16) expresses the probability that the $N$ (not distinguishable) particles are distributed into $N$ groups with the capacity of $a_t^{(1)}$, $a_t^{(2)}$, ..., $a_t^{(N)}$ by resampling. From (14) and (16), the characteristic eligibility is derived as follows:

$$e_t = \sum_{i=1}^{N} \left( \frac{a_t^{(i)} \times a_t^{(i)l} \times g(z_t, x_t^{(i)l-1}, m)}{w_t^{(i)}} \nabla_m g(z_t, x_t^{(i)l-1}, m) \right)$$

$$- \frac{N}{\sum_{i=1}^{N} w_t^{(i)}} \left( \sum_{i=1}^{N} \nabla_m g(z_t, x_t^{(i)l-1}, m) \right).$$  \hfill (17)

Equation (17) can be rewritten as a function of $z_t$, $m$, $x_t$, and $\bar{x}_l$, which are the inputs and outputs of Algorithm 1, as follows:

$$e_t = \zeta(z_t, m, x_t, \bar{x}_l)$$  \hfill (18)

where $\zeta : \mathbb{R}^{n \times L} \times \mathbb{R}^M \times \mathbb{R}^N \times \mathbb{R}^{(n+1) \times N} \rightarrow \mathbb{R}^M$ is a function equivalent to (17). In fact, the values of $a_t^{(i)}$ and $w_t^{(i)}$ in (17) are obtained from $x_t$ and $\bar{x}_l$, and $\nabla_m g$ can be calculated from $z_t$, $m$, and $\bar{x}_l$. For example, if $g$ is given by (2), $\nabla_m g$ is

$$\frac{\partial g}{\partial m(m)} \left( z_t, \hat{z}_l^{(i)}_{l=1}, m \right) = \beta^{(i,m)} \frac{k_2 w_t^{(i)}}{k_2 m(m) + k_3}$$  \hfill (19)

where $\beta^{(i,m)} \in \mathbb{N}_0$ is the order of the factor $(k_2 m(m) + k_3)$ in the function $g(z_t, x_t^{(i)l-1}, m)$ in (2), and it corresponds to the number of measurements $z_t^{(i)}$ projected onto the $\mu$-th cell. In the case of Fig. 7 and (3), $\beta^{(i,m)}$ takes the following values:

$$\beta^{(i,14)} = 1, \beta^{(i,20)} = 2, \beta^{(i,23)} = 1,$$

$$\beta^{(i,m)} = 0 \text{ if } \mu \neq 14, 20, 23.$$  \hfill (20)

### B. METHOD OF GENERATING HIGHLIGHTED MAP

We propose the following method for generating a highlighted map based on RL:

1. Prepare the initial value of the highlighted map $m(0)$, e.g., a likelihood field constructed from the occupancy grid map.
2. Drive the robot to record control data $u_{1:T}$, measurement data $z_{1:T}$, and true poses $x_{1:T}$. Set $k \leftarrow 0$.
3. Estimate the robot poses by performing MCL with $m(k)$ and $u_{1:T}, z_{1:T}$ sequentially from $t = 1$ to $T$. Construct the updated map $m(k+1)$ based on the result of the estimation and $x_{1:T}$ of Step 2).
4. If $k + 1 = k_{\text{max}}$, then stop. Otherwise, set $k \leftarrow k + 1$ and return to Step 3).

Note that $m(k) \in \mathbb{R}^M$ is the highlighted map at iteration $k \in \mathbb{N}_0$ and $k_{\text{max}} \in \mathbb{N}$ is the maximum iteration number. In Step 2), for example, the true poses $x_{1:T}$ can be obtained by any of the following procedures:

- Use a device that can measure the robot pose directly, such as GPS and indoor positioning system (IPS) [39].
- After performing Step 2), estimate the pose with $u_{1:T}, z_{1:T}$ by using a more complex algorithm than usual (e.g., MCL with a larger number of particles $N$, at a higher frequency), and use it as the approximate true pose. Note that real-time processing is not required for this computation.
- Add a higher performance sensor (e.g., a 3D sensor) to the robot temporarily, and perform Step 2) with it in addition to the original sensor to obtain a high precision pose estimate. Then, use it as the approximate true pose.

The specific method we propose for Step 3) is described in Algorithm 2, where $b \in \mathbb{R}$ is a reinforcement baseline [38] and $\eta(k) \in \mathbb{R}_+$ is a learning rate for updating the map.

In this algorithm, the following calculation is performed sequentially from $t = 1$ to $T$. At each time, MCL is executed first; then, the evaluation value of the estimation $r_t$
Algorithm 2 Update Highlighted Map

1: function UpdateMap($x_0, u_1:T, z_1:T, x_{1:T}, m(k))$
2: \[ v_0 = V_0 = 0 \]
3: for $t = 1$ to $T$ do
4: \[ \chi_t, \tilde{x}_t = \text{MCL}(\chi_{t-1}, u_t, z_t, m(k)) \]
5: \[ r_t = \rho(\chi_t, \tilde{x}_t) \]
6: \[ e_t = \xi(z_t, m(k), \chi_t, \tilde{x}_t) \]
7: \[ v_t = v_{t-1} + e_t \]
8: \[ V_t = V_{t-1} + (r_t - b)v_t \]
9: end for
10: $m(k + 1) = m(k) + \eta(k)V_T$
11: return $m(k + 1)$
12: end function

and the characteristic eligibility $e_t$ are computed. Next, the cumulative sum of $e_t$ denoted by $v_t \in \mathbb{R}^M$ and the cumulative sum of $(r_t - b)v_t$ denoted by $V_t \in \mathbb{R}^M$ are updated. After the calculation at time $T$, the map $m(k)$ is modified by adding $V_T$.

C. THEORETICAL ANALYSIS

We present two theorems for the proposed method. Let $X \subset \mathbb{R}^n$ be a bounded set, which is used in both theorems. The first theorem regards the gradient of the objective function.

**Theorem 1:** For Algorithm 2, assume that the following conditions hold:

(A1) For all $t \in \{1, 2, \ldots, T\}$ and all $i \in \{1, 2, \ldots, N\}$, $\tilde{x}_t^{(i)} \in X$ with probability 1.

(A2) There exists a $\delta_1 \in \mathbb{R}_+$ such that for all $t \in \{1, 2, \ldots, T\}$ and all $\tilde{x}_t \in X$, $|g(z_t, \tilde{x}, m(k))| \geq \delta_1$.

(A3) There exists a $\delta_2 \in \mathbb{R}_+$ such that for all $t \in \{1, 2, \ldots, T\}$, all $\tilde{x}_t \in X$, and all $\mu \in \{1, 2, \ldots, M\}$, $\nabla_m g(z_t, \tilde{x}, m(k))$ exists and $|\partial g(z_t, \tilde{x}, m(k)) / \partial m(\mu)| \leq \delta_2$.

(A4) There exists a $\delta_3 \in \mathbb{R}_+$ such that for all $t \in \{1, 2, \ldots, T\}$ and all $\chi \in X^N$, $|\rho(x_t, \chi)| \leq \delta_3$.

Then,

$$E\left( V_T | x_0, u_1:T, z_1:T, x_{1:T} \right) = \nabla_m \tilde{J}(m(k)).$$  \hspace{1cm} (21)

**Proof:** See Appendix A.

Algorithm 2 is based on the RL method of [35], and Theorem 1 can be proved in a similar way as in [35].

Theorem 1 implies that Algorithm 2 is equivalent to the stochastic approximation method [40], [41], which updates the parameter $m$ according to the gradient of $\tilde{J}$. The second theorem proves that the method converges.

**Theorem 2:** Let $v = m^* \in \mathbb{R}^M$ be an asymptotically stable equilibrium of the differential equation

$$\frac{dv(\tau)}{d\tau} = \nabla v \tilde{J}(v(\tau))$$ \hspace{1cm} (22)

where $\tau \in \mathbb{R}$ and $v \in \mathbb{R}^M$ are variables of the equation. For the method in Section III-B, assume that (A1)–(A4) hold for all $k \in \mathbb{N}_0$ and the following conditions also hold:

(A5) There exist $\delta_4, \delta_5 \in \mathbb{R}$ such that for all $k \in \mathbb{N}_0$, $m(k) \in \mathbb{M}$ where

$$\mathbb{M} := \left\{ m \mid \delta_4 \leq m(\mu) \leq \delta_5, \mu = 1, 2, \ldots, M \right\}.$$

(A6) For all $t \in \{1, 2, \ldots, T\}$, all $x_t \in X$, and all $m \in \mathbb{M}$, $\nabla_m g(z_t, x_t, m)$ exists and $\nabla_m g(z_t, x_t, m)$ is continuous in $m$.

(A7) $\sum_{k=0}^{\infty} \eta(k) = \infty$, $\sum_{k=0}^{\infty} \eta(k)^2 < \infty$, and $\eta(k) \geq 0$ for all $k \in \mathbb{N}_0$.

(A8) There exists a compact set $\mathbb{D} := \mathbb{D}(m^*)$ such that

$$\{ k \mid m(k) \in \mathbb{D}, k = 0, 1, 2, \ldots \} = \infty,$$

where $\mathbb{D}(m^*) := \left\{ v_0 \mid \lim_{k \to \infty} v(\tau|v_0) = m^* \right\}$ and $v(\tau|v_0) \in \mathbb{R}^M$ is the solution of (22) for an initial value $v_0 \in \mathbb{R}^M$.

Then,

$$\lim_{k \to \infty} m(k) = m^*$$ \hspace{1cm} (23)

with probability 1.

**Proof:** The equation in line 10 of Algorithm 2 can be rewritten as

$$m(k + 1) = m(k) + \eta(k)\nabla_m \tilde{J}(m(k)) + \eta(k)\xi(k)$$ \hspace{1cm} (24)

where

$$\xi(k) := V_T - \nabla_m \tilde{J}(m(k)).$$ \hspace{1cm} (25)

Then, if (A5), (A7), (A8) and the following conditions are satisfied, (23) holds by Theorem 2.3.1 in [40].

(B1) The derivative $\nabla_m \tilde{J}(m)$ is continuous at all $m \in \mathbb{M}$.

(B2) For all $\epsilon \in \mathbb{R}_+$,

$$\lim_{k \to \infty} P\left( \sup_{k \geq k} \left| \sum_{k' = k}^{K} \eta(k')\xi(k') \right| \geq \epsilon \right) = 0.$$ \hspace{1cm} (26)

Conditions (B1) and (B2) hold if (A1)–(A4), (A6), and (A7) are satisfied. Appendix B gives the proofs.

Theorem 2 reveals that if (A1)–(A8) are satisfied, the proposed method converges to a local optimum, which means that the method can provide an approximate solution to Problem 1.

Finally, we provide a supplementary explanation of the conditions in the theorems. Conditions (A1)–(A3) and (A6) are about the setting of MCL. The satisfaction of (A1) depends on the structure of $f$ and the boundedness of $x_0$ and $u_{1:T}$. Conditions (A2), (A3), and (A6) hold if, for example, $g$ is given by (2) and (A5) holds with $\delta_4 > -K_3/k_2$. Conditions (A4), (A5), (A7), and (A8) are for RL and stochastic approximation. For example, the function $\rho$ given by (5) or (6) satisfies (A4). Conditions (A5), (A7), and (A8) are generally assumed in stochastic approximation methods [40], [41]. Condition (A5) can be omitted by using the optimization algorithm proposed in [41], which explicitly keeps the solution in the constraint set.

IV. SIMULATION

To clarify the effectiveness of the highlighted map, we compare it with a conventional map (likelihood field) based on...
several MCL-based algorithms first. Furthermore, comparisons with other existing MCL algorithms are performed. These comparisons are carried out under harsh conditions, i.e., low computational resources and a low-performance sensor, because our method is especially effective for a low-cost robot.

In this section, our method is evaluated in the 3D dynamic simulator Gazebo [42]. A robot model and an environment model are shown in Fig. 10. The robot had a 2D LRF, and could obtain wheel odometry data \( u_t \) and LRF measurement data \( z_t \). Since the robot moved on a plane, its pose could be expressed as \( x_t := (x_t, y_t, \theta_t)^T \).

![Simulation environment and robot built using Gazebo. Cargo lines some of the corridors. The robot starts from the center of the left corridor and goes around.](image)

**FIGURE 10.** Simulation environment and robot built using Gazebo. Cargo lines some of the corridors. The robot starts from the center of the left corridor and goes around.

### A. GENERATION OF HIGHLIGHTED MAP

The (local) optimal highlighted map \( m^* \) was generated under the settings described below. The initial map \( m(0) \) is shown in Fig. 11 (represented on a color scale), which is a likelihood field generated by (1) with \( \kappa_1 = 0.1, \sigma = 0.15 \), and an occupancy grid map \( m_{occ} \). The data used for generating the map \( u_{1:T}, z_{1:T}, \) and \( x_{1:T} \) (denoted by \( u_{1:T}^{gen}, z_{1:T}^{gen}, \) and \( x_{1:T}^{gen} \)) were obtained by the following procedure:

- The control data \( u_{1:T}^{gen} \) and measurement data \( z_{1:T}^{gen} \) were measured while the robot followed the route in Fig. 10 at 1 m/s.
- The true poses \( x_{1:T}^{gen} \) were approximately obtained as follows: perform MCL with \( N = 10,000 \) particles (a larger number than usual for this robot) by using \( u_{1:T}^{gen}, z_{1:T}^{gen}, \) \( m(0) \); then, regard the following maximum likelihood pose as \( x_{1:T}^{gen} \),

\[
    x_{t}^{gen} = \hat{x}_{t|t-1}^{(i^*)} \quad \text{s.t.} \quad i^* = \arg \max_i w_t^{(i)} . \tag{27}
\]

Note that the true poses could be obtained from the simulator, however, we did not use it for generating the map in order to simulate a realistic situation. The initial particle set \( \chi_0 \) is given by

\[
    \chi_{0}^{(i)} = \chi_{ini}, \quad i = 1, 2, \ldots, N \tag{28}
\]

where \( \chi_{ini} \in \mathbb{R}^3 \) is the initial true pose. The function \( f \) is algorithm \textit{sample_motion_model_odometry} in Table 5.6 in [1] with sampling algorithm \textit{sample_normal_distribution} in Table 5.4 in [1], \( g \) is given by (2), and \( \rho \) is set as

\[
    \rho(x_t, \chi_t) := \exp \left\{ - (x_t - h(\chi_t))^\top \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix} (x_t - h(\chi_t)) \right\} \tag{29}
\]

where \( h \) is defined as (7). The learning rate \( \eta(k) \) in Algorithm 2 is

\[
    \eta(k) := \frac{1}{k + 3 \times 10^6} . \tag{30}
\]

The other parameters are shown in Table 1, where \( \alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathbb{R}_+ \) are the parameters included in algorithm \textit{sample_motion_model_odometry} [1]. The parameters of map and LRF, number of particles, and frequency are chosen assuming a low-cost robot. The parameters \( \alpha_1, \alpha_2, \alpha_3, \alpha_4, \kappa_2, \) and \( \kappa_3 \) are designed to make localization in the conventional map (Fig. 11) robust. The reinforcement baseline \( b \) is set to reduce the estimated variance of (21).

**TABLE 1. Parameters.**

| Map                                      | Grid scale                          | 240 × 140  |
|------------------------------------------|-------------------------------------|------------|
| Number of cells \( M \) (map size)      | 0.1 m/cell                           |            |
| LRF                                      | Scanning angle                      | 360 degree |
| Number of data \( L \)                  | 18 degree/rev                        |            |
| Possible maximum detection range         | 5 m                                  |            |
| MCL                                      | Number of particles \( N \)          | 100        |
| Frequency                                | Parameters of \( f \)                | All 0.5    |
| Parameters of \( g \)                   | Parameters of \( b \)                | All 0.5    |
| Algorithm 2                              | Maximum iteration number \( k_{max} \) | 4 × 10^6   |
|                                          | Reinforcement baseline \( b \)       | 0.85       |

Conditions (A1)–(A4), (A6), and (A7) hold under these settings, and it was numerically confirmed that (A5) (with \( \delta_4 > -\kappa_3/\kappa_2 \)) and (A8) held.

The highlighted map \( m^* \) generated by the method in Section III-B is shown in Fig. 12. In the upper corridor, the cargo at the center is highlighted most strongly (red spot).
In contrast, the cargo at the upper right in the corridor is highlighted only weakly. This is because it is more difficult to estimate pose in a long straight corridor than in a corner, and therefore, the central cargo (in the long straight corridor) is more important as a landmark. In the lower corridor, conversely, the location without cargo is highlighted (purple spot). Note that the red spot indicates that the corresponding object is an important landmark, and the purple spot denotes that the absence of an object is such a landmark. It can be seen that the way of highlighting differs depending on the surrounding environment even for the same-shaped objects, which implies that landmarks useful for localization can be automatically identified and emphasized.

Fig. 13 illustrates the change in the objective function, and it indicates that convergence is achieved.

Note that a similar highlighted map will be generated if \( \rho \) is given by (6) instead of (29). In this case, the process of obtaining true poses \( x_{1:T} \) can be omitted because (6) does not require \( x_t \).

B. PERFORMANCE OF HIGHLIGHTED MAP

Next, the generated highlighted map \( m^* \) was verified. The data for the verification \( u_{1:T}, z_{1:T}, \) and \( x_{1:T} \) (denoted by \( u_{1:T}^{\text{test}}, z_{1:T}^{\text{test}}, \) and \( x_{1:T}^{\text{test}} \)) were obtained as follows:

- The control data \( u_{1:T}^{\text{test}} \) and measurement data \( z_{1:T}^{\text{test}} \) were measured while the robot followed the same route at the same velocity as Section IV-A. Note that these values are different from \( u_{1:T}^{\text{gen}} \) and \( z_{1:T}^{\text{gen}} \) because the robot was operated manually and traced a route slightly different from that of Section IV-A.
- The true poses \( x_{1:T}^{\text{test}} \) were obtained from the simulator.

The performance of the map was evaluated by
- The objective function \( J \) with (29);
- Estimation error:

\[
\Delta_t := \sqrt{(x_{1:T}^{t\text{est}} - h(\chi_t))^{T} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} (x_{1:T}^{t\text{est}} - h(\chi_t))}
\]

with (7);
- Computation time for one estimation cycle \( \Gamma_t \in \mathbb{R}_+ \).

The values of \( J \) and \( \Delta_t \) were calculated with \( x_{1:T}^{t\text{est}} \) and \( \chi_{1:T} \). Also, \( \chi_{1:T} \) were computed by MCL with \( u_{1:T}^{t\text{est}}, z_{1:T}^{t\text{est}}, \) the parameters in Table 1, and the same \( f \) and \( g \) as in Section IV-A. The computation time was measured by using a Broadcom BCM2837 system on a chip (SoC) with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor. Since MCL contains randomness and the estimation result changes every iteration, we tested the map 20 times using the same \( u_{1:T}^{t\text{est}}, z_{1:T}^{t\text{est}}, \) and \( x_{1:T}^{t\text{est}} \).

The following two comparisons were carried out. Firstly, we compared the highlighted map with the conventional map in several MCL algorithms to verify the effectiveness of the highlighted map and the validity of combining the highlighted map with various MCL-based algorithms. Secondly, the combination of the basic MCL and the highlighted map were compared with that of the improved MCL-based algorithms and the conventional map to verify the significance of the improvement by the highlighted map. As the improved algorithms, we chose KLD-sampling-based MCL (KLD-MCL) [1], [5] and the algorithm introduced in [7], which we call scan-matching-based MCL (SM-MCL). KLD-MCL is the most widely used MCL-based algorithm and has been actively studied in recent years, and SM-MCL is one of the state-of-the-art methods.

For the above two comparisons, we performed six simulations denoted by S1–S6, whose configurations and results are shown in Table 2. Table 2 summarizes the values of the objective function \( J \), the mean and maximum values of the estimation error \( \Delta_t \) and computation time \( \Gamma_t \). All these values are the averages over 20 tests. To show how the highlighted map improved the performance, the estimation errors \( \Delta_t \) of 20 tests in the case of S1 and S2 are shown in Figs. 14 and 15, respectively. Furthermore, we evaluated the statistical significance of the improvement by box plots presented in Fig. 16 and t-tests. The mean and maximum values of \( \Delta_t \) of S2, S4, and S6 had statistically significant differences from those of S1, S3, and S5, respectively (t-test, \( \rho < 0.001 \)). The highlighted map (optimized for use in MCL) improved KLD-MCL and SM-MCL as well as MCL because of the similarity between these improved MCL algorithms and MCL algorithm. These results confirmed the effectiveness of the highlighted map and the validity of combining the highlighted map with other MCL-based algorithms.
TABLE 2. Localization accuracy and computation time for each configuration in the simulation. All values are the averages over 20 tests.

| Configuration | Objective function $J$ | Estimation error $\Delta_{t}$ (m) | Computation time $T_{r}$ (ms/cycle) |
|---------------|-------------------------|-----------------------------------|-----------------------------------|
| $S_1$ | MCL | Conventional map (Fig. 11) | 166.4 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.22 | 0.85 | 6.11 | 6.75 |
| $S_2$ | MCL | Highlighted map (Fig. 12) | 209.1 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.12 | 0.42 | 6.06 | 6.39 |
| $S_3$ | KLD-MCL | Conventional map (Fig. 11) | 171.1 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.20 | 0.61 | 49.47 | 97.89 |
| $S_4$ | KLD-MCL | Highlighted map (Fig. 12) | 217.1 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.11 | 0.29 | 40.87 | 73.14 |
| $S_5$ | SM-MCL | Conventional map (Fig. 11) | 179.4 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.18 | 0.51 | 41.72 | 73.00 |
| $S_6$ | SM-MCL | Highlighted map (Fig. 12) | 218.9 | $\frac{1}{T} \sum_{t=1}^{T} \Delta_{t}$ | 0.11 | 0.25 | 37.69 | 66.79 |

FIGURE 14. Estimation errors with the combination of MCL and the conventional likelihood field map $m(0)$ in the simulation. (a) Error at each time. (b) Error at each position.

FIGURE 15. Estimation errors with the combination of MCL and the highlighted map $m^*$ in the simulation. (a) Error at each time. (b) Error at each position.

In addition, by comparing the mean and maximum values of $\Delta_{t}$ of $S_2$ with those of $S_3$ and $S_5$, we find that the significance of the improvement by the highlighted map was comparable or greater than the other existing algorithms even though the highlighted map approach required less computation time.

Note that we generated a highlighted map from the actual robot data $u_{1:T}^{gen}$, $z_{1:T}^{gen}$, and $x_{1:T}^{gen}$; thus, the generated highlighted map should be specialized for situations such as sensor characteristics and robot dynamics. For example, since $u_{1:T}^{gen}$, $z_{1:T}^{gen}$, and $x_{1:T}^{gen}$ were measured when the robot followed a clockwise route, the generated highlighted map was especially effective for clockwise driving as shown in this section. On the other hand, by using the data recorded when the robot traced a counterclockwise route, we would obtain a highlighted map specialized for counterclockwise driving, which is slightly different from $m^*$ in Fig. 12. As described here, our method can generate various highlighted maps adapted to each situation.

V. EXPERIMENT

To verify the performance of the highlighted map in a practical situation, this section performs the real-world experiment in two environments: the corridor shown in Fig. 17 and the laboratory shown in Fig. 18. Fig. 19 shows...
the robot system. A 2D LRF mounted on the robot was RPLIDAR (SLAMTEC), whose scanning angle, number of data, and possible maximum detection range were 360 degree, 360 data/rev, and 12 m, respectively. However, we used only 1/20 of the data (18 data/rev) and excluded data that were more than 5 m in order to simulate a low-cost sensor.

First, we generated the highlighted maps $m^*$. The initial maps $m(0)$ in the corridor and laboratory experiments are illustrated in Figs. 20 and 21, which are likelihood fields generated by (1) with $\kappa_1 = 0.1$, $\sigma = 0.15$, and the occupancy grid maps $m_{\text{occ}}$ in Figs. 17 and 18, respectively. The data for generating the maps $u_{\text{gen}}$, $z_{\text{gen}}$, and $x_{\text{gen}}$ were measured in the same way as in Section IV-A, where the routes followed by the robot when collecting $u_{\text{gen}}$, $z_{\text{gen}}$, and $x_{\text{gen}}$ are shown in Figs. 17 and 18. The map sizes $M$ and reinforcement baselines $b$ were $240 \times 200$ and 0.85 in the corridor.
TABLE 3. Localization accuracy and computation time for each configuration in the corridor experiment. All values are the averages over 20 tests.

| Configuration | Objective function $J$ | Estimation error $\Delta_t$ (m) | Computation time $\Gamma_t$ (ms/cycle) |
|---------------|------------------------|-------------------------------|-------------------------------------|
| C1            | MCL                    | 143.6                         | 0.11 0.38 6.00 6.37                |
| C2            | MCL                    | 155.7                         | 0.07 0.36 6.00 6.54                |
| C3            | KLD-MCL                | 153.9                         | 0.08 0.41 43.47 68.67             |
| C4            | KLD-MCL                | 160.7                         | 0.06 0.24 39.58 64.36             |
| C5            | SM-MCL                 | 154.8                         | 0.07 0.40 39.23 58.34             |
| C6            | SM-MCL                 | 161.7                         | 0.05 0.28 37.00 58.52             |

FIGURE 24. Change in objective function in the corridor experiment. The thin blue line is the value of $J(m(k))$ at each iteration $k$, and the thick red line is its moving average. The data are thinned out to 1/1000, that is, only the data at $k = 0, 1000, 2000, \ldots$ are displayed.

FIGURE 25. Change in objective function in the laboratory experiment. The thin blue line is the value of $J(m(k))$ at each iteration $k$, and the thick red line is its moving average. The data are thinned out to 1/1000, that is, only the data at $k = 0, 1000, 2000, \ldots$ are displayed.

FIGURE 26. Estimation errors with the combination of MCL and the conventional likelihood field map $m(0)$ in the corridor experiment. (a) Error at each time. (b) Error at each position.

FIGURE 27. Estimation errors with the combination of MCL and the highlighted map $m^*$ in the corridor experiment. (a) Error at each time. (b) Error at each position.

FIGURE 28. Box plots of mean and maximum of estimation errors in the corridor experiment. (a) Mean error $\left(\frac{1}{T} \sum_{t=1}^{T} \Delta_t\right)$. (b) Maximum error $(\max_t \Delta_t)$. The items C1–C6 correspond to the configurations in Table 3.

experiment, 220 × 180 and 0.9 in the laboratory experiment, respectively. All other parameters and functions were the same as in Section IV-A. The generated highlighted maps and the changes in the objective function in the corridor and laboratory experiments are shown in Figs. 22, 23, 24 and 25, respectively. As in Section IV-A, it can be seen that the landmarks useful for localization are highlighted and convergence is achieved.

Next, we evaluated the generated highlighted maps $m^*$. The measurement procedure of $u_{1:T}^{test}$ and $z_{1:T}^{test}$, and the evaluation method were the same as in Section IV-B. The true poses $x_{1:T}^{test}$ were approximately calculated from $u_{1:T}^{test}$, $m(0)$, and the complete data obtained from RPLIDAR (number of data was 360 data/rev and possible maximum detection range was 12 m) instead of $z_{1:T}^{test}$ (number of data was 18 data/rev and possible maximum detection range was 5 m). The results of the corridor experiment are shown in Table 3, Figs. 26, 27, and 28. Table 3 summarizes the values of $J$, $\Delta_t$, and $\Gamma_t$. 
TABLE 4. Localization accuracy and computation time for each configuration in the laboratory experiment. All values are the averages over 20 tests.

| Index | Algorithm | Map | Objective function $J$ | Estimation error $\Delta t$ (m) | Computation time $T_t$ (ms/cycle) |
|-------|-----------|-----|------------------------|-------------------------------|----------------------------------|
|       |           |     | $\frac{1}{T} \sum_{t=1}^{T} \Delta t$ | $\frac{1}{T} \sum_{t=1}^{T} T_t$ | $\max_t T_t$ |
| L1    | MCL       | Conventional map (Fig. 21) | 213.3 | 0.12 | 0.58 | 6.08 | 6.71 |
| L2    | MCL       | Highlighted map (Fig. 23)  | 237.7 | 0.07 | 0.31 | 6.05 | 6.62 |
| L3    | KLD-MCL   | Conventional map (Fig. 21) | 223.5 | 0.10 | 0.45 | 48.33 | 94.69 |
| L4    | KLD-MCL   | Highlighted map (Fig. 23)  | 244.6 | 0.05 | 0.20 | 42.48 | 79.63 |
| L5    | SM-MCL    | Conventional map (Fig. 21) | 221.6 | 0.10 | 0.40 | 42.13 | 74.44 |
| L6    | SM-MCL    | Highlighted map (Fig. 23)  | 245.8 | 0.05 | 0.19 | 38.58 | 67.43 |

![FIGURE 29](image1) Estimation errors with the combination of MCL and the conventional likelihood field map $m(0)$ in the laboratory experiment. (a) Error at each time. (b) Error at each position.

![FIGURE 30](image2) Estimation errors with the combination of MCL and the highlighted map $m^*$ in the laboratory experiment. (a) Error at each time. (b) Error at each position.

For each configuration denoted by C1–C6, Figs. 26 and 27 show the errors (31) of 20 tests in the case of C1 and C2, respectively. Fig. 28 illustrates box plots of the errors. Similarly, the results of the laboratory experiment, denoted by L1–L6, are presented in Table 4, Figs. 29, 30, and 31. The mean and maximum values of $\Delta t$ of C2, C4, C6, L2, L4, and L6 had statistically significant differences from those of C1, C3, C5, L1, L3, and L5, respectively (t-test, $p < 0.001$). As in Section IV-B, it was confirmed that the highlighted maps improved those localization algorithms by replacing the conventional maps. By comparing C2 with C3 and C5, and L2 with L3 and L5, respectively, we find that the accuracy of the highlighted map approach was comparable or better than the other existing algorithms, regardless of the fact that our approach took less computation time. These results show the effectiveness of the highlighted map in a practical situation. Note that the reason why the degree of improvement differs for each experiment is mainly due to the difference in the shape of the environments.

VI. CONCLUSION

This article proposed a highlighted map for mobile robot localization and a method for generating such a map. The highlighted map can improve the localization accuracy without any need for updating their sensors or online computation, and moreover, it can be applied to various versions of MCL. Although this article only generated 2D highlighted maps, the proposed method can also create 3D highlighted maps.

Furthermore, the proposed method can be easily applied to other particle filter applications in addition to localization. For example, it can be used to optimize template images in object tracking that uses template matching.
In future work, we are going to consider the robustness of a highlighted map to resist disturbances. A highlighted map generated by our proposed method may be vulnerable to uncertainties because the map is generated from a single experiment. In order to solve this, we will develop an RL algorithm that targets disturbance environments.

**APPENDIX A**

**PROOF OF THEOREM 1**

To prove Theorem 1, several variables have to be defined first. From (11)–(13) and (15), the state transition probability and the expected reward under π are defined as

\[
P^{m}_{s_t, s_{t+1}} := p\left(s_{t+1} \mid s_t, \pi\right) = \sum_{a \in \mathcal{A}} \tilde{P}(a, s_t, m) p\left(s_{t+1} \mid s_t, a\right),
\]

\[
R^{m}_{s_t} := E\left(r_t \mid s_t, \pi\right) = \sum_{a \in \mathcal{A}} \tilde{P}(a, s_t, m) E\left(r_t \mid s_t, a\right),
\]

(32) where \(\mathcal{A} \subseteq \mathbb{R}^N_0\) is the action set. The probability distribution of the initial state is defined as

\[
P^{\mu_0}_{s_1} := p\left(s_1 \mid \mu_0, \mu_1\right).
\]

(33)

Then, from (4) and (9), the gradient of \(\tilde{J}\) is represented by (32)–(34), as follows:

\[
\nabla_m \tilde{J} (m) = \nabla_m E \left(\sum_{t=1}^{T} r_t \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
= \sum_{t=1}^{T} \nabla_m E \left(r_t \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
= \sum_{t=1}^{T} \nabla_m \left(\int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t} ds_t \cdots ds_1\right).
\]

(35)

Note that, from (A1), the integrals here mean to integrate over the space of all possible states \(S := \mathcal{X}^N\) (except for the space of state \(c_1\), the current time). For the integrand of (35), we define

\[
F_t (s_{1:t}, m) := p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t},
\]

(36)

for which the following properties hold:

- The integrand \(F_t\) is integrable, that is, \(\int_{S} \cdots \int_{S} |F_t (s_{1:t}, m)| ds_t \cdots ds_1 < \infty\) (because \(S\) is the bounded set, and \(|F_t (s_{1:t}, m)|\) are bounded from (A2) and (A4)).
- The derivative \(\nabla_m F_t (s_{1:t}, m)\) exists for \(\forall s_1, s_2, \ldots, s_t \in S\) (because \(F_t\) is a polynomial in \(\tilde{\pi}\), and \(\nabla_m \tilde{\pi}\) exists from (A2) and (A3)).
- There exists an integrable function \(\tilde{F} (s_{1:t})\) such that \(\left|\frac{\partial F_t (s_{1:t}, m)}{\partial m^{(\mu)}}\right| \leq \delta_6\) from (A2)–(A4), and \(\delta_6\) is integrable, that is, \(\int_{S} \cdots \int_{S} |\delta_6| ds_t \cdots ds_1 < \infty\).

Thus, the order of differentiation and integration can be interchanged by Proposition 5.9 in [43]. As a result,

\[
\nabla_m \left(\int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t} ds_t \cdots ds_1\right)
= \int_{S} \cdots \int_{S} \nabla_m \left(p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t}\right) ds_t \cdots ds_1
= \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} \nabla_m p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t} ds_t \cdots ds_1
+ \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} \nabla_m R^{m}_{s_t} ds_t \cdots ds_1
+ \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t} \nabla_m R^{m}_{s_t} ds_t \cdots ds_1.
\]

(37)

On the other hand, the expected value of \((r_t - b)\nu_t\) in Algorithm 2 is represented as follows:

\[
E \left((r_t - b)\nu_t \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
= E \left((r_t - b) \sum_{\tau=1}^{t} e_{\tau} \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
= \sum_{\tau=1}^{t} E \left(r_{\tau} e_{\tau} \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)

- b \sum_{\tau=1}^{t} E \left(e_{\tau} \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
\]

(38)

because \(e_1, e_2, \ldots, e_T\) exist from (A2) and (A3). The expected value of \(r_{\tau} e_{\tau}\) in the case of \(\tau < t\) is given by

\[
E \left(r_{\tau} e_{\tau} \mid \chi_0, u_{1:T}, z_{1:T}, x_{1:T}\right)
= \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t}
\times p^{m}_{s_{t-1}, s_t} R^{m}_{s_t} ds_t \cdots ds_1
\times \left\{ \sum_{a \in \mathcal{A}} \tilde{P}(a, s_{\tau}, m) p\left(s_{\tau+1} \mid s_{\tau}, a\right) \right\}
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
\times p^{m}_{s_{\tau+1}, s_{\tau+2}} \cdots p^{m}_{s_{t-1}, s_t} R^{m}_{s_t} ds_t \cdots ds_1
+ \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t}
\times \left\{ \sum_{a \in \mathcal{A}} \tilde{P}(a, s_{\tau}, m) p\left(s_{\tau+1} \mid s_{\tau}, a\right) \right\}
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
= \int_{S} \cdots \int_{S} p^{\mu_0}_{s_1} p^{m}_{s_1, s_2} \cdots p^{m}_{s_{t-1}, s_t}
\times \left\{ \sum_{a \in \mathcal{A}} \tilde{P}(a, s_{\tau}, m) p\left(s_{\tau+1} \mid s_{\tau}, a\right) \right\}
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
\times \nabla_m \ln \left(\tilde{\pi}(a, s_{\tau}, m)\right)
Then, for \( m \) and \( m' \) we have

\[
\begin{align}
= \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} p_{x_0}^{m_{s_1}} p_{y_0}^{m_{s_2}} \cdots p_{y_{r-1}}^{m_{s_r}} \\
\times \left( \nabla_m p_{s_r, s_{r+1}} \right) \\
\times \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} p_{s_r, s_{r+1}}^{m_{s_{r+1}}} \cdots p_{s_{r+1}, s_r}^{m_{s_r}} R_{s_{r+1}}^{m_{s_{r+1}}} ds_1 \cdots ds_1,
\end{align}
\]

and similarly,

\[
E \left( r_i \mid x_0, u_1, \ldots, s_{r-1}, x_1 \right) = \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} p_{x_0}^{m_{s_1}} p_{y_0}^{m_{s_2}} \cdots p_{y_{r-1}}^{m_{s_r}} \left( \nabla_m R_{s_{r+1}}^{m_{s_r}} \right) ds_1 \cdots ds_1,
\]

then from (35) and the fact that \( \nabla_m \mathbf{J}(m) \) is continuous. From this fact and (35), \( \nabla_m \mathbf{J}(m) \) is continuous.

**Lemma 2:** For the method in Section III-B, if (A2)–(A4) are satisfied for \( \forall k \in \mathbb{N}_0 \) and (A7) is also satisfied, then (B2) holds.

**Proof:** The following equation holds for \( \mathbb{E}_k := \sum_{k=0}^{\infty} \eta(k)\xi(k) \),

\[
E \left( \mathbb{E}_k \mid \mathbb{E}_{0:k-1} \right) = E \left( \mathbb{E}_{k-1} + \eta(k)\xi(k) \mid \mathbb{E}_{0:k-1} \right) = \mathbb{E}_{k-1} + \eta(k)E \left( \xi(k) \mid \mathbb{E}_{0:k-1} \right) = \mathbb{E}_{k-1}.
\]

Therefore, \( \mathbb{E}_k \) is a martingale sequence, and we obtain the martingale inequality of Doob [40]

\[
P \left( \sup_{k \geq k} \left\| \sum_{k=k}^{N} \eta(k)\xi(k') \right\| \geq \epsilon \right) \leq \frac{1}{\epsilon^2} E \left( \left\| \sum_{k=k}^{\infty} \eta(k')\xi(k') \right\|^2 \right). \tag{46}
\]

Since

\[
E \left( \xi(k_1)\xi(k_2) \right) = 0 \tag{47}
\]

if \( k_1 \neq k_2 \), the right side of (46) can be transformed as follows:

\[
\frac{1}{\epsilon^2} E \left( \left\| \sum_{k=k}^{\infty} \eta(k')\xi(k') \right\|^2 \right) = \frac{1}{\epsilon^2} \sum_{k=k}^{\infty} \eta(k')^2 E \left( \left\| \xi(k') \right\|^2 \right). \tag{48}
\]

In addition,

\[
E \left( \left\| \xi(k') \right\|^2 \right) = E \left( \left\| V_T - \nabla_m \mathbf{J}(m(k')) \right\|^2 \right)
\]

\[
= E \left( \left\| V_T \right\|^2 - \left\| \nabla_m \mathbf{J}(m(k')) \right\|^2 \right)
\]

\[
\leq E \left( \left\| V_T \right\|^2 \right)
\]

\[
\leq E \left( \sum_{t=1}^{T} (r_t - b) \sum_{t=1}^{T} e_t \right)^2
\]

\[
\leq E \left( \sum_{t=1}^{T} \left( r_t - b \right) \sum_{t=1}^{T} \left\| e_t \right\|^2 \right). \tag{49}
\]

Here, (A4) gives \( |r_t| \leq \delta_3 \), and we have \( \left\| e_t \right\| \leq \frac{2\delta_2 N N \sqrt{M}}{\delta_1} \) from (17) and (A2), (A3). Thus,

\[
\sum_{t=1}^{T} |r_t - b| \sum_{t=1}^{T} \left\| e_t \right\| \leq \frac{T(T + 1)(\delta_3 + |b|)\delta_2 N \sqrt{M}}{\delta_1}. \tag{50}
\]
From (46) and (48)–(50),

\[
P \left( \sup_{k \geq k_0} \left| \sum_{k' = k_0}^{k} \eta(k') \xi(k') \right| \geq \epsilon \right) \leq \left( \frac{1}{\delta_1 \epsilon} \right)^2 \sum_{k' = k_0}^{\infty} \eta(k')^2. \tag{51}
\]

Equation (51) and (A7) indicate that (B2) holds.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[2] F. Delaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo localization for mobile robots,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, MI, USA, May 1999, pp. 1322–1328.
[3] S. Thrun, D. Fox, W. Burgard, and F. Delaert. “Robust Monte Carlo localization for mobile robots,” *Artif. Intell.*, vol. 128, nos. 1–2, pp. 99–141, May 2001.
[4] J. Rowekamper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and R. Y. Yoshimura, “Real-time 6DoF localization for autonomous vehicle navigation and localization through precise map,” *Int. J. Automot. Eng.*, vol. 3, no. 1, pp. 19–25, 2012.
[5] D. Fox, “Adapting the sample size in particle filters through KLD-sampling,” *Int. J. Robot. Res.*, vol. 22, no. 12, pp. 985–1003, Dec. 2003.
[6] L. Zhang, R. Zapata, and P. Lépinay, “Self-adaptive Monte Carlo localization for mobile robots using range finders,” *Robotica*, vol. 30, no. 2, pp. 229–244, Mar. 2012.
[7] A. Ohsato, Y. Sasaki, and H. Mizoguchi, “Real-time 6DoF localization for a mobile robot using pre-computed 3D laser likelihood field,” in *Proc. IEEE Int. Conf. Robot. Biomimetcs (ROBIO)*, Zhuhai, China, Dec. 2015, pp. 2359–2364.
[8] H. Tehrani Niknejad, M. Seichi, H. Long, and H. Quoc, “Multi-sensor data fusion for autonomous vehicle navigation and localization through precise map,” *Int. J. Automot. Eng.*, vol. 3, no. 1, pp. 19–25, 2012.
[9] M. Rapp, M. Hahn, M. Thom, J. Dickmann, and K. Dietmayer, “Semi-supervised Monte Carlo localization based on effective combination of vision and range sensors,” *Int. J. Control. Automat. Syst.*, vol. 7, no. 1, pp. 97–104, Mar. 2009.
[10] Y.-J. Lee, B.-D. Yun, and J.-B. Song, “Mobile robot localization based on optimization of features,” in *Proc. 4th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Yokohama, Japan, Nov. 2015, pp. 002638–002643.
[11] K. Sokota, S. Nonaka, and K. Sekiguchi, “Experimental verification of a vehicle localization based on moving horizon estimation integrating LRS and odometry,” *J. Phys. Conf. Ser.*, vol. 744, Sep. 2016, Art. no. 012048.
[12] N. Roy and S. Thrun, “Coastal navigation with mobile robots,” in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1043–1049.
[13] M. Koizumi, S. Nonaka, and K. Sekiguchi, “Avoidance of singular localization environment using model predictive control for mobile robots,” in *Proc. 11th Asian Int. Conf. Robot. Autom. (ASRC)*, Gold Coast, QLD, Australia, Dec. 2017, pp. 2866–2871.
[14] J. Koher, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Aug. 2013.
[15] K. Sakaeta, K. Nonaka, and K. Sekiguchi, “Experimental verification of a vehicle localization based on moving horizon estimation integrating LRS and odometry,” *J. Phys. Conf. Ser.*, vol. 744, Sep. 2016, Art. no. 012048.
[16] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *Int. J. Robot. Res.*, vol. 5, no. 4, pp. 56–68, Dec. 1986.
[17] S. Huang and G. Dissanayake, “Convergence and consistency analysis for extended Kalman filter based SLAM,” *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1036–1049, Oct. 2007.
[18] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proc. 18th National Conf. Artif. Intell.*, Edmonton, AB, Canada, Jul. 2002, pp. 593–598.
[19] M. Lin, C. Yang, D. Li, and G. Zou, “Intelligent filter-based SLAM for mobile robots with improved localization performance,” *IEEE Access*, vol. 7, pp. 113284–113297, 2019.
[20] S. Thrun and M. Montemerlo, “The graph SLAM algorithm with applications to large-scale mapping of urban structures,” *Int. J. Robot. Res.*, vol. 25, nos. 5–6, pp. 403–429, May 2006.
[21] R. Murdoch, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular SLAM system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[23] H. Kimura, M. Yamamura, and S. Kobayashi, “Reinforcement learning by stochastic hill climbing on discounted reward,” in *Proc. 12th Int. Conf. Mach. Learn.*, Tahoe City, CA, USA, Jul. 1995, pp. 295–303.
[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, Dec. 1999, pp. 1057–1063.
[25] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Beijing, China, Oct. 2006, pp. 2219–2225.
[26] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.
[27] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Trans. Syst., Man, Cybern. C. Appl. Rev.*, vol. 37, no. 6, pp. 1087–1089, Nov. 2007.
[28] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York, NY, USA: Springer-Verlag, 1978.
[29] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York, NY, USA: Springer-Verlag, 2003.
[30] K. Koenig and A. Howard, “Design and use paradigms for gazerbo, an open-source multi-robot simulator,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sendai, Japan, Oct. 2004, pp. 2149–2154.
[31] M. Brokate and G. Kersting, *Measure and Integral*. Basel, Switzerland: Birkhauser, 2004.
RYOTA YOSHIMURA received the B.S. degree in engineering and the M.S. degree in informatics from Kyoto University, Kyoto, Japan, in 2009 and 2011, respectively, where he is currently pursuing the Ph.D. degree with the Graduate School of Engineering. He is currently a Researcher with the Regional Technology Support Division, Tokyo Metropolitan Industrial Technology Research Institute, Tokyo, Japan. His research interests include mobile robot navigation, optimal control, and machine learning.

ICHIRO MARUTA (Member, IEEE) received the Bachelor of Engineering, Master of Informatics, and Doctor of Informatics degrees from Kyoto University, Kyoto, Japan, in 2006, 2008, and 2011, respectively. He was a Research Fellow with the Japan Society for the Promotion of Science, from 2008 to 2011. From 2012 to 2017, he was an Assistant Professor with the Graduate School of Informatics, Kyoto University. In 2017, he joined the Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University, as a Lecturer, where he has been an Associate Professor, since 2019.

KENJI FUJIMOTO (Member, IEEE) received the B.Sc. and M.Sc. degrees in engineering and the Ph.D. degree in informatics from Kyoto University, Japan, in 1994, 1996, and 2001, respectively. From 1997 to 2004, he was a Research Associate with the Graduate School of Engineering and the Graduate School of Informatics, Kyoto University. From 2004 to 2012, he was an Associate Professor with the Graduate School of Engineering, Nagoya University, Japan. From 1999 to 2000, he was a Research Fellow with the Department of Electrical Engineering, Delft University of Technology, The Netherlands. He has held visiting research positions at Australian National University, Australia, in 1999, and the Delft University of Technology, in 2002. He is currently a Professor with the Graduate School of Engineering, Kyoto University. His research interests include nonlinear control and stochastic systems theory. He received the IFAC Congress Young Author Prize in IFAC World Congress 2005 and the SICE Pioneer Award in 2007.

KEN SATO received the B.S. degree in engineering from Shinshu University, Nagano, Japan, in 1990. He is currently a Senior Researcher with the Business Development Section, Tokyo Metropolitan Industrial Technology Research Institute, Tokyo, Japan. His research interests include the design of appearance and electrical systems for autonomous mobile robots, and recently, he has been engaged in the development of autonomous mobile cleaning robots.

YUSUKE KOBAYASHI received the M.S. degree in engineering from Tokyo Denki University, Tokyo, Japan, in 2011. He is currently a Deputy Senior Researcher with the Business Development Section, Tokyo Metropolitan Industrial Technology Research Institute, Tokyo. His current research interest includes the design and development of drive units for autonomous mobile robots.