# Reinforcement Learning Based Generation of Highlighted Map for Mobile Robot Localization and Its Generalization to Particle Filter Design

Ryota YOSHIMURA

2022

# Contents

# Chapter 1

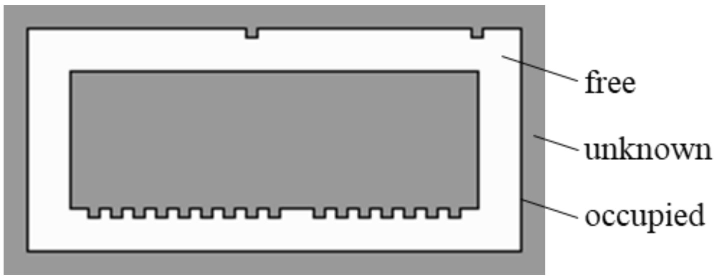
# Introduction

Autonomous navigation technology has been applied to various robots, including automated guided vehicles and cars. For robots, estimation of their pose, namely mobile robot localization, is essential [1–3].

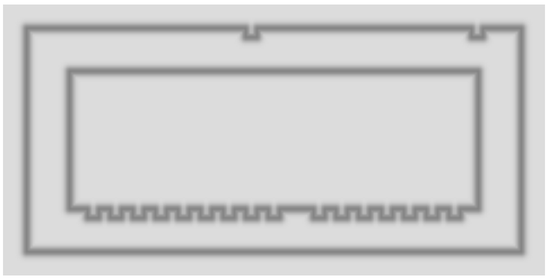
Monte Carlo localization (MCL) is one of the most popular localization methods [4–6]. It is a particle-filter-based algorithm, which scatters a large number of particles (hypotheses about the pose estimate) onto the environment map and calculates the likelihood of each particle (the likelihood of the corresponding pose estimate) to estimate a probability distribution of the robot pose. Since MCL can handle complex multimodal probability distributions, it enables highly reliable estimation in various environments. Therefore, it has been widely studied and used for over two decades; e.g., [7] improved MCL by adding the process of injecting random particles, and [8] proposed Kullback-Leibler divergence (KLD) sampling, which dynamically adapts the number of particles. In [9], a pre-caching technique was applied to MCL to reduce the computational burden and sample particles efficiently. The method in [10] 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 [11]. The major target of MCL is a wheeled robot [12], but it has also been applied to an underwater robot [13], an uncrewed aerial vehicle [14], and the field of wireless sensor networks [15].

Although many robots utilize MCL as above, 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



(a) Original map (occupancy grid map)

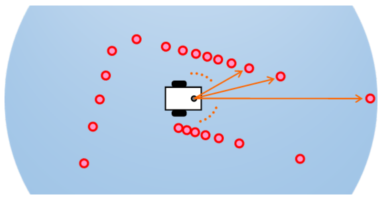


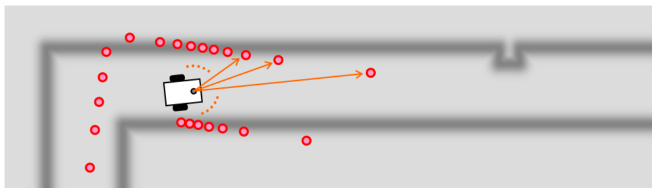
(b) Likelihood field map

Figure 1.1: Example of likelihood field map

and measurements of the robot's sensor, such as a light detection and ranging (LIDAR), ultrasonic sensor, or camera. The measurement model is often built based on a likelihood field (see, for instance, [16–21]). A likelihood field is a map that considers the sensor's measurement noise. Figure 1.1 shows an example. Figure 1.1(a) is the original map, where the color indicates the probability that the space is occupied by an object. Figure 1.1(b) is the corresponding likelihood field map, which is generated by blurring the original map by the magnitude of the measurement noise. The map of Fig. 1.1(b) denotes that the darker a location is, the more likely it is to be detected by the sensor. Figure 1.2 is an example demonstrating how the likelihood of a pose estimate is calculated from a likelihood field map in the case of using a LIDAR sensor. Figure 1.2(a) shows the LIDAR measurement data, and Fig. 1.2(b) and (c) show the projection of the measurement data onto the likelihood field map at each pose estimate. The measurement data in Fig. 1.2(c), which are the red points, overlap the dark locations more than in Fig. 1.2(b); thus, the pose estimate of Fig. 1.2(c) is judged to be more likely.

The likelihood calculation in MCL does not work well in monotonous environments. For example, in the case of a long straight corridor shown in Fig. 1.3, the pose estimate of Fig. 1.3(b) is greatly shifted to the left from the actual robot pose, whereas in the case of Fig. 1.3(c), it is slightly rotated clockwise from the actual pose. On the other hand, the


(a) Measurement data


(b) Case of pose estimate whose likelihood is low

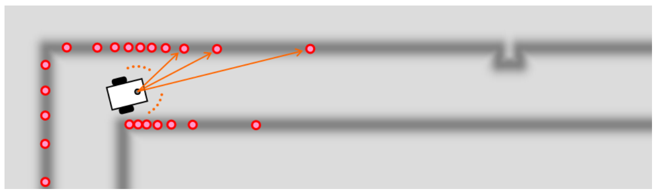

(c) Case of pose estimate whose likelihood is high

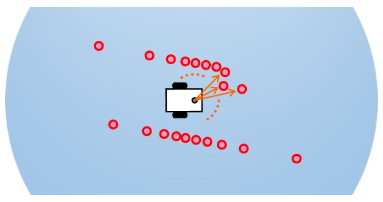
Figure 1.2: Projection of measurement data onto map

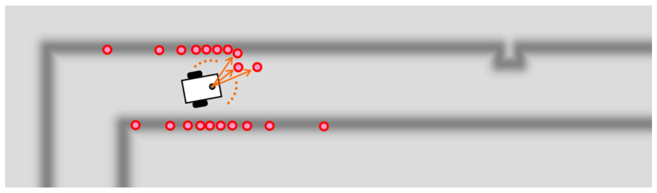
measurement data in Fig. 1.3(b) overlap the dark locations more than in Fig. 1.3(c). This means that the pose estimate of Fig. 1.3(b) is more deviated but is judged to be more likely than that of Fig. 1.3(c). If the robot does not have enough computational resources, it will get lost by this matter.

The most common conventional method for localization in such environments is to use additional sensors having different characteristics from those of a LIDAR, e.g., use a global navigation satellite system (GNSS) [22, 23] or a camera [24, 25], and fuse these data. The sensor fusion method in [26] 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 [27]. Methods that prevent a robot from entering such monotonous areas were presented in [28, 29].

## 1.1 Highlighted map

This thesis considers 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. The


(a) Measurement data


(b) Case of pose estimate whose error is large but likelihood is high

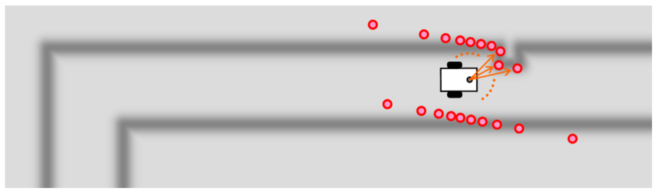
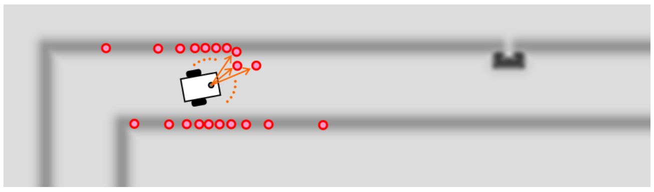

(c) Case of pose estimate whose error is small but likelihood is low

Figure 1.3: Projection of measurement data onto corridor map

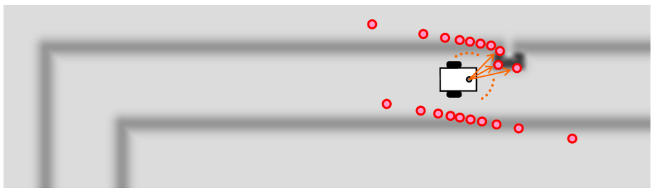
approach in this thesis is to generate a map on which the landmarks are emphasized and use the map for localization instead of a likelihood field map. For example, if the landmarks are darkened, the likelihood of the pose estimate increases when the measurement data overlap the landmarks. As a result, the localization accuracy improves. Figure 1.4 demonstrates the idea. The measurement data in Fig. 1.4(a) overlap only the light locations, whereas those in Fig. 1.4(b) also overlap the dark locations. Thus, the pose estimate of Fig. 1.4(b) is judged to be more likely than that of Fig. 1.4(a).

In this thesis, the map whose shade is modified for making a more accurate localization is called a highlighted map. The advantage of the highlighted map approach is that it requires no additional sensors 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



(a) Case of pose estimate whose error is large and likelihood is low



(b) Case of pose estimate whose error is small and likelihood is high

Figure 1.4: Projection of measurement data onto highlighted map

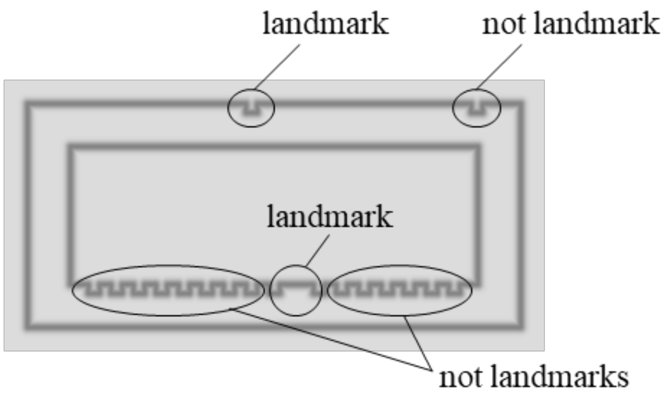


Figure 1.5: Landmarks and non-landmarks

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. 1.1, but, as indicated in Fig. 1.5,

- The protrusion at the center of the north corridor (in a long straight corridor) is a landmark.
- The protrusion at the northeast (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 south 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.

Note that the upper side of the figure is north in this thesis. 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 thesis 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, consider using reinforcement learning (RL), which has attracted much attention in various fields including robotics [30, 31], to optimize the shade of the map. The idea in this thesis is to regard the randomness of MCL 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.

In this thesis, the problem of generating a highlighted map is formulated, and a numerical optimization method based on RL is proposed for the solution. Furthermore, a theorem that guarantees the optimization will converge is derived. The numerical simulation and real-world experiment are performed to show the effectiveness of a highlighted map.

Note that this 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 [4]. The constructed map is generally used for path planning as well as localization. Various SLAM methods, such as EKF-SLAM [32, 33], particle-filter-based SLAM [34, 35], and graph-based SLAM [36, 37], 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 thesis 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.

## 1.2 Robust highlighted map

After the basic method of generating a highlighted map is proposed, the robustification of a highlighted map is considered. Since the proposed method generates a highlighted map by utilizing a limited number of the sensor measurement data, the generated highlighted map is vulnerable to unexpected sensor measurement noise.

In recent years, adversarial learning has received considerable attention in the machine learning community (e.g., generative adversarial networks [38]), and it has also been used in RL to acquire robust policies. The method proposed in [39], robust adversarial reinforcement learning (RARL), trains both the original system (protagonist) and extra disturbances (adversary) simultaneously in the framework of a two-player game. By the training, the protagonist becomes robust to the environmental changes and modeling errors. This method has shown good results in various problems [40, 41], and it is often applied to autonomous vehicle control [42, 43]. In related studies, [44] proposed a Q-learning-based RL algorithm for a two-player game, and [45] presented an actor-critic-based RL algorithm that learns a robust policy in a framework similar to adversarial learning.

This thesis also proposes a method of generating a robust highlighted map, which is a highlighted map that is robust against noise, by using adversarial RL. This method introduces a virtual obstacle causing measurement noise and learns both the worst-case obstacle behavior and the optimal highlighted map simultaneously. The numerical simulation is performed to verify the robustness of the map.

## 1.3 Particle filter design

Next, consider generalizing the proposed method of generating a highlighted map to the design of a particle filter (PF). A PF is a popular method for estimating the state of a dynamical system [46, 47]. Owing to its high versatility in handling nonlinear systems with non-Gaussian noise and ease of implementation, it has a wide range of applications. For example, it has been used for object tracking in computer vision [48] and for predicting remaining useful life of lithium-ion batteries [49] in addition to mobile robot localization.

The PF algorithm is outlined below. The PF approximates the probability distribution of the estimated state by using a set of many particles, which represent hypotheses about the state. This probability distribution is estimated by repeating the following steps: 1) predict the current particles based on the system model; 2) calculate the weight (likelihood) of each particle based on the measurement model; 3) resample the particles according to their weights. Here, the system and measurement models must be designed by the user for each target system. The design of these models is an important problem because the performance of a PF depends on it.

Therefore, several methods for estimating appropriate models, or the parameters that determine them, have been studied. For example, the Bayesian estimation method in [50] incorporates the parameters into the state vector and estimates both the state and parameters simultaneously using the PF algorithm. Another Bayesian approach in [51] uses Markov chain Monte Carlo methods. In [52], a PF was used to approximate the gradient and Hessian of the log-likelihood of the parameters, which enabled a gradient ascent for maximum likelihood estimation (MLE). These methods aim to provide plausible models but do not directly consider the performance of the PF (e.g., estimation accuracy).

Several methods directly optimize the estimation accuracy of a PF. The method in [53] uses particle swarm optimization (PSO) to optimize it. However, PSO cannot be expected to optimize high-dimensional parameters successfully. Another method to optimize the estimation accuracy implements the system and measurement models as neural networks to make the PF algorithm differentiable [54], so it lacks versatility in the structure of the models. Furthermore, the convergence of the optimization is not considered in [53, 54].

The proposed method of generating a highlighted map optimizes the estimation accuracy of MCL, whose algorithm is equivalent to a PF. In addition, it is proven that the optimization converges. This method regards every pixel of the map as parameters that determine the measurement model of the PF and optimizes them by using RL. As described here, this method considers only a specific PF application and lacks generality; e.g., the system model is not designed, and only the estimation accuracy can be optimized. Thus, this method needs to be extended.

In this thesis, a new PF design method is proposed. The advantage of the proposed method is that it is possible to optimize various objective functions, e.g., the likelihood of the parameters, the estimation accuracy of the PF, and the regularization term of the parameters. Moreover, the convergence of the optimization is guaranteed.

After presenting the proposed PF design method, it is applied to mobile robot localization, and the numerical simulation is performed, where more than 30,000-dimensional parameters are optimized simultaneously.

## 1.4 Summary

In summary, the main contributions of this thesis are as follows:

- This thesis newly proposed the idea of modifying the shade of a map that evaluates the likelihood of a pose estimate to improve the localization accuracy. The modified map is called a highlighted map. The map can improve the accuracy of the MCL algorithm without additional sensors 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.
- By applying the framework of the adversarial RL, the above method is extended to generate a highlighted map that is robust against noise.
- The above method is generalized to a PF design method. This method can design not only the measurement model but also the system model and accommodate various objective functions.

This thesis is organized as follows. First, Chapter 2 explains the MCL and RL algorithm. Next, Chapter 3, Chapter 4, and Chapter 5 propose the method of generating a highlighted map, the method of generating a robust highlighted map, and the method of designing a PF, respectively. Chapter 6 is the conclusion.

*Notation*: Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{N}$, $\mathbb{C}$, $\emptyset$, and $\mathbb{P}^n_{0+}$ denote the real number field, the set of positive real numbers, the set of natural numbers, the set of complex numbers, the empty set, and the set of the $n \times n$ positive semidefinite matrices, respectively. For $N$ data $x^{(1)}, x^{(2)}, \dots, x^{(N)}$, the notation $\{x^{(i)}\}_{i=1}^N$ denotes the set of them, i.e., $\{x^{(i)}\}_{i=1}^N := \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$. For time-varying data $x_t$, $x_{t_1:t_2}$ denotes sequences of data from time $t_1$ to time $t_2$, i.e., $x_{t_1:t_2} := (x_{t_1}, x_{t_1+1}, x_{t_1+2}, \dots, x_{t_2})$. For a vector $v$ and matrix $M$, $\|v\|$ and $\|M\|_F$ represent the Euclidean norm of $v$ and the Frobenius norm of $M$, respectively. For a vector $v$, scalar function $f$, and vector function $g$, $\nabla_v f$, $\nabla_v^2 f$, and $\nabla_v g$ denote the gradient of $f$, the Hessian matrix of $f$, and the Jacobian matrix of $g$, respectively.

# Chapter 2

# Preliminaries

This chapter describes the MCL algorithm and its components, the map and the measurement model. In addition, RL, which is utilized to solve the problems in the following chapters, is outlined.

## 2.1 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 $m := \left[m^{(1)}, m^{(2)}, \ldots, m^{(M)}\right]^\top$, 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. Figure 2.1 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. 1.1(a), which is called an occupancy grid map [4], $m^{(\mu)}$ indicates the probability that the $\mu$-th cell is occupied by an object. In the case of Fig. 1.1(b), which is called a likelihood field map, $m^{(\mu)}$ denotes the probability that the $\mu$-th cell is detected by a sensor. Although Fig. 1.1 is the case of a two-dimensional (2D) map, a three-dimensional (3D) map can also be expressed in the same form by regarding $m^{(\mu)}$ as a voxel value.

## 2.2 Monte Carlo localization

Although the highlighted map can be applied to various localization algorithms related to MCL, this thesis considers the combination with the most basic MCL for simplicity. This section describes the MCL algorithm.

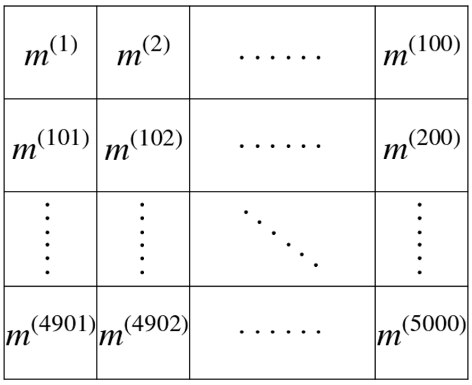


Figure 2.1: Grid map (size $100 \times 50$)

Let $x_t \in \mathbb{R}^n$ denote the true pose of the robot at time $t \in \{0\} \cup \mathbb{N}$. In the case of a robot moving on a plane, $x_t$ is given by

$$x_t := [\mathrm{x}_t, \mathrm{y}_t, \theta_t]^\top \tag{2.1}$$

where $(\mathrm{x}_t, \mathrm{y}_t) \in \mathbb{R}^2$ is the position, and $\theta_t \in \mathbb{R}$ is the orientation.

MCL is a PF-based algorithm for estimating $x_t$, which corresponds to Table 8.2 in [4]. Algorithm 1 illustrates MCL, where $\hat{x}_t^{(i)} \in \mathbb{R}^n$ is the state of the $i$-th particle, $N \in \mathbb{N}$ is the number of particles, and $\chi_t := \left\{\hat{x}_t^{(i)}\right\}_{i=1}^N$ is the particle set. Each particle $\hat{x}_t^{(i)}$ represents a candidate for the pose estimate, and the set $\chi_t$ approximates the probability distribution of the pose. The purpose of Algorithm 1 is to update $\chi_t$ at each time.

Algorithm 1 has four inputs $\chi_{t-1}$, $u_t$, $z_t$, and $m$, where $u_t \in \mathbb{R}^{n_u}$ are control data such as velocity commands or wheel odometry, $z_t := \left\{z_t^{(\ell)}\right\}_{\ell=1}^L$ are data measured by a sensor (e.g., LIDAR), 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 $z_t^{(\ell)} \in \mathbb{R}^{n_z}$, which is an $n_z$-dimensional vector of an object's position detected by the sensor in the sensor coordinates.

Algorithm 1 contains two functions, $f$ and $g$. The function $f : \mathbb{R}^{n_u} \times \mathbb{R}^n \to \mathbb{R}^n$ updates the state of each particle by applying control data $u_t$. The output $\hat{x}_{t|t-1}^{(i)} \in \mathbb{R}^n$ is the state of the particle after the update. The function $g : \mathbb{R}^{n_z \times L} \times \mathbb{R}^n \times \mathbb{R}^M \to \mathbb{R}_+$ corresponds to the measurement model, which provides the likelihood of a pose estimate by matching the measurement data $z_t$ to the map $m$. The output $w_t^{(i)} \in \mathbb{R}_+$ 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 flow of Algorithm 1 is as follows: 1) calculate the weighted particle set $\bar{\chi}_t :=$

**Algorithm 1** MCL

```
1: function MCL($\chi_{t-1}, u_t, z_t, m$)
2:     $\bar{\chi}_t = \chi_t = \emptyset$
3:     for $i = 1$ to $N$ do
4:         $\hat{x}^{(i)}_{t|t-1} = f\left(u_t, \hat{x}^{(i)}_{t-1}\right)$
5:         $w^{(i)}_t = g\left(z_t, \hat{x}^{(i)}_{t|t-1}, m\right)$
6:         $\bar{\chi}_t \leftarrow \bar{\chi}_t \cup \left\{\left(\hat{x}^{(i)}_{t|t-1}, w^{(i)}_t\right)\right\}$
7:     end for
8:     for $i = 1$ to $N$ do
9:         $\hat{x}^{(i)}_t = \begin{cases} \hat{x}^{(1)}_{t|t-1} & \text{with probability } \frac{w^{(1)}_t}{\sum_{j=1}^N w^{(j)}_t} \\ \hat{x}^{(2)}_{t|t-1} & \text{with probability } \frac{w^{(2)}_t}{\sum_{j=1}^N w^{(j)}_t} \\ \vdots & \vdots \\ \hat{x}^{(N)}_{t|t-1} & \text{with probability } \frac{w^{(N)}_t}{\sum_{j=1}^N w^{(j)}_t} \end{cases}$
10:        $\chi_t \leftarrow \chi_t \cup \left\{\hat{x}^{(i)}_t\right\}$
11:    end for
12:    return $\chi_t$, $\bar{\chi}_t$
13: end function
```

$\left\{\left(\hat{x}^{(i)}_{t|t-1}, w^{(i)}_t\right)\right\}_{i=1}^{N}$ (lines 3–7); 2) sample the particles randomly based on the ratio of the likelihood $w^{(i)}_t$ to compose the particle set at time $t$, $\chi_t$ (lines 8–11); 3) return $\chi_t$ and $\bar{\chi}_t$. The sampling process in Step 2) is called resampling. Note that the algorithm in [4] returns $\chi_t$ only, whereas Algorithm 1 also returns $\bar{\chi}_t$. This is because $\bar{\chi}_t$ is used for generating a highlighted map, as described later.

## 2.3 Measurement model and likelihood field map

The measurement model simulates the probability distribution $p\left(z_t \mid x_t, m\right)$. 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 [4]. The likelihood field model is often used, not only for 2D maps [16–19] but also for 3D maps [20, 21], because it tends to produce smoother distributions even in cluttered areas, and it is computationally efficient.

Here, the details of the likelihood field model are described. An example of the calculation based on it is presented in Table 6.3 in [4] 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 map so that most parts of the likelihood calculation can be skipped by referring it [4, 21].

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

$$m_{\mathrm{lf}}^{(\mu)} = \kappa_1 \, \mathcal{N}\left(D\left(x^\mu, y^\mu, m_{\mathrm{occ}}\right), \sigma\right) \tag{2.2}$$

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 $D : \mathbb{R} \times \mathbb{R} \times \mathbb{R}^M \to \{0\} \cup \mathbb{R}_+$ is the distance to the nearest object on the map, i.e.,

$$D(x, y, m) := \min_{x', y'} \left\{ \sqrt{(x - x')^2 + (y - y')^2} \,\middle|\, (x', y') \text{ occupied in } m \right\}. \tag{2.3}$$

The function $\mathcal{N} : \mathbb{R} \times \mathbb{R}_+ \to \mathbb{R}_+$ denotes a zero-centered normal distribution, and the value $\mathcal{N}(x, \sigma^2)$ is the probability density at $x \in \mathbb{R}$ with standard deviation $\sigma \in \mathbb{R}_+$. Note that the likelihood field model assumes that the sensor measurement noise is Gaussian.

The likelihood field map 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 [4] as the function $g$; $g$ includes only the following operation:

$$g(z_t, x_t, m) := \prod_{\ell=1}^{L} q\left(z_t^{(\ell)}, x_t, m\right) \tag{2.4}$$

where

$$q\left(z_t^{(\ell)}, x_t, m\right) := \begin{cases} 1 & \text{if } \ell\text{–th sensor does not detect object} \\ \left(\kappa_2 \, \mathcal{M}\left(z_t^{(\ell)}, x_t, m\right) + \kappa_3\right) & \text{otherwise,} \end{cases} \tag{2.5}$$

$m$ is the likelihood field map, and $\kappa_2, \kappa_3 \in \mathbb{R}_+$ are the model parameters. Also, $\mathcal{M} : \mathbb{R}^{n_z} \times \mathbb{R}^n \times \mathbb{R}^M \to \mathbb{R}$ is a function, and $\mathcal{M}\left(z_t^{(\ell)}, x_t, m\right)$ is the cell value $m^{(\mu)}$ of the likelihood field map containing the measurement data $\tilde{z}_t^{(\ell)}$, where $\tilde{z}_t^{(\ell)}$ is the position obtained by converting $z_t^{(\ell)}$ (the measurement data in the sensor coordinates) into the map coordinates based on the robot pose $x_t$. For example, in Fig. 2.2, the measurement data are located on

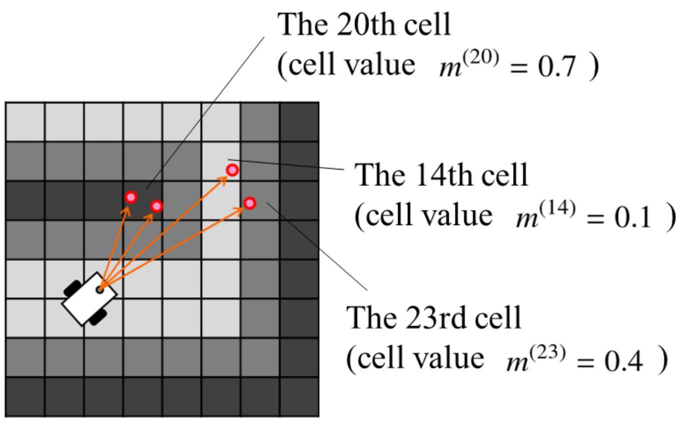

The 20th cell
(cell value $m^{(20)} = 0.7$ )
The 14th cell
(cell value $m^{(14)} = 0.1$ )
The 23rd cell
(cell value $m^{(23)} = 0.4$ )

Figure 2.2: Projection of measurement data onto likelihood field map and the corresponding cell values

the 14th, 20th, and 23rd cells; thus, the likelihood of the pose estimate is

$$w_t^{(i)} = \left(\kappa_2\, m^{(14)} + \kappa_3\right)\left(\kappa_2\, m^{(20)} + \kappa_3\right)^2\left(\kappa_2\, m^{(23)} + \kappa_3\right). \tag{2.6}$$

## 2.4 Reinforcement learning

This section outlines RL, which is an approach for finding the optimal policy through trial and error [55].

RL deals with the system in Fig. 2.3, composed of an agent and an environment. They repeat the following behaviors: 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 $a_t$. For this system, the purpose of RL is to determine the optimal policy $\pi$ that gives the highest cumulative reward.

RL mainly consists of two types of methods: value iteration methods, e.g., Q learning [56] and deep Q-network (DQN) [57], and policy gradient methods, e.g., REINFORCE [58] and stochastic hill climbing [59]. This thesis mostly uses the policy gradient method and also uses the value iteration method in Chapter 4.

The policy gradient methods directly optimize the parameter $\Theta$ that defines the policy. For optimizing $\Theta$, many policy gradient methods use the characteristic eligibility [58–61]. It is given by

$$e_t := \nabla_\Theta \ln\{\tilde{\pi}(a_t, s_t, \Theta)\} \tag{2.7}$$

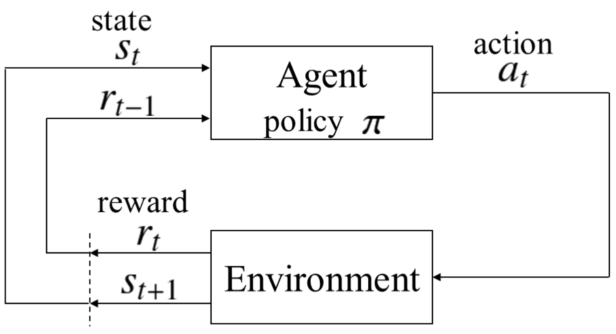


Figure 2.3: RL framework

where

$$\tilde{\pi}(a_t, s_t, \Theta) := p\left(a_t \,\middle|\, s_t, \Theta\right) \tag{2.8}$$

is the probability density function corresponding to $\pi$. 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 $\Theta$, $e_t$ becomes larger.

The value iteration methods learn the value of each pair of states and actions. For example, Q learning updates the state-action value function $Q(s_t, a_t)$ by the following equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \left\{ r_t + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right\}, \tag{2.9}$$

where $\eta \in \mathbb{R}_+$ is a learning rate, and $\lambda \in \mathbb{R}_+$ is a discount factor [56]. The optimal policy $\pi^*(s_t)$ is obtained indirectly from the value function, e.g.,

$$\pi^*(s_t) = \arg\max_a Q(s_t, a). \tag{2.10}$$

# Chapter 3

# Highlighted map generation

This chapter presents the generation method of a highlighted map. First, the problem of generating a highlighted map is formulated, and the solution for it is proposed. Next, the numerical simulation and experiment show the effectiveness of using a highlighted map.

## 3.1 Problem formulation

Consider modifying a map $m$ described in Chapter 2, which consists of $M$ values $m^{(1)}, m^{(2)}, \ldots, m^{(M)}$. If we obtain a suitably modified map $m$, using it in MCL in Section 2.2 may improve the localization accuracy. In this thesis, the modified map is called a highlighted map.

The objective function for constructing a highlighted map $m$ is defined as follows:

$$J(m) := \sum_{t=1}^{T} \rho\left(x_t, \chi_t\right) \tag{3.1}$$

where $T \in \mathbb{N}$ is the operating time, and $\rho : \mathbb{R}^n \times \mathbb{R}^{n\times N} \to \mathbb{R}$ evaluates the localization accuracy at each time from the true pose $x_t$ and the particle set (probability distribution of the pose estimate) $\chi_t$ in Section 2.2. For example, $\rho$ can be defined as the error between $x_t$ and the average of $\chi_t$

$$\rho\left(x_t, \chi_t\right) := \exp\left\{-\left(x_t - h\left(\chi_t\right)\right)^\top A \left(x_t - h\left(\chi_t\right)\right)\right\}, \tag{3.2}$$

or the variance of $\chi_t$

$$\rho(x_t, \chi_t) := \exp\left\{-\sum_{i=1}^{N}\left(\hat{x}_t^{(i)} - h(\chi_t)\right)^\top A\left(\hat{x}_t^{(i)} - h(\chi_t)\right)\right\}, \tag{3.3}$$

where $h : \mathbb{R}^{n\times N} \to \mathbb{R}^n$ is the average of the state of the particles, i.e.,

$$h(\chi_t) := \frac{1}{N}\sum_{i=1}^{N}\hat{x}_t^{(i)}, \tag{3.4}$$

and $A \in \mathbb{P}_{0+}^n$ is a weight coefficient matrix. Note that (3.3) does not use the true pose $x_t$.

The problem is formulated as follows.

**Problem 1.** *For Algorithm 1, assume that the number $N$ of particles, functions $f$, $g$, and $\rho$, 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*

$$\arg\max_{m}\ \bar{J}(m) \tag{3.5}$$

*where $\bar{J} : \mathbb{R}^M \to \mathbb{R}$ is the expected value of* (3.1)*, i.e.,*

$$\bar{J}(m) := E\left[J(m)\right]. \tag{3.6}$$

■

Note that the solution (3.5) is optimal just for the past data $u_{1:T}$, $z_{1:T}$, and $x_{1:T}$. 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 $u_{1:T}$, $z_{1:T}$, and $x_{1:T}$ were measured. In fact, a simulation and experiment verified that it is effective for future data (see Sections 3.3 and 3.4).

## 3.2 Highlighted map generation based on reinforcement learning

This section presents the solution to Problem 1 based on RL.

### 3.2.1 Interpretation of highlighted map generation problem into reinforcement learning

Let us associate MCL with RL described in Section 2.4. RL is characterized by a mechanism to search for the optimal policy 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 $a_t$ in RL, the highlighted map, which is the design parameter used in MCL, can be optimized in the RL framework.

This chapter uses the method in [59], a kind of policy gradient method, which directly optimizes the parameters of the policy.

First, MCL is converted into the RL framework by the following assignment:

- Policy $\pi$: Function $g$ and resampling process of MCL. The highlighted map $m$ is a parameter that determines the behavior of these processes.
- State $s_t = \left(\left(\hat{x}_{t|t-1}^{(1)}, \hat{x}_{t|t-1}^{(2)}, \dots, \hat{x}_{t|t-1}^{(N)}\right), c_t\right)$: States of all particles of MCL $\hat{x}_{t|t-1}^{(1)}, \hat{x}_{t|t-1}^{(2)}, \dots, \hat{x}_{t|t-1}^{(N)}$ and current time $c_t := t$.
- Action $a_t = \left[\gamma_t^{(1)}, \gamma_t^{(2)}, \dots, \gamma_t^{(N)}\right]^\top$: Resampling result of MCL, where $\gamma_t^{(i)} \in \mathbb{N}$ is the index $j$ of the state $\hat{x}_{t|t-1}^{(j)}$ to which the $i$-th particle is resampled in line 9 of Algorithm 1.
- Reward $r_t$: Output of the function $\rho$ in (3.1).

Figure 3.1 shows the relationship between MCL and RL in this case. In Fig. 3.1, $\mathcal{D}$ outputs $a_t = \left[\gamma_t^{(1)}, \gamma_t^{(2)}, \dots, \gamma_t^{(N)}\right]^\top$ as follows:

$$\gamma_t^{(i)} = \begin{cases} 1 & \text{with probability } \frac{w_t^{(1)}}{\sum_{j=1}^N w_t^{(j)}} \\ 2 & \text{with probability } \frac{w_t^{(2)}}{\sum_{j=1}^N w_t^{(j)}} \\ \vdots & \qquad \vdots \\ N & \text{with probability } \frac{w_t^{(N)}}{\sum_{j=1}^N w_t^{(j)}} \end{cases} \quad (i = 1, 2, \dots, N) \tag{3.7}$$

which generates $N$ random numbers from the probability distribution represented by $w_t^{(1)}, w_t^{(2)}, \dots, w_t^{(N)}$. In addition, $\mathcal{A}$ in Fig. 3.1 is the process to compose $\chi_t$, where the

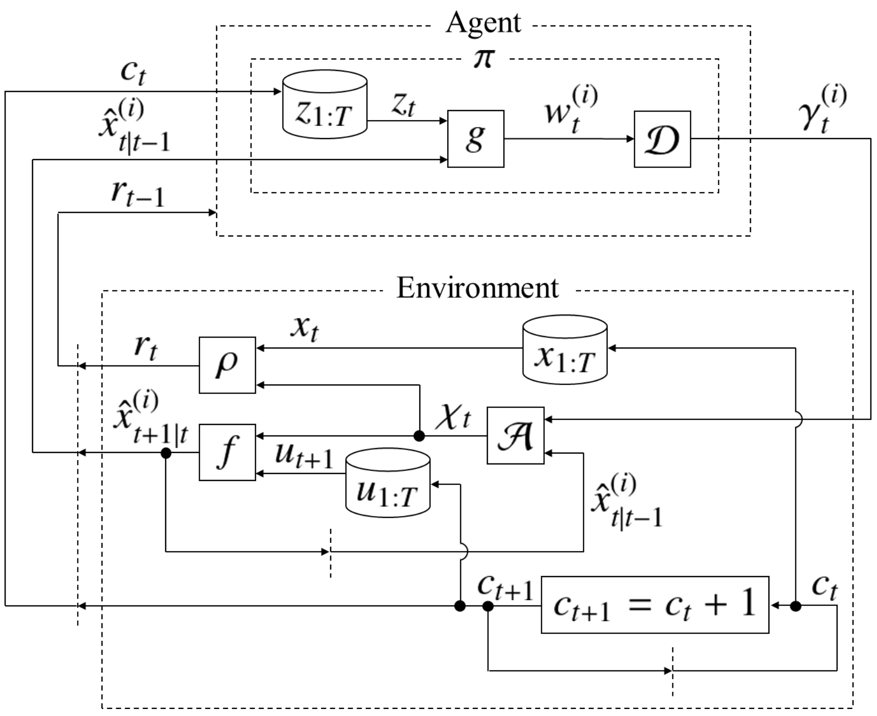


Figure 3.1: Relationship between MCL and RL

state of the $i$-th particle in $\chi_t$ is set to $\hat{x}^{(\gamma^{(i)}_t)}_{t|t-1}$ $(i = 1, 2, \ldots, N)$. Then,

$$p\left(s_{t+1} \,\middle|\, s_{1:t}, a_{1:t}, \pi\right) = p\left(s_{t+1} \,\middle|\, s_t, a_t\right), \tag{3.8}$$

$$p\left(a_t \,\middle|\, s_{1:t}, a_{1:t-1}, \pi\right) = p\left(a_t \,\middle|\, s_t, m\right), \tag{3.9}$$

$$E\left[r_t \,\middle|\, s_{1:t}, a_{1:t}, \pi\right] = E\left[r_t \,\middle|\, s_t, a_t\right] \tag{3.10}$$

are satisfied, and the whole system can be regarded as a Markov decision process (MDP), which is the target system for many RL algorithms [55, 59].

Next, the characteristic eligibility (2.7) is derived. The probability density function (2.8) is derived as follows:

$$\tilde{\pi}(a_t, s_t, m) = \prod_{i=1}^{N} \frac{w^{(\gamma^{(i)}_t)}_t}{\sum_{j=1}^{N} w^{(j)}_t}. \tag{3.11}$$

From (2.7) and (3.11), the characteristic eligibility is derived as follows:

$$\begin{aligned} e_t &:= \nabla_m \ln \left\{\tilde{\pi}(a_t, s_t, m)\right\} \\ &= \sum_{i=1}^{N} \left(\Gamma^{(i)}_t - \frac{N w^{(i)}_t}{\sum_{j=1}^{N} w^{(j)}_t}\right) \nabla_m \ln \left\{g\left(z_t, \hat{x}^{(i)}_{t|t-1}, m\right)\right\} \end{aligned} \tag{3.12}$$

where

$$\Gamma_t^{(i)} := \left|\left\{j \,\middle|\, \gamma_t^{(j)} = i,\ j = 1, 2, \ldots, N\right\}\right| \tag{3.13}$$

is the number of particles resampled to the state $\hat{x}_{t|t-1}^{(i)}$. The values of $\Gamma_t^{(i)}$ and $w_t^{(i)}$ in (3.12) are obtained from $\chi_t$ and $\bar{\chi}_t$, and $\nabla_m \ln g$ can be calculated from $\bar{\chi}_t$, $z_t$, and $m$. For example, if $g$ is given by (2.4), $\nabla_m \ln g$ is

$$\frac{\partial}{\partial m^{(\mu)}} \ln \left\{ g\left(z_t, \hat{x}_{t|t-1}^{(i)}, m\right)\right\} = \frac{\kappa_2\, \beta^{(i,\mu)}}{\kappa_2\, m^{(\mu)} + \kappa_3} \tag{3.14}$$

where $\beta^{(i,\mu)} \in \{0\} \cup \mathbb{N}$ is the order of the factor $\left(\kappa_2\, m^{(\mu)} + \kappa_3\right)$ in the function $g$ in (2.4), and it corresponds to the number of measurement data $z_t^{(\ell)}$ projected onto the $\mu$-th cell. In the case of Fig. 2.2 and (2.6), $\beta^{(i,\mu)}$ takes the following values:

$$\beta^{(i,\mu)} = \begin{cases} 1 & \text{if } \mu = 14 \\ 2 & \text{if } \mu = 20 \\ 1 & \text{if } \mu = 23 \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

### 3.2.2 Method of generating highlighted map

This chapter proposes Algorithm 2 for generating a highlighted map based on RL. Note that $m(k) \in \mathbb{R}^M$ is the highlighted map at iteration $k \in \{0\} \cup \mathbb{N}$, $k_{\max} \in \mathbb{N}$ is the maximum iteration number, $b \in \mathbb{R}$ is a reinforcement baseline [58], and $\eta(k) \in \mathbb{R}_+$ is a learning rate for updating the map.

To execute Algorithm 2, it is necessary to drive the robot to record control data $u_{1:T}$, measurement data $z_{1:T}$, and true poses $x_{1:T}$. The true poses $x_{1:T}$ can be obtained, e.g., by temporarily adding a sensor that can measure the robot pose directly, such as a GNSS or indoor positioning system [62], to the robot. Alternatively, $x_{1:T}$ are approximately obtained by executing a more complex localization algorithm than usual (e.g., MCL with a larger number of particles, at a higher frequency) after recording $u_{1:T}$ and $z_{1:T}$. Note that real-time processing is not required for this computation.

In this algorithm, the following calculation is performed $k_{\max}$ times. At each time from $t = 1$ to $T$, MCL is executed first; then, the evaluation value of the estimation $r_t$ and the characteristic eligibility $e_t$ are computed. Next, the cumulative sum of $e_t$ denoted by $U_t \in \mathbb{R}^M$ and the cumulative sum of $(r_t - b)U_t$ denoted by $V_t \in \mathbb{R}^M$ are updated. After the calculation

**Algorithm 2** Highlighted map generation

```
 1: function HighlightedMapGen($\chi_0, u_{1:T}, z_{1:T}, x_{1:T}, m(0)$)
 2:     for $k = 0$ to $k_{\max} - 1$ do
 3:         $U_0 = V_0 = 0$
 4:         for $t = 1$ to $T$ do
 5:             $\chi_t, \bar{\chi}_t = \mathrm{MCL}\left(\chi_{t-1}, u_t, z_t, m(k)\right)$
 6:             $r_t = \rho\left(x_t, \chi_t\right)$
 7:             calculate $e_t$ according to (3.12) from $\chi_t$, $\bar{\chi}_t$, $z_t$, and $m(k)$
 8:             $U_t = U_{t-1} + e_t$
 9:             $V_t = V_{t-1} + (r_t - b)U_t$
10:         end for
11:         $m(k+1) = m(k) + \eta(k)V_T$
12:     end for
13:     return $m(k_{\max})$
14: end function
```

at time $T$, the map $m(k)$ is modified by adding $\eta(k)V_T$.

### 3.2.3 Theoretical analysis

The following theorem shows the optimality of the proposed method.

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

*(C1) There exists a bounded set $\mathbb{X} \subset \mathbb{R}^n$ such that for all $(k,t,i) \in (\{0\} \cup \mathbb{N}) \times \{1,2,\ldots,T\} \times \{1,2,\ldots,N\}$, $\hat{x}^{(i)}_{t|t-1} \in \mathbb{X}$ with probability 1.*

*(C2) There exist $\delta_1, \delta_2 \in \mathbb{R}$ such that for all $k \in \{0\} \cup \mathbb{N}$, $m(k) \in \mathbb{M}$ where $\mathbb{M} := \left\{m \,\middle|\, \delta_1 \leq m^{(\mu)} \leq \delta_2,\ \mu = 1,2,\ldots,M\right\}$.*

*(C3) There exists a $\delta_3 \in \mathbb{R}_+$ such that for all $(t,\hat{x},m) \in \{1,2,\ldots,T\} \times \mathbb{X} \times \mathbb{M}$, $|g\left(z_t,\hat{x},m\right)| \geq \delta_3$.*

*(C4) For all $(t,\hat{x},m) \in \{1,2,\ldots,T\} \times \mathbb{X} \times \mathbb{M}$, $\nabla_m g\left(z_t,\hat{x},m\right)$ exists, $\|\nabla_m g\left(z_t,\hat{x},m\right)\| < \infty$, and $\nabla_m g\left(z_t,\hat{x},m\right)$ is continuous in m.*

*(C5) For all $(t,\chi) \in \{1,2,\ldots,T\} \times \mathbb{X}^N$, $|\rho\left(x_t,\chi\right)| < \infty$.*

*(C6) The learning rate $\eta(k) \in \mathbb{R}_+$ satisfies $\sum_{k=0}^{\infty} \eta(k) = \infty$ and $\sum_{k=0}^{\infty} \eta(k)^2 < \infty$.*

*(C7) For variables $\mathbf{t} \in \mathbb{R}$ and $\nu \in \mathbb{R}^M$, the differential equation*

$$\frac{d\nu(\mathbf{t})}{d\mathbf{t}} = \nabla_\nu \bar{J}\left(\nu(\mathbf{t})\right) \tag{3.16}$$

*has an asymptotically stable equilibrium* $v = m^* \in \mathbb{M}$.

*(C8) There exists a compact set* $\mathbb{B}_1 \subseteq \mathbb{B}(m^*)$ *such that* $\left|\left\{k \,\middle|\, m(k) \in \mathbb{B}_1,\ k = 0, 1, 2, \dots\right\}\right| = \infty$*, where* $\mathbb{B}(m^*) := \left\{v_0 \,\middle|\, \lim_{\mathbf{t}\to\infty} v(\mathbf{t}|v_0) = m^*\right\}$*, and* $v(\mathbf{t}|v_0) \in \mathbb{R}^M$ *is the solution of* (3.16) *for an initial value* $v_0 \in \mathbb{R}^M$.

*Then,*

$$\lim_{k\to\infty} m(k) = m^*, \quad \text{with probability 1.} \tag{3.17}$$

*Proof.* The equation in line 11 of Algorithm 2 can be rewritten as

$$m(k+1) = m(k) + \eta(k)\nabla_m \bar{J}\left(m(k)\right) + \eta(k)\varphi(k) \tag{3.18}$$

where

$$\varphi(k) := V_T - \nabla_m \bar{J}\left(m(k)\right). \tag{3.19}$$

Then, if (C2), (C6)–(C8), and the following conditions are satisfied, (3.17) holds by Theorem 2.3.1 in [63].

(D1) For all $k \in \{0\} \cup \mathbb{N}$, $E\left[V_T\right] = \nabla_m \bar{J}\left(m(k)\right)$.

(D2) The function $\nabla_m \bar{J}(m)$ is continuous at all $m \in \mathbb{M}$.

(D3) For all $\epsilon \in \mathbb{R}_+$, $\lim_{k\to\infty} P\left(\sup_{K\geq k} \left\|\sum_{k'=k}^{K} \eta(k')\varphi(k')\right\| \geq \epsilon\right) = 0$.

Conditions (D1)–(D3) hold if (C1)–(C6) are satisfied. Appendix A gives the proofs. ■

Since Algorithm 2 is based on the RL method of [59], (D1) holds. Furthermore, (D1) implies that Algorithm 2 is also equivalent to the stochastic approximation method [63, 64], which updates the parameter according to the gradient of the objective function. The theorem in [63] allows us to prove Theorem 1.

Theorem 1 reveals that if (C1)–(C8) are satisfied, the proposed method converges to a local optimum, which means that the method can provide an approximate solution to Problem 1.

Finally, a supplementary explanation of the conditions in the theorems is provided. Conditions (C1), (C3), and (C4) are about the setting of MCL. The satisfaction of (C1) depends on the structure of $f$ and the boundedness of $\chi_0$ and $u_{1:T}$. Conditions (C3) and (C4) hold if, e.g., $g$ is given by (2.4), and (C2) holds with $\delta_1 > -\kappa_3/\kappa_2$. Conditions (C2) and (C5)–(C8) are for RL and stochastic approximation. For example, the function

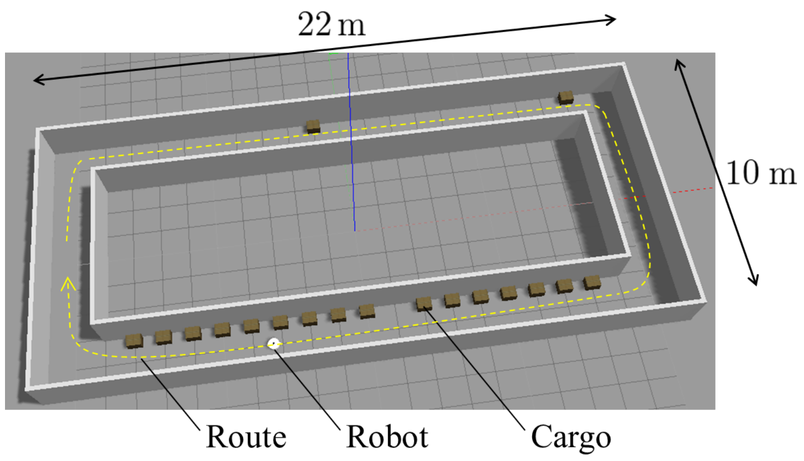


Figure 3.2: Simulation environment

$\rho$ given by (3.2) or (3.3) satisfies (C5). Condition (C6) holds by setting the learning rate $\eta(k)$ appropriately. Conditions (C2), (C7), and (C8) are generally assumed in stochastic approximation methods [63, 64]. Condition (C2) can be omitted by using the optimization algorithm proposed in [64], which explicitly keeps the solution in the constraint set.

## 3.3 Numerical simulation

To clarify the effectiveness of the highlighted map, this section compares it with a conventional likelihood field map 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 the proposed method is especially effective for a low-cost robot.

In this section, the proposed method is evaluated in the 3D dynamic simulator Gazebo [65]. The simulation environment is shown in Fig. 3.2. Cargo lines some of the corridors. The robot moves on a plane, so its pose is expressed as (2.1). The robot has a 2D LIDAR and can obtain wheel odometry data $u_t$ and LIDAR measurement data $z_t$. The specification of the LIDAR is as follows: possible maximum detection range is 5 m, scanning angle is 360 degrees, and the number of data is $L := 18$ data/rev. The MCL setting is as follows: the number of the particles is $N := 100$, frequency is 5 Hz, the function $f$ is algorithm *sample_motion_model_odometry* in Table 5.6 in [4] with sampling algorithm *sample_normal_distribution* in Table 5.4 in [4], and $g$ is given by (2.4) where $\kappa_2 := 0.6$ and $\kappa_3 := 0.4$. These specifications were chosen assuming a low-cost robot.

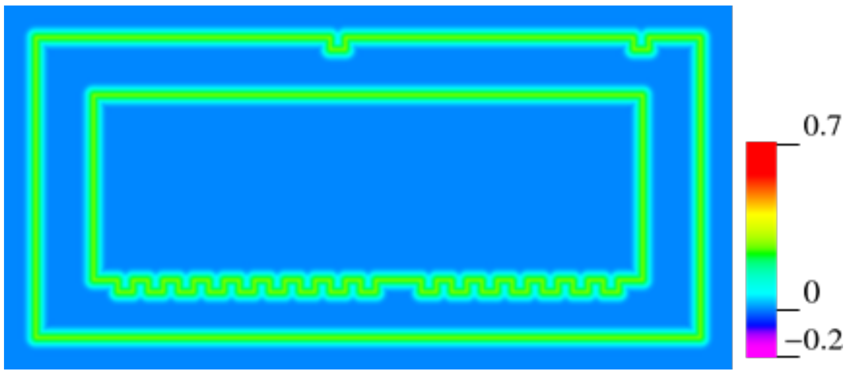


Figure 3.3: Initial map $m(0)$ (conventional likelihood field map) in simulation

### 3.3.1 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. 3.3, which is the same map as in Fig. 1.1(b), except that it is represented on a color scale. This is a likelihood field map generated by (2.2) with $\kappa_1 := 0.1$, $\sigma := 0.15$, and an occupancy grid map $m_{\text{occ}}$ in Fig. 1.1(a). The map size (number of cells) is $M := 240 \times 140$, and the grid scale is 0.1 m/cell. The training data used for generating the map $u_{1:T}$, $z_{1:T}$, and $x_{1:T}$ (denoted by $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$) were obtained by the following procedure:

- The control data $u_{1:T}^{\text{train}}$ and measurement data $z_{1:T}^{\text{train}}$ were measured while the robot followed the route in Fig. 3.2 at 1 m/s.

- The true poses $x_{1:T}^{\text{train}}$ were approximately obtained as follows: perform MCL with $N := 10{,}000$ particles (a larger number than usual for this robot) by using $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $m(0)$; then, regard the following maximum likelihood pose as $x_t^{\text{train}}$,

$$x_t^{\text{train}} := \hat{x}_{t|t-1}^{(i^*)} \qquad \text{s.t.} \;\; i^* = \arg\max_i \; w_t^{(i)}. \tag{3.20}$$

  Note that the true poses could be obtained from the simulator, however, it was not used for generating the map in order to simulate a realistic situation.

The initial particle set is given by $\chi_0 := \left\{\hat{x}^{\text{ini}}, \hat{x}^{\text{ini}}, \ldots, \hat{x}^{\text{ini}}\right\}$ where $\hat{x}^{\text{ini}} \in \mathbb{R}^3$ is the initial true pose. The function $\rho$ in (3.1) is set as

$$\rho\left(x_t, \chi_t\right) := \exp\left\{-\left(x_t - h\left(\chi_t\right)\right)^\top \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(x_t - h\left(\chi_t\right)\right)\right\} \tag{3.21}$$

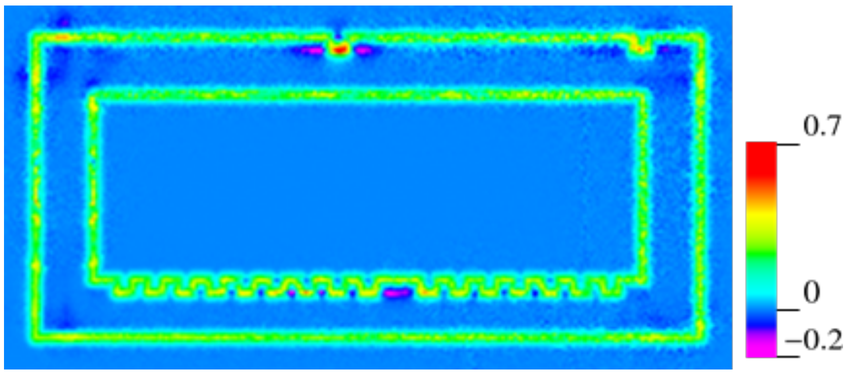


Figure 3.4: Highlighted map $m^*$ in simulation

where $h$ is defined as (3.4). In Algorithm 2, the learning rate $\eta(k)$ is

$$\eta(k) := \frac{1}{k + 3 \times 10^6}, \tag{3.22}$$

maximum iteration number is $k_{\max} := 4 \times 10^6$, and reinforcement baseline is $b := 0.85$. The reinforcement baseline is set to reduce the estimated variance of the equation in (D1).

Conditions (C1) and (C3)–(C6) hold under these settings, and it was numerically confirmed that (C2), (C7), and (C8) held.

The highlighted map $m^*$ generated by Algorithm 2 is shown in Fig. 3.4. In the north corridor, the cargo at the center is highlighted most strongly (red spot). In contrast, the cargo at the northeast 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 south 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 are properly identified and emphasized.

The change in the objective function is illustrated in Fig. 3.5. 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, i.e., only the data at $k = 0, 1000, 2000, \ldots$ are displayed. Figure 3.5 indicates that convergence is achieved.

Note that a similar highlighted map will be generated if $\rho$ is given by (3.3) instead of (3.21). In this case, the process of obtaining true poses $x_{1:T}$ can be omitted because (3.3) does not require $x_t$.

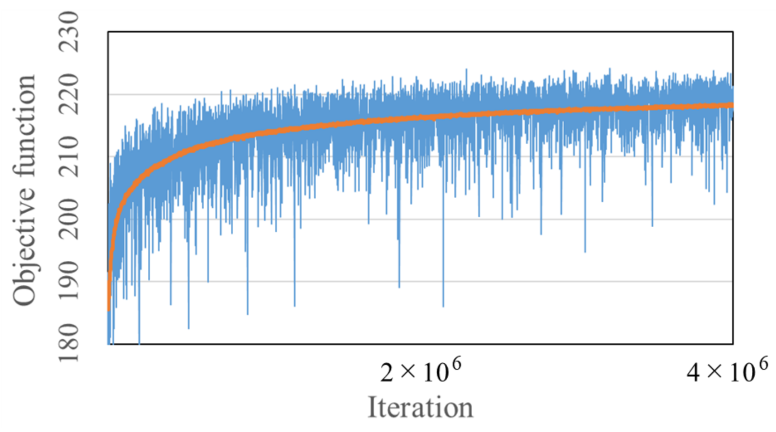


Figure 3.5: Change in objective function in simulation

### 3.3.2 Performance of highlighted map

Next, the generated highlighted map $m^*$ was verified. The test 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 3.3.1. Note that these values are different from $u_{1:T}^{\text{train}}$ and $z_{1:T}^{\text{train}}$ because the robot was operated manually and traced a route slightly different from that of Section 3.3.1.
- 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 (3.21),
- Estimation error:

$$\Delta_t := \sqrt{\left(x_t^{\text{test}} - h\left(\chi_t\right)\right)^\top \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(x_t^{\text{test}} - h\left(\chi_t\right)\right)} \tag{3.23}$$

  with (3.4),
- Computation time for one estimation cycle $\Upsilon_t \in \mathbb{R}_+$.

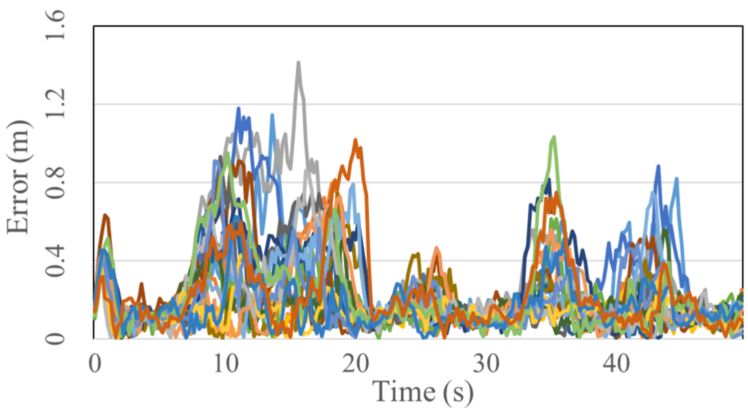
The values of $J$ and $\Delta_t$ were calculated with $x_{1:T}^{\text{test}}$ and $\chi_{1:T}$. Also, $\chi_{1:T}$ were computed by MCL with $u_{1:T}^{\text{test}}$ and $z_{1:T}^{\text{test}}$. The computation time was measured by using a Broadcom BCM2837 system on a chip (SoC) with a 1.2 GHz 64-bit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

The following two comparisons were carried out. Firstly, the highlighted map was compared 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 was 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, this section chose KLD-sampling-based MCL (KLD-MCL) [4, 8] and the algorithm introduced in [10], which this section calls 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.

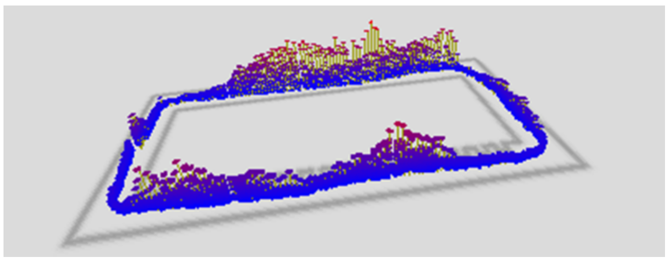
For the above two comparisons, this section performed six simulations denoted by S1–S6, whose configurations and results are shown in Table 3.1. Table 3.1 summarizes the values of the objective function $J$, the mean and maximum values of the estimation error $\Delta_t$, and the mean and maximum values of the computation time $\Upsilon_t$. 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. 3.6 and 3.7, respectively. Furthermore, the statistical significance of the improvement was evaluated by box plots presented in Fig. 3.8 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, $p < 0.001$). 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 combing the highlighted map with other MCL-based algorithms. In addition, by comparing the mean and maximum values of $\Delta_t$ of S2 with those of S3 and S5, 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.

Table 3.1: Localization accuracy and computation time for each configuration in simulation

| Configuration | | | Objective function $J$ | Estimation error $\Delta_t$ (m) | | Computation time $\Upsilon_t$ (ms/cycle) | |
|---|---|---|---|---|---|---|---|
| Index | Algorithm | Map | | $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ | $\max_t \Delta_t$ | $\frac{1}{T}\sum_{t=1}^{T}\Upsilon_t$ | $\max_t \Upsilon_t$ |
| S1 | MCL | Conventional map (Fig. 3.3) | 166.4 | 0.22 | 0.83 | 6.11 | 6.75 |
| S2 | MCL | Highlighted map (Fig. 3.4) | 209.1 | 0.12 | 0.42 | 6.06 | 6.59 |
| S3 | KLD-MCL | Conventional map (Fig. 3.3) | 171.1 | 0.20 | 0.61 | 49.47 | 97.89 |
| S4 | KLD-MCL | Highlighted map (Fig. 3.4) | 217.1 | 0.11 | 0.29 | 40.87 | 73.14 |
| S5 | SM-MCL | Conventional map (Fig. 3.3) | 179.4 | 0.18 | 0.51 | 41.72 | 73.00 |
| S6 | SM-MCL | Highlighted map (Fig. 3.4) | 218.9 | 0.11 | 0.25 | 37.69 | 66.79 |



(a) Error at each time



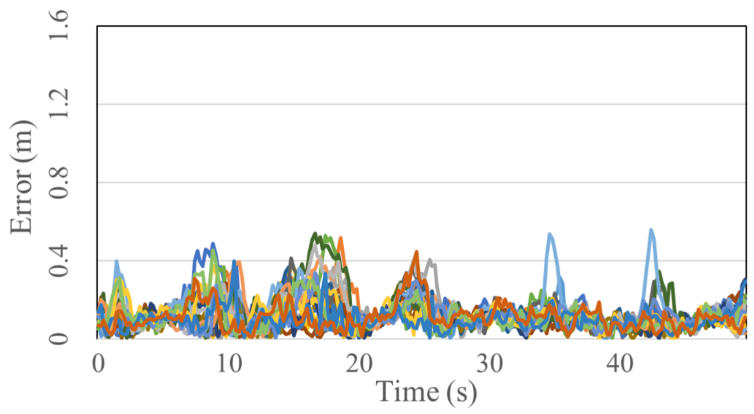
(b) Error at each position

Figure 3.6: Estimation errors with combination of MCL and conventional likelihood field map $m(0)$ in simulation

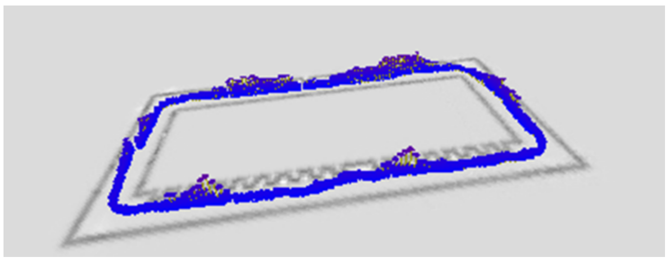
Note that a highlighted map was generated from the actual robot data $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$; thus, the generated highlighted map should be specialized for situations such as sensor characteristics and robot dynamics. For example, since $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$ 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. 3.4. As described here, the proposed method can generate various highlighted maps adapted to each situation.

## 3.4 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. 3.9 and the laboratory shown in Fig. 3.10. Each picture in these figures is a landscape from the viewpoint

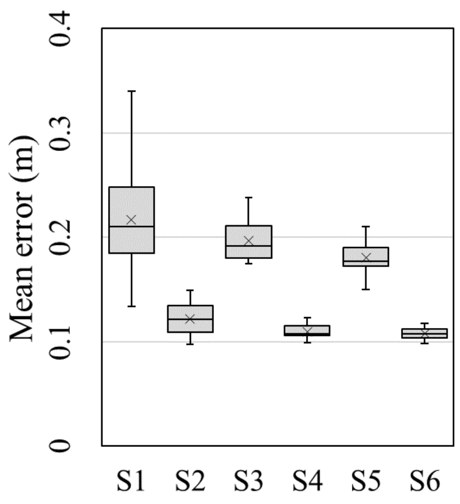


(a) Error at each time

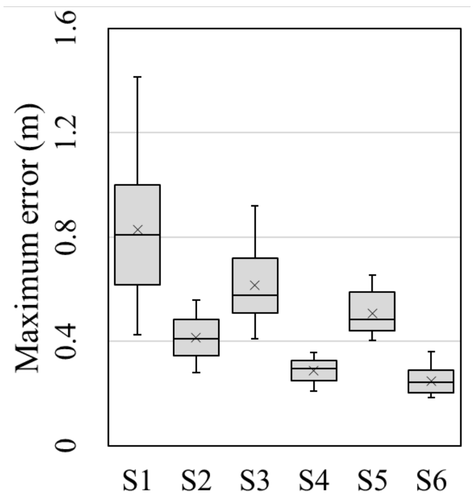


(b) Error at each position

Figure 3.7: Estimation errors with combination of MCL and highlighted map $m^*$ in simulation



(a) Mean error $\left(\frac{1}{T}\sum_{t=1}^{T}\Delta_t\right)$



(b) Maximum error ($\max_t \Delta_t$)

Figure 3.8: Box plots of mean and maximum of estimation errors in simulation

of the arrow. Figure 3.11 shows the robot system. A 2D LIDAR mounted on the robot was RPLIDAR (SLAMTEC), whose possible maximum detection range, scanning angle, and the

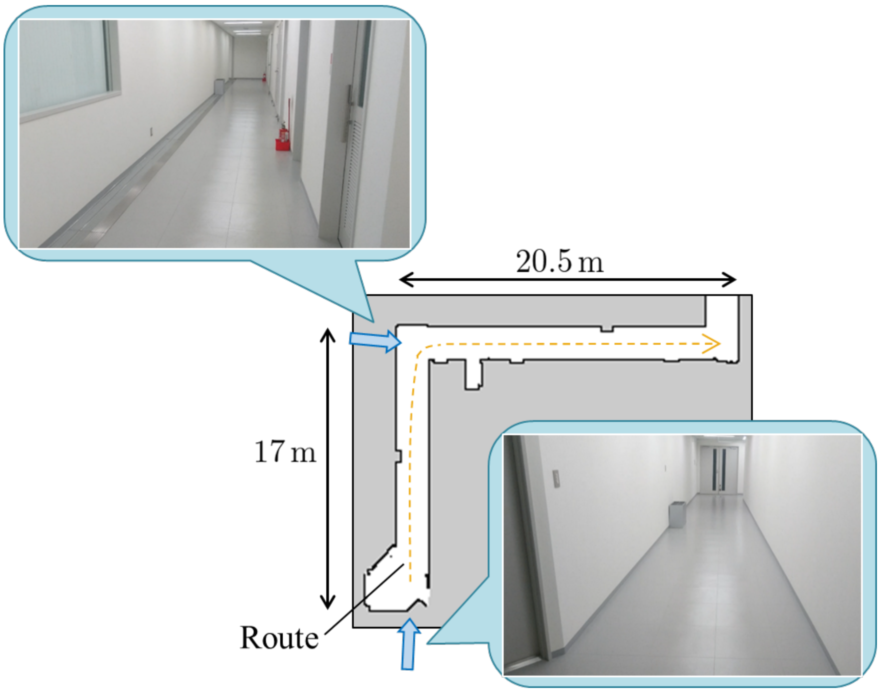


Figure 3.9: Experimental environment (corridor)

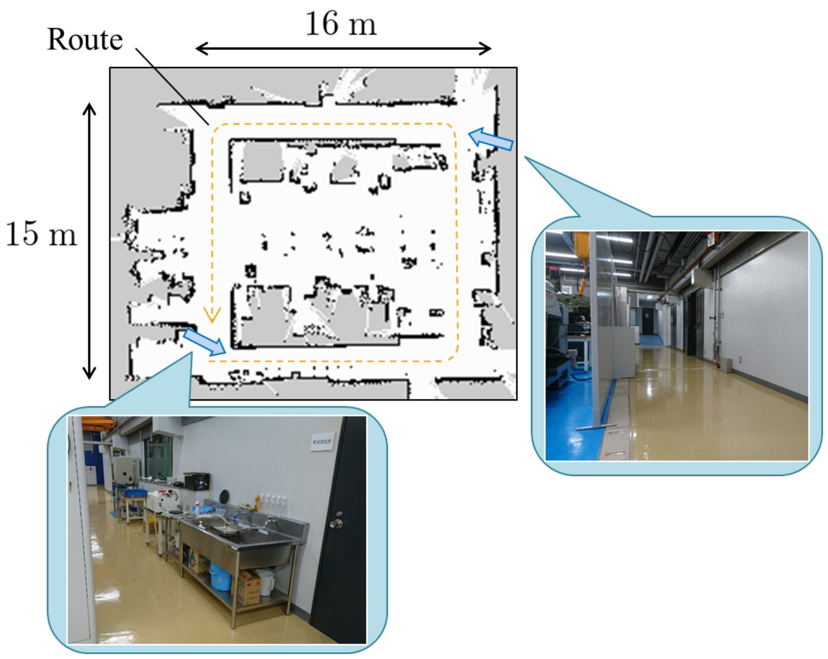


Figure 3.10: Experimental environment (laboratory)

number of data were 12 m, 360 degrees, and 360 data/rev, respectively. However, this section 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.

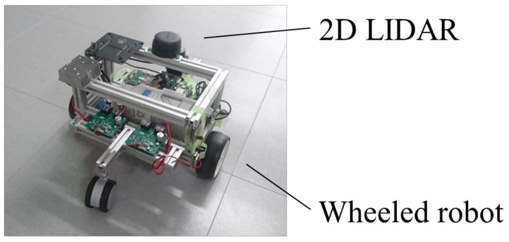


Figure 3.11: Experimental robot system

First, the highlighted maps $m^*$ were generated. The initial maps $m(0)$ in the corridor and laboratory experiments are illustrated in Figs. 3.12 and 3.13, which are likelihood field maps generated by (2.2) with $\kappa_1 := 0.1$, $\sigma := 0.15$, and the occupancy grid maps $m_{\mathrm{occ}}$ in Figs. 3.9 and 3.10, respectively. The data for generating the maps $u_{1:T}^{\mathrm{train}}$, $z_{1:T}^{\mathrm{train}}$, and $x_{1:T}^{\mathrm{train}}$ were measured in the same way as in Section 3.3.1, where the routes followed by the robot when collecting $u_{1:T}^{\mathrm{train}}$ and $z_{1:T}^{\mathrm{train}}$ are shown in Figs. 3.9 and 3.10. The map sizes $M$ and reinforcement baselines $b$ were $240 \times 200$ and 0.85 in the corridor experiment, $220 \times 180$ and 0.9 in the laboratory experiment, respectively. All other parameters and functions were the same as in Section 3.3.1. The generated highlighted maps and the changes in the objective function in the corridor and laboratory experiments are shown in Figs. 3.14, 3.15, 3.16, and 3.17, respectively. In Fig. 3.14, objects in the middle of the corridors, where localization is difficult, are highlighted. Also, in Fig. 3.15, important landmarks are highlighted, e.g., the object in the south corridor (indicated by a circle below) and the place without walls in the north corridor (indicated by a circle above). The meanings of the lines in Figs. 3.16 and 3.17 are the same as in Fig. 3.5. As in Section 3.3.1, it can be seen that the landmarks useful for localization are highlighted, and convergence is achieved in both environments.

Next, the generated highlighted maps $m^*$ was evaluated. The measurement procedure of $u_{1:T}^{\mathrm{test}}$ and $z_{1:T}^{\mathrm{test}}$, and the evaluation method were the same as in Section 3.3.2. The true poses $x_{1:T}^{\mathrm{test}}$ were approximately calculated from $u_{1:T}^{\mathrm{test}}$, $m(0)$, and the complete data obtained from RPLIDAR (possible maximum detection range was 12 m, and the number of data was 360 data/rev) instead of $z_{1:T}^{\mathrm{test}}$ (possible maximum detection range was 5 m, and the number of data was 18 data/rev). The results of the corridor experiment are shown in Table 3.2, Figs. 3.18, 3.19, and 3.20. Table 3.2 summarizes the values of $J$, $\Delta_t$, and $\Upsilon_t$ for each configuration denoted by C1–C6. All these values are the averages over 20 tests. Figures 3.18 and 3.19 show the errors (3.23) of 20 tests in the case of C1 and C2, respectively. Figure 3.20 illustrates box plots of the errors. Similarly, the results of the laboratory experiment, denoted by L1–L6, are presented in Table 3.3, Figs. 3.21, 3.22, and 3.23. The mean

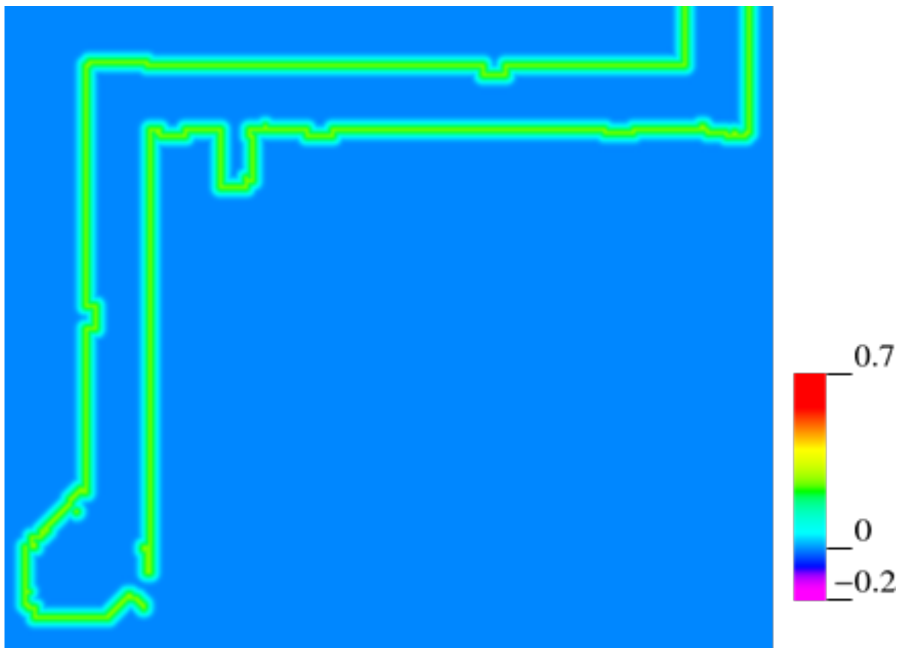


Figure 3.12: Initial map $m(0)$ (conventional likelihood field map) in corridor experiment

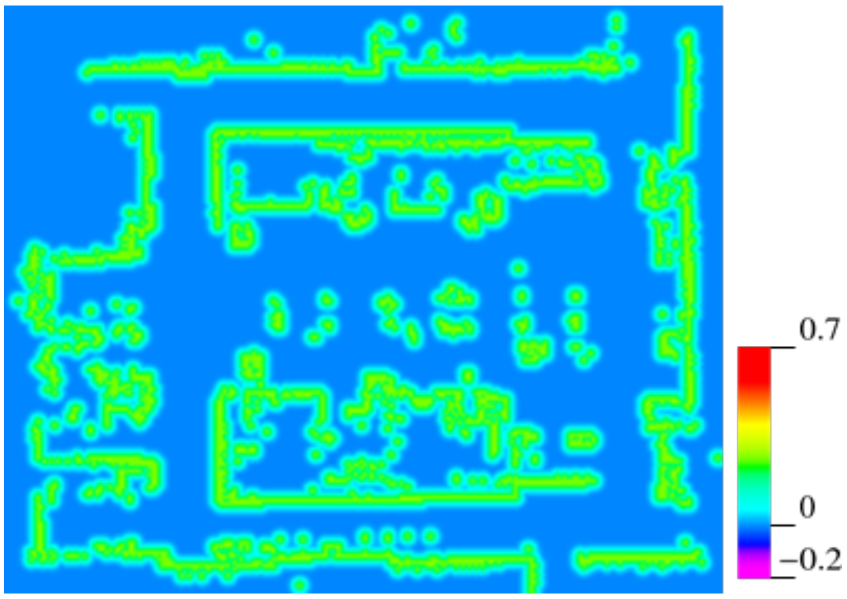


Figure 3.13: Initial map $m(0)$ (conventional likelihood field map) in laboratory experiment

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 3.3.2, 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

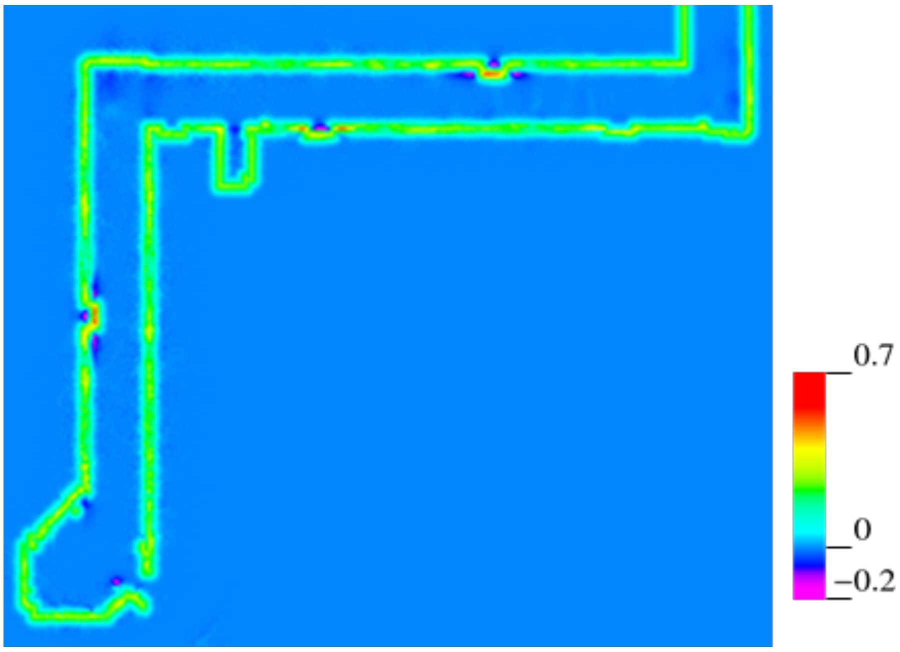


Figure 3.14: Highlighted map $m^*$ in corridor experiment

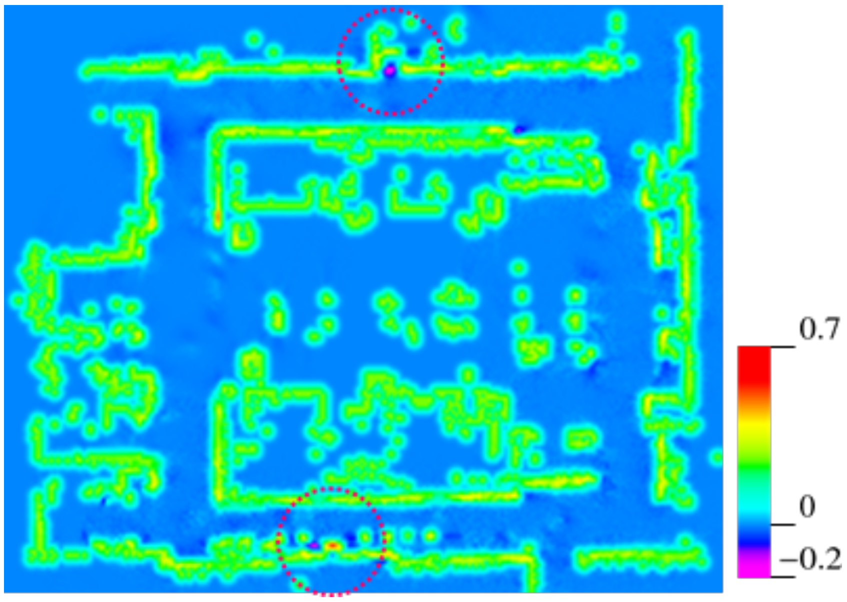


Figure 3.15: Highlighted map $m^*$ in laboratory experiment

was comparable or better than the other existing algorithms, regardless of the fact that the proposed 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.

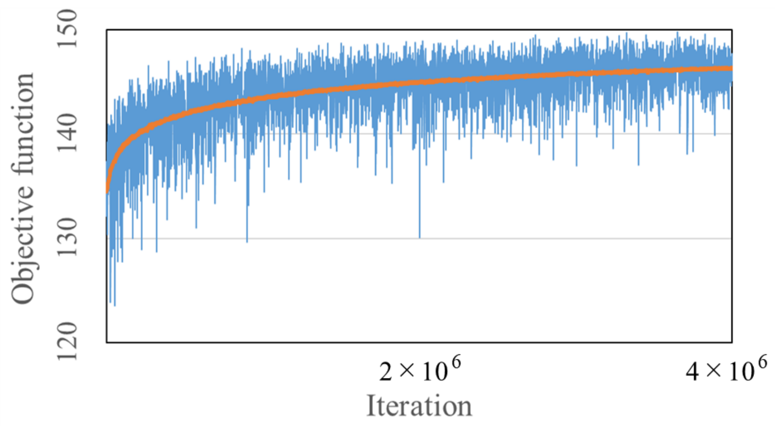


Figure 3.16: Change in objective function in corridor experiment

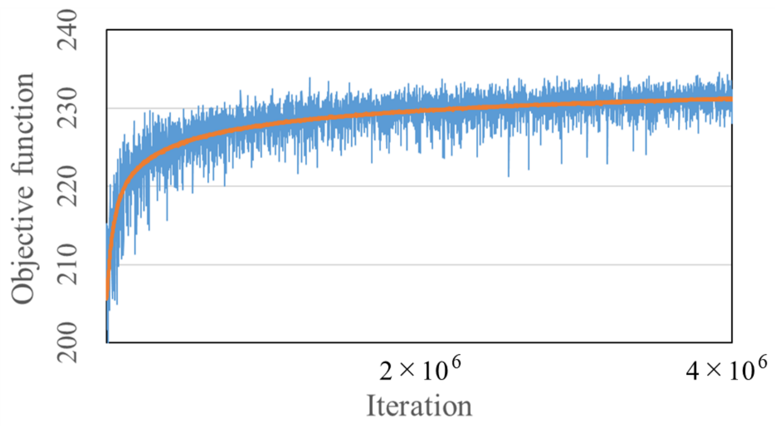
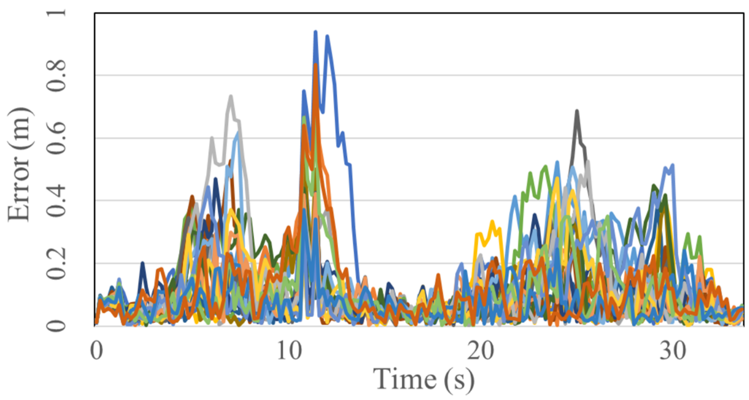


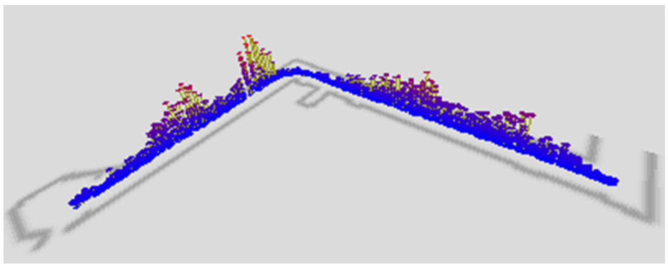
Figure 3.17: Change in objective function in laboratory experiment

Table 3.2: Localization accuracy and computation time for each configuration in corridor experiment

| Configuration | | | Objective function $J$ | Estimation error $\Delta_t$ (m) | | Computation time $\Upsilon_t$ (ms/cycle) | |
|---|---|---|---|---|---|---|---|
| Index | Algorithm | Map | | $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ | $\max_t \Delta_t$ | $\frac{1}{T}\sum_{t=1}^{T}\Upsilon_t$ | $\max_t \Upsilon_t$ |
| C1 | MCL | Conventional map (Fig. 3.12) | 143.6 | 0.11 | 0.58 | 6.00 | 6.57 |
| C2 | MCL | Highlighted map (Fig. 3.14) | 155.7 | 0.07 | 0.36 | 6.00 | 6.54 |
| C3 | KLD-MCL | Conventional map (Fig. 3.12) | 153.9 | 0.08 | 0.41 | 43.47 | 68.67 |
| C4 | KLD-MCL | Highlighted map (Fig. 3.14) | 160.7 | 0.06 | 0.24 | 39.58 | 64.56 |
| C5 | SM-MCL | Conventional map (Fig. 3.12) | 154.8 | 0.07 | 0.40 | 39.25 | 58.34 |
| C6 | SM-MCL | Highlighted map (Fig. 3.14) | 161.7 | 0.05 | 0.28 | 37.00 | 58.52 |

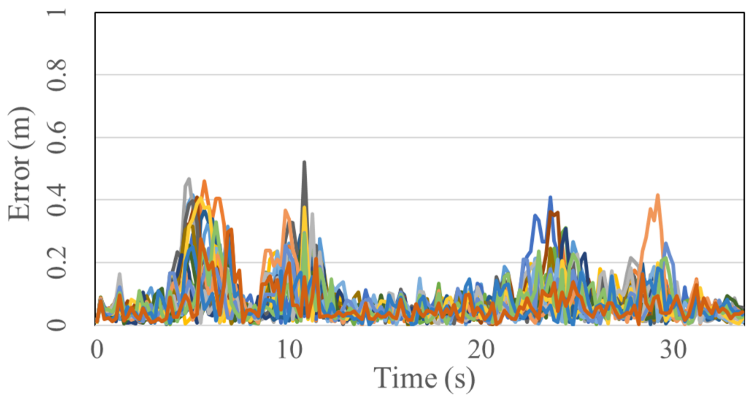


(a) Error at each time

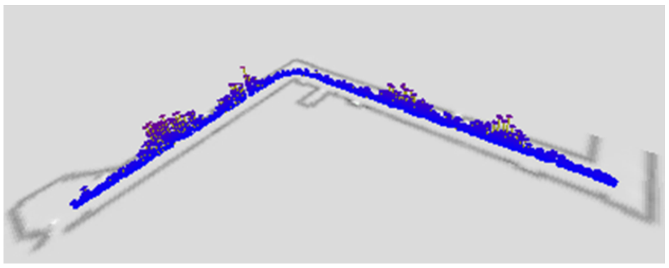


(b) Error at each position

Figure 3.18: Estimation errors with combination of MCL and conventional likelihood field map $m(0)$ in corridor experiment

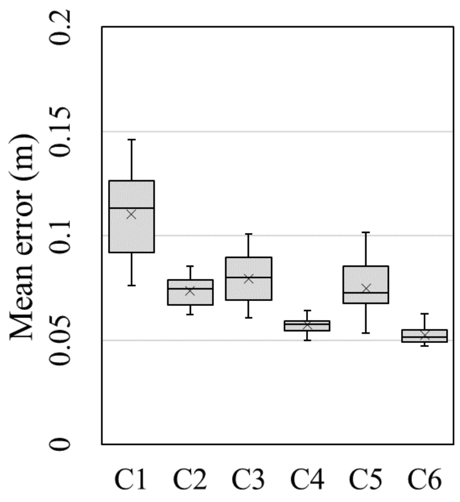


(a) Error at each time

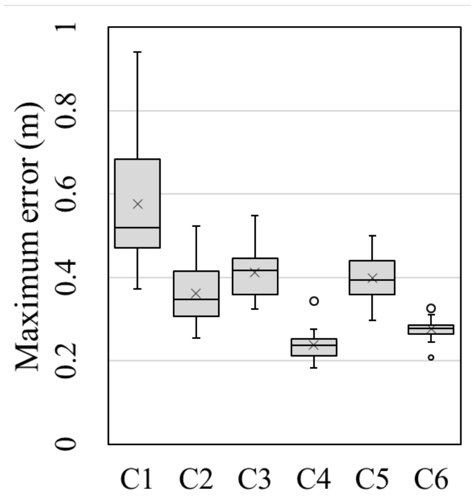


(b) Error at each position

Figure 3.19: Estimation errors with combination of MCL and highlighted map $m^*$ in corridor experiment



(a) Mean error $\left(\frac{1}{T}\sum_{t=1}^{T}\Delta_t\right)$

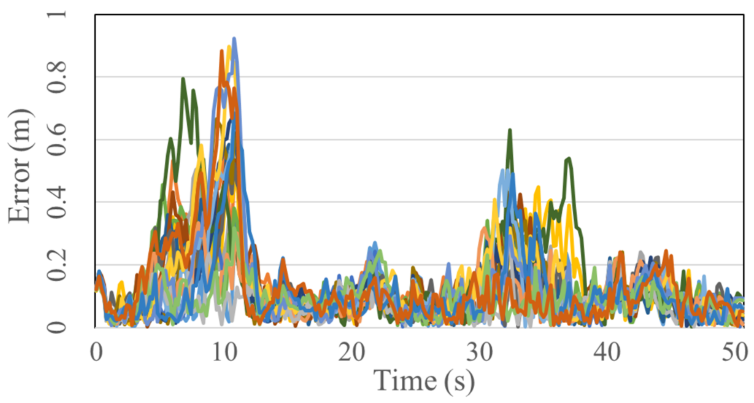


(b) Maximum error $(\max_t \Delta_t)$

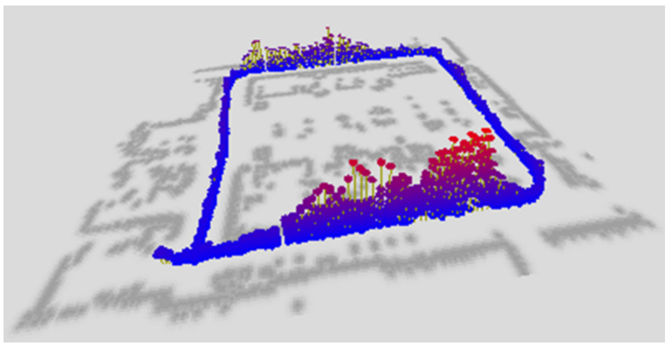
Figure 3.20: Box plots of mean and maximum of estimation errors in corridor experiment

Table 3.3: Localization accuracy and computation time for each configuration in laboratory experiment

| Configuration | | | Objective function $J$ | Estimation error $\Delta_t$ (m) | | Computation time $\Upsilon_t$ (ms/cycle) | |
|---|---|---|---|---|---|---|---|
| Index | Algorithm | Map | | $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ | $\max_t \Delta_t$ | $\frac{1}{T}\sum_{t=1}^{T}\Upsilon_t$ | $\max_t \Upsilon_t$ |
| L1 | MCL | Conventional map (Fig. 3.13) | 213.3 | 0.12 | 0.58 | 6.08 | 6.71 |
| L2 | MCL | Highlighted map (Fig. 3.15) | 237.7 | 0.07 | 0.31 | 6.05 | 6.62 |
| L3 | KLD-MCL | Conventional map (Fig. 3.13) | 223.5 | 0.10 | 0.45 | 48.35 | 94.69 |
| L4 | KLD-MCL | Highlighted map (Fig. 3.15) | 244.6 | 0.05 | 0.20 | 42.48 | 79.63 |
| L5 | SM-MCL | Conventional map (Fig. 3.13) | 221.6 | 0.10 | 0.40 | 42.13 | 74.44 |
| L6 | SM-MCL | Highlighted map (Fig. 3.15) | 245.8 | 0.05 | 0.19 | 38.58 | 67.43 |



(a) Error at each time

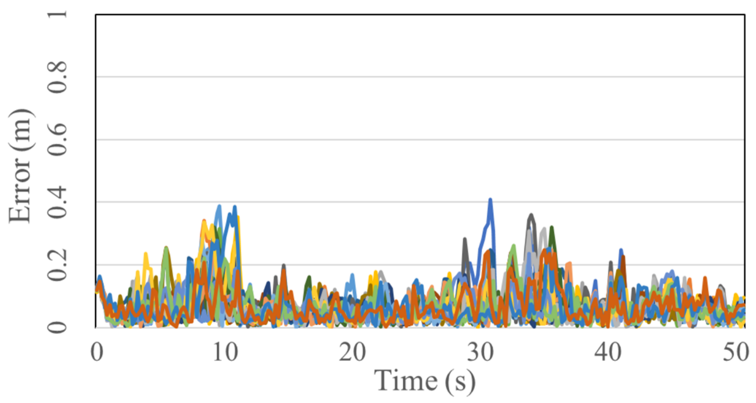


(b) Error at each position

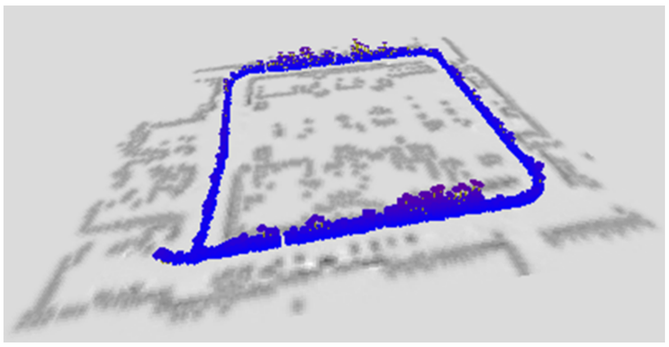
Figure 3.21: Estimation errors with combination of MCL and conventional likelihood field map $m(0)$ in laboratory experiment

## 3.5 Conclusion

This chapter 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 chapter only generated 2D highlighted maps, the proposed method can also create 3D highlighted maps.

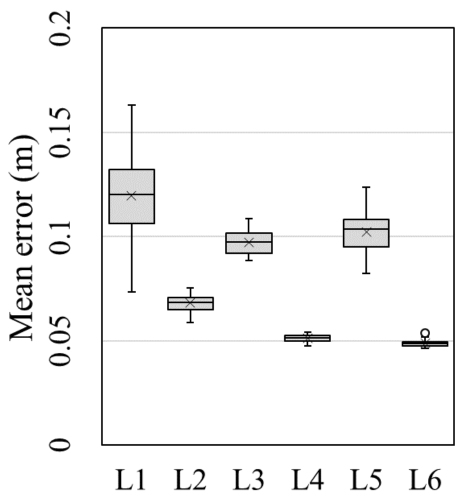


(a) Error at each time

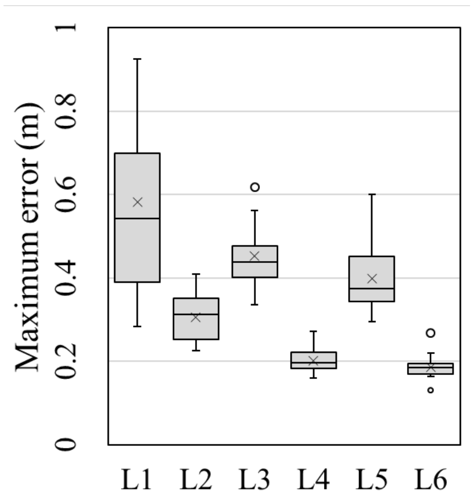


(b) Error at each position

Figure 3.22: Estimation errors with combination of MCL and highlighted map $m^*$ in laboratory experiment



(a) Mean error $\left(\frac{1}{T}\sum_{t=1}^{T}\Delta_t\right)$



(b) Maximum error $(\max_t \Delta_t)$

Figure 3.23: Box plots of mean and maximum of estimation errors in laboratory experiment

# Chapter 4

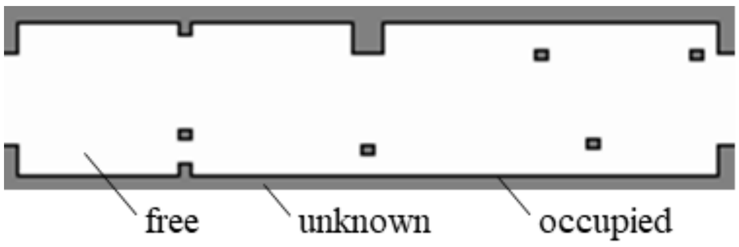
# Robust highlighted map generation

In this chapter, a method of generating a robust highlighted map based on adversarial RL is proposed. First, a virtual obstacle (e.g., assuming a pedestrian) that generates measurement noise to interfere with the localization is considered. Next, this chapter proposes an algorithm that simultaneously learns the worst-case obstacle behavior and the optimal highlighted map in a competitive manner to output a robust highlighted map. Finally, it is confirmed by performing a numerical simulation that the proposed robust highlighted map gives highly accurate estimation even in the presence of measurement noise.

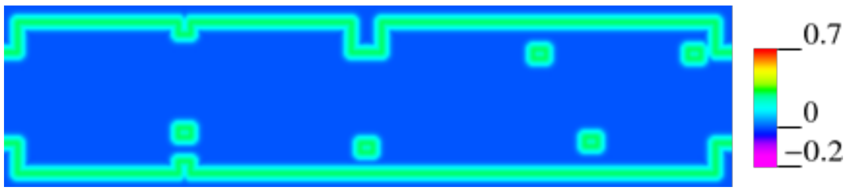
As in Chapter 3, this chapter also treats MCL described in Section 2.2 as a localization algorithm.

## 4.1 Problem in highlighted map

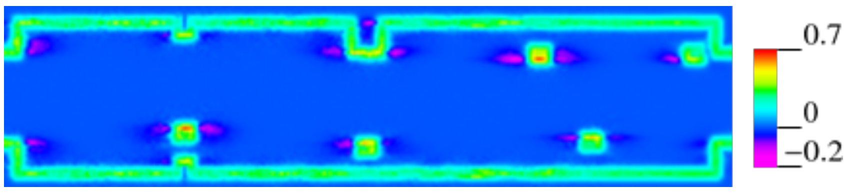
The highlighted map proposed in Chapter 3 is a grid map where landmarks useful for localization are highlighted. The example of the map is shown in Fig. 4.1. Figure 4.1(a) is the original map (occupancy grid map), where the color indicates the probability that the space is occupied by an object. Figure 4.1(b) is the likelihood field map, which is generated by blurring the original map by the magnitude of measurement noise. Figure 4.1(c) is the highlighted map. The environment in the map is a corridor with a large pillar at the north center and several small square objects. Outside the vicinity of the pillar, localization with Fig. 4.1(b) tends to fail since there are only small landmarks. In Fig. 4.1(c), the objects at the southwest and the northeast, which are useful landmarks, are highlighted in red. By using the highlighted map as $m$ in MCL, the pose estimates (particles) where the measurement data $z_t$ overlaps these landmarks will be evaluated more in line 5 of Algorithm 1, and as a



(a) Original map (occupancy grid map)



(b) Likelihood field map



(c) Highlighted map

Figure 4.1: Example of highlighted map

result, the localization accuracy will improve.

Chapter 3 also proposed the method of generating a highlighted map. This is an RL-based method that performs the following two steps repeatedly: 1) replay the control and measurement data recorded by the actual robot and perform (offline) localization; 2) modify the shade of the map gradually according to the localization results. This method can automatically detect and highlight landmarks useful for localization.

However, the method in Chapter 3 does not take into account any measurement noise caused by dynamic obstacles that are not drawn on the map. If highlighted landmarks in the highlighted map are occluded by dynamic obstacles during localization, the map may not be able to perform well.

## 4.2 Virtual obstacle

In order to solve this problem, this chapter consider introducing a virtual obstacle, which simulates, e.g., a pedestrian wandering around the robot. If the highlighted map is generated by RL where localization is performed under the influence of noise caused by the virtual obstacle, the generated highlighted map should be robust against the same kind of actual noise.

The dynamics of the virtual obstacle is defined by the following state equation:

$$\begin{cases} \xi_t = f^{\mathrm{o}}\left(\xi_{t-1}, d_t\right) \\ \tilde{z}_t = g^{\mathrm{o}}\left(z_t, \xi_t\right) \end{cases} \tag{4.1}$$

where $\xi_t \in \mathbb{R}^{n_\xi}$ and $d_t \in \mathbb{R}^{n_d}$ are the state and input of the virtual obstacle, respectively, and $\tilde{z}_t \in \mathbb{R}^{n_z \times L}$ is the measurement data affected by the obstacle. The function $f^{\mathrm{o}} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_\xi}$ represents the state transition, and $g^{\mathrm{o}} : \mathbb{R}^{n_z \times L} \times \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_z \times L}$ inserts the noise caused by the obstacle into $z_t$, which is the raw measurement data. Figure 4.2 shows an example, where the shape of the obstacle is a simple circle. In Fig. 4.2(b), $\xi_t := \left[r_t^{\mathrm{o}}, \theta_t^{\mathrm{o}}\right]^\top \in \mathbb{R}^2$ corresponds to the position of the obstacle in the sensor coordinates (polar coordinate representation), and $d_t := [\Delta r^{\mathrm{o}}, \Delta\theta^{\mathrm{o}}]^\top \in \mathbb{R}^2$ means the amount of movement of the obstacle. Then, $f^{\mathrm{o}}$ is given by

$$f^{\mathrm{o}}\left(\xi_{t-1}, d_t\right) := \xi_{t-1} + d_t. \tag{4.2}$$

The function $g^{\mathrm{o}}$ converts $z_t$ in Fig. 4.2(c) into $\tilde{z}_t$ in Fig. 4.2(d) based on $\xi_t$.

## 4.3 Problem formulation

Let $\mathbb{D} \subset \mathbb{R}^{n_d}$ be the set of values of the obstacle input $d_t$, and consider generating a highlighted map $m$ that best improves the localization accuracy when $d_t$ takes the worst-case value for localization at each time.

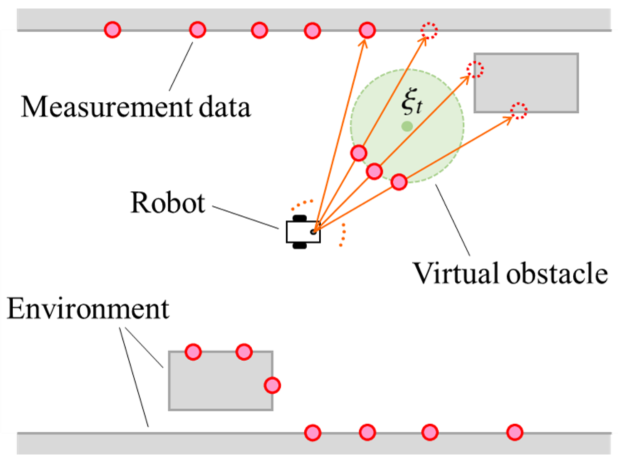
This section defines the objective function for generating a robust highlighted map in the same way as in Section 3.1, i.e.,

$$J(m, d_{1:T}) := \sum_{t=1}^{T} \rho\left(x_t, \chi_t\right). \tag{4.3}$$

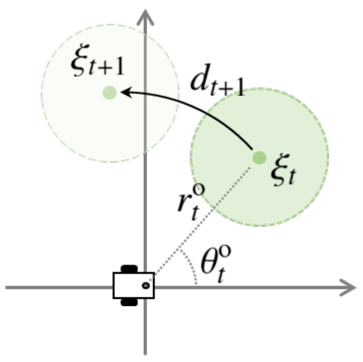
Here, the particle set $\chi_t$ is calculated by Algorithm 1 whose inputs are $\chi_{t-1}$, $u_t$, $\tilde{z}_t$, and highlighted map $m$, i.e., $\chi_t = \mathrm{MCL}\left(\chi_{t-1}, u_t, \tilde{z}_t, m\right)$.

Then, the problem in this chapter is formulated as follows.

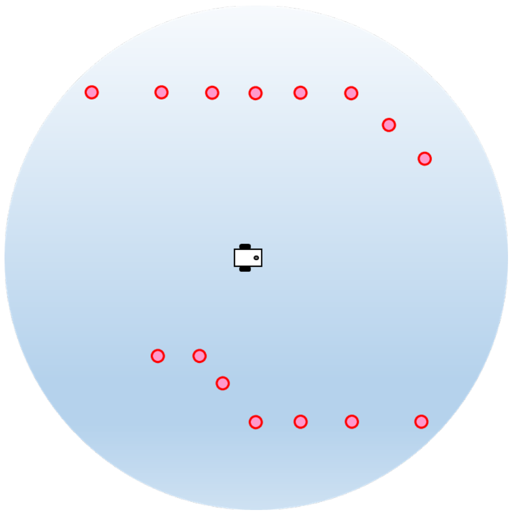
**Problem 2.** *For Algorithm 1, assume that the number $N$ of particles, functions $f$, $g$, $f^{\mathrm{o}}$, $g^{\mathrm{o}}$, and $\rho$, an initial particle set $\chi_0$, an initial obstacle state $\xi_0$, control data $u_{1:T}$, measurement*



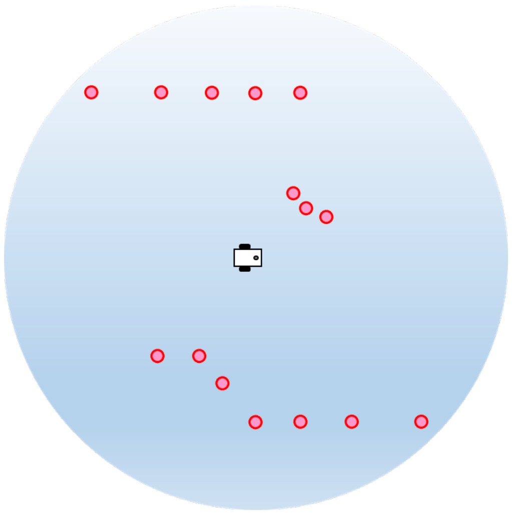
(a) Situation



(b) State $\xi_t$ of virtual obstacle and its transition



(c) Original measurement data $z_t$



(d) Measurement data $\tilde{z}_t$ affected by virtual obstacle

Figure 4.2: Relationship between virtual obstacle and measurement data

*data $z_{1:T}$, true poses $x_{1:T}$, and a set $\mathbb{D}$ of the obstacle input are given. Then, find the optimal highlighted map*

$$\arg\max_{m} \min_{d_1,d_2,\ldots,d_T\in\mathbb{D}} \bar{J}(m, d_{1:T}) \tag{4.4}$$

*where $\bar{J} : \mathbb{R}^M \times \mathbb{R}^{n_d\times T} \to \mathbb{R}$ is the expected value of* (4.3)*, i.e.,*

$$\bar{J}(m, d_{1:T}) := E\left[J(m, d_{1:T})\right]. \tag{4.5}$$

■

## 4.4 Robust highlighted map generation based on adversarial reinforcement learning

This section derives the solution to Problem 2. Problem 2 has the difficulty that as the map $m$ is optimized, the worst-case obstacle inputs $d_{1:T}$ also changes. Therefore, this section uses the framework of the adversarial RL such as RARL [39] to learn the optimal highlighted map and the worst-case obstacle behavior simultaneously in a competitive manner. The worst-case obstacle behavior is learned by a value iteration method outlined in Section 2.4, and the highlighted map is optimized by the method proposed in Chapter 3, which is based on the policy gradient method in [59].

The state of the whole system is uniquely determined from the state of all particles $\chi_t$ (or $\bar{\chi}_t$), the state of the virtual obstacle $\xi_t$, and current time $c_t := t$. Thus, the input of the obstacle should be selected from these values, i.e.,

$$d_t = \tilde{\pi}^{\mathrm{o}}\left(\chi_{t-1}, \xi_{t-1}, c_t\right) \tag{4.6}$$

where $\tilde{\pi}^{\mathrm{o}} : \mathbb{R}^{n\times N} \times \mathbb{R}^{n_\xi} \times \mathbb{N} \to \mathbb{R}^{n_d}$ is a function that selects the value of $d_t$ from the set $\mathbb{D}$.

### 4.4.1 Optimizing of highlighted map

First, focus on how to optimize the highlighted map $m$. The target system in this chapter is different from Chapter 3 because it includes a virtual obstacle. However, the map can actually be optimized in the similar way as in Chapter 3.

This section interprets MCL into the RL framework in Section 2.4 by the following assignment:

- Policy $\pi^{\mathrm{m}}$: Functions $g$ and $g^{\mathrm{o}}$, and resampling process of MCL. The highlighted map $m$ is a parameter that determines the behavior of these processes.
- State $s_t^{\mathrm{m}} = \left(\left(\hat{x}_{t|t-1}^{(1)}, \hat{x}_{t|t-1}^{(2)}, \dots, \hat{x}_{t|t-1}^{(N)}\right), \xi_t, c_t\right)$: States of all particles of MCL $\hat{x}_{t|t-1}^{(1)}, \hat{x}_{t|t-1}^{(2)}, \dots, \hat{x}_{t|t-1}^{(N)}$, state of virtual obstacle $\xi_t$, and current time $c_t := t$.
- Action $a_t^{\mathrm{m}} = \left[\gamma_t^{(1)}, \gamma_t^{(2)}, \dots, \gamma_t^{(N)}\right]^\top$: Resampling result of MCL, where $\gamma_t^{(i)} \in \mathbb{N}$ is the index $j$ of the state $\hat{x}_{t|t-1}^{(j)}$ to which the $i$-th particle is resampled in line 9 of Algorithm 1.
- Reward $r_t^{\mathrm{m}}$: Output of the function $\rho$ in (4.3).

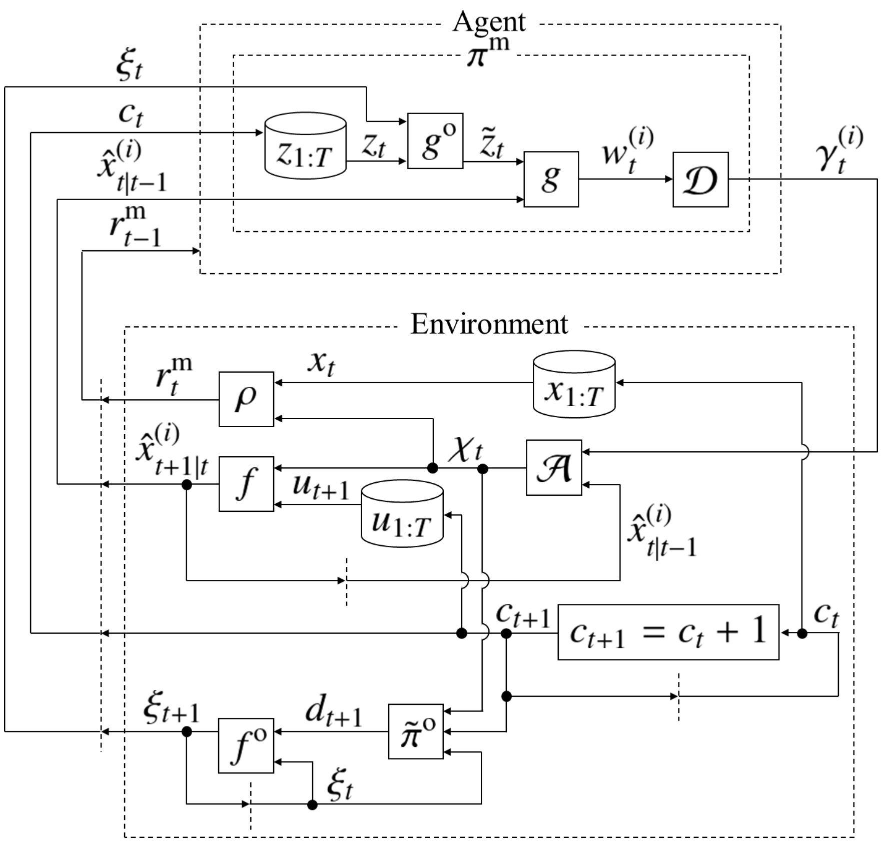


Figure 4.3: Relationship between MCL and RL in case of optimizing highlighted map

Figure 4.3 shows the relationship between MCL and RL in this case. In Fig. 4.3, $\mathcal{D}$ and $\mathcal{A}$ compose the resampling process as in Section 3.2.1. Then, the whole system can be regarded as an MDP.

Next, the characteristic eligibility (2.7) is derived. From Section 3.2.1, $\tilde{\pi}^{\mathrm{m}} := p\left(a^{\mathrm{m}}_t \,\middle|\, s^{\mathrm{m}}_t, \Theta\right)$ and $e_t := \nabla_m \ln\left\{\tilde{\pi}(a^{\mathrm{m}}_t, s^{\mathrm{m}}_t, m)\right\}$ are derived as follows:

$$\tilde{\pi}^{\mathrm{m}}(a^{\mathrm{m}}_t, s^{\mathrm{m}}_t, m) = \prod_{i=1}^{N} \frac{w_t^{(\gamma_t^{(i)})}}{\sum_{j=1}^{N} w_t^{(j)}}, \tag{4.7}$$

$$e_t = \sum_{i=1}^{N} \left( \Gamma^{(i)}_t - \frac{N w^{(i)}_t}{\sum_{j=1}^{N} w^{(j)}_t} \right) \nabla_m \ln\left\{ g\left(z_t, \hat{x}^{(i)}_{t|t-1}, m\right) \right\}. \tag{4.8}$$

By using $e_t$, we can optimize $m$ based on the policy-gradient-based algorithm corresponding to Algorithm 2.

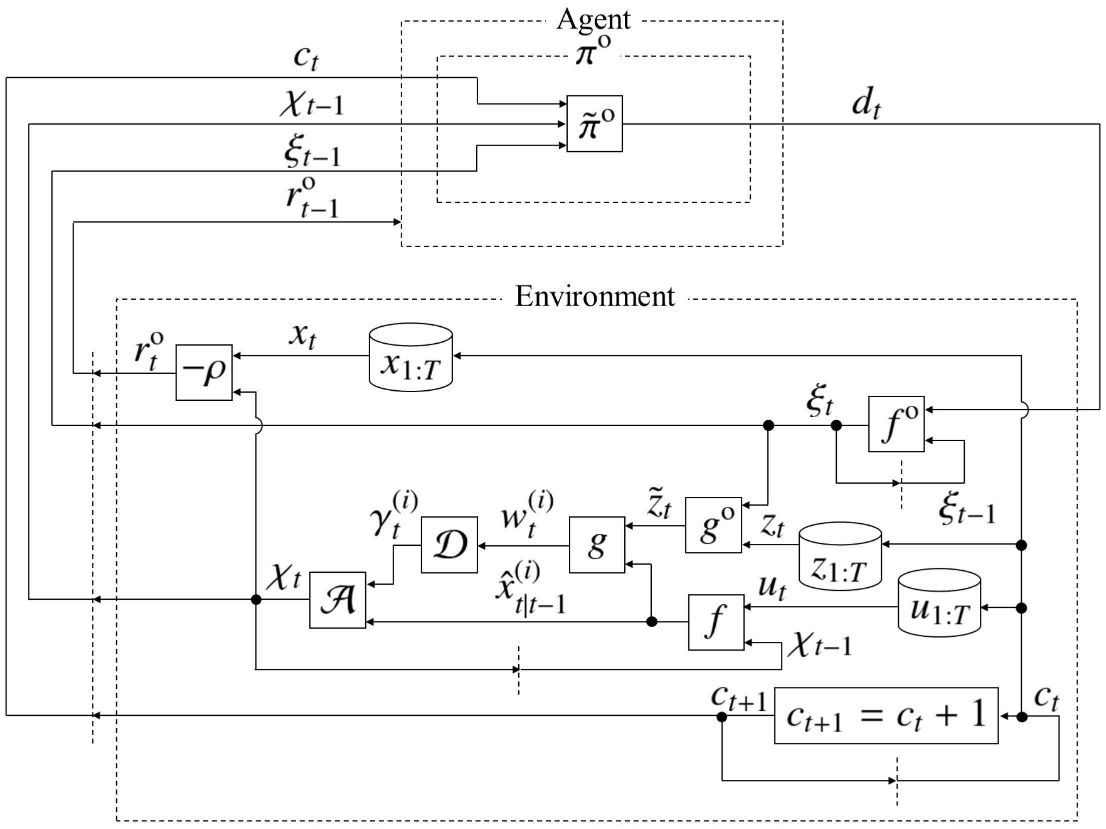


Figure 4.4: Relationship between MCL and RL in case of learning the worst-case obstacle behavior

### 4.4.2 Learning of the worst-case obstacle behavior

Next, let us consider learning the worst-case inputs $d_{1:T}$ of the virtual obstacle. From (4.6), the RL assignment for learning the worst-case $d_{1:T}$ is as follows:

- Policy $\pi^{\mathrm{o}}$: Function (4.6) which selects the obstacle input $d_t$ from $\mathbb{D}$.
- State $s^{\mathrm{o}}_t = (\chi_{t-1}, \xi_{t-1}, c_t)$: States of all particles of MCL $\chi_{t-1}$, state of virtual obstacle $\xi_{t-1}$, and current time $c_t := t$.
- Action $a^{\mathrm{o}}_t = d_t$: Input of virtual obstacle $d_t$.
- Reward $r^{\mathrm{o}}_t = -r^{\mathrm{m}}_t$: Output of the function $\rho$ in (4.3).

The relationship in this case is shown in Fig. 4.4. The policy $\pi^{\mathrm{o}}$ can be optimized by a value iteration method such as the classical Q learning [56] or DQN [57].

Here, an example implementation of the obstacle learning based on Q learning is presented. Q learning requires a table of Q values in the space of the state $s^{\mathrm{o}}_t$ and action $a^{\mathrm{o}}_t$.

To satisfy this condition, it is assumed that the set of obstacle's inputs consists of a finite number of elements. In the case of the obstacle in Fig. 4.2(b), an example of the set is

$$\mathbb{D} = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Delta r^{\mathrm{o}} \\ 0 \end{bmatrix}, \begin{bmatrix} -\Delta r^{\mathrm{o}} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \Delta \theta^{\mathrm{o}} \end{bmatrix}, \begin{bmatrix} 0 \\ -\Delta \theta^{\mathrm{o}} \end{bmatrix} \right\} \tag{4.9}$$

where $\Delta r^{\mathrm{o}} \in \mathbb{R}$ and $\Delta \theta^{\mathrm{o}} \in \mathbb{R}$ are constants. Since the particle set $\chi_t$, which is included in the state $s_t^{\mathrm{o}}$, has a large dimension ($n \times N$ dimension) for Q learning, its dimension is reduced to $n + 1$ dimensions, i.e., average estimation error ($x_t - h(\chi_t)$) and variance ($\sum_{i=1}^{N} \left\| \hat{x}_t^{(i)} - h(\chi_t) \right\|^2$).

### 4.4.3 Method of generating robust highlighted map

The proposed method of generating a robust highlighted map is shown in Algorithm 3, which is an extension of Algorithm 2.

Algorithm 3 repeats the following processes $k_{\max}$ times. Firstly, the obstacle behavior is learned in lines 3–10. At each time from $t = 1$ to $T$, $d_t$ and $\xi_t$ are calculated to obtain measurement data including noise, $\tilde{z}_t$, and then MCL is executed by using $\tilde{z}_t$. After calculating $r_t$, $\pi^{\mathrm{o}}$ is updated by, e.g., Q learning. Secondly, the highlighted map is updated in lines 11–22. At each time from $t = 1$ to $T$, as in the first half of the algorithm, $d_t$, $\xi_t$, and $\tilde{z}_t$ are computed, MCL is executed, and $r_t$ is calculated. Next, $e_t$, the cumulative sum $U_t$ of $e_t$, and the cumulative sum $V_t$ of $(r_t - b)U_t$ are computed. After the calculation at time $T$, the map $m(k)$ is updated by adding $\eta(k)V_T$.

Since the expected value of $V_T$ is equal to the gradient of (4.5), i.e.,

$$E[V_T] = \nabla_m \bar{J}(m, d_{1:T}), \tag{4.10}$$

the equation in line 22 can be regarded as a stochastic approximation method [63,64], similiar to Algorithm 2. Therefore, this algorithm gives an approximate solution to Problem 2.

## 4.5 Numerical simulation

The effectiveness of the robust highlighted map is demonstrated in the 3D dynamic simulator Gazebo. The experimental environment is the corridor shown in Fig. 4.5, whose maps are equivalent to Fig. 4.1. There is a large pillar at the center and several small objects in the

**Algorithm 3** Robust highlighted map generation

```
 1: function RobustHighlightedMapGen($\chi_0, \xi_0, u_{1:T}, z_{1:T}, x_{1:T}, m(0)$)
 2:     for $k = 0$ to $k_{\max} - 1$ do
 3:         for $t = 1$ to $T$ do
 4:             $d_t = \tilde{\pi}^{\mathrm{o}}(\chi_{t-1}, \xi_{t-1}, c_t)$
 5:             $\xi_t = f^{\mathrm{o}}(\xi_{t-1}, d_t)$
 6:             $\tilde{z}_t = g^{\mathrm{o}}(z_t, \xi_t)$
 7:             $\chi_t, \bar{\chi}_t = \mathrm{MCL}(\chi_{t-1}, u_t, \tilde{z}_t, m(k))$
 8:             $r_t = \rho(x_t, \chi_t)$
 9:             update $\pi^{\mathrm{o}}$ from $s_t^{\mathrm{o}} = (\chi_{t-1}, \xi_{t-1}, c_t)$, $s_{t+1}^{\mathrm{o}} = (\chi_t, \xi_t, c_{t+1})$, and $r_t$
10:         end for
11:         $U_0 = V_0 = 0$
12:         for $t = 1$ to $T$ do
13:             $d_t = \tilde{\pi}^{\mathrm{o}}(\chi_{t-1}, \xi_{t-1}, c_t)$
14:             $\xi_t = f^{\mathrm{o}}(\xi_{t-1}, d_t)$
15:             $\tilde{z}_t = g^{\mathrm{o}}(z_t, \xi_t)$
16:             $\chi_t, \bar{\chi}_t = \mathrm{MCL}(\chi_{t-1}, u_t, \tilde{z}_t, m(k))$
17:             $r_t = \rho(x_t, \chi_t)$
18:             calculate $e_t$ according to (4.8) from $\chi_t$, $\bar{\chi}_t$, $\tilde{z}_t$, and $m(k)$
19:             $U_t = U_{t-1} + e_t$
20:             $V_t = V_{t-1} + (r_t - b)U_t$
21:         end for
22:         $m(k+1) = m(k) + \eta(k)V_T$
23:     end for
24:     return $m(k_{\max})$
25: end function
```

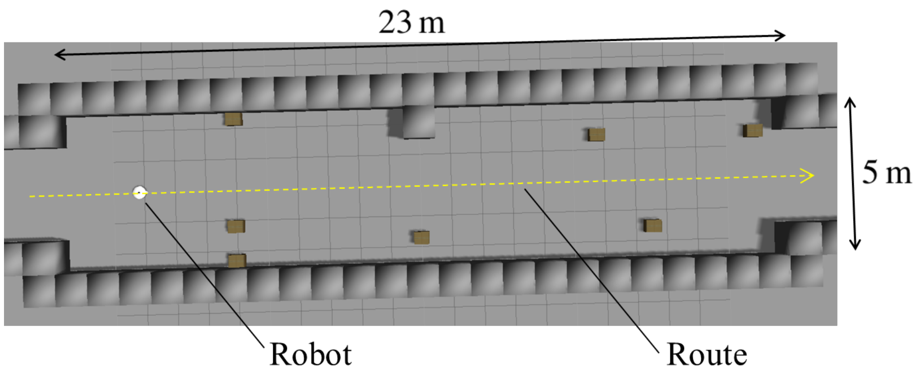


Figure 4.5: Simulation environment

corridor. The robot pose is expressed as (2.1). The specification of the LIDAR is as follows: possible maximum detection range is 3 m, scanning angle is 360 degrees, and the number of data is $L := 36$ data/rev. The MCL setting is as follows: the number of the particles is

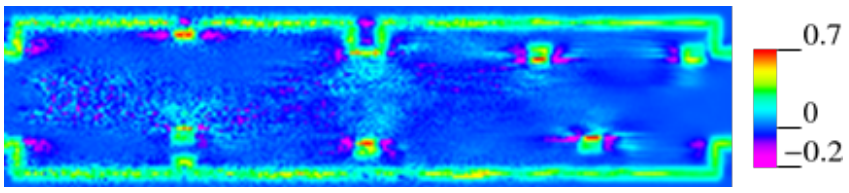


Figure 4.6: Robust highlighted map $m^*$

$N := 100$, frequency is 5 Hz, and functions $f$ and $g$ are the same as in Section 3.3.

The robust highlighted map was generated by the proposed method under the following settings. The initial map $m(0)$ is the likelihood field map shown in Fig. 4.1(b). The map size (number of cells) is $M := 240 \times 60$, and the grid scale is 0.1 m/cell. The training data $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$ were obtained as follows: 1) record $u_t^{\text{train}}$ and $z_t^{\text{train}}$ while the robot follows the route in Fig. 4.5 at 2 m/s; 2) calculate $x_t^{\text{train}}$ approximately by executing MCL with $N := 10{,}000$ particles (a larger number than usual for this robot). The function $\rho$ in (4.3) is set as (3.21). The virtual obstacle is the one in Fig. 4.2, where the shape of the obstacle is a circle with a diameter of 0.5 m. The set of $d_t$ is

$$\mathbb{D} := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.2 \\ 0 \end{bmatrix}, \begin{bmatrix} -0.2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 30^\circ \end{bmatrix}, \begin{bmatrix} 0 \\ -30^\circ \end{bmatrix} \right\}, \tag{4.11}$$

and the obstacle wanders 360 degrees around the robot, keeping the distance to the robot between 1 m and 3 m ($r_t^{\text{o}} \in \{1, 1.2, 1.4, \ldots, 3\}$, $\theta_t^{\text{o}} \in \{0^\circ, 30^\circ, 60^\circ, \ldots, 330^\circ\}$). For learning the the worst-case obstacle behavior, Q learning in Section 4.4.2 was used. In Algorithm 2, the learning rate, maximum iteration number, and reinforcement baseline are

$$\eta(k) := \frac{3}{k + 3 \times 10^6}, \tag{4.12}$$

$k_{\max} := 4 \times 10^6$, and $b := 0.9$, respectively.

The robust highlighted map generated by the proposed method is shown in Fig. 4.6. Compared to the conventional highlighted map (Fig. 4.1(c)), more objects, i.e., the objects at the northwest, south center, and southeast, are highlighted (red spots) in addition to the one at the northeast. From the fact, even if some objects are occluded by the obstacle, MCL can estimate the robot pose by observing other highlighted objects. On the other hand, the object at the southwest is no longer highlighted. This is because this object was often confused with the obstacle and interfered with the estimation. Based on these observations, it seems that the differences from the conventional highlighted map are reasonable ones that contribute to

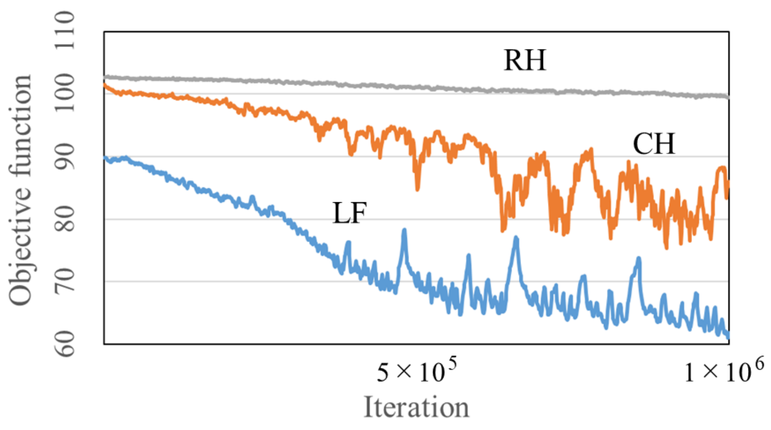


Figure 4.7: Change in objective function while policy of virtual obstacle is trained

the robustness of MCL.

To confirm this robustness, the performance of the generated robust highlighted map are evaluated as follows. The test data for the evaluation $u_{1:T}^{\text{test}}$, $z_{1:T}^{\text{test}}$, and $x_{1:T}^{\text{test}}$ were collected in the similar way as in Section 3.3.2. Firstly, only the virtual obstacle $\pi^{\text{o}}$ was trained (while the map remained fixed) in the case of using the likelihood field map in Fig. 4.1(b), conventional highlighted map in Fig. 4.1(c), and robust highlighted map in Fig. 4.6 (denoted by LF, CH, and RH, respectively). The change in the objective function $J$ is plotted in Fig. 4.7, where each line shows a moving average of $J$. It can be seen that the performance of the robust highlighted map hardly deteriorates even if the training of the virtual obstacle is advanced. Secondly, the estimation errors in the presence of the trained (worst-case) obstacle were measured. Each map was tested 100 times by using the corresponding trained obstacle. If the error exceeded 4 m along the way, it was judged that the robot got lost, and the test was terminated as a failure. In the case of LF, CH, and RH, the percentages of times the robot got lost is 6/100, 3/100, 0/100, respectively. The three failures when using CH all resulted from the confusion between the highlighted object at the southwest and the obstacle. In the case of RH, the robot did not get lost due to the confusion since the object at the southwest were not highlighted. Figure 4.8 shows box plots of the maximum estimation errors for each of the 100 tests, except for the failed tests. The maximum estimation errors when using CH were mainly caused by the fact that the highlighted object at the northeast was occluded by the obstacle. In the case of RH, the object at the southeast were also highlighted; thus, even if one side was occluded by the obstacle, the robot could estimate accurately by observing the other side. These results suggest that the robust highlighted map has the highest robustness against the obstacle.

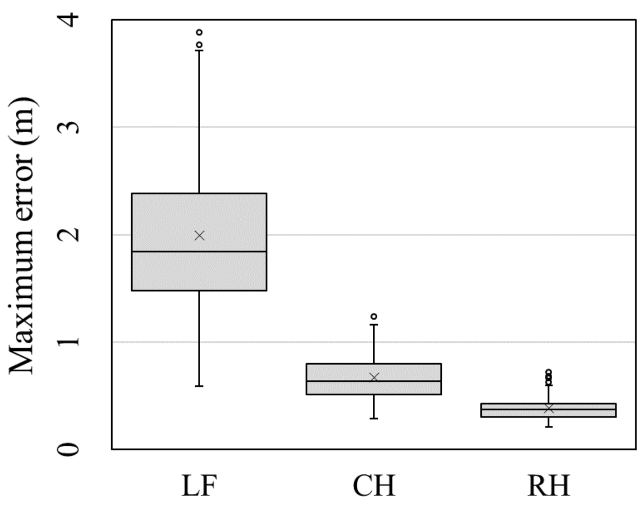


Figure 4.8: Box plots of maximum estimation errors in presence of the worst-case obstacle

## 4.6 Conclusion

This chapter proposed the method of generating a robust highlighted map. Whereas it needs complex computation such as Q learning and the policy gradient method during generating the map, it only requires ordinary MCL computation after the map generation. Although the classical Q learning was used for learning the obstacle behavior, it is possible to learn more sophisticated behavior by using deep RL such as DQN. In future work, it is necessary to devise a robustification method that can consider not only measurement noise but also system noise.

# Chapter 5

# Particle filter design

This chapter proposes a new PF design method by extending the method in Chapter 3. Similar to Chapter 3, it is an RL-based method, where the parameters of the PF are optimized in the RL framework by assigning the two randomnesses in the PF to the randomness required for RL. Specifically, this method performs the following two steps repeatedly: 1) estimate by the PF; 2) update the parameters based on the estimation results. This method can design both the system and measurement models and accommodate various objective functions.

First, the PF design method based on RL is proposed. Next, it is applied to the mobile robot localization problem. The numerical simulation confirms that the proposed method makes the localization even more accurate than the method in Chapter 3, where only the measurement model is designed.

## 5.1 Target system

The target system in this chapter is the general stochastic state-space model denoted by

$$\begin{cases} x_{t+1} = F(x_t, u_t, v_t) \\ z_t = G(x_t, \omega_t) \end{cases} \tag{5.1}$$

where $x_t \in \mathbb{R}^n$ is the state, $u_t \in \mathbb{R}^{n_u}$ is the input, $z_t \in \mathbb{R}^{n_z}$ is the output or measurement data, $v_t \in \mathbb{R}^{n_v}$ is the system noise, and $\omega_t \in \mathbb{R}^{n_\omega}$ is the measurement noise. The functions $F : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \to \mathbb{R}^n$ and $G : \mathbb{R}^n \times \mathbb{R}^{n_\omega} \to \mathbb{R}^{n_z}$ are possibly nonlinear.

## 5.2 Particle filter

Consider estimating the probability distribution of $x_t$ from $u_t$ and $z_t$ by the PF shown in Algorithm 4. This algorithm is a generalization of MCL in Section 2.2. In the algorithm, $\hat{x}_t^{(i)} \in \mathbb{R}^n$ is the state of the $i$-th particle, $N \in \mathbb{N}$ is the number of particles, and $\chi_t := \left\{\hat{x}_t^{(i)}\right\}_{i=1}^N$ is the particle set. Each particle $\hat{x}_t^{(i)}$ is a hypothesis about the true state $x_t$, and the set $\chi_t$ represents the probability distribution of $x_t$ approximately. The algorithm has inputs $u_t$, $z_t$, and $\Theta$ in addition to $\chi_{t-1}$, where $u_t$ and $z_t$ are the input and measurement data of the target system (5.1), respectively, and $\Theta \in \mathbb{R}^{n_\Theta}$ is the parameter vector that characterizes the system model and the measurement model in the algorithm. Line 4 in the algorithm predicts the state of each particle based on the system model $p\left(x_t \,\middle|\, x_{t-1}, u_t, \Theta\right)$, which expresses the function $F$. The state of the $i$-th particle after the prediction step is denoted by $\hat{x}_{t|t-1}^{(i)} \in \mathbb{R}^n$. Line 5 calculates the weight (likelihood) of each particle from the measurement model $p\left(z_t \,\middle|\, x_t, \Theta\right)$, which corresponds to the function $G$. The output $w_t^{(i)} \in \mathbb{R}_+$ is the weight of the particle $\hat{x}_{t|t-1}^{(i)}$. The system and measurement models, which are both probability density functions formed by the parameter vector $\Theta$, are hereinafter referred to as

$$f^{\mathrm{pf}}\left(x_t, x_{t-1}, u_t, \Theta\right) := p\left(x_t \,\middle|\, x_{t-1}, u_t, \Theta\right), \tag{5.2}$$

$$g^{\mathrm{pf}}\left(z_t, x_t, \Theta\right) := p\left(z_t \,\middle|\, x_t, \Theta\right). \tag{5.3}$$

This algorithm performs the following processes. After calculating the weighted particle set $\bar{\chi}_t := \left\{\left(\hat{x}_{t|t-1}^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^N$, the particles are randomly sampled based on the ratio of the weight $w_t^{(i)}$ to compose $\chi_t$. This sampling process is called resampling. Finally, $\chi_t$ is returned. In addition, the algorithm returns the weighted particle set before resampling, $\bar{\chi}_t$, which is used for designing the PF.

## 5.3 Problem formulation

For the purpose of optimizing the parameter vector $\Theta$ in the PF, the objective function is defined as follows:

$$J(\Theta) := \sum_{t=1}^{T} \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) \tag{5.4}$$

where $T \in \mathbb{N}$ is the operating time, and $\rho : \mathbb{R}^n \times \mathbb{R}^{n\times N} \times \left(\mathbb{R}^n \times \mathbb{R}_+\right)^N \times \mathbb{R}^{n_\Theta} \to \mathbb{R}$ evaluates the estimation result at each time from the true state $x_t$, the particle sets $\chi_t$ and $\bar{\chi}_t$, and the

**Algorithm 4** Particle filter

```
1:  function PF($\chi_{t-1}, u_t, z_t, \Theta$)
2:      $\bar{\chi}_t = \chi_t = \emptyset$
3:      for $i = 1$ to $N$ do
4:          sample $\hat{x}^{(i)}_{t|t-1} \sim p\left(x_t \,\middle|\, \hat{x}^{(i)}_{t-1}, u_t, \Theta\right)$
5:          $w^{(i)}_t = p\left(z_t \,\middle|\, \hat{x}^{(i)}_{t|t-1}, \Theta\right)$
6:          $\bar{\chi}_t \leftarrow \bar{\chi}_t \cup \left\{\left(\hat{x}^{(i)}_{t|t-1}, w^{(i)}_t\right)\right\}$
7:      end for
8:      for $i = 1$ to $N$ do
9:          $\hat{x}^{(i)}_t = \begin{cases} \hat{x}^{(1)}_{t|t-1} & \text{with probability } \frac{w^{(1)}_t}{\sum_{j=1}^N w^{(j)}_t} \\ \hat{x}^{(2)}_{t|t-1} & \text{with probability } \frac{w^{(2)}_t}{\sum_{j=1}^N w^{(j)}_t} \\ \vdots & \vdots \\ \hat{x}^{(N)}_{t|t-1} & \text{with probability } \frac{w^{(N)}_t}{\sum_{j=1}^N w^{(j)}_t} \end{cases}$
10:         $\chi_t \leftarrow \chi_t \cup \left\{\hat{x}^{(i)}_t\right\}$
11:     end for
12:     return $\chi_t$, $\bar{\chi}_t$
13: end function
```

parameter vector $\Theta$. Note that $\rho$ has more inputs than that in the previous chapters, i.e., $\bar{\chi}_t$ and $\Theta$ in addition to $x_t$ and $\chi_t$. This allows us to set up a variety of objective functions.

The PF design problem is formulated as follows.

**Problem 3.** *For Algorithm 4, assume that the number $N$ of particles, functions $f^{\mathrm{pf}}$, $g^{\mathrm{pf}}$, and $\rho$, an initial particle set $\chi_0$, input data $u_{1:T}$, measurement data $z_{1:T}$, and true states $x_{1:T}$ are given. Then, find the optimal parameter vector*

$$\arg\max_{\Theta} \ \bar{J}(\Theta) \tag{5.5}$$

*where $\bar{J} : \mathbb{R}^{n_\Theta} \to \mathbb{R}$ is the expected value of* (5.4), *i.e.,*

$$\bar{J}(\Theta) := E\left[J(\Theta)\right]. \tag{5.6}$$

■

Note the following two facts: First, as with the previous problems, the solution (5.5) is optimal only for the past data ($u_{1:T}$, $z_{1:T}$, and $x_{1:T}$) and may not be optimal for future data.

However, it is expected that the same performance is obtained when the state changes in a trajectory similar to the one in which the training data ($u_{1:T}$, $z_{1:T}$, and $x_{1:T}$) were collected. The simulation in Section 5.5 confirmed that it was also effective for future data. Second, this study focuses on time-invariant systems for simplicity, but its contributions can be easily extended to time-varying systems. In other words, if the functions in the target system (5.1), PF, and objective function (5.4) are time-varying ($F_t$, $G_t$, $f_t^{\mathrm{pf}}$, $g_t^{\mathrm{pf}}$, and $\rho_t$), all the results described below can be applied in the same way.

In Problem 3, various optimal PFs can be obtained by appropriately assigning $\rho$. For example, $\rho$ can be given as one or a combination of the following.

**Example 1** (Estimation Error)**.** *Assume that the weighted average of the particles*

$$\bar{h}\left(\bar{\chi}_t\right) := \frac{\sum_{i=1}^{N} w_t^{(i)} \hat{x}_{t|t-1}^{(i)}}{\sum_{i=1}^{N} w_t^{(i)}} \tag{5.7}$$

*is a representative state estimate at each time. Then, maximizing the sum of*

$$\rho_1\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) := -\left(x_t - \bar{h}\left(\bar{\chi}_t\right)\right)^\top A \left(x_t - \bar{h}\left(\bar{\chi}_t\right)\right) \tag{5.8}$$

*gives the PF that minimizes the error between the true state and the estimate, where* $A \in \mathbb{P}_{0+}^n$ *is a weight coefficient matrix.* ■

**Example 2** (Particle Dispersion)**.** *Roy and Thrun presented the mobile robot motion planning that minimized the uncertainty of pose estimation along the way [28]. They used a Bayes filter and represented the uncertainty as the entropy of the probability distribution of the pose estimate. In the case of the PF, the variance of the particles*

$$\rho_2\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) := -\sum_{i=1}^{N} \left(\hat{x}_t^{(i)} - h\left(\chi_t\right)\right)^\top A \left(\hat{x}_t^{(i)} - h\left(\chi_t\right)\right) \tag{5.9}$$

*can represent the uncertainty of estimation, where h is the average of the state of the particles, defined as* (3.4). ■

**Example 3** (Likelihood)**.** *As the log-likelihood of the parameters* $\ln\left\{p\left(z_{1:T} \,\middle|\, \Theta\right)\right\}$ *can be approximated by the sum of*

$$\rho_3\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) := \ln\left(\frac{1}{N}\sum_{i=1}^{N} w_t^{(i)}\right), \tag{5.10}$$

*the MLE of* $\Theta$ *can be performed by maximizing them [47].* ■

**Example 4** (Regularization)**.** *By adding the regularization term*

$$\rho_4 \left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) := \kappa \left\|\Theta\right\|^2 \tag{5.11}$$

*to functions such as the above* $\rho_1$*–*$\rho_3$*, the parameters will not be extremely large, and overfitting can be avoided [66], where* $\kappa \in \mathbb{R}_+$ *is a weight coefficient for the term.* ■

## 5.4 Particle filter design based on reinforcement learning

This section gives the solution to Problem 3 based on RL.

### 5.4.1 Interpretation of particle filter design problem into reinforcement learning

Consider associating PF with RL. Many RL algorithms, such as in [59], optimizes the parameters included in the policy by using a probabilistic policy that selects an action at random. On the other hand, a PF has two random processes; the prediction of the state from the system model $f^{\mathrm{pf}}$ and the resampling based on the ratio of $w_t^{(i)}$. Therefore, by assigning these random processes of PF to the policy $\pi$ of RL, the PF parameters $\Theta$ can be optimized based on such RL algorithms. Thus, this section interprets PF into the RL framework in Section 2.4 by the following assignment:

- Policy $\pi$: Prediction process based on the system model $f^{\mathrm{pf}}$ and resampling process including the weight calculation based on the measurement model $g^{\mathrm{pf}}$. The parameter vector $\Theta$ determines the behavior of these processes.
- State $s_t = (\chi_{t-1}, c_t)$: Particle set $\chi_{t-1}$ and current time $c_t := t$.
- Action $a_t = \left(a_t^{\mathrm{p}}, a_t^{\mathrm{r}}\right)$: Prediction result $a_t^{\mathrm{p}} := \left(\hat{x}_{t|t-1}^{(1)}, \hat{x}_{t|t-1}^{(2)}, \ldots, \hat{x}_{t|t-1}^{(N)}\right)$ and resampling result $a_t^{\mathrm{r}} := \left[\gamma_t^{(1)}, \gamma_t^{(2)}, \ldots, \gamma_t^{(N)}\right]^\top$, where $\gamma_t^{(i)} \in \mathbb{N}$ is the index $j$ of the state $\hat{x}_{t|t-1}^{(j)}$ to which the $i$-th particle is resampled in line 9 of Algorithm 4.
- Reward $r_t$: Output of the function $\rho$ in (5.4).

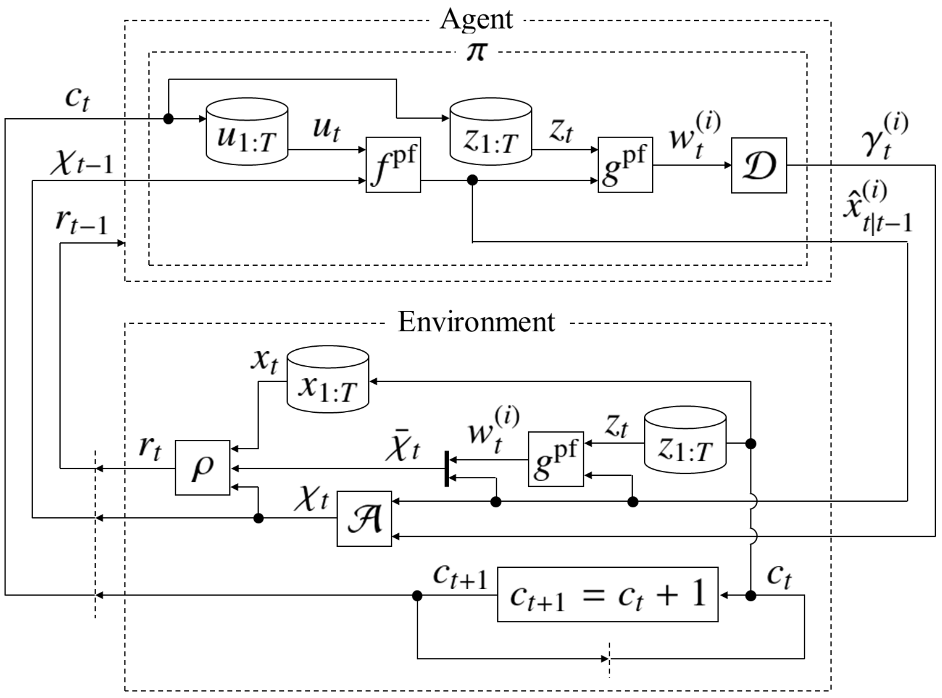


Figure 5.1: Relationship between PF and RL

Note that this chapter differs from Chapter 3 mainly in the following aspects: not only the resampling but also the prediction process are regarded as the policy, and the reward depends on $\bar{\chi}_t$ and $\Theta$ in addition to $x_t$ and $\chi_t$.

The relationship between PF and RL is shown in Fig. 5.1, where the systems $\mathcal{D}$ and $\mathcal{A}$ are the same as in Section 3.2.1.

Then, the equations

$$p\left(s_{t+1} \,\middle|\, s_{1:t}, a_{1:t}, \pi\right) = p\left(s_{t+1} \,\middle|\, s_t, a_t\right), \tag{5.12}$$

$$p\left(a_t \,\middle|\, s_{1:t}, a_{1:t-1}, \pi\right) = p\left(a_t \,\middle|\, s_t, \Theta\right), \tag{5.13}$$

$$E\left[r_t \,\middle|\, s_{1:t}, a_{1:t}, \pi\right] = E\left[r_t \,\middle|\, s_t, a_t, \Theta\right] \tag{5.14}$$

are satisfied, and the entire system can be treated in a similar way as an MDP. Note that in a general MDP, the reward does not depend on the policy parameters $\Theta$; however, some RL methods [67, 68] consider the parameter-dependent reward (5.14).

To apply the policy gradient method in [59], the characteristic eligibility (2.7) is derived. In the case of this system, $\tilde{\pi}$ in (2.8) is derived as:

$$\tilde{\pi}(a_t, s_t, \Theta) = p\left(a_t^{\mathrm{p}} \,\middle|\, s_t, \Theta\right) p\left(a_t^{\mathrm{r}} \,\middle|\, s_t, a_t^{\mathrm{p}}, \Theta\right) \tag{5.15}$$

where

$$p\left(a_t^{\mathrm{p}} \,\middle|\, s_t, \Theta\right) = \prod_{i=1}^{N} f^{\mathrm{pf}}\left(\hat{x}_{t|t-1}^{(i)}, \hat{x}_{t-1}^{(i)}, u_t, \Theta\right), \tag{5.16}$$

$$p\left(a_t^{\mathrm{r}} \,\middle|\, s_t, a_t^{\mathrm{p}}, \Theta\right) = \prod_{i=1}^{N} \frac{w_t^{(\gamma_t^{(i)})}}{\sum_{j=1}^{N} w_t^{(j)}}. \tag{5.17}$$

Equations (5.16) and (5.17) are the probability distributions corresponding to the prediction from $f^{\mathrm{pf}}$ and resampling based on $w_t^{(i)}$, respectively. From (5.15)–(5.17), $e_t$ is derived as:

$$\begin{aligned} e_t = \sum_{i=1}^{N} \Bigg\{ & \nabla_\Theta \ln\left\{f^{\mathrm{pf}}\left(\hat{x}_{t|t-1}^{(i)}, \hat{x}_{t-1}^{(i)}, u_t, \Theta\right)\right\} \\ & + \left(\Gamma_t^{(i)} - \frac{N g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right)}{\sum_{j=1}^{N} g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(j)}, \Theta\right)}\right) \nabla_\Theta \ln\left\{g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right)\right\} \Bigg\} \end{aligned} \tag{5.18}$$

where $\Gamma_t^{(i)}$ is defined as (3.13).

In addition, for the conciseness in discussing the parameter-dependent reward (5.14), the gradient of the reward function is denoted by

$$\zeta_t := \nabla_\Theta \tilde{\rho}\left(s_t, a_t, \Theta\right), \tag{5.19}$$

where $\tilde{\rho} : (s_t, a_t, \Theta) \mapsto \rho\left(x_t(s_t), \chi_t(a_t), \bar{\chi}_t(s_t, a_t, \Theta), \Theta\right)$ is a rewriting of the function $\rho$ in (5.4) with RL variables. This can be transformed into

$$\begin{aligned} \zeta_t &= \sum_{i=1}^{N} \frac{\partial \rho}{\partial w_t^{(i)}}\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) \cdot \nabla_\Theta w_t^{(i)} + \nabla_\Theta \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) \\ &= \sum_{i=1}^{N} \frac{\partial \rho}{\partial w_t^{(i)}}\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) \cdot \nabla_\Theta g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right) + \nabla_\Theta \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right). \end{aligned} \tag{5.20}$$

It should be noted that $w_t^{(i)}$ is included in $\bar{\chi}_t$ $\left(\bar{\chi}_t := \left\{\left(\hat{x}_{t|t-1}^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}\right)$. For example, if $\rho$ is given by (5.8),

$$\zeta_t = \frac{2}{\sum_{i=1}^N w_t^{(i)}} \left\{ \sum_{i=1}^N \left(x_t - \bar{h}(\bar{\chi}_t)\right)^\top A \hat{x}_{t|t-1}^{(i)} \nabla_\Theta g^{\mathrm{pf}} \left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right) \right.$$
$$\left. - \left(x_t - \bar{h}(\bar{\chi}_t)\right)^\top A \bar{h}(\bar{\chi}_t) \sum_{i=1}^N \nabla_\Theta g^{\mathrm{pf}} \left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right) \right\}. \tag{5.21}$$

Also, in the case of (5.10),

$$\zeta_t = \frac{1}{\sum_{i=1}^N w_t^{(i)}} \sum_{i=1}^N \nabla_\Theta g^{\mathrm{pf}} \left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right). \tag{5.22}$$

By using the values $e_t$ and $\zeta_t$, the parameter vector $\Theta$ can be optimized based on the RL-based algorithm described in the next section.

### 5.4.2 Particle filter design method

The PF design method proposed in this chapter is shown in Algorithm 5, where $\Theta(k) \in \mathbb{R}^{n_\Theta}$ is the parameter vector at the iteration $k$. This algorithm is a generalization of Algorithm 2.

In Algorithm 5, the following process is repeated $k_{\max}$ times. First, at each time from $t = 1$ to $T$, the PF is executed, $r_t$, $e_t$, and $\zeta_t$ are calculated, and then the cumulative sum $U_t$ of $e_t$ and the cumulative sum $V_t$ of $(r_t - b)U_t + \zeta_t$ are updated. Second, the parameters are modified by adding $\eta(k)V_T$.

Before executing Algorithm 5, it is necessary to operate the system and record its inputs $u_{1:T}$, measurement data $z_{1:T}$, and true states $x_{1:T}$. The true states $x_{1:T}$ can be acquired, e.g., by temporarily adding a high-performance sensor that can observe the true state to the system. In [53], which optimized the estimation accuracy of a PF for estimating a robot pose, the true poses were measured by a camera mounted in the environment. Alternatively, $x_{1:T}$ can be estimated after recording $u_{1:T}$ and $z_{1:T}$ using a more complex estimation algorithm than usual (e.g., PF with a larger number of particles), or a smoothing algorithm [69] which utilizes future measurement data to improve past estimates. Although $x_{1:T}$ estimated by these methods will naturally contain some errors, the simulation in Section 5.5.3 confirms that this $x_{1:T}$ is sufficient to design a practical PF. Note that it is not necessary to obtain $x_{1:T}$ if the objective function consists of (5.9)–(5.11).

Another example of the procedure for preparing $u_{1:T}$, $z_{1:T}$, and $x_{1:T}$ as follows: 1) record $u_{1:T}$ and $z_{1:T}$ and estimate $x_{1:T}$ from these data; 2) add the extra noise to $u_{1:T}$ and $z_{1:T}$

**Algorithm 5** Particle filter design

```
1: function PFDesign($\Theta(0), \chi_0, u_{1:T}, z_{1:T}, x_{1:T}$)
2:     for $k = 0$ to $k_{\max} - 1$ do
3:         $U_0 = V_0 = 0$
4:         for $t = 1$ to $T$ do
5:             $\chi_t, \bar{\chi}_t = \mathrm{PF}(\chi_{t-1}, u_t, z_t, \Theta(k))$
6:             $r_t = \rho(x_t, \chi_t, \bar{\chi}_t, \Theta(k))$
7:             calculate $e_t$ according to (5.18) from $\chi_t$, $\bar{\chi}_t$, $u_t$, $z_t$, and $\Theta(k)$
8:             calculate $\zeta_t$ according to (5.20) from $\chi_t$, $\bar{\chi}_t$, $z_t$, $x_t$, and $\Theta(k)$
9:             $U_t = U_{t-1} + e_t$
10:            $V_t = V_{t-1} + (r_t - b)U_t + \zeta_t$
11:        end for
12:        $\Theta(k+1) = \Theta(k) + \eta(k)V_T$
13:    end for
14:    return $\Theta(k_{\max})$
15: end function
```

intentionally and input these and $x_{1:T}$ to Algorithm 5. This procedure is expected to design the PF that is robust to noise.

For the algorithm, the following theorem holds.

**Theorem 2.** *For Algorithm 5, assume that the following conditions hold:*

*(C9) There exists a bounded convex set* $\mathbb{P} \subset \mathbb{R}^{n_\Theta}$ *such that for all* $k \in \{0\} \cup \mathbb{N}$, $\Theta(k) \in \mathbb{P}$.

*(C10) For all* $(t, \hat{x}, \hat{x}', \chi_t, \bar{\chi}_t, \Theta) \in \{1, 2, \ldots, T\} \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^{n \times N} \times (\mathbb{R}^n \times \mathbb{R}_+)^N \times \mathbb{P}$, $\nabla^2_\Theta f^{\mathrm{pf}}(\hat{x}, \hat{x}', u_t, \Theta)$, $\nabla^2_\Theta g^{\mathrm{pf}}(z_t, \hat{x}, \Theta)$, $\nabla^2_{w_t} \rho(x_t, \chi_t, \bar{\chi}_t, \Theta)$, *and* $\nabla^2_\Theta \rho(x_t, \chi_t, \bar{\chi}_t, \Theta)$ *exists, where* $w_t := \left[w_t^{(1)}, w_t^{(2)}, \ldots, w_t^{(N)}\right]^\top$.

*(C11) For all* $(t, \tau, \Theta) \in \{1, 2, \ldots, T\} \times \{1, 2, \ldots, t\} \times \mathbb{P}$, $E[|r_t|] < \infty$, $E\left[\|e_t\|^2\right] < \infty$, $E\left[\|r_t e_\tau\|^2\right] < \infty$, $E[\|r_t \nabla_\Theta e_\tau\|_F] < \infty$, $E\left[\|\zeta_t\|^2\right] < \infty$, *and* $E[\|\nabla_\Theta \zeta_t\|_F] < \infty$.

*(C12) The learning rate* $\eta(k) \in \mathbb{R}_+$ *satisfies* $\sum_{k=0}^{\infty} \eta(k) = \infty$ *and* $\sum_{k=0}^{\infty} \eta(k)^2 < \infty$.

*(C13) For variables* $\mathbf{t} \in \mathbb{R}$ *and* $v \in \mathbb{R}^{n_\Theta}$, *the differential equation*

$$\frac{dv(\mathbf{t})}{d\mathbf{t}} = \nabla_v \bar{J}(v(\mathbf{t})) \tag{5.23}$$

*has an asymptotically stable equilibrium* $v = \Theta^* \in \mathbb{P}$.

*(C14) There exists a compact set* $\mathbb{B}_1 \subseteq \mathbb{B}(\Theta^*)$ *such that* $\left|\left\{k \,\middle|\, \Theta(k) \in \mathbb{B}_1,\ k = 0, 1, 2, \ldots\right\}\right| = \infty$, *where* $\mathbb{B}(\Theta^*) := \left\{v_0 \,\middle|\, \lim_{\mathbf{t}\to\infty} v(\mathbf{t}|v_0) = \Theta^*\right\}$, *and* $v(\mathbf{t}|v_0) \in \mathbb{R}^{n_\Theta}$ *is the solution of* (5.23) *for an initial value* $v_0 \in \mathbb{R}^{n_\Theta}$.

*Then,*

$$\lim_{k\to\infty} \Theta(k) = \Theta^*, \quad \text{with probability 1.} \tag{5.24}$$

*Proof.* By defining $\varphi(k) := V_T - \nabla_\Theta \bar{J}(\Theta(k))$, line 12 in Algorithm 5 can be rewritten as follows:

$$\Theta(k+1) = \Theta(k) + \eta(k)\nabla_\Theta \bar{J}(\Theta(k)) + \eta(k)\varphi(k). \tag{5.25}$$

This equation corresponds to the stochastic approximation method [63, 64], and Theorem 2.3.1 in [63] reveals that (5.24) holds if (C9), (C12)–(C14), and the following conditions are satisfied.

(D4) For all $k \in \{0\} \cup \mathbb{N}$, $E[V_T] = \nabla_\Theta \bar{J}(\Theta(k))$.

(D5) The function $\nabla_\Theta \bar{J}(\Theta)$ is continuous at all $\Theta \in \mathbb{P}$.

(D6) For all $\epsilon \in \mathbb{R}_+$, $\lim_{k\to\infty} P\left(\sup_{K\geq k} \left\|\sum_{k'=k}^{K} \eta(k')\varphi(k')\right\| \geq \epsilon\right) = 0$.

Conditions (D4)–(D6) hold if (C9)–(C12) are satisfied (see Appendix B). ■

Theorem 2 clarifies that the proposed method gives a local optimum $\Theta^*$, i.e., an approximate solution to Problem 3.

Finally, here are some comments on the conditions in Theorem 2. Conditions (C9), (C13), and (C14) pertain to the boundedness, convergence value, and trajectory of the parameters, respectively. These conditions are generally assumed in stochastic approximation methods [63, 64], and (C9) can be omitted using the method in [64]. Condition (C10) is about the differentiability of the functions $f^{\mathrm{pf}}$, $g^{\mathrm{pf}}$, and $\rho$, and (C11) means the boundedness of the expected values. Although the expected values in (C11) are difficult to calculate, the corresponding samples (e.g., $|r_t|$, $\|e_t\|^2$, and $\|\zeta_t\|^2$) are easily available. Therefore, (C11) can be roughly assessed by running Algorithm 5 several times to obtain the samples and confirming that each sample does not diverge. If it is difficult to obtain the samples of $\|r_t \nabla_\Theta e_\tau\|_F$ and $\|\nabla_\Theta \zeta_t\|_F$, the inequalities should be decomposed as follows:

$$\begin{aligned} & E\left[\|r_t \nabla_\Theta e_\tau\|_F\right] < \infty \\ \Leftrightarrow\ & E\left[\left\|r_t \nabla_\Theta^2 \ln\left\{f^{\mathrm{pf}}\left(\hat{x}^{(i)}_{\tau|\tau-1}, \hat{x}^{(i)}_{\tau-1}, u_\tau, \Theta\right)\right\}\right\|_F\right] < \infty \\ & \quad \text{and } E\left[\left\|r_t \nabla_\Theta^2 \ln\left\{g^{\mathrm{pf}}\left(z_\tau, \hat{x}^{(i)}_{\tau|\tau-1}, \Theta\right)\right\}\right\|_F\right] < \infty, \end{aligned} \tag{5.26}$$

$$\begin{aligned}
&E\left[\|\nabla_\Theta \zeta_t\|_F\right] < \infty \\
&\Leftrightarrow E\left[\left\|\nabla_\Theta \left(\nabla_{w_t} \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right)\right)\right\|_F \cdot \left\|\nabla_\Theta g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right)\right\|\right] < \infty, \\
&\quad E\left[\left\|\nabla_{w_t}^2 \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right)\right\|_F \cdot \left\|\nabla_\Theta g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right)\right\|^2\right] < \infty, \\
&\quad E\left[\left\|\nabla_{w_t} \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right)\right\| \cdot \left\|\nabla_\Theta^2 g^{\mathrm{pf}}\left(z_t, \hat{x}_{t|t-1}^{(i)}, \Theta\right)\right\|_F\right] < \infty, \\
&\quad \text{and } E\left[\left\|\nabla_\Theta^2 \rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right)\right\|_F\right] < \infty.
\end{aligned} \tag{5.27}$$

Condition (C12) is used for designing the learning rate $\eta(k)$.

### 5.4.3 Illustrative example

To demonstrate how to use the proposed method, this section designs a PF for estimating the state $x_t \in \mathbb{R}$ of the Weiner process:

$$\begin{cases} x_{t+1} = x_t + v_t \\ z_t = x_t + \omega_t \end{cases} \tag{5.28}$$

where the system noise $v_t \in \mathbb{R}$ and measurement noise $\omega_t \in \mathbb{R}$ are zero-mean Gaussian noise with standard deviations $\sigma_v \in \mathbb{R}_+$ and $\sigma_\omega \in \mathbb{R}_+$, respectively. Consider finding the parameter vector $\Theta := [\sigma_v, \sigma_\omega]^\top$ optimal with respect to the objective function consisting of (5.10) (MLE). The system and measurement models are described as

$$f^{\mathrm{pf}}\left(x_t, x_{t-1}, u_t, \Theta\right) = \mathcal{N}\left(x_t - x_{t-1}, \sigma_v^2\right), \tag{5.29}$$

$$g^{\mathrm{pf}}\left(z_t, x_t, \Theta\right) = \mathcal{N}\left(z_t - x_t, \sigma_\omega^2\right), \tag{5.30}$$

where the function $\mathcal{N}$ is a zero-centered normal distribution, as in (2.2). From (5.18) and (5.22), the characteristic eligibility and gradient of the reward are derived as follows:

$$e_t = \begin{bmatrix} \dfrac{1}{\sigma_v^3} \displaystyle\sum_{i=1}^{N} \left(\hat{x}_{t|t-1}^{(i)} - \hat{x}_{t-1}^{(i)}\right)^2 - \dfrac{N}{\sigma_v} \\ \dfrac{1}{\sigma_\omega^3} \displaystyle\sum_{i=1}^{N} \left(\Gamma_t^{(i)} - \dfrac{N w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}\right) \left(z_t - \hat{x}_{t|t-1}^{(i)}\right)^2 \end{bmatrix}, \tag{5.31}$$

$$\zeta_t = \left[0, \frac{1}{\sigma_\omega^3 \sum_{i=1}^{N} w_t^{(i)}} \sum_{i=1}^{N} w_t^{(i)} \left(z_t - \hat{x}_{t|t-1}^{(i)}\right) - \frac{1}{\sigma_\omega}\right]^\top. \tag{5.32}$$

By executing Algorithm 5 with (5.31) and (5.32), we can obtain the (local) maximum likelihood estimate of the parameters.

To confirm this, simulations were performed to record the training data $z_{1:T}$ ($x_{1:T}$ are not required for (5.10)) from the system (5.28) with $\sigma_v = 3$ and $\sigma_\omega = 6$ (unknown parameters), and then the parameters $\Theta$ were designed. The initial parameters, the number of particles, learning rate, and maximum iteration number were $\Theta(0) := [4, 4]^\top$, $N := 100$,

$$\eta(k) := \frac{1}{k + 1 \times 10^5}, \tag{5.33}$$

and $k_{\max} := 1 \times 10^5$, respectively. The reinforcement baseline, which reduces the estimated variance of the equation in (D4), was $b := -3.4$. By performing the simulations 10 times, 10 sets of training data were collected, and $\Theta$ were designed for each set (the designed parameters for the $i$-th set are denoted by $\Theta_i^*$). The results obtained were as follows: the mean $\bar{\Theta}^* := \sum_{i=1}^{10} \Theta_i^* / 10 = [3.00, 6.03]^\top$ and the variance $\sigma_\Theta^2 := \sum_{i=1}^{10} \left(\Theta_i^* - \bar{\Theta}^*\right)^2 / 10 = \left[0.325^2, 0.320^2\right]^\top$.

## 5.5 Application to mobile robot localization

This section applies the proposed PF design method to MCL, whose algorithm corresponds to the PF, to show that the proposed method can be applied to practical-scale problems.

This section considers a wheeled robot equipped with a 2D LIDAR. The state $x_t$ of the robot is represented by (2.1). The robot uses wheel odometry data as $u_t$ and LIDAR measurement data as $z_t$ to estimate the state.

### 5.5.1 System and measurement models in localization

As typical examples of a system model $f^{\mathrm{pf}}$ and measurement model $g^{\mathrm{pf}}$, this section uses *motion_model_odometry* and *likelihood_field_range_finder_model* in [4]. Each model involves parameters $\alpha$ and $m$, respectively; therefore, $\Theta := \left[\alpha^\top, m^\top\right]^\top$.

The system model *motion_model_odometry* calculates the probability density of the

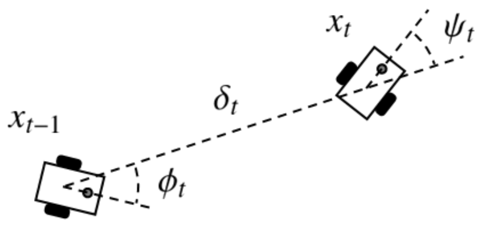


Figure 5.2: Odometry data in system model

current pose $x_t$ from the previous pose $x_{t-1}$ and odometry data $u_t$ as follows [4]:

$$\begin{aligned} f^{\mathrm{pf}}\left(x_t, x_{t-1}, u_t, \Theta\right) = &\ \mathcal{N}\left(\phi_t - \hat{\phi}_t,\ \alpha^{(1)}\phi_t^2 + \alpha^{(2)}\delta_t^2\right) \\ &\times \mathcal{N}\left(\delta_t - \hat{\delta}_t,\ \alpha^{(3)}\delta_t^2 + \alpha^{(4)}\phi_t^2 + \alpha^{(4)}\psi_t^2\right) \\ &\times \mathcal{N}\left(\psi_t - \hat{\psi}_t,\ \alpha^{(1)}\psi_t^2 + \alpha^{(2)}\delta_t^2\right), \end{aligned} \tag{5.34}$$

where $u_t := [\phi_t, \delta_t, \psi_t]^\top$ represents the odometry data at each time in three steps: rotation ($\phi_t$), translation ($\delta_t$), and rotation ($\psi_t$), as shown in Fig. 5.2. In addition, $\hat{\phi}_t$, $\hat{\delta}_t$, and $\hat{\psi}_t$ are computed from $x_t$ and $x_{t-1}$ as follows:

$$\hat{\phi}_t = \mathrm{Arg}\left((\mathrm{x}_t - \mathrm{x}_{t-1}) + \mathbf{i}\,(\mathrm{y}_t - \mathrm{y}_{t-1})\right) - \theta_{t-1}, \tag{5.35}$$

$$\hat{\delta}_t = \sqrt{(\mathrm{x}_{t-1} - \mathrm{x}_t)^2 + (\mathrm{y}_{t-1} - \mathrm{y}_t)^2}, \tag{5.36}$$

$$\hat{\psi}_t = \theta_t - \theta_{t-1} - \hat{\phi}_t \tag{5.37}$$

where $\mathbf{i}$ is the imaginary unit, and $\mathrm{Arg} : \mathbb{C} \to \mathbb{R}$ is the argument of the input complex number. In (5.34), zero-centered normal distributions $\mathcal{N}$ are used to model odometry errors, such as wheel slippage. The parameter vector $\alpha := \left[\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, \alpha^{(4)}\right]^\top \in \mathbb{R}_+^4$ represents the relationship between the amount of movements and variance of odometry errors.

The measurement model *likelihood_field_range_finder_model*

$$g^{\mathrm{pf}}(z_t, x_t, \Theta) = \prod_{\ell=1}^{L} q\left(z_t^{(\ell)}, x_t, m\right) \tag{5.38}$$

uses LIDAR measurement data $z_t$ and the environment map $m \in \mathbb{R}^{n_m}$ to calculate the likelihood of a pose estimate $x_t$ as described in Section 2.3. In the previous chapters, the cell values of $m$ were optimized to improve the estimation accuracy, and this chapter also treats them as the parameters of the measurement model. Note that the measurement model (2.4) has parameters $\kappa_2$ and $\kappa_3$ in addition to $m$, but in this chapter, these values are fixed as $\kappa_2 = 1$

and $\kappa_3 = 0$ for simplicity.

### 5.5.2 Derivation of characteristic eligibility for applying proposed method

To design $\Theta := \left[\alpha^\top, m^\top\right]^\top$ using the proposed method, the characteristic eligibility (5.18) is derived for the models in Section 5.5.1. From (5.34), the log-derivative of $f^{\mathrm{pf}}$ in (5.18) is

$$\begin{aligned}
&\nabla_\Theta \ln\left\{f^{\mathrm{pf}}\left(\hat{x}^{(i)}_{t|t-1}, \hat{x}^{(i)}_{t-1}, u_t, \Theta\right)\right\} \\
&= \nabla_\alpha \ln\left\{\mathcal{N}\left(\phi_t - \hat{\phi}^{(i)}_t,\ \alpha^{(1)}\phi_t^2 + \alpha^{(2)}\delta_t^2\right)\right\} \\
&\quad +\nabla_\alpha \ln\left\{\mathcal{N}\left(\delta_t - \hat{\delta}^{(i)}_t,\ \alpha^{(3)}\delta_t^2 + \alpha^{(4)}\phi_t^2 + \alpha^{(4)}\psi_t^2\right)\right\} \\
&\quad +\nabla_\alpha \ln\left\{\mathcal{N}\left(\psi_t - \hat{\psi}^{(i)}_t,\ \alpha^{(1)}\psi_t^2 + \alpha^{(2)}\delta_t^2\right)\right\}
\end{aligned} \tag{5.39}$$

where $\hat{\phi}^{(i)}_t$, $\hat{\delta}^{(i)}_t$, and $\hat{\psi}^{(i)}_t$ denote the amount of movements (5.35)–(5.37) for the $i$-th particle, and each term in (5.39) can be calculated, e.g., as follows:

$$\frac{\partial}{\partial\alpha^{(1)}} \ln\left\{\mathcal{N}\left(\phi_t - \hat{\phi}^{(i)}_t,\ \alpha^{(1)}\phi_t^2 + \alpha^{(2)}\delta_t^2\right)\right\} = \frac{\left\{(\phi_t - \hat{\phi}^{(i)}_t)^2 - \alpha^{(1)}\phi_t^2 - \alpha^{(2)}\delta_t^2\right\}\phi_t^2}{2\left(\alpha^{(1)}\phi_t^2 + \alpha^{(2)}\delta_t^2\right)^2}. \tag{5.40}$$

The log-derivative of $g^{\mathrm{pf}}$ in (5.18) is derived as:

$$\frac{\partial}{\partial m^{(\mu)}} \ln\left\{g^{\mathrm{pf}}\left(z_t, \hat{x}^{(i)}_{t|t-1}, \Theta\right)\right\} = \frac{\beta^{(i,\mu)}}{m^{(\mu)}} \tag{5.41}$$

where $\beta^{(i,\mu)} \in \{0\} \cup \mathbb{N}$ is the order of the factor $m^{(\mu)}$ in (5.38), or the number of measurement data $z^{(\ell)}_t$ projected onto the $\mu$-th cell as described in Section 3.2.1. By substituting (5.39) and (5.41) into (5.18), the characteristic eligibility is calculated.

### 5.5.3 Numerical simulation

#### 5.5.3.1 Configuration

This section verifies the effectiveness of the proposed method using the 3D dynamic simulator Gazebo. Figure 5.3 shows the environment and robot. The measurement data $z_t$ are obtained from the LIDAR whose possible maximum detection range is 5 m, scanning angle is 360 degrees, and the number of data is $L := 18$ data/rev. In the PF, the number of particles

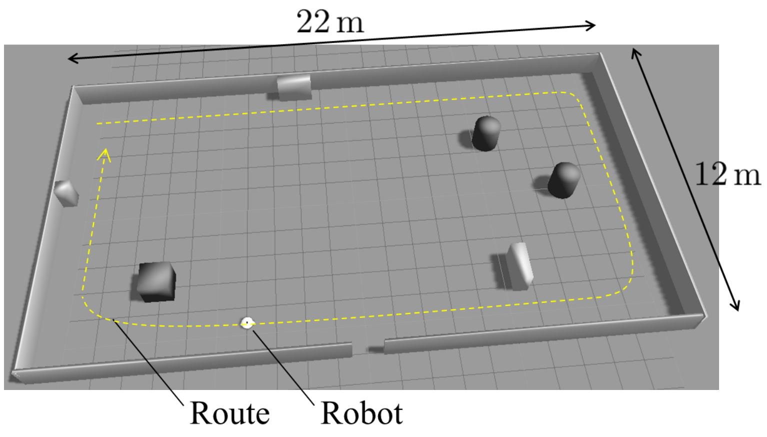


Figure 5.3: Simulation environment

is $N := 100$, and the frequency is 5 Hz. The system model $f^{\text{pf}}$ and measurement model $g^{\text{pf}}$ are given by (5.34) and (5.38), respectively.

The parameter vector $\Theta = \left[\alpha^\top, m^\top\right]^\top$ of the PF was optimized under the following settings. The initial parameter vector of the system model is $\alpha(0) := [0.5, 0.5, 0.5, 0.5]^\top$, and that of the measurement model $m(0)$ is shown in Fig. 5.4, where the map size (number of cells) is $n_m := 240 \times 140$, and the grid scale is 0.1 m/cell. The training data $u_{1:T}^{\text{train}}$, $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$ were collected as follows. First, the robot followed the route in Fig. 5.3 at 2 m/s to obtain $u_{1:T}^{\text{train}}$ and $z_{1:T}^{\text{train}}$. Second, the PF was performed with $N := 10{,}000$ particles (a larger number than usual for this robot) by using $\alpha(0)$, $m(0)$, $u_{1:T}^{\text{train}}$, and $z_{1:T}^{\text{train}}$, and $x_{1:T}^{\text{train}}$ were approximately computed as the following maximum likelihood pose:

$$x_t^{\text{train}} := \hat{x}_{t|t-1}^{(i^*)} \quad \text{s.t.} \quad i^* = \arg\max_i w_t^{(i)}. \tag{5.42}$$

Note that the true poses, which could be obtained from the simulator, was not used in the design of the PF to simulate a realistic scenario. The initial particle set is $\chi_0 := \left\{\hat{x}^{\text{ini}}, \hat{x}^{\text{ini}}, \ldots, \hat{x}^{\text{ini}}\right\}$ where $\hat{x}^{\text{ini}} \in \mathbb{R}^3$ is the initial true state. The function $\rho$ in (5.4) is set as follows, using the estimation error (5.8),

$$\rho\left(x_t, \chi_t, \bar{\chi}_t, \Theta\right) := \exp\left\{-\left(x_t - \bar{h}\left(\bar{\chi}_t\right)\right)^\top \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(x_t - \bar{h}\left(\bar{\chi}_t\right)\right)\right\}, \tag{5.43}$$

where the exponential function is used to prevent the magnitude of the value from becoming too large. The gradient $\zeta_t$ of $\rho$ can be calculated similar to (5.21). The learning rate,

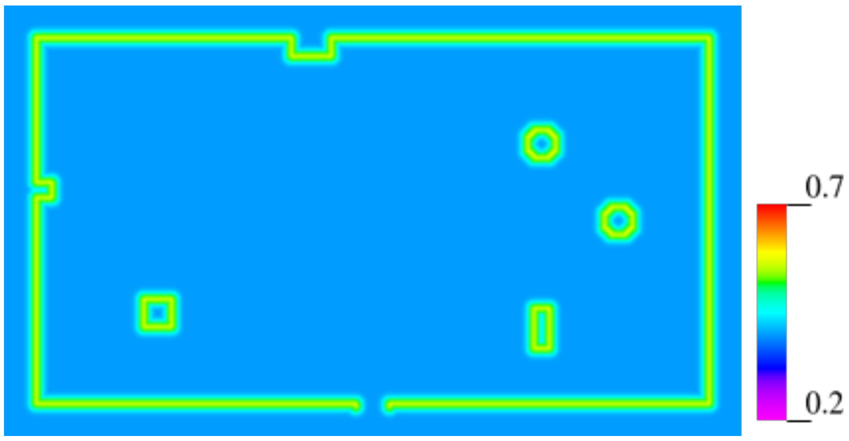


Figure 5.4: Initial map $m(0)$

maximum iteration number, and reinforcement baseline are

$$\eta(k) := \frac{0.4}{k + 4 \times 10^6}, \tag{5.44}$$

$k_{\max} := 4 \times 10^6$, and $b := 0.8$, respectively. As the parameters $\alpha$ in $f^{\mathrm{pf}}$ are positive numbers close to zero and change sensitively during optimization, this section optimized $\alpha' := \left[\ln \alpha^{(1)}, \ln \alpha^{(2)}, \ln \alpha^{(3)}, \ln \alpha^{(4)}\right]^\top \in \mathbb{R}^4$ instead of $\alpha$ to keep $\alpha$ positive.

#### 5.5.3.2 Result

The result of optimizing the parameters under this configuration is as follows. The change in the objective function is shown in Fig. 5.5, where the thin blue line is the value of $J(\Theta(k))$ (only the data at $k = 0, 1000, 2000, \dots$ are displayed), and the thick red line is its moving average. It can be seen that the parameters converged to the (local) optimum $\Theta^* := \left[\alpha^{*\top}, m^{*\top}\right]^\top$. The parameter vector $\alpha$ changed as shown in Fig. 5.6, and the final value was $\alpha^* = [0.490, 0.079, 0.339, 0.505]^\top$. The final value $m^*$ of $m$ is shown in Fig. 5.7, where uniquely shaped objects in the map are emphasized (red or purple spots). By using $m^*$, the robot can use these spots more effectively as clues for localization.

Next, the performance of the designed PF was evaluated. The test data $u^{\mathrm{test}}_{1:T}$, $z^{\mathrm{test}}_{1:T}$, and $x^{\mathrm{test}}_{1:T}$ were obtained as follows. The odometry data $u^{\mathrm{test}}_{1:T}$ and measurement data $z^{\mathrm{test}}_{1:T}$ were measured while the robot followed the similar trajectory as Section 5.5.3.1, and the true poses $x^{\mathrm{test}}_{1:T}$ were obtained from the simulator. The PF estimation was performed 20 times using these

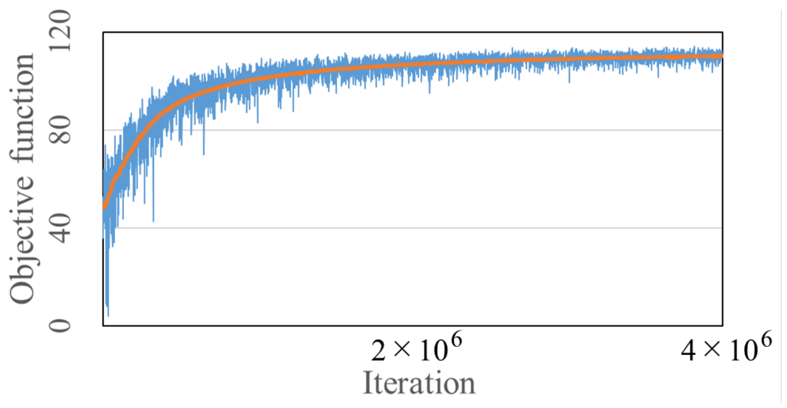


Figure 5.5: Change in objective function

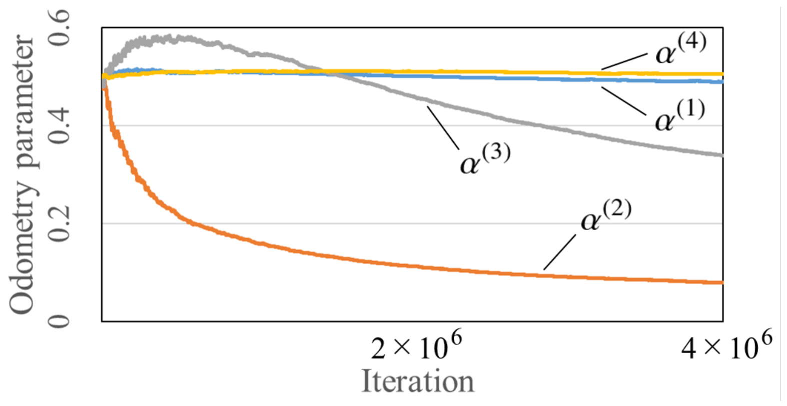


Figure 5.6: Change in odometry parameters $\alpha$

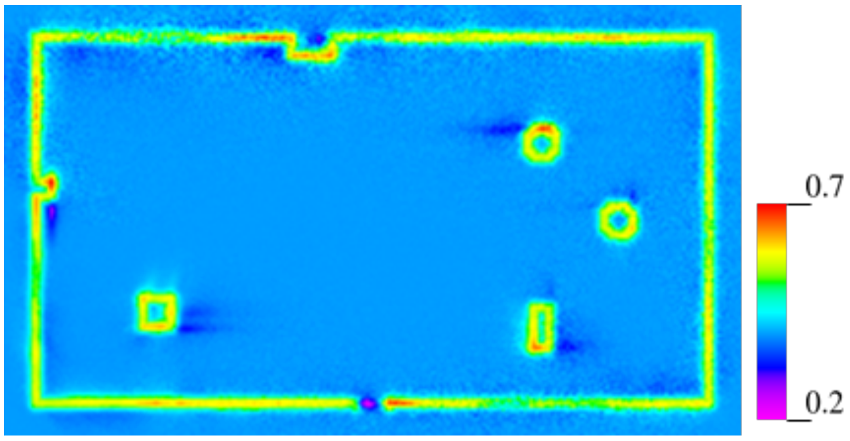


Figure 5.7: Optimized map $m^*$

test data to calculate the objective function $J$ consisting of (5.43) and the estimation error

$$\Delta_t := \sqrt{\left(x_t^{\text{test}} - \bar{h}\left(\bar{\chi}_t\right)\right)^\top \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left(x_t^{\text{test}} - \bar{h}\left(\bar{\chi}_t\right)\right)}. \tag{5.45}$$

Table 5.1: Performance of each PF

| PF | Mean of $J$ | Mean of $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ (m) |
|---|---|---|
| Initial parameters | 22.2 | 0.816 |
| Only $\alpha$ was optimized | 45.3 | 0.407 |
| Only $m$ was optimized | 72.5 | 0.263 |
| Both $\alpha$ and $m$ were optimized | 84.0 | 0.212 |

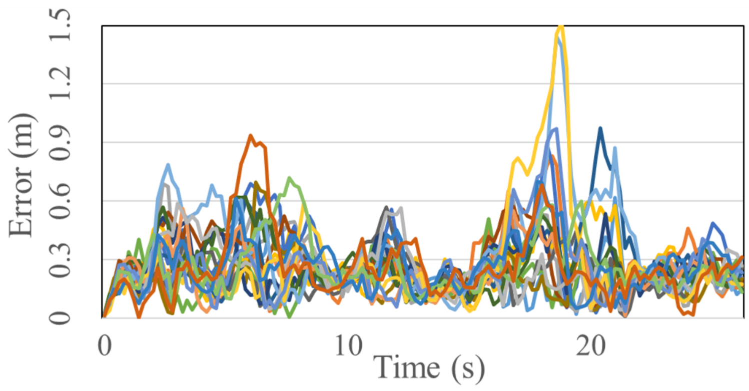


Figure 5.8: Estimation error with PF equivalent to Chapter 3

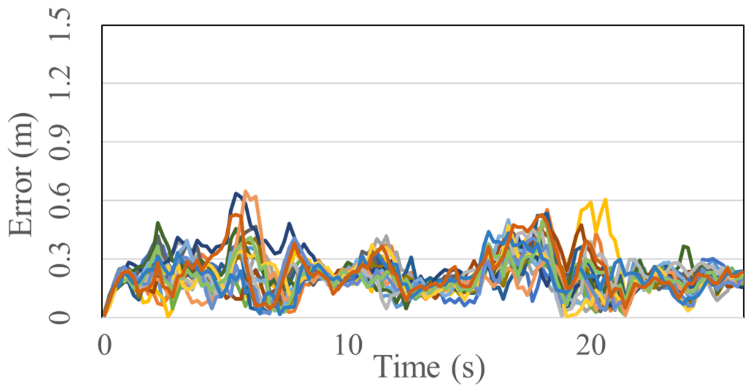


Figure 5.9: Estimation error with PF designed by proposed method

Table 5.1 shows the mean of $J$ and the mean of the mean absolute error $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ over 20 estimations. In addition to the designed PF where both $\alpha$ and $m$ were optimized, the results of the PF with the initial parameters $\Theta(0)$, PF where only $\alpha$ was optimized, and PF where only $m$ was optimized (equivalent to the method in Chapter 3) are presented for comparison. In addition, Figs. 5.8 and 5.9 illustrate the errors $\Delta_t$ of 20 estimations for the PF equivalent to Chapter 3 (only $m$ was optimized) and the designed PF (both $\alpha$ and $m$ were optimized), respectively. These results suggest that the proposed method further reduces estimation errors compared to the conventional method.

Note that there were no objects that were difficult to detect by the LIDAR (e.g., glass),

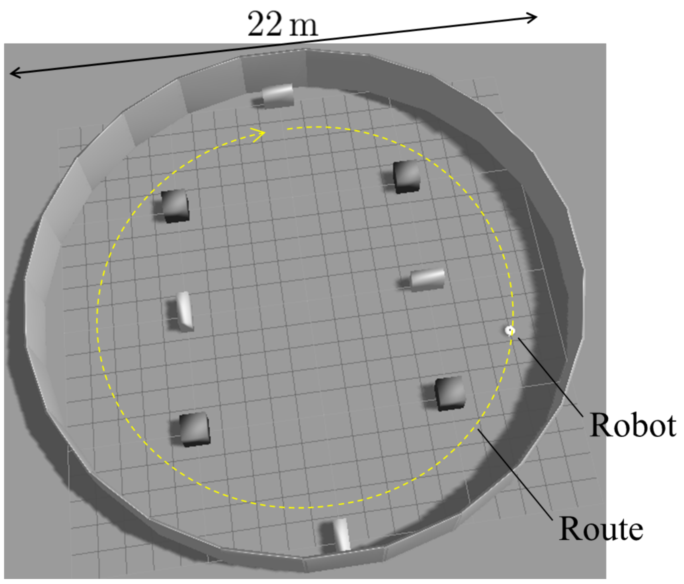


Figure 5.10: Simulation environment in additional experiment

so the color differences in Fig. 5.7 were not caused by the ease of detection by the LIDAR. Therefore, Fig. 5.4 is a more accurate representation of the environment; whereas Fig. 5.7 provides the PF better estimation accuracy. This is because the estimation accuracy (5.43) was optimized; therefore, Fig. 5.7 cannot be obtained by the conventional method that optimizes the likelihood of the parameters, such as [50–52].

#### 5.5.3.3 Verification in different environment

To show the robustness of the proposed method, an additional experiment in a different environment was performed. The environment and the initial map $m(0)$ are shown in Figs. 5.10 and 5.11, respectively. The map size is $n_m := 220 \times 220$, and the grid scale is 0.1 m/cell. Other settings are the same as in Section 5.5.3.1.

The convergence of the optimization was achieved as shown in Fig. 5.12, which illustrates the change in the objective function. The odometry parameters $\alpha$ changed as shown in Fig. 5.13, and the final values were $\alpha^* = [0.445, 0.059, 0.292, 0.490]^\top$. Figure 5.14 shows the optimized map $m^*$.

The performance of the designed PF was evaluated in the same way as in Section 5.5.3.2, and the results are shown in Table 5.2, Figs. 5.15, and 5.16. Table 5.2 lists the values of $J$ and $\Delta_t$ for four PFs. Figures 5.15 and 5.16 demonstrate the transition of $\Delta_t$ in the case of the PF equivalent to Chapter 3 and the PF designed by the proposed method, respectively. These results confirm that the proposed method produced a high-performance PF as in the

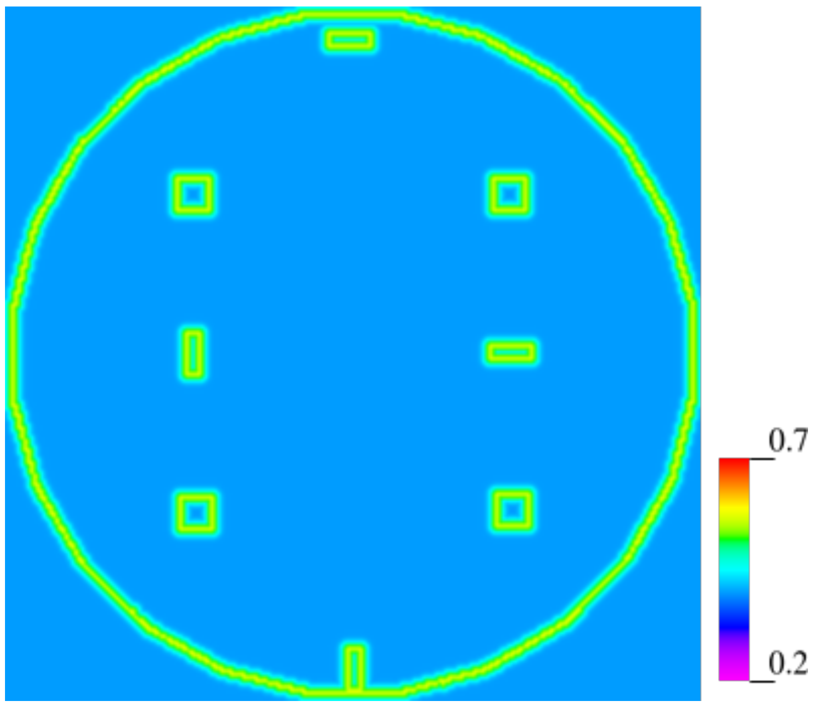


Figure 5.11: Initial map $m(0)$ in additional experiment

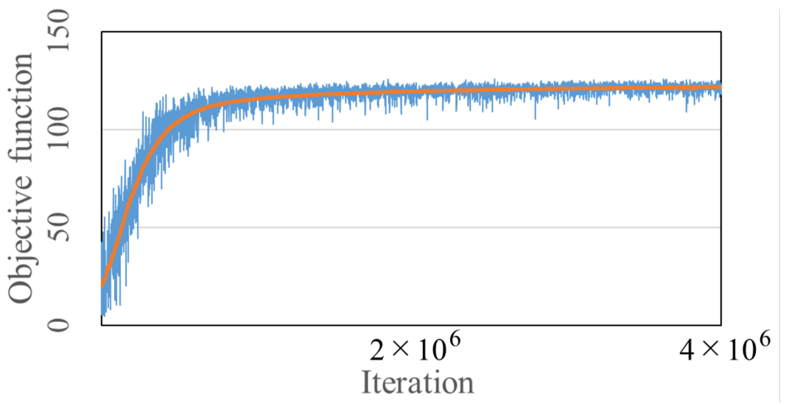


Figure 5.12: Change in objective function in additional experiment

first experiment. Even though the proposed method designed the PFs with almost the same settings, it showed good performance in both environments; thus, the proposed method is considered to be robust to changes in an environment.

## 5.6 Conclusion

This chapter proposed a method of designing the system and measurement models for a PF. As the proposed method can optimize various objective functions, it has multiple

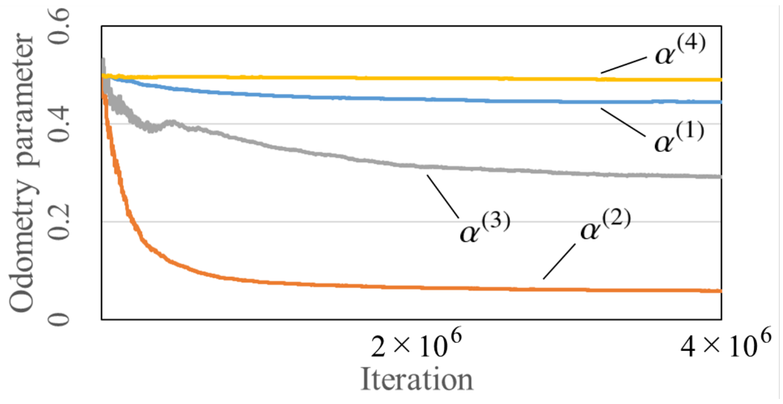


Figure 5.13: Change in odometry parameters $\alpha$ in additional experiment

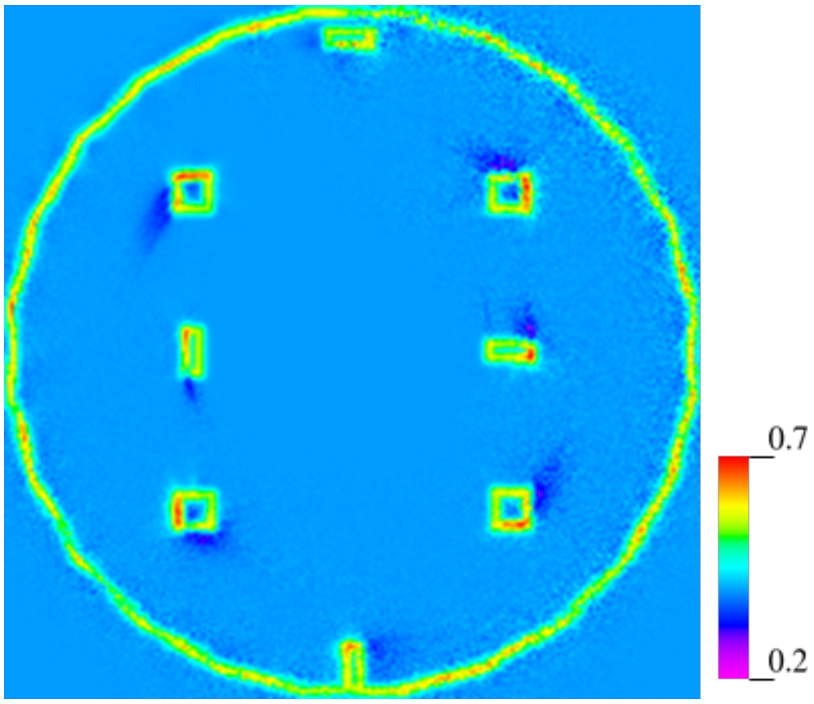


Figure 5.14: Optimized map $m^*$ in additional experiment

applications, such as nonlinear system identification and improvement of the performance of a PF. The conditions to guarantee the convergence of the optimization were derived. Furthermore, the method was applied to mobile robot localization to confirm that it provided better performance than the method proposed in the previous chapter. Although this study considered the most basic PF for simplicity, there are many variations of PF, such as auxiliary PF [70], Rao-Blackwellized PF [71,72], and cluster PF [73,74]. The future work is to develop design methods for these.

Table 5.2: Performance of each PF in additional experiment

| PF | Mean of $J$ | Mean of $\frac{1}{T}\sum_{t=1}^{T}\Delta_t$ (m) |
|---|---|---|
| Initial parameters | 8.7 | 1.325 |
| Only $\alpha$ was optimized | 22.5 | 0.512 |
| Only $m$ was optimized | 76.6 | 0.267 |
| Both $\alpha$ and $m$ were optimized | 83.9 | 0.232 |

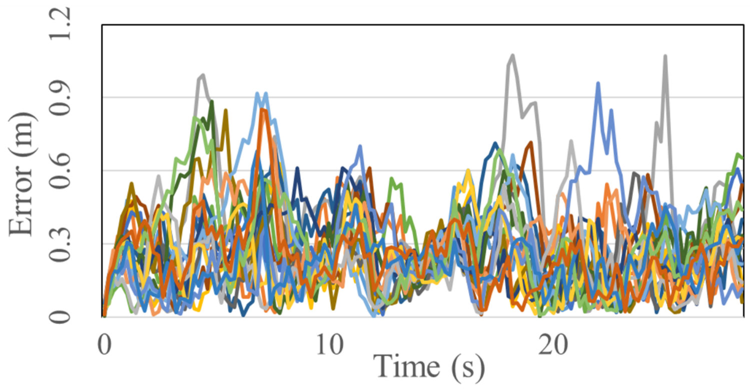


Figure 5.15: Estimation error with PF equivalent to Chapter 3 in additional experiment

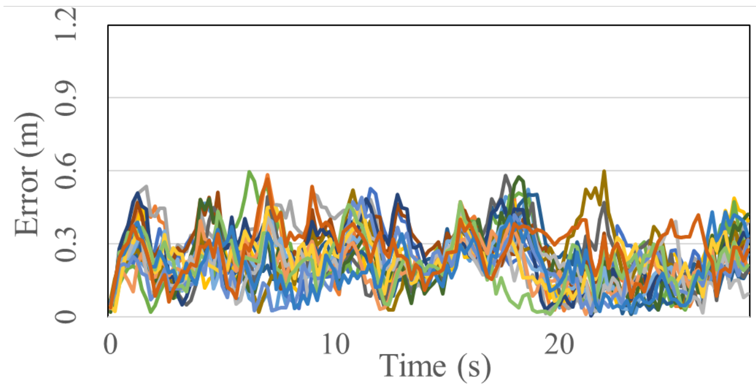


Figure 5.16: Estimation error with PF designed by proposed method in additional experiment

# Chapter 6

# Conclusion

## 6.1 Summery

This thesis proposed a new kind of map for mobile robot localization, a highlighted map, and its generation method. Furthermore, the method was generalized to PF design problem.

In Chapter 3, the concept of the highlighted map was introduced. By using this map, the localization performance can be improved without having to update robot's sensors or online computation. Furthermore, this map can be easily combined with many other existing MCL-based algorithms. This chapter formulated the problem of generating a highlighted map and proposed a numerical optimization method based on RL as a solution. This 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 was proven that the optimization converges under certain technical assumptions.

Chapter 4 constructed the robustification method of a highlighted map. This method introduces a virtual obstacle causing measurement noise and learns both the worst-case obstacle behavior and the optimal highlighted map simultaneously based on RARL. The highlighted map generated by this method improves the localization performance even in the presence of measurement noise.

Chapter 5 presented a novel method to design the system and measurement models in a PF. First, the particle filter design problem was formulated, and a solution was proposed. This is a numerical optimization method, which is a generalization of the method in Chapter 3. Compared to other PF design methods, the advantage of the method is that it can accommodate various objective functions, such as the estimation accuracy of the PF, the variance of the particles, the likelihood of the parameters that determine both models, and the regularization term of the parameters. Moreover, the conditions to guarantee that the optimization converges with probability 1 were derived. Next, the PF for mobile robot localization was designed by this method to show that this PF design method can be applied to practical-scale problems. It should be noted that the robustification method in Chapter 4 can be applied to this PF design method if an effective virtual obstacle for the target system is introduced.

## 6.2 Future work

To propose the first highlighted map generation method and a novel PF design method, this thesis simplified these problems as much as possible. Hence, several issues should be considered to make the proposed methods more general and practical. For example, the estimators designed by the proposed methods (MCL with the generated highlighted map or the designed PF) are specific to a certain situation because they are designed from a single training data set. Therefore, the estimators are not guaranteed to be effective for all situations. Also, the proposed methods need to measure the true states as training data in order to minimize the estimation errors. Several procedures for acquiring true states are presented in Chapter 3 and 5, but these procedures are sometimes difficult to use.

To design an estimator effective for various possible situations, the following approaches could be used. The first is to prepare a large number of training data and use all of them by turns to learn. This is expected to extract common patterns for each data set and yield a highly versatile estimator. Another approach designs an optimal estimator for each training data set, and then switches between these designed estimators according to the situation.

It is also necessary to devise a method of designing an estimator that minimizes the estimation errors without using true states as training data. In [75], RL was used to learn a policy for estimating hidden states and parameters of nonlinear dynamical systems. They defined the accumulated errors of observable states as the objective function to search for an estimation policy that reduces the estimation errors; so their method uses only measurement data and does not need true states. By applying the strategy of the method, it is expected that we can obtain an estimator design method that minimizes estimation errors without measuring true states.

Solving these issues and deriving a more general and practical method of designing estimators is an important task for the future.

# Appendices

## A Complement to proof of Theorem 1

This appendix provides lemmas about conditions (D1)–(D3) in the proof of Theorem 1.

**Lemma 1.** *For Algorithm 2, if (C1)–(C5) are satisfied, then (D1) holds.*

*Proof.* Several variables have to be defined first. From (2.8) and (3.8)–(3.10), the state transition probability and the expected reward under $\pi$ are derived as

$$P_{s_t}^{s_{t+1}} := p\left(s_{t+1} \,\middle|\, s_t, \pi\right) = \sum_{a \in \mathbb{A}} \tilde{\pi}(a, s_t, m)\, p\left(s_{t+1} \,\middle|\, s_t, a\right), \tag{A.1}$$

$$R_{s_t} := E\left[r_t \,\middle|\, s_t, \pi\right] = \sum_{a \in \mathbb{A}} \tilde{\pi}(a, s_t, m)\, E\left[r_t \,\middle|\, s_t, a\right], \tag{A.2}$$

where $\mathbb{A} \subset \{0\} \cup \mathbb{N}^N$ is the action set. The probability distribution of the initial state is defined as

$$P_{\chi_0, u_1}^{s_1} := p\left(s_1 \,\middle|\, \chi_0, u_1\right). \tag{A.3}$$

Then, from (3.1) and (3.6), the gradient of $\bar{J}$ is represented by (A.1)–(A.3), as follows:

$$\begin{aligned} \nabla_m \bar{J}(m) &= \nabla_m E\left[\sum_{t=1}^{T} r_t\right] \\ &= \sum_{t=1}^{T} \nabla_m E\left[r_t\right] \\ &= \sum_{t=1}^{T} \nabla_m \left(\int_{\mathbb{S}^t} P_{\chi_0, u_1}^{s_1} P_{s_1}^{s_2} \cdots P_{s_{t-1}}^{s_t} R_{s_t}\, ds_{1:t}\right), \end{aligned} \tag{A.4}$$

where $\mathbb{S} := \mathbb{X}^N$ is the space of all possible states at each time. For the integrand of (A.4), this appendix defines

$$\mathcal{E}_t\left(s_{1:t}, m\right) := P_{\chi_0, u_1}^{s_1} P_{s_1}^{s_2} \cdots P_{s_{t-1}}^{s_t} R_{s_t}. \tag{A.5}$$

From (C1)–(C5), the following properties hold:

- The function $\mathcal{E}_t$ is integrable, i.e., $\int_{\mathbb{S}^t} |\mathcal{E}_t(s_{1:t}, m)|\, ds_{1:t} < \infty$, for all $m \in \mathbb{M}$ because $\mathbb{S}$ is the bounded set, and $|\mathcal{E}_t(s_{1:t}, m)|$ is bounded.
- The function $\mathcal{E}_t$ is differentiable with respect to $m$, and there exists an integrable function $\tilde{\mathcal{E}}(s_{1:t})$ such that $\|\nabla_m \mathcal{E}_t(s_{1:t}, m)\| \le \tilde{\mathcal{E}}(s_{1:t})$ for all $(s_{1:t}, m) \in \mathbb{S}^t \times \mathbb{M}$. This is because there exists a $\delta_4 \in \mathbb{R}_+$ such that $\|\nabla_m \mathcal{E}_t(s_{1:t}, m)\| \le \delta_4$ for all $(s_{1:t}, m) \in \mathbb{S}^t \times \mathbb{M}$, and $\delta_4$ is integrable, i.e., $\int_{\mathbb{S}^t} |\delta_4|\, ds_{1:t} < \infty$.

Thus, the order of differentiation and integration can be interchanged by Proposition 5.9 in [76]. As a result,

$$\begin{aligned}
&\nabla_m \left( \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t} \right) \\
&= \int_{\mathbb{S}^t} \nabla_m \left( P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_t}_{s_{t-1}} R_{s_t} \right) ds_{1:t} \\
&= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} \left( \nabla_m P^{s_2}_{s_1} \right) \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t} \\
&\quad + \cdots \\
&\quad + \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots \left( \nabla_m P^{s_t}_{s_{t-1}} \right) R_{s_t}\, ds_{1:t} \\
&\quad + \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_t}_{s_{t-1}} \left( \nabla_m R_{s_t} \right) ds_{1:t}.
\end{aligned} \tag{A.6}$$

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

$$E\left[(r_t - b)U_t\right] = E\left[ (r_t - b) \sum_{\tau=1}^{t} e_\tau \right] = \sum_{\tau=1}^{t} E\left[r_t e_\tau\right] - b \sum_{\tau=1}^{t} E\left[e_\tau\right]. \tag{A.7}$$

The expected value of $r_t e_\tau$ in the case of $\tau < t$ is given by

$$\begin{aligned}
E\left[r_t e_\tau\right] &= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left( P^{s_{\tau+1}}_{s_\tau} \cdot e_\tau \right) P^{s_{\tau+2}}_{s_{\tau+1}} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t} \\
&= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left\{ \sum_{a \in \mathbb{A}} \tilde{\pi}(a, s_\tau, m)\, p\left(s_{\tau+1} \,\middle|\, s_\tau, a\right) \cdot \nabla_m \ln\left\{\tilde{\pi}(a, s_\tau, m)\right\} \right\} \\
&\quad \times P^{s_{\tau+2}}_{s_{\tau+1}} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t} \\
&= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left\{ \sum_{a \in \mathbb{A}} \nabla_m \tilde{\pi}(a, s_\tau, m)\, p\left(s_{\tau+1} \,\middle|\, s_\tau, a\right) \right\} P^{s_{\tau+2}}_{s_{\tau+1}} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t} \\
&= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0, u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left( \nabla_m P^{s_{\tau+1}}_{s_\tau} \right) P^{s_{\tau+2}}_{s_{\tau+1}} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t},
\end{aligned} \tag{A.8}$$

and similarly,

$$E\left[r_t e_t\right] = \int_{\mathbb{S}^t} P^{s_1}_{\chi_0,u_1} P^{s_2}_{s_1} \cdots P^{s_t}_{s_{t-1}} \left(\nabla_m R_{s_t}\right) ds_{1:t}, \tag{A.9}$$

$$\begin{aligned} E\left[e_\tau\right] &= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0,u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left\{\sum_{a\in\mathbb{A}} \tilde{\pi}(a, s_\tau, m) \cdot \nabla_m \ln\left\{\tilde{\pi}(a, s_\tau, m)\right\}\right\} ds_{1:\tau} \\ &= \int_{\mathbb{S}^t} P^{s_1}_{\chi_0,u_1} P^{s_2}_{s_1} \cdots P^{s_\tau}_{s_{\tau-1}} \left\{\nabla_m \sum_{a\in\mathbb{A}} \tilde{\pi}(a, s_\tau, m)\right\} ds_{1:\tau} \\ &= 0. \end{aligned} \tag{A.10}$$

From (A.6)–(A.10),

$$\nabla_m \left(\int_{\mathbb{S}^t} P^{s_1}_{\chi_0,u_1} P^{s_2}_{s_1} \cdots P^{s_t}_{s_{t-1}} R_{s_t}\, ds_{1:t}\right) = E\left[(r_t - b)U_t\right]. \tag{A.11}$$

Then, from (A.4) and the fact that $V_T = \sum_{t=1}^{T}(r_t - b)U_t$, (D1) holds. ■

**Lemma 2.** *For Algorithm 2, if (C1)–(C5) are satisfied, then (D2) holds.*

*Proof.* From (C3) and (C4), the derivative of (A.5) is continuous in $m$ for all $s_{1:t} \in \mathbb{S}^t$, which means that for all $(m', \epsilon) \in \mathbb{M} \times \mathbb{R}_+$, there exists a $\delta_5 \in \mathbb{R}_+$ such that for all $m \in \mathbb{M}$,

$$\|m - m'\| < \delta_5 \Rightarrow \max_{s_{1:t}} \|\nabla_m \mathcal{E}_t(s_{1:t}, m) - \nabla_m \mathcal{E}_t(s_{1:t}, m')\| < \epsilon. \tag{A.12}$$

Then, for $\mathcal{F}_t(m) := \nabla_m \left(\int_{\mathbb{S}^t} \mathcal{E}_t(s_{1:t}, m)\, ds_{1:t}\right)$,

$$\|\mathcal{F}_t(m) - \mathcal{F}_t(m')\| \leq \int_{\mathbb{S}^t} \|\nabla_m \mathcal{E}_t(s_{1:t}, m) - \nabla_m \mathcal{E}_t(s_{1:t}, m')\|\; ds_{1:t} < \delta_6\, \epsilon \tag{A.13}$$

where $\delta_6 \in \mathbb{R}_+$ is some constant. Note that the order of differentiation and integration can be interchanged from (C1)–(C5), as explained in Lemma 1. Therefore, $\mathcal{F}_t(m)$ is also continuous. From this fact and (A.4), $\nabla_m \bar{J}(m)$ is continuous. ■

**Lemma 3.** *For Algorithm 2, if (C3)–(C6) are satisfied, then (D3) holds.*

*Proof.* The following equation holds for the random variable $\Xi_K := \sum_{k'=k}^{K} \eta(k')\varphi(k')$,

$$\begin{aligned} E\left[\Xi_K \,\middle|\, \Xi_{k:K-1}\right] &= E\left[\Xi_{K-1} + \eta(K)\varphi(K) \,\middle|\, \Xi_{k:K-1}\right] \\ &= \Xi_{K-1} + \eta(K)\, E\left[\varphi(K) \,\middle|\, \Xi_{k:K-1}\right] \\ &= \Xi_{K-1}. \end{aligned} \tag{A.14}$$

Therefore, the stochastic process $\Xi_K$ ($K = k, k+1, k+2, \dots$) is a martingale, and we obtain the martingale inequality of Doob [63]

$$P\left(\sup_{K \geq k}\left\|\sum_{k'=k}^{K}\eta(k')\varphi(k')\right\| \geq \epsilon\right) \leq \frac{1}{\epsilon^2}E\left[\left\|\sum_{k'=k}^{\infty}\eta(k')\varphi(k')\right\|^2\right]. \tag{A.15}$$

Since $E\left[\varphi(k_1)^\top\varphi(k_2)\right] = 0$ if $k_1 \neq k_2$, the right side of (A.15) can be transformed as follows:

$$\frac{1}{\epsilon^2}E\left[\left\|\sum_{k'=k}^{\infty}\eta(k')\varphi(k')\right\|^2\right] = \frac{1}{\epsilon^2}\sum_{k'=k}^{\infty}\eta(k')^2 E\left[\|\varphi(k')\|^2\right]. \tag{A.16}$$

In addition,

$$\begin{aligned} E\left[\|\varphi(k')\|^2\right] &= E\left[\left\|V_T - \nabla_m \bar{J}(m(k'))\right\|^2\right] \\ &= E\left[\|V_T\|^2\right] - \left\|\nabla_m \bar{J}(m(k'))\right\|^2 \\ &\leq E\left[\|V_T\|^2\right] \\ &= E\left[\left\|\sum_{t=1}^{T}(r_t - b)\sum_{\tau=1}^{t} e_\tau\right\|^2\right] \\ &\leq E\left[\left(\sum_{t=1}^{T}|r_t - b|\sum_{\tau=1}^{t}\|e_\tau\|\right)^2\right]. \end{aligned} \tag{A.17}$$

Here, (C5) gives $|r_t| < \infty$, and we have $\|e_\tau\| < \infty$ from (3.12) and (C3), (C4). Thus, there exists a $\delta_7 \in \mathbb{R}_+$ such that

$$\sum_{t=1}^{T}|r_t - b|\sum_{\tau=1}^{t}\|e_\tau\| < \delta_7. \tag{A.18}$$

From (A.15)–(A.18),

$$P\left(\sup_{K \geq k}\left\|\sum_{k'=k}^{K}\eta(k')\varphi(k')\right\| \geq \epsilon\right) < \frac{\delta_7^2}{\epsilon^2}\sum_{k'=k}^{\infty}\eta(k')^2. \tag{A.19}$$

Equation (A.19) and (C6) indicate that (D3) holds. ■

## B Complement to proof of Theorem 2

This appendix gives the three lemmas to prove Theorem 2.

**Lemma 4.** *For Algorithm 5, if (C9)–(C11) are satisfied, then (D4) holds.*

*Proof.* The particle set $\chi_{t-1}$, which composes the RL state $s_t = (\chi_{t-1}, c_t)$, can be denoted as $\chi(a_{t-1})$ since it is uniquely determined from $a_{t-1}$. Therefore,

$$\tilde{\pi}(a_t, s_t, \Theta) = \tilde{\pi}\left(a_t, (\chi(a_{t-1}), c_t), \Theta\right), \tag{B.1}$$

$$\tilde{\rho}\left(s_t, a_t, \Theta\right) = \tilde{\rho}\left((\chi(a_{t-1}), c_t), a_t, \Theta\right). \tag{B.2}$$

With the notation, the following functions are defined:

$$P^{a_t}_{t,a_{t-1}} := p\left(a_t \,\middle|\, t, a_{t-1}, \pi\right) = \tilde{\pi}\left(a_t, (\chi(a_{t-1}), c_t), \Theta\right), \tag{B.3}$$

$$R_{t,a_{t-1:t}} := E\left[r_t \,\middle|\, t, a_{t-1}, a_t, \pi\right] = \tilde{\rho}\left((\chi(a_{t-1}), c_t), a_t, \Theta\right), \tag{B.4}$$

where in the case of $t = 1$,

$$P^{a_1}_{1,a_0} := p\left(a_1 \,\middle|\, \chi_0, \pi\right) = \tilde{\pi}\left(a_1, (\chi_0, 1), \Theta\right), \tag{B.5}$$

$$R_{1,a_{0:1}} := E\left[r_1 \,\middle|\, a_1, \pi\right] = \tilde{\rho}\left((\chi_0, 1), a_1, \Theta\right). \tag{B.6}$$

Then, the gradient of $\bar{J}$ is represented as:

$$\begin{aligned}\nabla_\Theta \bar{J}(\Theta) &= \nabla_\Theta E\left[\sum_{t=1}^{T} r_t\right] \\ &= \sum_{t=1}^{T} \nabla_\Theta E\left[r_t\right] \\ &= \sum_{t=1}^{T} \nabla_\Theta \left(\sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \mathcal{G}_t\left(a_{1:t}, \Theta\right) da^{\mathrm{p}}_{1:t}\right)\end{aligned} \tag{B.7}$$

where $\mathbb{A}^{\mathrm{p}} \subseteq \mathbb{R}^{n \times N}$ and $\mathbb{A}^{\mathrm{r}} \subset \mathbb{N}^N$ are the sets of $a^{\mathrm{p}}_t$ and $a^{\mathrm{r}}_t$, respectively, and

$$\mathcal{G}_t\left(a_{1:t}, \Theta\right) := P^{a_1}_{1,a_0} P^{a_2}_{2,a_1} \cdots P^{a_t}_{t,a_{t-1}} R_{t,a_{t-1:t}}. \tag{B.8}$$

From (C10) and (C11), the following properties hold for (B.8):

- The function $\mathcal{G}_t$ is integrable for all $\Theta \in \mathbb{P}$ because $\int_{(\mathbb{A}^{\mathrm{p}})^t} |\mathcal{G}_t(a_{1:t}, \Theta)|\, da^{\mathrm{p}}_{1:t} = E[|r_t|] < \infty$.

- The function $\mathcal{G}_t$ is differentiable with respect to $\Theta$, and there exists an integrable function $\tilde{\mathcal{G}}(a_{1:t})$ such that $\|\nabla_\Theta \mathcal{G}_t(a_{1:t}, \Theta)\| \leq \tilde{\mathcal{G}}(a_{1:t})$ for all $(a_{1:t}, \Theta) \in (\mathbb{A}^{\mathrm{p}} \times \mathbb{A}^{\mathrm{r}})^t \times \mathbb{P}$. This is because for the functions

$$\mathcal{P}^{(\tau)}_t(a_{1:t}, \Theta) := P^{a_1}_{1,a_0} \cdots \left(\nabla_\Theta P^{a_\tau}_{\tau,a_{\tau-1}}\right) \cdots P^{a_t}_{t,a_{t-1}} R_{t,a_{t-1:t}}, \tag{B.9}$$

$$\mathcal{R}_t(a_{1:t}, \Theta) := P^{a_1}_{1,a_0} \cdots P^{a_t}_{t,a_{t-1}} \left(\nabla_\Theta R_{t,a_{t-1:t}}\right), \tag{B.10}$$

the following inequalities hold:

$$\|\nabla_\Theta \mathcal{G}_t(a_{1:t}, \Theta)\| \leq \sum_{\tau=1}^{t} \left\|\mathcal{P}^{(\tau)}_t(a_{1:t}, \Theta)\right\| + \|\mathcal{R}_t(a_{1:t}, \Theta)\|, \tag{B.11}$$

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \left\|\mathcal{P}^{(\tau)}_t(a_{1:t}, \Theta)\right\| da^{\mathrm{p}}_{1:t} = E[\|r_t e_\tau\|] < \infty, \tag{B.12}$$

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \|\mathcal{R}_t(a_{1:t}, \Theta)\|\, da^{\mathrm{p}}_{1:t} = E[\|\zeta_t\|] < \infty, \tag{B.13}$$

where the equality in (B.12) results from

$$\nabla_\Theta P^{a_\tau}_{\tau,a_{\tau-1}} = P^{a_\tau}_{\tau,a_{\tau-1}} \times \nabla_\Theta \ln P^{a_\tau}_{\tau,a_{\tau-1}} = P^{a_\tau}_{\tau,a_{\tau-1}} \times e_\tau. \tag{B.14}$$

From these properties, the order of differentiation and integration can be interchanged by Proposition 5.9 in [76], and thus,

$$\nabla_\Theta \left( \sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \mathcal{G}_t(a_{1:t}, \Theta)\, da^{\mathrm{p}}_{1:t} \right) = \sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \nabla_\Theta \mathcal{G}_t(a_{1:t}, \Theta)\, da^{\mathrm{p}}_{1:t}. \tag{B.15}$$

On the other hand, the expected value of $(r_t - b)U_t + \zeta_t$ in Algorithm 5 can be transformed as follows:

$$E[(r_t - b)U_t + \zeta_t] = \sum_{\tau=1}^{t} E[r_t e_\tau] - b \sum_{\tau=1}^{t} E[e_\tau] + E[\zeta_t]. \tag{B.16}$$

Each term in the right side of (B.16) is calculated similar to (B.12), as follows:

$$E\left[r_t e_\tau\right] = \sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \mathcal{P}^{(\tau)}_t\left(a_{1:t}, \Theta\right) da^{\mathrm{p}}_{1:t}, \tag{B.17}$$

$$E\left[e_\tau\right] = 0, \tag{B.18}$$

$$E\left[\zeta_t\right] = \sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \mathcal{R}_t\left(a_{1:t}, \Theta\right) da^{\mathrm{p}}_{1:t}. \tag{B.19}$$

Therefore,

$$E\left[(r_t - b)U_t + \zeta_t\right] = \sum_{a^{\mathrm{r}}_{1:t} \in (\mathbb{A}^{\mathrm{r}})^t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \nabla_\Theta \mathcal{G}_t\left(a_{1:t}, \Theta\right) da^{\mathrm{p}}_{1:t}. \tag{B.20}$$

From (B.7), (B.15), (B.20), and the fact that $V_T = \sum_{t=1}^{T} \left((r_t - b)U_t + \zeta_t\right)$, (D4) holds. ∎

**Lemma 5.** *For Algorithm 5, if (C9)–(C11) are satisfied, then (D5) holds.*

*Proof.* From (B.7), it is sufficient to prove that the function

$$\mathcal{H}_t(\Theta) := \nabla_\Theta \left( \int_{(\mathbb{A}^{\mathrm{p}})^t} \mathcal{G}_t\left(a_{1:t}, \Theta\right)\ da^{\mathrm{p}}_{1:t} \right) \tag{B.21}$$

is continuous. Since the order of differentiation and integration can be interchanged as explained in Lemma 4, for all $(\Theta', \Theta) \in \mathbb{P}^2$, the following inequality holds:

$$\begin{aligned} \left\|\mathcal{H}_t(\Theta') - \mathcal{H}_t(\Theta)\right\| &\le \int_{(\mathbb{A}^{\mathrm{p}})^t} \left\|\nabla_\Theta \mathcal{G}_t\left(a_{1:t}, \Theta'\right) - \nabla_\Theta \mathcal{G}_t\left(a_{1:t}, \Theta\right)\right\| da^{\mathrm{p}}_{1:t} \\ &\le \sum_{\tau=1}^{t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \left\|\mathcal{P}^{(\tau)}_t\left(a_{1:t}, \Theta'\right) - \mathcal{P}^{(\tau)}_t\left(a_{1:t}, \Theta\right)\right\| da^{\mathrm{p}}_{1:t} \\ &\quad + \int_{(\mathbb{A}^{\mathrm{p}})^t} \left\|\mathcal{R}_t\left(a_{1:t}, \Theta'\right) - \mathcal{R}_t\left(a_{1:t}, \Theta\right)\right\| da^{\mathrm{p}}_{1:t}. \end{aligned} \tag{B.22}$$

First, it is proven that there exists a $\delta_8 \in \mathbb{R}_+$ such that

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \left\|\mathcal{P}^{(\tau)}_t\left(a_{1:t}, \Theta'\right) - \mathcal{P}^{(\tau)}_t\left(a_{1:t}, \Theta\right)\right\| da^{\mathrm{p}}_{1:t} \le \delta_8 \left\|\Theta' - \Theta\right\|. \tag{B.23}$$

From (C10) and Taylor's theorem, there exists a $\lambda \in (0, 1)$ such that

$$\begin{aligned} &\mathcal{P}^{(\tau,i)}_t\left(a_{1:t}, \Theta'\right) - \mathcal{P}^{(\tau,i)}_t\left(a_{1:t}, \Theta\right) \\ &\quad = \sum_{j=1}^{n_\Theta} \left(\Theta'^{(j)} - \Theta^{(j)}\right) \frac{\partial \mathcal{P}^{(\tau,i)}_t}{\partial \Theta^{(j)}}\left(a_{1:t}, \Theta + \lambda\left(\Theta' - \Theta\right)\right) \end{aligned} \tag{B.24}$$

where $\mathcal{P}_t^{(\tau,i)}$ is the $i$-th component of $\mathcal{P}_t^{(\tau)}$, and $\Theta^{(j)}$ is the $j$-th component of $\Theta$. Therefore,

$$\begin{aligned}\int_{(\mathbb{A}^{\mathrm{p}})^t} &\left|\mathcal{P}_t^{(\tau,i)}(a_{1:t},\Theta') - \mathcal{P}_t^{(\tau,i)}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} \\ &\leq \sum_{j=1}^{n_\Theta} \left|\Theta'^{(j)} - \Theta^{(j)}\right| \int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\frac{\partial \mathcal{P}_t^{(\tau,i)}}{\partial \Theta^{(j)}}(a_{1:t},\Theta + \lambda(\Theta' - \Theta))\right| da_{1:t}^{\mathrm{p}}. \end{aligned} \tag{B.25}$$

Also,

$$\begin{aligned}\int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\frac{\partial \mathcal{P}_t^{(\tau,i)}}{\partial \Theta^{(j)}}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} \leq &\sum_{\tau'=1}^{t} \int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\mathcal{V}_t^{(\tau,i,\tau',j)}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} \\ &+ \int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\mathcal{W}_t^{(\tau,i,j)}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} \end{aligned} \tag{B.26}$$

where

$$\mathcal{V}_t^{(\tau,i,\tau',j)}(a_{1:t},\Theta) := P_{1,a_0}^{a_1} \cdots \frac{\partial P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(i)}} \cdots \frac{\partial P_{\tau',a_{\tau'-1}}^{a_{\tau'}}}{\partial \Theta^{(j)}} \cdots P_{t,a_{t-1}}^{a_t} R_{t,a_{t-1:t}}, \tag{B.27}$$

$$\mathcal{W}_t^{(\tau,i,j)}(a_{1:t},\Theta) := P_{1,a_0}^{a_1} \cdots \frac{\partial P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(i)}} \cdots P_{t,a_{t-1}}^{a_t} \frac{\partial R_{t,a_{t-1:t}}}{\partial \Theta^{(j)}}. \tag{B.28}$$

Each term in the right side of (B.26) is calculated as:

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\mathcal{V}_t^{(\tau,i,\tau',j)}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} = \begin{cases} E\left[\left|r_t\left(\dfrac{\partial e_\tau^{(i)}}{\partial \Theta^{(j)}} + e_\tau^{(i)} e_\tau^{(j)}\right)\right|\right] & \text{if } \tau = \tau' \\ E\left[\left|r_t\, e_\tau^{(i)} e_{\tau'}^{(j)}\right|\right] & \text{otherwise,} \end{cases} \tag{B.29}$$

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\mathcal{W}_t^{(\tau,i,j)}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} = E\left[\left|e_\tau^{(i)} \zeta_t^{(j)}\right|\right], \tag{B.30}$$

where (B.29) is derived from the fact that

$$\begin{aligned}\frac{\partial e_\tau^{(i)}}{\partial \Theta^{(j)}} &= \frac{\partial}{\partial \Theta^{(j)}} \frac{\partial \ln P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(i)}} \\ &= \frac{1}{P_{\tau,a_{\tau-1}}^{a_\tau}} \frac{\partial^2 P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(j)} \partial \Theta^{(i)}} - \frac{\partial \ln P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(i)}} \frac{\partial \ln P_{\tau,a_{\tau-1}}^{a_\tau}}{\partial \Theta^{(j)}}. \end{aligned} \tag{B.31}$$

Hence, we have $\int_{(\mathbb{A}^{\mathrm{p}})^t} \left|\frac{\partial \mathcal{P}_t^{(\tau,i)}}{\partial \Theta^{(j)}}(a_{1:t},\Theta)\right| da_{1:t}^{\mathrm{p}} < \infty$ because (C11) results in $E\left[\|r_t \nabla_\Theta e_\tau\|_F\right] < \infty$, $E\left[\|r_t e_\tau\| \cdot \|e_{\tau'}\|\right] < \infty$, and $E\left[\|e_\tau\| \cdot \|\zeta_t\|\right] < \infty$. Note that for any two random variables

$(X, Y) \in \mathbb{R}^2$, we have

$$E\left[X^2\right] < \infty \text{ and } E\left[Y^2\right] < \infty \Rightarrow E\left[XY\right] < \infty \tag{B.32}$$

from the Cauchy-Schwarz inequality $|E\left[XY\right]|^2 \le E\left[X^2\right] E\left[Y^2\right]$. Thus, from (B.25), (B.23) holds.

Similarly, from (C11), there exists a $\delta_9 \in \mathbb{R}_+$ such that

$$\int_{(\mathbb{A}^{\mathrm{p}})^t} \|\mathcal{R}_t\left(a_{1:t}, \Theta'\right) - \mathcal{R}_t\left(a_{1:t}, \Theta\right)\| \, da^{\mathrm{p}}_{1:t} \le \delta_9 \|\Theta' - \Theta\|. \tag{B.33}$$

From (B.22), (B.23), and (B.33),

$$\lim_{\Theta' \to \Theta} \|\mathcal{H}_t(\Theta') - \mathcal{H}_t(\Theta)\| = 0, \tag{B.34}$$

which means that $\mathcal{H}_t(\Theta)$ is continuous. ■

**Lemma 6.** *For Algorithm 5, if (C9)–(C12) are satisfied, then (D6) holds.*

*Proof.* For the random variable $\Xi_K := \sum_{k'=k}^{K} \eta(k')\varphi(k')$, the stochastic process $\Xi_K$ $(K = k, k+1, k+2, \dots)$ is a martingale (because the equation $E\left[\Xi_K \,\middle|\, \Xi_{k:K-1}\right] = \Xi_{K-1}$ holds from Lemma 4), and then the martingale inequality of Doob holds [63]. Therefore,

$$\begin{aligned} P\left(\sup_{K \ge k} \left\| \sum_{k'=k}^{K} \eta(k')\varphi(k') \right\| \ge \epsilon \right) &\le \frac{1}{\epsilon^2} E\left[ \left\| \sum_{k'=k}^{\infty} \eta(k')\varphi(k') \right\|^2 \right] \\ &= \frac{1}{\epsilon^2} \sum_{k'=k}^{\infty} \eta(k')^2 E\left[\|\varphi(k')\|^2\right] \end{aligned} \tag{B.35}$$

where the equality follows by the fact that $E\left[\varphi(k_1)^\top \varphi(k_2)\right] = 0$ if $k_1 \ne k_2$. In addition,

$$\begin{aligned} E\left[\|\varphi(k')\|^2\right] &= E\left[\|V_T\|^2\right] - \left\|\nabla_\Theta \bar{J}(\Theta(k'))\right\|^2 \\ &\le E\left[\|V_T\|^2\right] \\ &\le E\left[ \left\{ \sum_{t=1}^{T} \left( |r_t - b| \sum_{\tau=1}^{t} \|e_\tau\| + \|\zeta_t\| \right) \right\}^2 \right] \\ &< \infty \end{aligned} \tag{B.36}$$

where the last inequality follows from (C11), which leads to $E\left[\|e_\tau\| \cdot \|e_{\tau'}\|\right] < \infty$,

$E\left[\|\zeta_t\| \cdot \|\zeta_{t'}\|\right] < \infty$, $E\left[\|r_t e_\tau\| \cdot \|r_{t'} e_{\tau'}\|\right] < \infty$, $E\left[\|e_\tau\| \cdot \|\zeta_t\|\right] < \infty$, $E\left[\|r_t e_\tau\| \cdot \|e_{\tau'}\|\right] < \infty$, and $E\left[\|r_t e_\tau\| \cdot \|\zeta_{t'}\|\right] < \infty$ from (B.32). From (B.35), (B.36), and (C12), (D6) holds. ■

# References

[1] S. M. Malagon-Soldara, M. Toledano-Ayala, G. Soto-Zarazua, R. V. Carrillo-Serrano, and E. A. Rivas-Araiza, "Mobile robot localization: A review of probabilistic map-based techniques," *International Journal of Robotics and Automation*, vol. 4, no. 1, pp. 73–81, 2015.

[2] G. Nirmala, S. Geetha, and S. Selvakumar, "Mobile robot localization and navigation in artificial intelligence: Survey," *Computational Methods in Social Sciences*, vol. 4, no. 2, pp. 12–22, 2016.

[3] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge (MA): MIT Press, 2005.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999, pp. 1322–1328.

[6] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1–2, pp. 99–141, 2001.

[7] J. S. Gutmann and D. Fox, "An experimental comparison of localization methods continued," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 454–459.

[8] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.

[9] 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, 2012.

[10] G. Peng, W. Zheng, Z. Lu, J. Liao, L. Hu, G. Zhang, and D. He, "An improved AMCL algorithm based on laser scanning match in a complex and unstructured environment," *Complexity*, vol. Article ID 2327637, 2018.

[11] H. Jo and E. Kim, "New Monte Carlo localization using deep initialization: A three-dimensional LiDAR and a camera fusion approach," *IEEE Access*, vol. 8, pp. 74 485–74 496, 2020.

[12] N. Akai, T. Hirayama, and H. Murase, "3D Monte Carlo localization with efficient distance field representation for automated driving in dynamic environments," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1859–1866.

[13] N. Y. Ko, T. G. Kim, and S. W. Noh, "Monte Carlo localization of underwater robot using internal and external information," in *Proceedings of the 2011 IEEE Asia-Pacific Services Computing Conference*, 2011, pp. 410–415.

[14] F. J. Perez-Grau, F. Caballero, L. Merino, and A. Viguria, "Multi-modal mapping and localization of unmanned aerial robots based on ultra-wideband and RGB-D sensing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3495–3502.

[15] A. Baggio and K. Langendoen, "Monte Carlo localization for mobile wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 5, pp. 718–733, 2008.

[16] E. B. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 4387–4393.

[17] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3158–3164.

[18] S. W. Yang and C. C. Wang, "Feasibility grids for localization and mapping in crowded urban scenes," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 2322–2326.

[19] M. Rapp, M. Hahn, M. Thom, J. Dickmann, and K. Dietmayer, "Semi-Markov process based localization using radar in dynamic environments," in *Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 423–429.

[20] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte Carlo localization in outdoor terrains using multilevel surface maps," *Journal of Field Robotics*, vol. 25, no. 6–7, pp. 346–359, 2008.

[21] A. Ohsato, Y. Sasaki, and H. Mizoguchi, "Real-time 6DoF localization for a mobile robot using pre-computed 3D laser likelihood field," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2015, pp. 2359–2364.

[22] T. N. Hossein, L. H. S. Mita, and Q. H. Do, "Multi-sensor data fusion for autonomous vehicle navigation and localization through precise map," *International Journal of Automotive Engineering*, vol. 3, no. 1, pp. 19–25, 2012.

[23] M. Hentschel, O. Wulf, and B. Wagner, "A GPS and laser-based localization for urban and non-urban outdoor environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 149–154.

[24] Y. J. Lee, B. D. Yim, and J. B. Song, "Mobile robot localization based on effective combination of vision and range sensors," *International Journal of Control, Automation and Systems*, vol. 7, no. 1, pp. 97–104, 2009.

[25] S. Xu, W. Chou, and H. Dong, "A robust indoor localization system integrating visual localization aided by CNN-based image retrieval with Monte Carlo localization," *Sensors*, vol. 19, no. 2, p. 249, 2019.

[26] K. Suyama, Y. Funabora, S. Doki, and K. Doki, "Robust localization for mobile robot by K-L divergence-based sensor data fusion," in *Proceedings of the 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 002 638–002 643.

[27] K. Sakaeta, K. Nonaka, and K. Sekiguchi, "Experimental verification of a vehicle localization based on moving horizon estimation integrating LRS and odometry," *Journal of Physics: Conference Series*, vol. 744, p. 012048, 2016.

[28] N. Roy and S. Thrun, “Coastal navigation with mobile robots,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 1043–1049.

[29] M. Koizumi, K. Nonaka, and K. Sekiguchi, “Avoidance of singular localization environment using model predictive control for mobile robots,” in *Proceedings of the 11th Asian Control Conference*, 2017, pp. 2866–2871.

[30] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[31] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[32] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[33] S. Huang and G. Dissanayake, “Convergence and consistency analysis for extended Kalman filter based SLAM,” *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[34] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2003, pp. 593–598.

[35] M. Lin, C. Yang, D. Li, and G. Zhou, “Intelligent filter-based SLAM for mobile robots with improved localization performance,” *IEEE Access*, vol. 7, pp. 113 284–113 297, 2019.

[36] S. Thrun and M. Montemerlo, “The graph SLAM algorithm with applications to large-scale mapping of urban structures,” *International Journal of Robotics Research*, vol. 25, no. 5–6, pp. 403–429, 2006.

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM: A versatile and accurate monocular SLAM system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[38] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[39] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, “Robust adversarial reinforcement learning,” in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2817–2826.

[40] H. Shioya, Y. Iwasawa, and Y. Matsuo, “Extending robust adversarial reinforcement learning considering adaptation and diversity,” in *Proceedings of the International Conference on Learning Representations*, 2018.

[41] K. Zhang, B. Hu, and T. Basar, “On the stability and convergence of robust adversarial reinforcement learning: A case study on linear quadratic systems,” in *Proceedings of the 33th International Conference on Neural Information Processing Systems*, 2020.

[42] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer, “Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1665–1671.

[43] X. Pan, D. Seita, Y. Gao, and J. Canny, “Risk averse robust adversarial reinforcement learning,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 8522–8528.

[44] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *Proceedings of the 11th International Conference on Machine Learning*, 1994, pp. 157–163.

[45] J. Morimoto and K. Doya, “Robust reinforcement learning,” *Neural Computation*, vol. 17, no. 2, pp. 335–359, 2005.

[46] N. J. Gordon, D. J. Salmond, and A. F. M. Smit, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” *IEE Proceedings-F*, vol. 140, no. 2, pp. 107–113, 1993.

[47] G. Kitagawa, “Monte Carlo filter and smoother for non-Gaussian nonlinear state space models,” *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[48] C. Choi and H. I. Christensen, “RGB-D object tracking: A particle filter approach on GPU,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1084–1091.

[49] J. Wei, G. Dong, and Z. Chen, “Remaining useful life prediction and state of health diagnosis for lithium-ion batteries using particle filter and support vector regression,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5634–5643, 2018.

[50] G. Kitagawa, “A self-organizing state-space model,” *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1203–1215, 1998.

[51] C. Andrieu, A. Doucet, and R. Holenstein, “Particle Markov chain Monte Carlo methods,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[52] G. Poyiadjis, A. Doucet, and S. S. Singh, “Particle approximations of the score and observed information matrix in state space models with application to parameter estimation,” *Biometrika*, vol. 98, no. 1, pp. 65–80, 2011.

[53] A. Burchardt, T. Laue, and T. Röfer, “Optimizing particle filter parameters for self-localization,” in *RoboCup 2010: Robot Soccer World Cup XIV*, 2011, pp. 145–156.

[54] R. Jonschkowski, D. Rastogi, and O. Brock, “Differentiable particle filters: End-to-end learning with algorithmic priors,” in *Proceedings of Robotics: Science and Systems*, 2018.

[55] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge (MA): MIT Press, 1998.

[56] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, pp. 279–292, 1992.

[57] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015.

[58] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, pp. 229–256, 1992.

[59] H. Kimura, M. Yamamura, and S. Kobayashi, “Reinforcement learning by stochastic hill climbing on discounted reward,” in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 295–303.

[60] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 1057–1063.

[61] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2219–2225.

[62] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[63] H. J. Kushner and D. S. Clark, *Stochastic Approximaiton Methods for Constrained and Unconstrained Systems*. New York (NY): Springer, 1978.

[64] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*. New York (NY): Springer, 2003.

[65] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[66] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York (NY): Springer, 2005.

[67] L. Baird and A. Moore, “Gradient descent for general reinforcement learning,” in *Proceedings of the 11th International Conference on Neural Information Processing Systems*, 1998, pp. 968–974.

[68] J. Baxter and P. L. Bartlett, “Infinite-horizon policy-gradient estimation,” *Journal of Artificial Intelligent Research*, vol. 15, pp. 319–350, 2001.

[69] P. Fearnhead, D. Wyncoll, and J. Tawn, “A sequential smoothing algorithm with linear computational cost,” *Biometrika*, vol. 97, no. 2, pp. 447–464, 2010.

[70] M. K. Pitt and N. Shephard, “Filtering via simulation: Auxiliary particle filters,” *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[71] K. Murphy, “Bayesian map learning in dynamic environments,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 1015–1021.

[72] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, “Rao-Blackwellised particle filtering for dynamic bayesian networks,” in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 176–183.

[73] A. Milstein, J. N. Sánchez, and E. T. Williamson, “Robust global localization using clustered particle filtering,” in *Proceedings of the 18th National Conference on Artificial Intelligence*, 2002, pp. 581–586.

[74] G. Cen, N. Matsuhira, J. Hirokawa, H. Ogawa, and I. Hagiwara, “An improved particle filter for service robot self-localization,” *SICE Journal of Control, Measurement, and System Integration*, vol. 2, no. 1, pp. 056–063, 2009.

[75] J. Morimoto and K. Doya, “Reinforcement learning state estimator,” *Neural Computation*, vol. 19, no. 3, pp. 730–756, 2007.

[76] M. Brokate and G. Kersting, *Measure and Integral.* Basel: Springer, 2015.

# Publications

This thesis is based on the following publications:

1. R. Yoshimura, I. Maruta, K. Fujimoto, K. Sato, and Y. Kobayashi, "Highlighted map for mobile robot localization and its generation based on reinforcement learning," *IEEE Access*, vol. 8, pp. 201527–201544, 2020.

2. R. Yoshimura, I. Maruta, K. Fujimoto, K. Sato, and Y. Kobayashi, "Particle filter design based on reinforcement learning and its application to mobile robot localization," *IEICE Transactions on Information and Systems*, vol. E105-D, no. 5, 2022 (in press).

3. R. Yoshimura, I. Maruta, K. Fujimoto, K. Sato, and Y. Kobayashi, "Adversarial reinforcement learning based robustification of highlighted map for mobile robot localization," in *Proceedings of the SICE Annual Conference 2021*, 2021, pp. 596–602.

4. R. Yoshimura, I. Maruta, K. Fujimoto, K. Sato, and Y. Kobayashi, "Optimization of maps for autonomous vehicles based on reinforcement learning," in *Proceedings of the 6th Multi-symposium on Control Systems*, 2019, 2G1-3 (in Japanese).

5. R. Yoshimura, I. Maruta, K. Fujimoto, K. Sato, and Y. Kobayashi, "Particle filter design based on reinforcement learning and its application to localization," in *Proceedings of the 7th Multi-symposium on Control Systems*, 2020, 3A2-2 (in Japanese).

# Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors, Professor Kenji Fujimoto and Associate Professor Ichiro Maruta. Professor Fujimoto graciously accepted me as a doctoral student in his laboratory and gave me excellent teachings on nonlinear control, optimal control, and more. I was very stimulated by discussions on state-of-the-art control engineering topics in his laboratory meetings. Associate Professor Maruta kindly guided me at all times and supplied me with a variety of insightful ideas for my research. In particular, I could not have proved the theorems in this thesis without his help. Their guidance was the greatest encouragement for me to pursue my research and helped me grow as a researcher.

I greatly appreciate Professor Kei Senda and Professor Yoshito Ohta for helpful discussions and important suggestions. Their advice allowed me to consider my research in depth and, as a result, to broaden my research horizons.

I would like to express my sincere gratitude to my mentor, Professor Toshiharu Sugie of Osaka University, who has always given me kind guidance and encouragement since my undergraduate days.

I am indebted to the staff in the System & Control Laboratory, Ms. Chihiro Inamoto and Ms. Rei Maki, for their kind administrative support. I also gratefully acknowledge interesting discussions with all the students of the laboratory.

This research could not have been carried out without the support and cooperation of the staff at my workplace, the Tokyo Metropolitan Industrial Technology Research Institute. I would like to offer my special appreciation to Dr. Kensuke Kawarada for his warm-hearted guidance and help in advancing to the doctoral program. I am deeply grateful to Dr. Atsushi Mitsuo, Mr. Masanao Kuramochi, Mr. Ken Sato, and Mr. Yusuke Kobayashi for their fruitful help and advice on my research. I would like to express my heartfelt acknowledgment to Dr. Minoru Tanaka, Dr. Yutaka Arakawa, Ms. Misako Yamaguchi, and the members of the Joto Branch for their continued great support and cooperation.

Finally, I would like to thank to my wife, Emi, my son, Yuki, and my parents for their constant support and encouragement.