Master's Thesis

Title

Mobile Robot Control through Real-Time Identification of Physical/Network Environments by Bayesian Inference

Supervisor

Professor Masayuki Murata

Author

Takumi Matsuda

February 5th, 2019

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Mobile Robot Control through Real-Time Identification
of Physical/Network Environments by Bayesian Inference

Takumi Matsuda

Abstract

Robots such as drones, disaster rescue robots, and automatic guided vehicles, have been developed and become widely used. Many of them are controlled via a network. Such systems are constructed of a controller, robots, and sensors that are connected via a network. The controller predicts the network delay and the state of the robots at the time of receiving the command. Then the commands are calculated for the predicted state of the robot.

The predicted state of the robot includes errors, and commands are required to be calculated, considering the errors. Such a prediction error depends on the environment. Therefore, the controller needs to identify the current environment. Using machine learning is one approach to considering the error depending on the environment. In this approach, the commands and states of the robot after performing the commands are monitored at the environment where robots are controlled. Then, the relation between them is learned. However, environments can change. The controller is required to identify the current environment immediately after the change of the environments. However, methods based on machine learning require a sufficient amount of monitored data and takes time to identify the current environment.

In this thesis, we establish a framework that enables controllers to identify the current environment from a small number of observation results. Our framework is inspired by the organisms that evolve to become able to identify the current environment by using Bayesian inference with the prior distribution obtained through evolution.

In this thesis, we propose an environment identification method using Bayesian inference for remote control. In this method, the following operation is performed by a controller. Based on observation obtained from the robot and control commands sent to the

robot, the controller calculates an error that occurred when running the commands.Based on the obtained error, the controller updates the error model for the current environment by Bayesian inference and calculates the control commands based on the error model.

In our framework, the prior distribution of the error model is obtained by the evolution under various environments; in each generation, we evaluate the controller with each prior distribution by scoring the results of tasks done by the controller and evolve the parameters of prior distribution based on the evaluation results.

In this thesis, we implement a controller for a two-wheel mobile robot based on our framework. Through experiments, we demonstrate that the controller based on our framework identified the current environment and control the robot properly. As a result, our method reduces the deviation from the intended path by about 75% and the control time by about 35% compared to the method that identifies the current environment only from the observed information.

Keywords

Bayesian Inference
Network Control System
Mobile Robot
Genetic Algorithm
Environmental Identification

Contents

List of Figures

List of Tables

1 Introduction

Robots such as drones, disaster rescue robots, and automatic guided vehicles, have been developed and become widely used. Many of them are controlled via a network [2–7]. Such systems are constructed of a controller, robots, and sensors that are connected via a network. The robots have only limited computational resources. On the other hand, the controller has powerful computation resources. The controller collects and analyzes the sensor data, and calculates the command for robots. The commands are received by the robot after network delay. Thus, the controller predicts the network delay and the state of the robots at the time of receiving the command. Then the commands are calculated for the predicted state of the robot.

The prediction of the state of the robots is important to control the robots via a network. If the actual state of the robots is different from the predicted one, the commands sent by the controller are not suitable to the actual state. One approach to predicting the state of the robots is to model robots. Based on the robots, we can predict the future state of robots. However, the actual state of the robots may be different from the model. For example, the fluctuation of the network delay causes that some commands are performed at a different time than intended. The slip of wheels also causes the prediction errors of a wheel-based robot [10].

Such a prediction error depends on the environment. For example, the amount of slip depends on the surface of the floor, the weight of the baggage if the robots carry the baggage, and so on. The network delay also depends on the environments; when there are many devices using the same channel of the wireless network, the delay becomes large.

Using machine learning is one approach to considering the error depending on the environment [11–14]. In this approach, the commands and states of the robot after performing the commands are monitored at the environment where robots are controlled. Then, the relation between them is learned. By using the learned relation, the controller can accurately predict the state of future robots, and calculate the suitable commands.

However, environments can change. For example, the robots move to the area with the different type of floor surface or the area that is far from the access points of the wireless network. The change of the task such as picking up baggage may also cause the change of

environment. The controller is required to identify the current environment immediately after the change of the environments. However, methods based on machine learning require a sufficient amount of monitored data and takes time to identify the current environment.

In this thesis, we establish a framework that enables controllers to identify the current environment from a small number of observation results. Our framework is inspired by the organisms that evolve to become able to identify the current environment. Camilo et al. demonstrated that evolution makes organisms able to estimate the current environment properly [1]. They simulated the evolution of organisms under the condition that each individual predicts the occurrence probability p_A of the event A by bayesian inference. The results show that the individuals obtain the proper prior distribution to predicts the occurrence probability of p_A as they evolve.

In this thesis, we propose an environment identification method using Bayesian inference for remote control. In this method, the following operation is performed by a controller. (1) Based on observation obtained from the robot and control commands sent to the robot, the controller calculates an error that occurred when running the commands. (2) Based on the obtained error, the controller updates the error model for the current environment by Bayesian inference and calculates the control commands based on the error model.

In our framework, the prior distribution of the error model is obtained by the evolution under various environments; in each generation, we evaluate the controller with each prior distribution by scoring the results of tasks done by the controller and evolve the parameters of prior distribution based on the evaluation results. By evolving the prior distribution under the various environments, we obtain the prior distribution that can properly identify various environments.

In this thesis, we implement a controller for a two-wheel mobile robot based on our framework. Through experiments, we demonstrate that the controller based on our framework identified the current environment and control the robot properly.

The rest of this thesis is organized as follows. Chapter 2 describes related work. Chapter 3 proposes a framework to identify the current environment and control the remote robot based on the identified environment, and Chapter 4 explains the implementation of the framework for a two-wheel mobile robot. Chapter 5 evaluates our framework based

on the experiments. Finally, Chapter 6 concludes this thesis.

2 Related Work

2.1 Networked Control System

A network control system is a system in which a controller, sensors, and actuators are connected via a network. In this system, the controller receives sensor data and transmits commands via a network shared with other systems. As the many devices are being connected to and controlled via a network, the network control system becomes important. Thus, many methods to control devices in a network control system have been proposed.

Cuenca et al. developed an autonomous vehicle system [7]. In this system, the remote controller makes decision based on the monitored data and generates reference path. The autonomous vehicle receives the reference path and controls its wheels so as to follow the reference path. This system handles the delays by predicting future state using extended Kalman filter, and handles the packet-disorder by applying dual-rate control.

Yasuda et al. designed a method to control a device via network with high responsibility by minimizing the sum of the control cycle and the network delay time [6].

Dohyun et al. proposed a method to operate unmanned aerial vehicles in a network environment with a time-varying network delay. They applied model predictive control and applied a Gaussian process to learn the model of unmanned aerial vehicles. By applying them, the method successfully compensates network delay.

Zhang et al. proposed a predictive sliding mode controller for networked control systems with time delay and packet dropout [8]. They modeled the time delay and packet dropout by using a Markov chain. Then, they proposed a new controller with a delay compensator.

Most of the existing research on the network control system assume that the dynamics of the devices and networks are static. Thus, they model or learn the dynamics, and calculate the commands based on the modeled or learned dynamics. However, environments can change. For example, the robots move to the area with the different type of floor surface or the area that are far from the access points of the wireless network. The change of the task such as picking up baggage may also cause the change of environment. Therefore, we propose a method to identify the current environment immediately after the change of the environments.

2.2 Evolution of Bayesian Learners

Organisms make decisions based on estimates of the state of their environment. The mechanisms of how organisms estimate the state of their environment can be used to estimation of the environment in the remote control of robots.

Camilo et al. demonstrates that evolution makes organisms able to estimate the current environment properly [1]. Their demonstration is based on the simulation. In this simulaation, each individual predicts the occurrence probability p_A of the event A. The prediction is performed by bayesian inference, and a beta distribution with parameters $[\alpha, \beta]$ is used as a prior distribution. Each individual performs a Bernoulli trial of event A for n times, updates the probability distribution as follows based on the number of times event A was obtained k, and then uses updated distribution to predict the occurrence probability of A.

$$\phi_x(p_A) = \frac{k + \alpha}{n + \alpha + \beta} \tag{1}$$

In this simulation, evolution is performed by mutation and selection based on the fitness function. The fitness function $f_s(x)$ of the individual x is defined below.

$$f_s(x) = \frac{1}{1 + \sum_{p_i \in S} [p_i - \phi_x(p_i)]^2} \tag{2}$$

This value increases when the prediction of the individual and the occurrence of the actual event A match. That is, by performing selection so that individuals with high values survive, evolution can be performed so that appropriate individuals are generated.

The simulation results show that the individual obtained as a result of the above-mentioned evolution can predict A with higher accuracy than the individual that predicts the event based only on the events observed by itself.

In this thesis, we apply the above evolution to a mechanism to make the controller able to identify the current environment.

3 Framework for Robot Control through Identifying Physical/Network Environments

3.1 Remote Robot Control Considering Identified Environments

In this thesis, we discuss the systems where robots are controlled via a network. In this system, we assume that the robots have only limited computational resources while the controller has powerful computation resources. Thus, the tasks requiring powerfull computation resources are done by the controller.

Figure 1 shows the overview of remote control of a robot. As shown in this figure, the controller periodically receive the sensor data A_{t-d_1} at the time t from sensors, where d_1 is the network delay between the sensors and the controller. Then, the controller estimates the state of the robot X_{t-d_1} at the time $t - d_1$. Due to the network delay, the command calculated at time t is received by the robot at time $t + d_2$. So, the controller predicts the future state of the robot X_{t+d_2}. Then, the controller calculates the command B_{t+d_2} based on X_{t+d_2} and sends it to the robot. In the above steps, the controller cannot accurately estimate and predict the state of robots, because the controlling robots includes errors due to time-varying network delay, slip of the wheels and so on. In this framework, we handle such uncertainty by modeling the state of the robots as the probability function. That is, we model $P(A_t|X_t)$ and $P(X_{t+1}|X_t, B_t)$, estimate and predict state of the robot X_t as a probability function, and calculate B_t by usinf the probability function.

$P(A_t|X_t)$ can be modeled by the relation between sensors and the state of the robots, and considered to be static. However, $P(X_{t+1}|X_t, B_t)$ depends on the environment. For example, the amount of slip depends on the surface of the floor, the weight of the baggage if the robots carry the baggage, and so on. The network delay also depends on the environments; when there are many devices using the same channel of the wireless network, the delay becomes large. Therefore, we should identify $P(X_{t+1}|X_t, B_t)$ for the current environment.

In this subsection, we explain how to estimate and predict the state of the robot and calculate the commands when $P(X_{t+1}|X_t, B_t)$ is given.

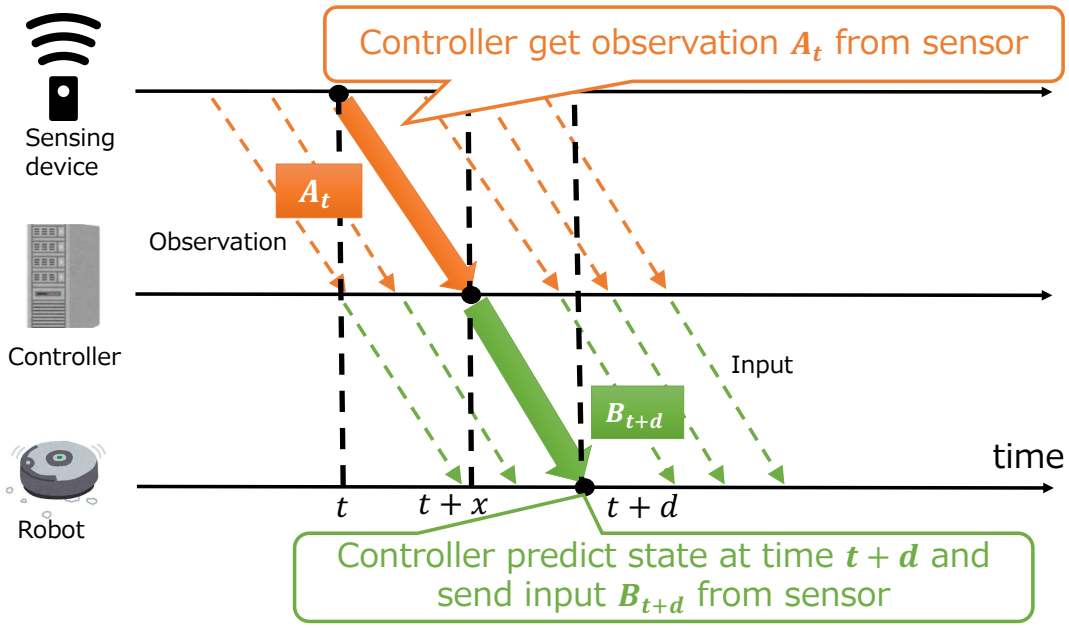

Figure 1: Overveiw of Remote Control

3.1.1 Estimation of the Current Robot State

The controller periodically receive the sensor data A_{t-d_1}. Each time the controllre recieves A_{t-d_1},it estimates the state of the robot at time t. To estimate the state, we use the baysian inference.

$$P(X_t) = \alpha P(A_t|X_t)\hat{P}(X_t) \tag{3}$$

where $\hat{P}(X_t)$ is the probability distribution of the state X_t predicted by using the information obtained by the time $t-1$. In this frame work, we assume that $P(A_t|X_t)$ is given. α is a constant that is introduced to make $\sum P(X_t) = 1$.

3.1.2 Prediction of the Future Robot State

The controller predicts state of the robots at the time $t+d_1$ from $P(X_t)$. By using $P(X_{t+1}|X_t, B_t)$, the predicted probability distribution of the state of the robot at the

time $t+1$, $\hat{P}(X_{t+1})$ is obtained by

$$\hat{P}(X_{t+1}) = \int_{X_t} P(X_{t+1}|X_t, B_t)P(X_t)$$

By repeating the above prediction, the state of the robot X_{t+d} at time $t+d$ is obtained.

3.1.3 Calculation of commands

The controller calculates the commands B_{t+d} so as to make the state of X_{t+d+1} the desired state $X_{t+d+1}^{\text{desired}}$ under the condition that the variance of the probability distribution of X_{t+d+1} is less the threshold V^{th}. That is, B_{t+d} is obtained by solving the following optimization problem.

$$\text{minimize}|E(\hat{P}(X_{t+d+1}|B_{t+d}) - X_{t+d+1}^{\text{desired}}|$$

$$\text{s.t.}Var(P(X_{t+d+1}|B_{t+d})) \leq V^{th}$$

where $E()$ and $Var()$ are the expected value and the variance of the probability distribution, and $\hat{P}(X_{t+d+1}|B(t+d)$ is

$$\hat{P}(X_{t+d+1}|B(t+d)) = \int_{X_{t+d}} P(X_{t+d+1}|X_{t+d}, B_{t+d})P(X_{t+d}).$$

3.2 Environment Identification using Bayesian Inference

The controller should identify $P(X_{t+1}|X_t, B_t)$ of the current environment. One approach to identifying $P(X_{t+1}|X_t, B_t)$ is to use machine learning techniques. But it requires a large amount of data. That is, this approach cannot identify the current environment immediately after the environment changes.

Therefore, we use the model of the robots. We assume that the dynamics of the robots are modeled in advance. The model is defined by

$$X_{t+1} = f(X_t, B_t, \epsilon)$$

where f is the function to obtain the next state of the robot. f has three parameters, the current state of robot X_t, the current command B_t, and the error term ϵ. We assume that the error term ϵ reflects the environment. Therefore, the current environment can be identified by identifying the probablistic distribution of ϵ.

The controller can calculate the current error term ϵ_t each time it receives the sensor data and estimate the current state of robots; if the current state of robots $P(X_t)$ is obtained, ϵ_{t-1} can be obtained by solving the following equation, assuming that the controller can accurately estimate the current robot state.

$$E(X_t) = f(E(X_{t-1}), B_{t-1}, \epsilon_{t-1})$$

By using the obtained $epsilon_t$, the distribution of $epsilon_t$ is estimated. But when we obtain only a small number of $epsilon_t$, the estimated distribution of $epsilon_t$ may be inaccurate. To avoid inaccurate esimation, we introduce, the prior distribution $P^{\text{prior}}(\epsilon)$ [15]. The distribution of $P(\epsilon)$ is obtained by

$$P(\epsilon) \propto P(\epsilon|\epsilon_{0:t})P^{\text{prior}}(\epsilon)$$

where $P(\epsilon|\epsilon_{0:t})$ is the distribution of the monitored *epsilon*.

3.3 Evolution of Environment Identification

In the above-mentioned step to identify the current environment, $P^{\text{prior}}(\epsilon)$ is important. In this thesis, we obtain $P^{\text{prior}}(\epsilon)$ by evolution, inspired by that evolution makes organisms able to estimate the current environment properly [1].

We model $P^{\text{prior}}(\epsilon)$ as a probability distribution such as a normal distribution, and we evolve its parameters.

In the evolution process, each individual corresponds to a set of parameters. First, we generate initial individuals by randomly setting their parameters. Then in each generation, new individuals are generated by crossover and/or mutation. Then, the individuals are evaluated. To evaluate individuals, we perform the tasks by using the controller with $P^{\text{prior}}(\epsilon)$ whose parameters are set to the values corresponding to the individual. This evaluation can be done by using actual robots or using simulation. Then, the performance of the controller is obtained. Finally, the individuals with high performance are selected to the next generation.

By repeating the above process by changing the environment used to evaluate the individual, we can obtain prior distribution $P^{\text{prior}}(\epsilon)$ that can identify various environment.

4 Implementation of Mobile Robot Control With Environment Identification

In this thesis, we implement the controller for two-wheel mobile robot based on our framework.

4.1 Mobile Robot Control Considering Identified Environments

4.1.1 Model of Mobile Robot

The state of the mobile robot is represented by its position and angle. That is the state of the robot X_t is defined by

$$X_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \tag{4}$$

$$\tag{5}$$

where x_t and y_t are the position of the robot, and θ_t is the angle of the robot.

To simply model the two-wheel mobile robot, we separate X_t into the position L_t and angle θ_t, and define L_t by

$$L_t = \begin{pmatrix} x_t \\ y_t \end{pmatrix} \tag{6}$$

The control command B_t includes the speed of wheels. That is,

$$B_t = \begin{pmatrix} w_r \\ w_l \end{pmatrix} \tag{7}$$

where w_r is the speed of right wheel, and w_l is the speed of the left wheel.

By using the above variables, the position and angle of the robot after running the command B_{t-1} is obtained by

$$L_t = L_{t-1} + F(\theta_{t-1})B_{t-1} + \epsilon_{t-1}^{XY} \tag{8}$$

$$\theta_t = \theta_{t-1} + RB_{t-1} + \epsilon_t^{\theta} \tag{9}$$

$$F(\theta_{t-1}) = \begin{pmatrix} \frac{r}{2}\cos\theta_{t-1} & \frac{r}{2}\cos\theta_{t-1} \\ \frac{r}{2}\sin\theta_{t-1} & \frac{r}{2}\sin\theta_{t-1} \end{pmatrix} \quad (10)$$

$$R = \begin{pmatrix} \frac{r}{W} & -\frac{r}{W} \end{pmatrix} \quad (11)$$

where W is the distance between two wheels, ϵ_{t-1}^{XY} and ϵ_t^{Θ} are the control errors of the robot position and angle, respectively.

4.1.2 Estimation of the current state of the mobile robot

The controller first estimates X_t based on the observed value A_t. In this thesis, we use the systems that can detect the current position and angle of the mobile robot from the camera images. For simplicity, we assume that the system accurately estimate the current position and angle. That is,

$$X_t = F(A_t) \quad (12)$$

$$\theta_t = G(A_t) \quad (13)$$

where $F(A_t)$ is a function for estimating X_t from A_t, and $G(A_t)$ is a function for estimating θ_t from observed values.

4.1.3 Prediction

The controller estimates the probability distribution $\hat{P}(X_{t+d})$ at the time $t+d$, based on X_t and $P(\epsilon)$ estimated by the controller. The movement of the mobile robot is nonlinear calculation, and it is difficult to analytically obtain the probability distribution of the state of the mobile robot at time $t+d$. In this thesis, we estimate $\hat{P}(X_{t+d})$ by using Monte Carlo simulation as shown in Figure2.

To estimate $\hat{P}(X_{t+1})$, we create a table whose cells corresponding to the state of robots. By counting the number of robots in each cell in the table, we estimate $\hat{P}(X_{t+1})$. The table for $\hat{P}(X_{t+1})$ is created by the following steps.

1. Generate E agents based on robot state X_t at time t.

2. For each agent generated in Step 1, ϵ are generated with probability $P(\epsilon)$. For each *epsilon*, the state of the robot at time $t+1$ is obtained by simulation.

3. Create the table by counting the egent whose state corresponds to each cell

By repeating the above steps, we can obtain the distribution of $\hat{P}(X_{t+1})$.

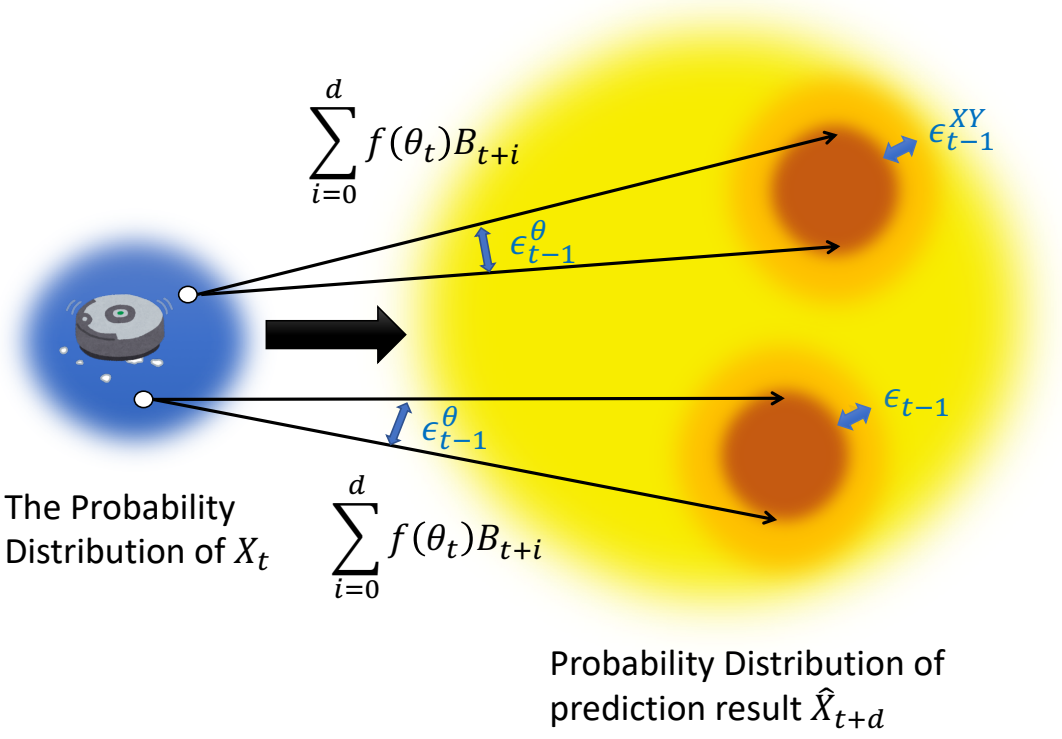

Figure 2: Prediction of mobile robot movement

4.1.4 Calculate the commands

In this thesis, we calculate the commands for the robot based on the method proposed by Lazoya et al. [16]. This method calculates the command to follow the target point.

The speed of wheels B'_{t+d} to follow the target point X^{target} is obtained by the following steps. First, the difference between target and current position of the robot $[e_x, e_y]$ is calculated by

$$\begin{pmatrix} e_x \\ e_y \end{pmatrix} = \begin{pmatrix} \cos\theta^{\text{ave}}_{t+d} & \sin\theta^{\text{ave}}_{t+d} \\ -\sin\theta^{\text{ave}}_{t+d} & \cos\theta^{\text{ave}}_{t+d} \end{pmatrix} \begin{pmatrix} x^{\text{target}} - x^{\text{ave}}_{t+d} \\ y^{\text{target}} - y^{\text{ave}}_{t+d} \end{pmatrix} \tag{14}$$

where $[x^{\text{ave}}_{t+d}, y^{\text{ave}}_{t+d}, \theta^{\text{ave}}_{t+d}]$ is the expected value of the current state of the robot, and $[x^{\text{target}}, y^{\text{target}}]$ is the target position.

Then the velocity and angular velocity (v, w) to follow the target is obtained by

$$v = K_{t+d} \tag{15}$$

$$w = 2D_{t+d}K_{t+d} \tag{16}$$

where

$$D_{t+d} = \text{sign}(e_x)\frac{e_y}{e_x^2} \tag{17}$$

$$K_{t+d} = \text{sign}(e_x)\frac{\alpha}{1+|A_t|} \tag{18}$$

where α is a positive constant, and the larger the value, the faster the robot moves.

Finally B'_{t+d} is obtained by

$$B'_{t+d} = \begin{pmatrix} \frac{v}{r} + \frac{Ww}{2r} \\ \frac{v}{r} - \frac{Ww}{2r} \end{pmatrix} \tag{19}$$

B'_{t+d} obtained by the above may be different from B'_{t+d-1}. In our controller, we avoid sudden acceleration by introducing the maximum acceleration ratio at each time slot $a_{\text{threshold}}$. By using $a_{\text{threshold}}$, the command B_{t+d} should satisfy the following condition.

$$|v(B_{t+d}) - v(B_{t+d-1})| \leq a_{\text{threshold}} \tag{20}$$

$$|X^{target} - \hat{X}_{t+d}| \geq \frac{1}{2a_{\text{threshold}}}|B_{t+d}|^2 \tag{21}$$

where $v(B_t)$ is a function representing the speed when the robot run the command B_t.

In addition, the robot moving first causes uncertain state of the robot. That is, we also should limit the speed in order to avoid uncertain state. In our framework, the uncertainty of the robot is evaluated by $Var(P(X_t)$ and $Var(P(X_{t+d+1}|B_{t+d}))$ should be less than a threshold.

We obtain B_{t+d} satisfying above condition by scaling B'_{t+d} by the following steps.

1. Obtain α so that $B_{t+d} = \alpha B'_{t+d}$ satisfies Eq.(20) and (21).

2. Obtain the distribution of $P(X_{t+d+1}|\alpha B'_{t+d})$by Monte Carlo simulation descrived above.

3. If $\mathrm{Var}(P(X_{t+d+1}|\alpha B'_{t+d}))$ is less than or equal to the threshold, set $B_{t+d+1} = \alpha B'_{t+d+1}$. Otherwise, go back to Step2 after $\alpha \leftarrow \alpha - \delta\alpha$

4.2 Estimation of Error Distribution using Bayesian Inference

4.2.1 Control Error Model

The distribution of the control error depends on the command. If the robot moves faster, the error ϵ^{XY} also becomes large. For simplicity, we model ϵ^{XY} so that it is proportional to the velocity. That is,

$$\epsilon_t^{\mathrm{XY}} = a_t^{\mathrm{XY}} v_t \tag{22}$$

$$a_t^{\mathrm{XY}} \sim N(\mu_{XY}, \sigma_{XY}) \tag{23}$$

where v_t is the velocity of the robot, and a_t is a random value. In this model, we simply model a so that it follows a normal distribution.

Similarly ϵ^{XY} is modeled by

$$\epsilon_t^{\theta} = a_t^{\theta} w_t \tag{24}$$

$$a_t^{\theta} \sim N(\mu_{\theta}, \sigma_{\theta}) \tag{25}$$

where w_t is the angle velocity of the robot.

4.2.2 Estimation of Control Error

Each time the controller recieves the sensor data and estimates the current state of the robot, the controller calculates error. The error can be calculated by using the estimated states of the robot. The error that occur in each time slot is small. Thus, the accurate calculation of the error in each time slot is difficult.

Therefore, we estimate the error by using the state of the robot N time slot ago. That is, we use the estimated state of robot $\hat{X}_t$ and $\hat{X}_{t-N}$, and the control inputs $B_{t-N}, B_{t-N+1}, \cdots, and B_t$. By using them, we have the following equations.

$$\dot{L}_t - \dot{L}_{t-N} = \sum_{i=0}^{N} f(\dot{\theta}_{t-i}) B_{t-i} + \sum_{i=0}^{N} a_{t-i}^{XY} V(B_{t-i}) \tag{26}$$

$$\dot{\Theta}_t - \dot{\theta}_{t-N} = \sum_{i=0}^{N} G_{\theta} W_{t-i} + \sum_{i=0}^{N} a_{t-i}^{theta} W(B_{t-i}) \tag{27}$$

where $V(B)$ and $W(B)$ are the velocity and angle velocity when the command B is run. ¿From Eqs. (26) and (27), we have

$$\sum_{i=0}^{N} a_{t-i}^{XY} V(B_{t-i}) = \hat{L}_t - \hat{L}_{t-N} - \sum_{i=0}^{N} f(\theta) B_{t-i} \tag{28}$$

$$\sum_{i=0}^{N} a_{t-i}^{\theta} W(B_{t-i}) = \hat{\theta}_t - \hat{\theta}_{t-N} - \sum_{i=0}^{N} G_{\Theta} B_{t-i} \tag{29}$$

If all of $V(B_{t-i})$ for i is from 0 to N nearly equal $\bar{V}$, the average of a^{XY} is obtained by

$$a^{XY} = \frac{1}{N\bar{V}} \left(\hat{L}_t - \hat{L}_{t-N} - \sum_{i=0}^{N} f(\theta) B_{t-i} \right). \tag{30}$$

Similarly a^{θ} is obtained by

$$a^{\theta} = \frac{1}{N\bar{W}} \hat{\theta}_t - \hat{\theta}_{t-N} - \sum_{i=0}^{N} G_{\Theta} B_{t-i}. \tag{31}$$

4.2.3 Update Distribution

In this thesis, we model $P(a^{XY})$ and $P(a^{\theta})$ by using the "hierarchical probability distribution. Hereafter, we explain the model of $P(a^{XY})$ only, but we use the same model for $P(a^{\theta})$.

$$P(a^{\mathrm{XY}}) \sim N(\mu, \sigma) \tag{32}$$

$$\mu|\sigma \sim N(\mu_n, \frac{\sigma^2}{k_n}) \tag{33}$$

$$\sigma^2 \sim IG(\frac{r_n}{2}, \frac{s_n}{2}) \tag{34}$$

Here, $k_n, s_n, r_n, \mu_n (k_n > 0, s_n > 0, r_n > 0)$ are parameters for determining the distribution, and n is the number of samples. $k_n, s_n, r_n, and \mu_n$ are derived as follows.

$$\mu_n = \frac{k_0}{k_0 + n} \mu_0 + \frac{k_0 + n}{n} \mu_{\epsilon} \tag{35}$$

$$k_n = k_0 + n \tag{36}$$

$$r_n = r_n + n \tag{37}$$

$$s_n = s_0 + (n-1)\sigma_{\epsilon}^2 + \frac{k_0 + n}{k_0 n} (\mu_0 - \mu_{\epsilon})^2 \tag{38}$$

where k_0, s_0, r_0, μ_0 are prior distribution parameters given at the start of control.

4.3 Evolution of Prior Distribution used Bayesian Inference

In our framework, the parameters of the prior distribution is obtained by evolution.To evolve the parameters, the processes of selection, crossover, and mutation are required to define. The rest of this subsection explains them.

4.3.1 Selection

We use the roulette selection. We define the fitness of each individual by the results of the task done by the controller using the prior distribution with the corresponding parameters. We denote the fitness of the individual i by f_i.Based on f_i, the individual i is selected with the probability p_i defined by

$$p_i = \frac{f_i}{\Sigma_{k=1}^{N} f_k}. \tag{39}$$

4.3.2 Crossover

After selecting two parent individuals by the selection, the new individuals with the parameters (μ_c, k_c, r_c, s_c) are generated. The new parameters are generated by

$$\mu_{\mathrm{c}} = \mathrm{random}(\min(\mu_1, \mu_2) - aI_\mu, \max(\mu_1, \mu_2) + aI_\mu) \tag{40}$$

$$k_{\mathrm{c}} = \mathrm{random}(\min(k_1, k_2) - aI_k, \max(k_1, k_2) + aI_k) \tag{41}$$

$$r_{\mathrm{c}} = \mathrm{random}(\min(r_1, r_2) - aI_r, \max(r_1, r_2) + aI_r) \tag{42}$$

$$s_{\mathrm{c}} = \mathrm{random}(\min(s_1, s_2) - aI_s, \max(s_1, s_2) + aI_s) \tag{43}$$

$$I_\mu = |\mu_1 - \mu_2| \tag{44}$$

$$I_k = |k_1 - k_2| \tag{45}$$

$$I_r = |r_1 - r_2| \tag{46}$$

$$I_s = |s_1 - s_2|. \tag{47}$$

where the parameters of the prior distribution of the two parent individuals are (μ_1, k_1, r_1, s_1) and (μ_2, k_2, r_2, s_2), respectively. a is a constant, and $\mathrm{random}(a, b)$ represents a uniform random number in the section $[a, b]$.

4.3.3 Mutation

The parameters of the prior distribution are set at random and newly generated. The new parameters (μ, k, r, s) are generated by

$$\mu = \text{random}(0, MAX_c) \tag{48}$$

$$k = \text{random}(0, MAX_c) \tag{49}$$

$$r = \text{random}(0, MAX_c) \tag{50}$$

$$s = \text{random}(0, MAX_c) \tag{51}$$

where MAX_c is a constant, and $\text{random}(a, b)$ is a random number in the section $[a, b]$.

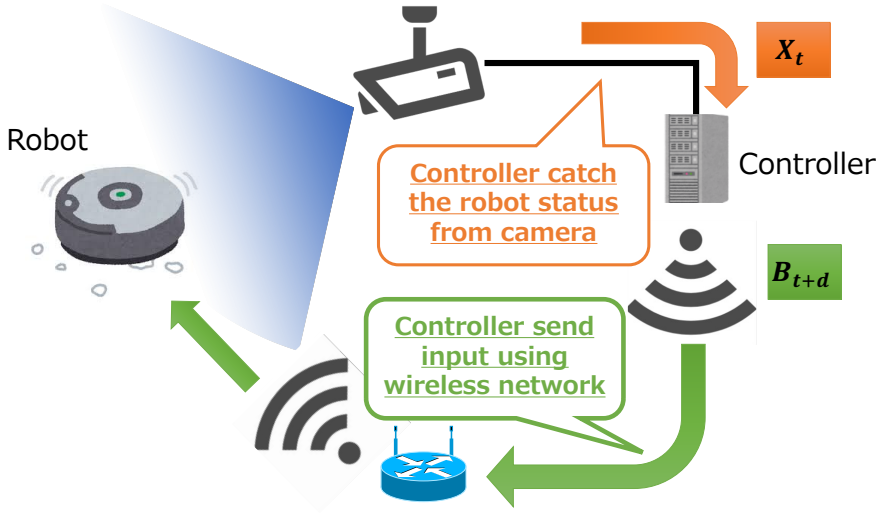

Figure 3: Expriment environment

5 Experiments

5.1 Senarios

Figure 3 shows the environment used in our experiment. In this environment, the camera is used as a sensor to identify the state of the robot. The controller receives the image and estimates the state of the robot every 30 ms. The camera is connected via the wired network. The delay between the camera and the controller is 130 ms and stable. On the other hand, the controller and the robot are connected via the wireless network, whose delay may be unstable.

5.1.1 Tasks

In this evaluation, we perform the task of round trip movement of straight line.In the simulation used to evolve the prrior distribution, we set the stating point to (0 , 0), turning point to (200, 0), and the goal to (-500, 0). In the actual environment, we set the stating point to (-35 , -865), turning point to (-145, 975), and the goal to (-35, -865).

Before starting the command to complete the above task, we perform the task to rotate the robot with a predefined speed for a predefined time. The roration and stop are

repeated 10 times. During this task, the controller obtains the information required to identify the current environemtn.

5.1.2 Physical/Network Environment

In this evaluation, we control the moving robot in the corridor of our graduate school. In this environment, we confirmed that the slip of wheels hardly occurs. The robot and the controller are connected via a wireless network. To know the network condition, we monitored the RTT between the controller and the robot. Figure 4 shows the monitored RTT. In most time slots, RTT is less than 100 ms.

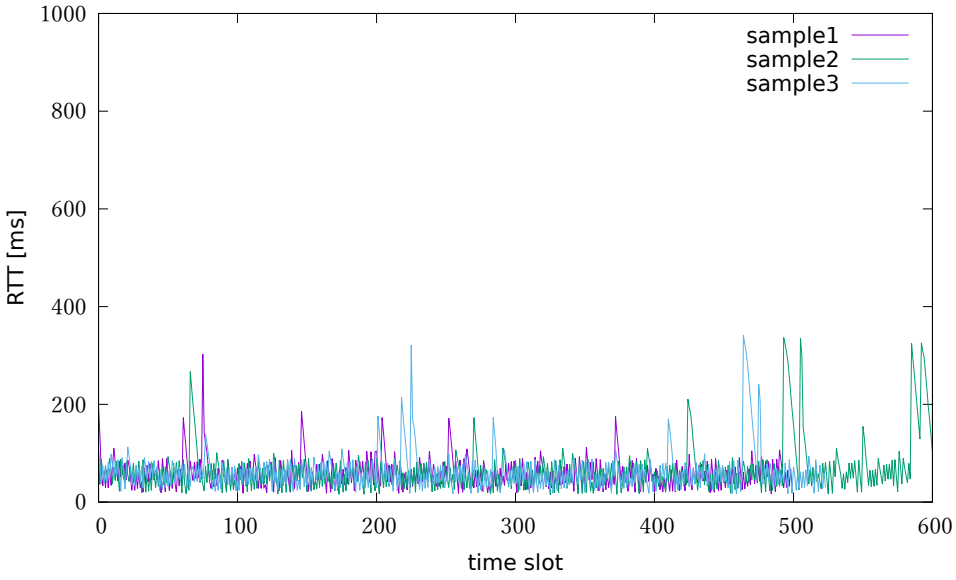

Figure 4: RTT (Normal Condition)

In this evaluation, we also control under the condition that the errors become large, to demonstrate our framework identifies the condition properly. As such a condition, we generate the case of the bad network condition. Assuming that many devices use the same channel and collision occurs, we artificially add the delay with a predefined probability. We set the added delay to 400 ms and the probability to 5 %. Figure 5 shows the example of the RTT generated by the above steps.

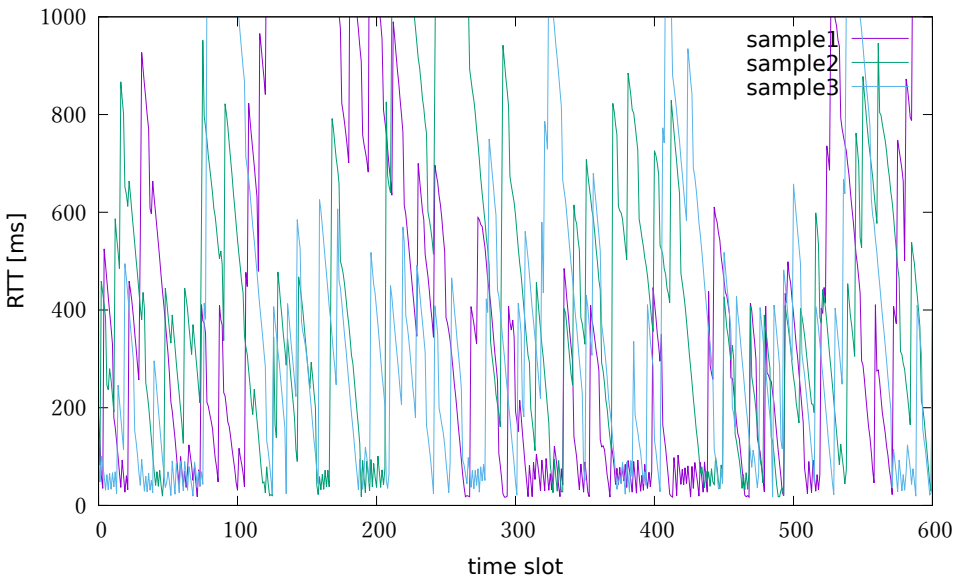

Figure 5: RTT (Bad Condition)

5.1.3 Compared Method

In this evaluation, we compare the following methods.

Control with Bayesian Inference (Bayesian) This method is a method based on our framework. The controller has the prior distribution obtained by evolution, and identify the current environment by Bayesian inference using the prior distribution. Then, the controller calculates the command based on the identified environment.

Control with Statistical Inference (Statistic) This method identifies the current environment by using only the monitored error. By calculating the distribution of the monitored error, this method identifies the current error model, and calculates the command based on the model. By comparing with this method, we demonstrate the advantage of the identification based on the Bayesian inference.

Control with Fixed Error Model This method uses the predefined error model. In this evaluation, we use two predefined error models; one is for the normal condition, and the other is for the bad condition. By comparing with this method, we demonstrate the

advantages of the identification of the current environment.

5.2 Parameter Settings

Table 1 shows the parameters of the controller used in this evaluation.

Table 1: parameters in controller

Parameter Name	value
Control cycle	30 [ms]
Robot position variance threshold	100.0 [mm * mm]
Robot angle variance threshold	0.10 [rad * rad]
Error acquisition cycle	100 [ms]
Maximum speed	130 [mm / s]
Minimum speed	23 [mm / s]
Maximum angular velocity	1.25 [rad / s]
MinimumAngular velocity	0.20 [rad / s]
Threshold used for discriminating between rotation and straight ahead	0.01 [rad / s]

We also set parameters for evolving the prior distribution. Table 2 shows the parameters.

In addition, we set the fitness of each individual during evolution. In our framework, the fitness is defined by the results of the task done by each individual. In this evaluation, we use the following indicators to evaluate the individuals.

- Maximum deviation from the defined path: $d_i^{\max}$
- Time to complete the task: t_i^{complete}

In this evaluation, we set two kinds of the fitness functions E_i^{normal} and E_i^{bad}. E_i^{normal} is used for the evaluaton under the normal network condition, while E_i^{bad} is used for the evaluation under the bad condition.

Table 2: parameters for evolution

Parameter Name	value
Population: N	100
Maximum generations: G	100
Crossover probability	49 %
Reproduce probability	49 %
Mutation probability	2 %
Expansion of parameter at intersection: a	1.2
Maximum deviation allowed: E_{max}	100.0 $[mm]$

By using the above indicators, we define the fitness E_i by

$$E_i^{\text{normal}} = \begin{cases} \frac{T}{t_i^{\text{complete}}}, & d_i^{\max} \leq D^{\max} \\ 0, & \text{otherwise} \end{cases} \tag{52}$$

where T and $D^{\max}$ are constant. E_i^{normal} becomes 0, if the devition from the path is larger than $D^{\max}$. Otherwise, E_i^{normal} becomes large as the time to complete the task is small.

On the other hand, E^{bad} is defined by

$$E_i^{\text{bad}} = \begin{cases} \frac{D}{d_i^{\max}}, & t_i^{\text{complete}} \leq T_{\max} \\ 0, & \text{otherwise} \end{cases} \tag{53}$$

where D and $T^{\max}$ are constants. E_i^{bad} becomes 0 if the time required to complete the task is larger than $T^{\max}$. Otherwise, E_i^{bad} becomes large as the deviation from the path is small. In the case of the bad network condition, the deviation from the path becomes large. Therefore, E_i^{bad} focuses on the deviation from the path.

5.3 Results

5.3.1 Evolution of Environment Identification

In this subsection, we show the results of evolution. Figures 6 and 7 show the average and minimum values of the control time and the deviation from the route in each generation.

Figure 6 shows that the deviation from the path becomes small as the generations go by; the deviation from the path fluctuates significantly at early stages, but the fluctuations become small as the generations go by. This is because the deviation from the path becomes small even in the case of the bad network condition.

Figure 7 shows that the time to complete the task becomes small as the generations go by. There are two types of individual that takes a long time to complete the task; One is the individual who unncessary limits the speed of the robot, and another one is the individual who sends the command to move robots with high speed. The later one takes a long time to complete the task, because it cannot accurately control the robot. As the generations go by, the above individuals are eliminated. Instead of them, the individuals that can accurately identify the current environment and control the robots properly considering the environemt are generated and survive. As a reult, the time required to complete the task becomes short.

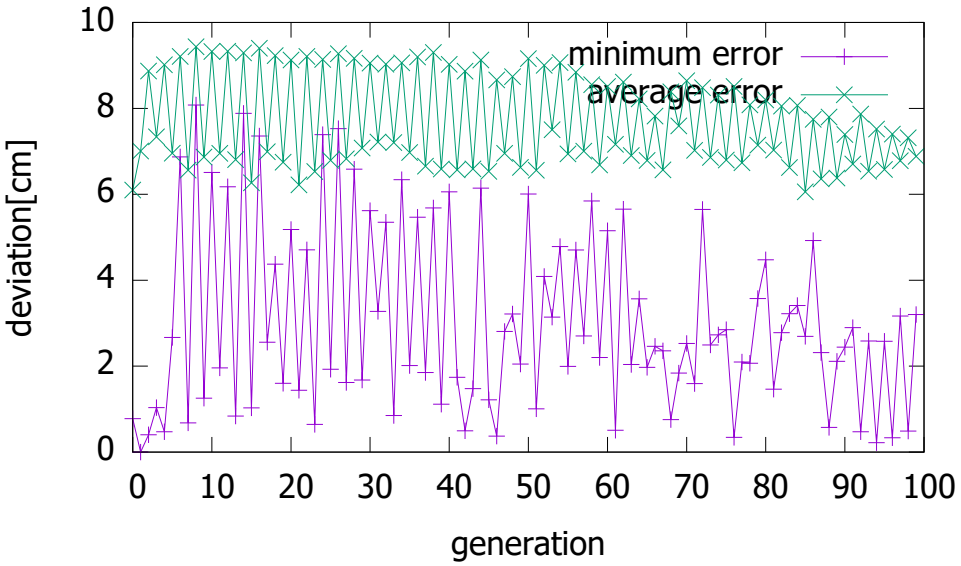

Figure 6: Max deviation from target trajectory in each generation

5.3.2 Comparison among method

Figures 9 and 8 show the cumulative complementary distribution of the maximum value of the deviation from the path. In our evaluation, we perform the task 15 times for each

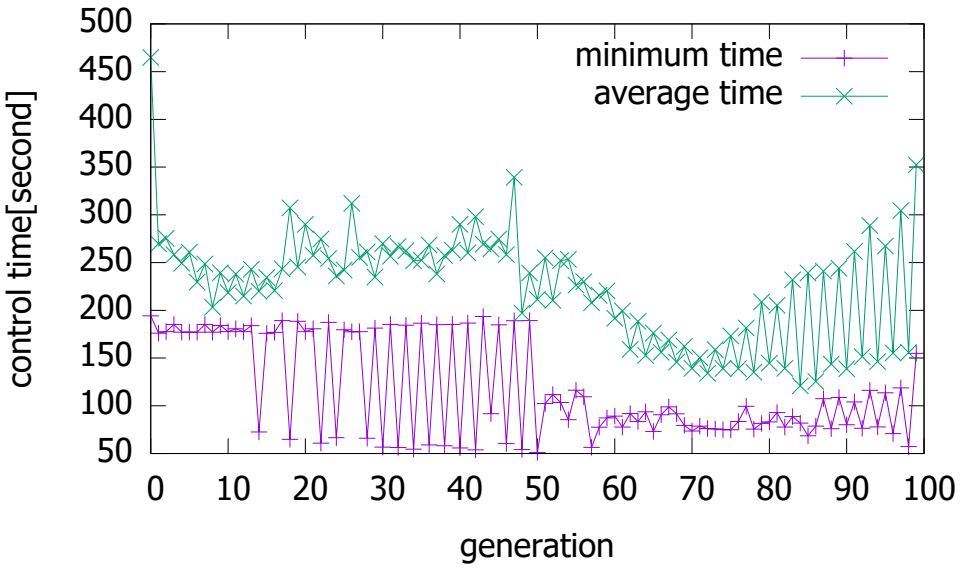

Figure 7: Task completion time in each generation

method. The results indicate that our method suppresses the maximum values of the deviation even in the case of the bad network condition. This is because our method identifies the current environment properly. As a result, our method limits the speed of robots in the case of the bad network condition to avoid large deviation. On the other hand, the deviation for the other method except for the method with the fixed model whose parameters are set to fit the bad network condition becomes large. This is because these methods cannot identify the bad network condition properly. Even the method with statistic inference may mistakenly identify the current environment as the environment with small errors. As a result, the methods control robots too fast, which causes a large deviation from the path.

Figures 11 and 10 shows the cumulative complementary distribution of time to complete the delay. The results show that the time to complete the task by our method is the smallest in both cases. This is because our method identifies the current environment properly. As a result, the controller controls the robot with high speed in the case of the normal network condition, while it limits the speed in the case of the bad network condition. On the other hand, other methods cannot control properly. The method using a fixed error model whose parameters are set to fit the normal network condition controls

the robots too fast in the case of the bad network condition. As a result, it takes a long time to set the angle of the robots to the target point. The method using a fixed error model whose parameters are set to fit the bad network condition unnecessarily limits the speed of robots and takes a long time to complete the task.

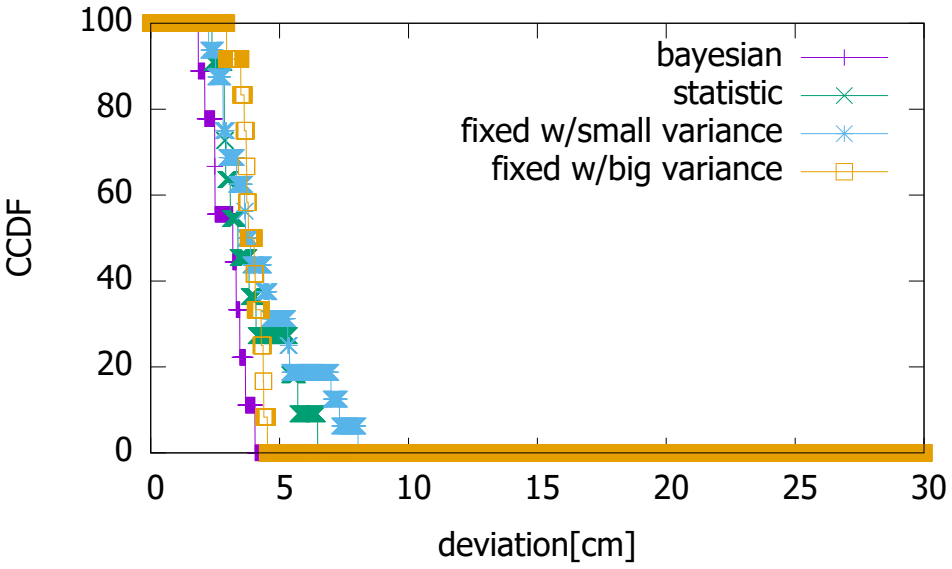

Figure 8: Deviation from target route without delay

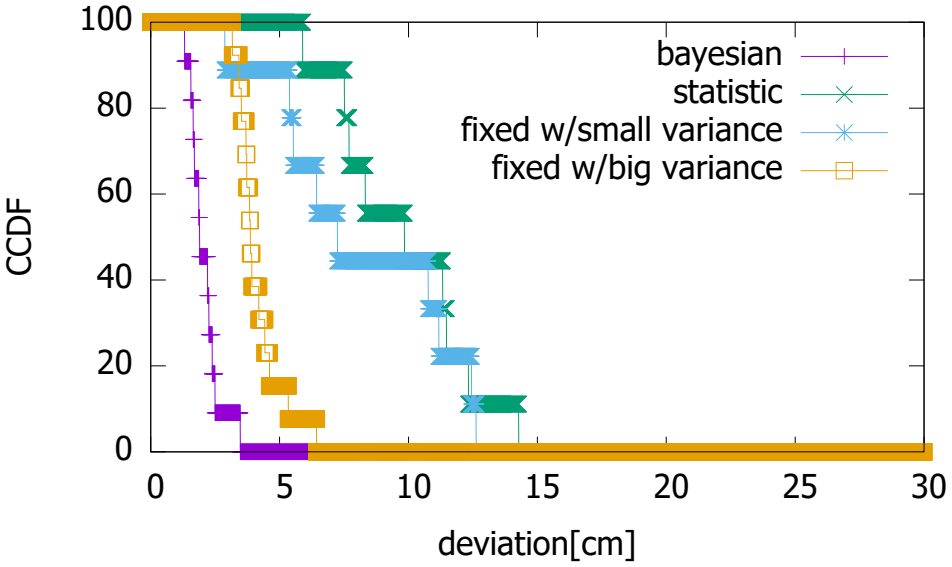

Figure 9: Deviation from target route with delay

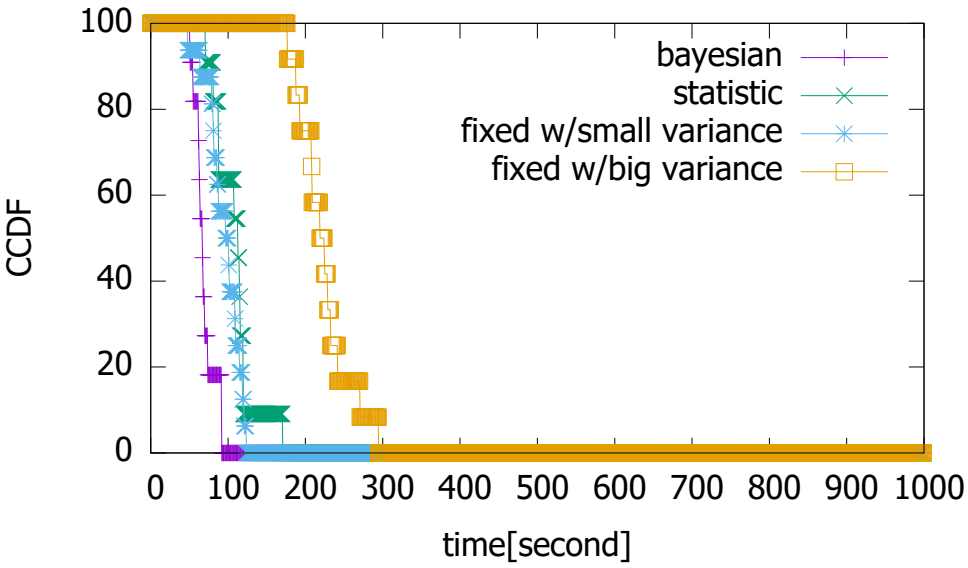

Figure 10: Task completion time without delay

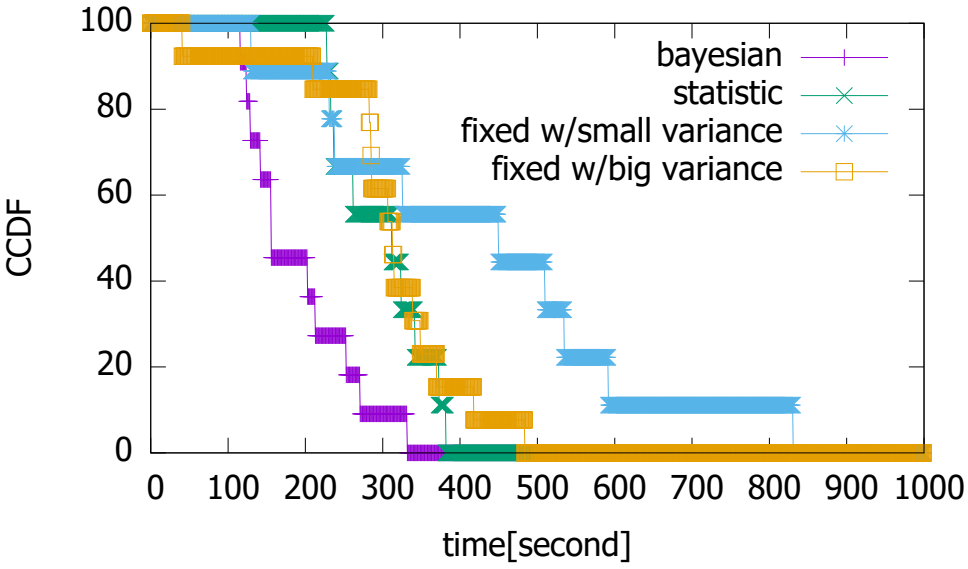

Figure 11: Task completion time with delay

6 Conclusion and Future Work

In this thesis, we established a framework that enables controllers to identify the current environment from a small number of observation results. Our framework is inspired by the organisms that evolve to become able to identify the current environment by using Bayesian inference with the prior distribution obtained through evolution.

In this thesis, we proposed an environment identification method using Bayesian inference for remote control. In this method, the following operation is performed by a controller. Based on observation obtained from the robot and control commands sent to the robot, the controller calculates an error that occurred when running the commands.Based on the obtained error, the controller updates the error model for the current environment by Bayesian inference and calculates the control commands based on the error model.

In our framework, the prior distribution of the error model is obtained by the evolution under various environments; in each generation, we evaluate the controller with each prior distribution by scoring the results of tasks done by the controller and evolve the parameters of prior distribution based on the evaluation results.

In this thesis, we implemented a controller for a two-wheel mobile robot based on our framework. Through experiments, we demonstrate that the controller based on our framework identified the current environment and control the robot properly. As a result, our method reduces the deviation from the intended path by about 75% and the control time by about 35% compared to the method that identifies the current environment only from the observed information.

In this report, in order to obtain prior distribution, we performed evolutionary computation using genetic algorithm in simulation. When applied to an actual device, it is necessary to obtain a prior distribution that can cope with an error generated in an actual environment while performing an operation test. In order to realize such a method, it is important to show that evolutionary computation can be realized by real machine experiments.

Futher our future research topics include applying our framework to the other tasks and/or the other robots.

Acknowledgments

We would like to thank Professor Masayuki Murata of the Graduate School of Information Science and Technology at Osaka University. He took the time for me and gave me creative suggestions, appropriate advice and words of encouragement. I would like to deeply thank again for your great support in promoting my research.

I would also like to express my sincere appreciation to Associate Professor Yuichi Ohsita of the Graduate School of Information Science and Technology, Osaka University for taking the time to conduct this research and providing appropriate advice and guidance. Despite he was busy, he eagerly provided me with detailed comments on my research, how to write my thesis, etc. Without his guidance, I could not write this thesis. I would like to thank again.

I would also like to show great appreciation to Associate Professor Shin'ichi Arakawa of Graduate School of Information Science and Technology of Osaka University, Assistant Professor Daichi Kominami of Graduate School of Economics of Osaka University and Specially Appointed Assistant Professor Tatsuya Otoshi of Graduate School of Information Science and Technology of Osaka University. Their support in conducting my research on a daily basis was essential to me.

We also thank Saeko Shigaki of the Graduate School of Information Science and Technology at Osaka University for creating and providing illustrations of the mobile robot used in this research. Finally, We would like to thank Mr. Masaaki Yamauchi, who has been supported in various aspects on a daily basis, and everyone in Murata Lab.

References

[1] J. C. Ramírez and J. A. Marshall, "Can natural selection encode bayesian priors?" *Journal of Theoretical Biology*, vol. 426, pp. 57–66, Aug 2017.

[2] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," in *Proceeding of the IEEE*, vol. 95, Jan 2007, pp. 138–162.

[3] M.-F. R. Lee, F.-H. S. Chiu, H.-C. Huang, and C. Ivancsits, "Generalized predictive control in a wireless networked control system," *International Journal of Distributed Sensor Networks*, vol. 9, no. 12, p. 475730, Jan 2013.

[4] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, Mar 2015.

[5] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, "Networked control systems: a survey of trends and techniques," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 1–17, 2019.

[6] S. Yasuda and H. Yoshida, "Dynamic optimization of a remote control cycle for better responsiveness," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, Oct 2018, pp. 2986–2991.

[7] Á. Cuenca, W. Zhan, J. Salt, J. Alcaina, C. Tang, and M. Tomizuka, "A remote control strategy for an autonomous vehicle with slow sensor using kalman filtering and dual-rate control," *Sensors*, vol. 19, no. 13, p. 2983, 2019.

[8] Y. Zhang, S. Xie, L. Ren, and L. Zhang, "A new predictive sliding mode control approach for networked control systems with time delay and packet dropout," *IEEE Access*, vol. 7, pp. 134 280–134 292, 2019.

[9] A. Stenman, "Model-free predictive control," in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 4. IEEE, Dec 1999, pp. 3712–3717.

[10] K. Nagatani, D. Endo, and K. Yoshida, "Improvement of the odometry accuracy of a crawler vehicle with consideration of slippage," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, Apr 2007, pp. 2752–2757.

[11] J.-B. Mouret, "Micro-data learning: The other end of the spectrum," *ERCIM News*, vol. 107, pp. 18–19, Oct 2016.

[12] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[13] R. Martinez-Cantin, N. De Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, vol. 27, no. 2, pp. 93–103, 2009.

[14] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty." in *Robotics: Science and Systems*, vol. 3, 2007, pp. 321–328.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*, C. M. Bishop, Ed. NSpringer, 2006.

[16] C. Lozoya, P. Martí, M. Velasco, and J. Fuertes, "Effective real-time wireless control of an autonomous guided vehicle," in *Proceeding of 2007 IEEE International Symposium on Industrial Electronics*. IEEE, June 2007, pp. 2876–2881.