Doctoral Dissertation

Gaussian Process Policy Search with Latent Variables in Uncertain Environments

Hikaru Sasaki
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Kenji Sugimoto
Robot Learning Lab. (Division of Information Science)

Submitted on December 12, 2021

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial ful llment of the requirements for the degree of
Doctor of Engineering

Hikaru Sasaki

Gaussian Process Policy Search with Latent Variables in Uncertain Environments*

Hikaru Sasaki

Abstract

Policy search reinforcement learning has been drawing much attention as a method for learning robot control. In particular, Gaussian process policy search (GP-PS) using Gaussian process regression as the policy model can learn optimal actions from high-dimensional and redundant sensors as input. However, it is difficult to naively apply such a GP-PS to real-world tasks because such tasks often involve various uncertainties. This is because the uncertainty of the environment often requires a very complex state-to-action mapping as a policy to obtain high-performance actions. To overcome such difficulties, this dissertation incorporates the concept of latent variable models in supervised/unsupervised learning. A latent variable model is a statistical model that describes the relation of observable variables to latent variables. If a latent variable model is appropriately designed, it may be possible to acquire a practical model that captures complex data while maintaining the simplicity of the learning algorithm. This dissertation proposes policy search methods to address the following three issues: 1) multiple optimal actions emerging from a reward function with ambiguous specification, 2) weak observations from the environment that contain little information about the state, 3) unpredictable fluctuations in the task performance evaluation. For issues 1) and 2), we designed a policy model for each issue by introducing latent variables into the Gaussian process and derived the policy update schemes based on variational Bayesian learning. The performance of the proposed policy search

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, December 12, 2021.

method was verified by simulation and a task using a robot manipulator. Finally, for issue 3), a policy learning framework based on Bayesian optimization (BO) with latent variables is proposed for application to actual heavy machinery, and its performance was verified using a real waste crane.

Keywords:

Policy Search, Reinforcement Learning, Gaussian Process, Latent Variable Model, Variational Bayesian Learning

不確実な環境下における
潜在変数を持つガウス過程方策探索[*]

佐々木光

内容梗概

ロボットの制御方策を自律的な試行錯誤から学習する方法として、方策探索型の強化学習が提案されている．特に，ガウス過程回帰を方策モデルとして用いたガウス過程方策探索は，高次元で冗長なセンサー値を入力としたロボットの制御方策が学習可能である．一方で，ガウス過程方策探索の実世界タスクへの応用は実世界タスクの環境が持つさまざまな不確実性によって，依然として困難である．不確実性の高い環境下において性能の高い方策を獲得するためには，方策が状態から行動への複雑な関数関係を捉える必要がある．しかし，シンプルな方策モデルを仮定してるガウス過程方策探索は複雑な関数を学習できない．このような方策の学習に対する問題を克服するために，教師あり/教師なし学習における潜在変数モデルに着目する．潜在変数モデルは，観測可能な変数と観測できない潜在変数を関連付ける確率モデルで，潜在変数を適切に設計することで，アルゴリズムのシンプルさを維持しながら複雑なデータを捉えるモデルを学習することができる．この博士論文は，不確実な実世界環境で引き起こされる方策学習の困難さに対して，ガウス過程方策探索に潜在変数を取り入れるアプローチを探求する．不確実性をもつ実世界タスクへの方策探索を応用することを目的とし，潜在変数の推論と方策の学習を同時に行うアルゴリズムを導出する．特に，実世界タスクが持つ，1) 曖昧に設計された報酬関数によって生じる複数の最適行動と 2) 環境の情報が十分に得られない観測の 2 つの複雑さ，3) タスク性能の評価に対する予測できない影響に焦点を当てる．1) と 2) に対して，潜在変数をガウス過程方策モデルに導入することにより，実世界タスクが持つ複雑さに対する新たなガウス過程方策モデルを設計し，変分ベイズ学習に基づいて潜在変数付き方策の更新則を導出する．提案された方策探索手法の性能をシミュレーションとロボットマニピュレータを使用したタスクによって検証した．また，3) に対して，潜在変数を

[*]奈良先端科学技術大学院大学 先端科学技術研究科 博士論文, 2021 年 12 月 12 日.

導入したベイズ最適化に基づく方策学習フレームワークを提案し，提案法のゴミ焼却施設のゴミクレーンタスクへの有効性を検証した.

キーワード

方策探索，強化学習，ガウス過程，潜在変数モデル，変分ベイズ学習

Contents

List of Figures

List of Tables

1. Introduction

1.1. Background

Automation of robot tasks requires a control policy to accomplish the tasks through consecutive action decisions from states observed from an environment. Reinforcement learning is a promising data-driven approach that acquires a control policy to accomplish tasks by repeating trial and error using the control policy and learning of the control policy using trial and error data [1]. Policy search reinforcement learning, one of the reinforcement learning methods, directly learns a robot control policy for performing tasks from trial and error data by a robot [2,3]. In particular, Gaussian process policy search (GP-PS), which employs Gaussian process regression [4] as a control policy model, can learn continuous robot control policies without designing a policy model [5–7]. GP-PS learns a control policy as a function by assuming a function for the state-to-action mapping. Gaussian process (GP) is generally used for a prior distribution for a function between input and output data and regresses the function probabilistically using data. Also, GP can learn a non-linear function with a small amount of data by implicitly handling high-dimensional features by kernel tricks. For example, the GP-based policy successfully used image-related, high-dimensional (800 dim.) features as inputs [6].

However, it is challenging to naively apply such GP-PS to real-world tasks like robot control in a factory and heavy machinery control in a vast workspace. Real-world tasks are often accompanied by the uncertainty of the environment due to the difficulty of observing the state of various manipulating objects and the environment-dependent constraints on installing sensors. The uncertainty of the environment often requires a very complex state-to-action mapping as a policy to obtain high-performance actions. GP-PS cannot capture this complex mapping due to the poor representability of the policy model.

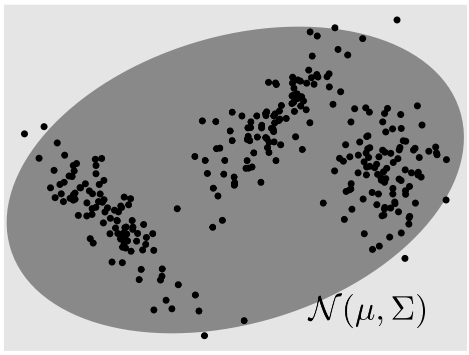
1.2. Approach

To overcome such difficulties, this dissertation incorporates the concept of latent variable models in supervised/unsupervised learning to GP-PS. A latent variable model is a statistical model that describes the relation of observable variables to latent variables. A typical example of a latent variable model is the Gaussian mixture model (GMM) [8]. GMM is proposed for multimodal data that is generated from multiple components. For the uncertainty about components, GMM employs a latent variable $\mathbf{Z}$ that identifies the data points generated by the same component to capture multimodal data with some Gaussian distributions (Fig. 1.1). If a latent variable model is appropriately designed for data, it may be possible to acquire a practical model that captures complex data while maintaining the simplicity of the learning algorithm.

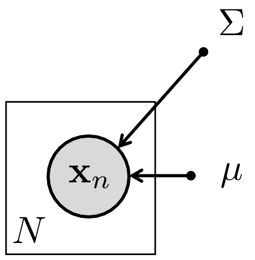
This dissertation explores a latent variable modeling approach in GP-PS to cope with difficulty caused in uncertain environments (Fig. 1.2). We then derive an algorithm that simultaneously performs latent variable inference and policy learning and aims to make policy search applicable to various real-world control tasks with uncertainty.

1.3. Contribution

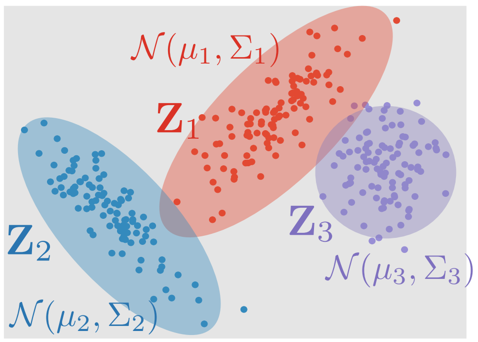
This dissertation proposes novel policy search methods to address complex state-action mapping caused by uncertainty in real-world tasks. Mainly, we focus on the automation of waste cranes as a real-world control task. A waste crane is a vast machine that can grasp several tons of waste at a time. The waste crane manages various garbage collected every day in the vast pit of the waste incineration plant (Fig. 1.3). The main tasks of the waste crane are to put the garbage into the incinerator and homogenize the garbage by grabbing and scattering the garbage. While handling a variety of garbage in a vast environment, the waste crane is only equipped with a sensor that observes the weight of the grasped garbage due to the high maintenance cost caused by the pollution of the garbage. Therefore, in



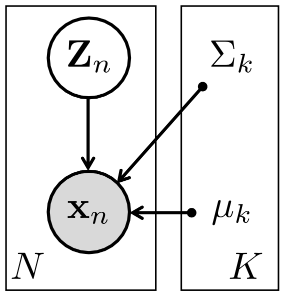
(a) Gaussian distribution



(b) Graphical model of Gaussian distribution



(c) Gaussian mixture model

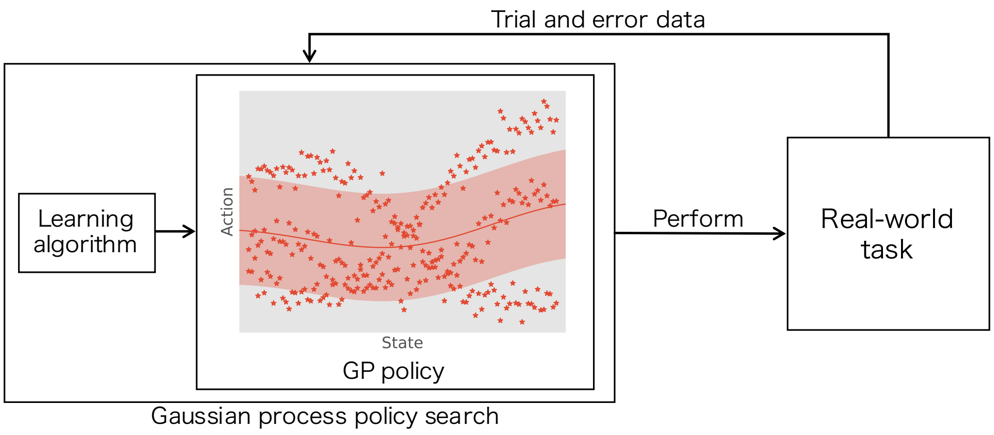


(d) Graphical model of Gaussian mixture model

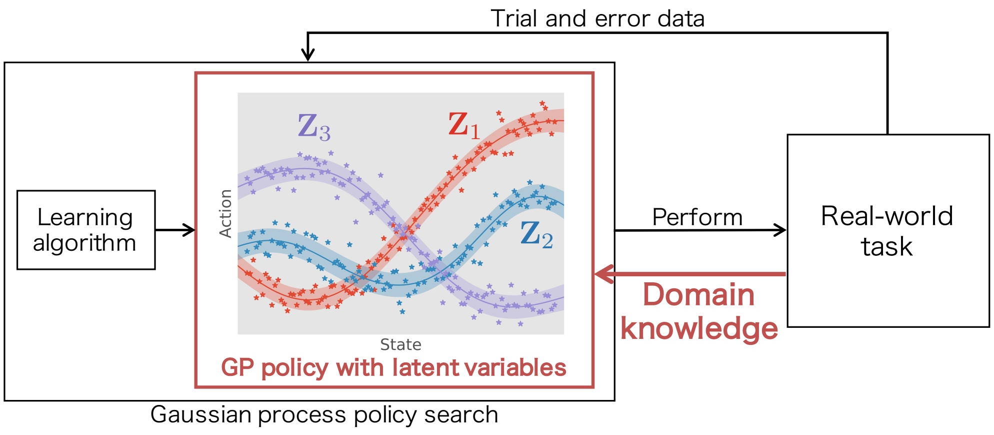
Figure 1.1.: A latent variable modeling approach for complex data. (a) (b) The Gaussian distribution cannot capture data generated by multiple components. (c) (d) By introducing a latent variable **Z** that identifies the data points generated by the same component, GMM captures multimodal data with some Gaussian distributions.

order to automate the waste crane, the policy search must learn a control policy in an environment with high uncertainty by a few sensors. The uncertainty of the environment due to such a shortage of sensors can also occur in other real-world tasks.

In particular, this dissertation proposes methods for the following three issues: 1) multiple optimal actions emerging from a reward function with ambiguous specification, 2) weak observations from the environment that contain little information about the state, 3) unpredictable fluctuations in the task performance evaluation.



(a) GP-PS



(b) GP-PS with latent variables

Figure 1.2.: Overview of GP-PS with latent variables. (a) GP policy cannot capture the complexity of trial and error data indicated by the dots. (b) GP policy with latent variables can capture complex data.

1) **Multiple optimal actions emerging from a reward function with ambiguous specification**

In an environment where complete information cannot be observed, it is difficult to design an evaluation function for the performance of a policy appropriately. An evaluation function inappropriate for a task generates multiple optimal actions for any state. Since Gaussian process policy search assumes one function as a policy model, it is not possible to learn a policy to select an appropriate action in a task in which multiple optimal actions exist.

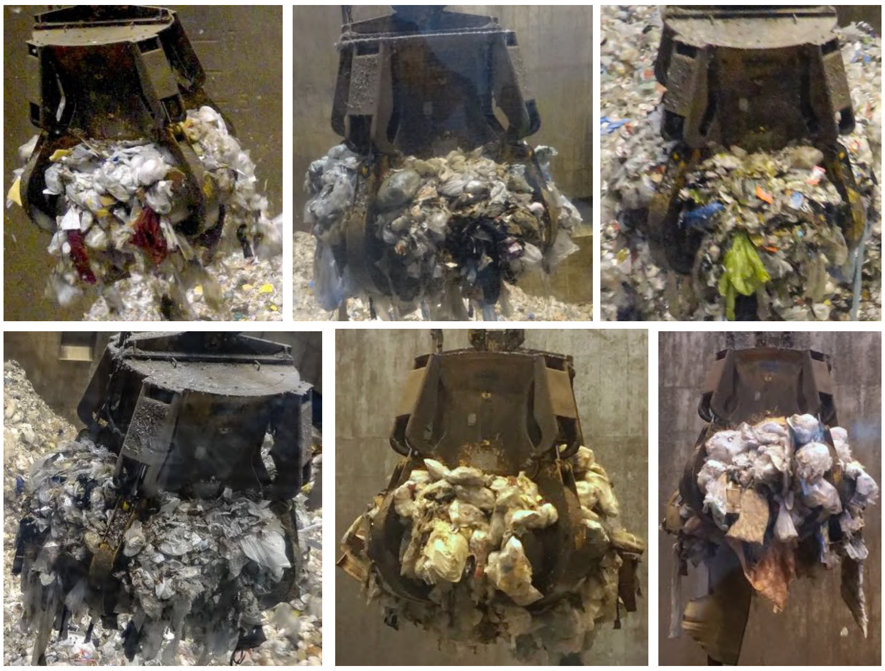


Figure 1.3.: Garbage grasped by a waste crane

2) **Weak observations from the environment that contain little information about the state**
   Generally, the policy acquired by the reinforcement learning method is to select the optimal action for the observed current state. However, if there are insufficient sensors to recognize changes in the actual environment, the control policy cannot select the optimal action.

3) **Unpredictable fluctuations in the task performance evaluation**
   Similar to 1), designing a function that appropriately evaluates a policy's performance in a real-world task is challenging. Such an evaluation function underestimates or overestimates the policy. Inappropriate evaluation of control policies prevents the learning of optimal control policies because policies are optimized based on policy evaluation.

For issues 1) and 2), we designed a policy model for each issue by introducing latent variables into the Gaussian process and derived the policy update schemes based on variational Bayesian learning. The performance of the proposed policy search method was verified by simulation and a task using a robot manipulator.

Finally, for issue 3), a policy learning framework based on Bayesian optimization (BO) with latent variables is proposed for application to actual heavy machinery, and its performance was verified using a real waste crane.

1.3.1. Multimodal and Mode-seeking Policy Search for Multiple Optimal Actions

To alleviate the problem of multiple optimal actions, we propose novel approaches in a GP-PS with multiple optimal actions and offer two different algorithms: a multimodal sparse Gaussian process policy search (multimodal SGP-PS) and a mode-seeking sparse Gaussian process policy search (mode-seeking SGP-PS). We introduce the following key ideas: 1) multimodality for capturing multiple optimal actions and 2) mode-seeking for capturing one optimal action by ignoring the others.

We validated the effectiveness of these algorithms through robotic manipulation tasks with multiple optimal actions in simulations: 1) hand-posture adjustment task using a robot simulator, V-REP [9] and 2) table-sweeping task using MuJoCo simulator [10]. The results of the hand-posture adjustment task demonstrate that our methods can efficiently learn optimal actions even with multiple optimal actions that previous methods cannot. The multimodal SGP-PS captures multiple optimal actions with the multimodal policy model. The mode-seeking SGP-PS learns an effective unimodal policy by seeking an optimal action. In the table-sweeping task, we confirmed the performance of our methods for more challenging situations.

1.3.2. Self-triggered Policy Search for a Weakly Observable Environment

Human operator's behavior in a weakly observable environment, such as a garbage crane, seems different from general sensory feedback control that selects an action based on sensor values at regular time intervals which is a typical policy model in policy search and reinforcement learning (e.g., [7,11]). Instead, a human operator selects an action, and duration based on sensor values creates robust behavior and stabilizes it even with weak observations. However, these policies need to be adjusted according to the garbage's characteristics. Such thoughts motivated us to explore a novel policy search framework with a specific type of policy applicable in weakly observable environments.

For a limitation, we propose GP self-triggered policy search (GPSTPS), which is a novel GP policy search algorithm with an action duration time as a latent variable. GPSTPS has two types of control policies: action and duration. The gating mechanism either maintains the action selected by the action policy for the duration specified by the duration policy or updates the action and duration by passing new observations to the policy; therefore, it is described as *self-triggered*. GPSTPS simultaneously learns both policies by trial and error based on sparse GP priors [12] and variational learning [7] to maximize the return. We experimentally verified the performance of our proposed method on a garbage-grasping-scattering task with a waste crane with weak observations using a simulation and a robotic waste crane system. Our experimental results suggest that our proposed method acquired suitable policies to determine the action and duration based on the garbage's characteristics.

1.3.3. Multi-task Robust Bayesian Optimization for The Inhomogeneity of Garbage

Finally, as an application for waste crane automation with a weakly observable environment, we propose a data-efficient policy learning framework. In a garbage incineration plant, little information about the garbage characteristics causes unpredictable fluctuations in the crane's task performance. As another issue, obtaining data samples by executing tasks is very costly due to the slow-moving system (it takes several minutes to execute one task) and limited plant downtime. Therefore, a large amount of trial and error is infeasible for a large-scale industrial waste crane, unlike in [13, 14].

We propose a robust BO-based policy learning framework to learn a policy stably for unpredictable performance fluctuations. Although conventional BO algorithms are sensitive to outliers inevitable, and its performance may deteriorate significantly. Therefore, our framework employs a novel BO algorithm, Multi-Task Robust Bayesian Optimization (MTRBO). The MTRBO has the following characteristics: (1) outlier robustness against garbage inhomogeneity and (2) sample reuse from previously solved tasks to enhance the sample efficiency further. To investigate our framework's effectiveness, we conducted experiments on garbage-scattering tasks with a robot waste crane and an actual waste crane. Experimental results demonstrate that our MTRBO framework robustly optimized the control policies of the garbage cranes, even with a significantly reduced number of data under the influence of garbage inhomogeneity.

1.4. Organization of Dissertation

The remainder of this dissertation is organized as follows:

Chaper 2
Derivation of sparse GP policy search based on variational Bayesian learning as the preliminaries.

Chaper 3
Proposal of multimodal SGP-PS and mode-seeking SGP-PS for multiple optimal actions, and verification of each method by robotics tasks using simulator.

Chaper 4
Proposal of GPSTPS for a weak observable environment and verification using a simulator and a robot waste crane system.

Chaper 5
Proposal of MTRBO for unpredictable fluctuation of performance and verification using a robot waste crane system and an actual waste crane.

Chaper 6
Discussions of this dissertation.

Chapter 7
Conclusions of this dissertation.

Appendix
Details of update laws of each method.

2. Preliminaries

In this chapter, as a preliminary, we describe two policy learning frameworks, policy search, and Bayesian optimization-based policy learning. Each method aims to learn the optimal policy iteratively by repeating execution and updating policy.

2.1. Policy Search

Policy search is an algorithm that acquires a policy that maximizes the expected return in the environment formulated by the Markov decision process. In this algorithm, usually, a policy is defined as a function of the state to action. The policy is optimized by repeating the following two processes as shown in Fig. 2.1:

1. Trial and error of the policy by executing tasks,
2. Policy update using trial and error data.

Episode $d = \{\mathbf{s}_1, a_1, \cdots, \mathbf{s}_T, a_T, \mathbf{s}_{T+1}\}$, which is a series of state $\mathbf{s}_t$, action a_t, and reward $r(\mathbf{s}_t, a_t)$, is collected in a trial and error fashion by the policy. Here we assume the following derivation where the action is one-dimensional for clarity without a loss of generality. Let the initial state probability and the state transition probability of the environment be $p(\mathbf{s}_1)$ and $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$, and express the probability of episode $p(d)$:

$$p(d \mid \pi) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi(a_t \mid \mathbf{s}_t) p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t). \tag{2.1}$$

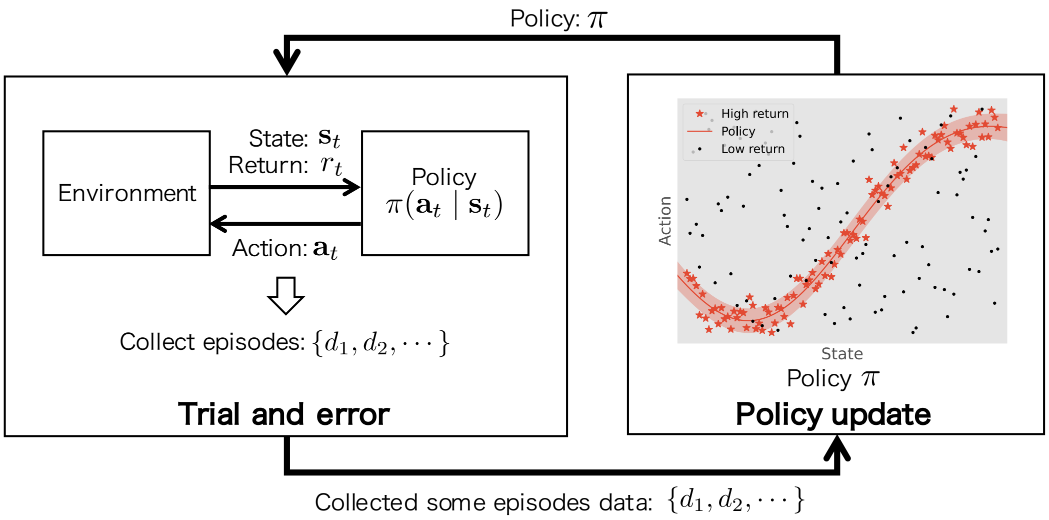


Figure 2.1.: The learning process of policy search. The policy is optimized gradually by repeating trial-and-error and policy updates.

The policy is improved by calculating it to maximize the expected return using return $R(d) = \sum_{t=1}^{T} r(\mathbf{s}_t, a_t)$ and probability $p(d)$:

$$\pi^* = \arg\max_{\pi} J(\pi), \tag{2.2}$$

$$J(\pi) = \int R(d)p(d \mid \pi) \, \mathrm{d}d. \tag{2.3}$$

2.2. Sparse Gaussian process policy search (SGP-PS)

SGP-PS is a policy search method that employs SGP as a policy model to regress a state-to-action function. We derive a policy search that optimizes the SGP policy model by maximizing a lower bound on the expected return. We denote a policy model that determines the action from the state as $p(a_t \mid \mathbf{s}_t) = \mathcal{N}(a_t \mid f(\mathbf{s}_t), \sigma^2)$, where f is the mean function of the policy. We place a GP prior on mean function f:

$$f \sim \mathcal{GP}(\mathbf{0}, \mathrm{k}(\mathbf{s}, \mathbf{s}')), \tag{2.4}$$

where $\mathrm{k}(\mathbf{s}, \mathbf{s}')$ is a kernel function.

As preparation for a policy search, we reduce the computational complexity by *augmenting* the prior distribution by introducing common pseudo inputs $\bar{\mathbf{s}} = \{\bar{\mathbf{s}}_l\}_{l=1}^{L}$ and corresponding pseudo outputs $\bar{\mathbf{f}} = \{\bar{f}_l\}_{l=1}^{L}$ [12, 15]. We place a GP prior on the pseudo outputs:

$$p(\bar{\mathbf{f}} \mid \bar{\mathbf{s}}) = \mathcal{N}(\bar{\mathbf{f}} \mid \mathbf{0}, \mathbf{K}_{\bar{\mathbf{s}}}), \tag{2.5}$$

where $\mathbf{K}_{\bar{\mathbf{s}}}$ is the kernel gram matrix computed with $\bar{\mathbf{s}}$. A prior distribution of $f_t = f(\mathbf{s}_t)$ is augmented:

$$p(f_t \mid \bar{\mathbf{f}}, \bar{\mathbf{s}}, \mathbf{s}_t) = \mathcal{N}(f_t \mid \mathbf{K}_{\mathbf{s}_t,\bar{\mathbf{s}}}\mathbf{K}_{\bar{\mathbf{s}}}^{-1}\bar{\mathbf{f}}, \lambda_t), \tag{2.6}$$

where $\lambda_t = \mathbf{K}_{\mathbf{s}_t} - \mathbf{K}_{\mathbf{s}_t,\bar{\mathbf{s}}}\mathbf{K}_{\bar{\mathbf{s}}}^{-1}\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_t}$, $\mathbf{K}_{\bar{\mathbf{s}}} = \mathrm{k}(\bar{\mathbf{s}}, \bar{\mathbf{s}})$, $\mathbf{K}_{\mathbf{s}_t,\bar{\mathbf{s}}} = \mathrm{k}(\mathbf{s}_t, \bar{\mathbf{s}})$.

To derive a policy search algorithm, we define the probability of the trajectory using a SGP policy model:

$$p(d, \mathbf{f}, \bar{\mathbf{f}} \mid \bar{\mathbf{s}}) \equiv p(\mathbf{s}_1)p(\bar{\mathbf{f}} \mid \bar{\mathbf{s}}) \prod_{t=1}^{T} p(a_t \mid f_t)p(f_t \mid \bar{\mathbf{f}}, \bar{\mathbf{s}}, \mathbf{s}_t)p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t), \tag{2.7}$$

where $\mathbf{f} = \{f_t\}_{t=1}^{T}$. For clarity, we omit the conditioning on $\mathbf{s}$ and $\bar{\mathbf{s}}$ in the remainder of this section. The expected return is defined:

$$J(\boldsymbol{\theta}) = \int R(d)p(d, \mathbf{f}, \bar{\mathbf{f}})\mathrm{d}d\mathrm{d}\mathbf{f}\mathrm{d}\bar{\mathbf{f}}. \tag{2.8}$$

The graphical model of SGP-PS is shown in Fig. 2.2.

The goal of a policy search is to find the hyperparameters of policy $\boldsymbol{\theta}$ to maximize the expected return. However, solving this optimization is difficult due to the analytical intractability of the integral. We derive the lower bound of expected return $\log J_L(\boldsymbol{\theta})$ from Jensen's inequality by introducing variational distribution $q(d, \mathbf{f}, \bar{\mathbf{f}})$:

$$\begin{aligned}
\log \frac{J(\boldsymbol{\theta})}{J_{\text{old}}} &= \log \int \frac{R(d)}{J_{\text{old}}} \frac{q(d, \mathbf{f}, \bar{\mathbf{f}})}{q(d, \mathbf{f}, \bar{\mathbf{f}})} p(d, \mathbf{f}, \bar{\mathbf{f}})\mathrm{d}d\mathrm{d}\mathbf{f}\mathrm{d}\bar{\mathbf{f}} \\
&\geq \int \frac{R(d)}{J_{\text{old}}} q(d, \mathbf{f}, \bar{\mathbf{f}}) \log \frac{p(\mathbf{a} \mid \mathbf{f})p(\mathbf{f} \mid \bar{\mathbf{f}})p(\bar{\mathbf{f}})}{q(d, \mathbf{f}, \bar{\mathbf{f}})} \mathrm{d}d\mathrm{d}\mathbf{f}\mathrm{d}\bar{\mathbf{f}} \\
&\equiv \log J_L(\boldsymbol{\theta}, q),
\end{aligned} \tag{2.9}$$

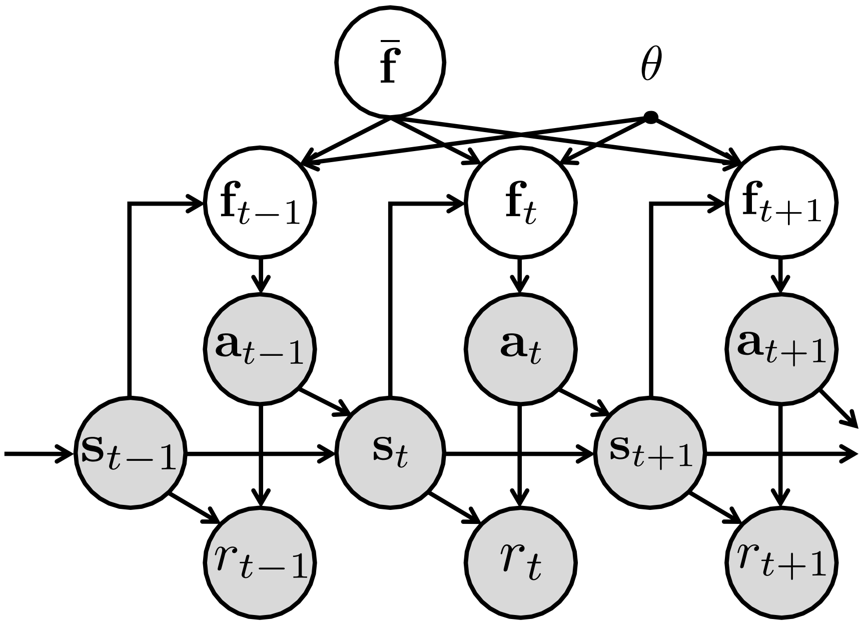


Figure 2.2.: Graphical model of SGP-PS

where J_{old} is the expected return using sampler p_{old} and p_{old} is a sampler of the training data using the previous policy. We introduce variational distribution $q(d, \mathbf{f}, \bar{\mathbf{f}}) = p_{\text{old}}(d)p(\mathbf{f} \mid \bar{\mathbf{f}})p(\bar{\mathbf{f}})$ and apply Monte Carlo approximation regarding expectation w.r.t. $p_{\text{old}}(d)$, and the lower bound becomes:

$$\log J_L(\boldsymbol{\theta}, q) \approx \int p(\mathbf{f} \mid \bar{\mathbf{f}})q(\bar{\mathbf{f}}) \log \frac{p(\tilde{\mathbf{a}} \mid \mathbf{f})p(\bar{\mathbf{f}})}{q(\bar{\mathbf{f}})} \mathrm{d}\mathbf{f}\mathrm{d}\bar{\mathbf{f}}, \tag{2.10}$$

where $p(\tilde{\mathbf{a}} \mid \mathbf{f})$ is the following return weighted likelihood using Gaussian distribution: $p(\tilde{\mathbf{a}} \mid \mathbf{f}) = \mathcal{N}(\tilde{\mathbf{a}} \mid \mathbf{W}\mathbf{f}, \sigma^2\mathbf{I})$, $\mathbf{W}$ is a weight based on the return express, $\mathbf{W} = \text{diag}\left\{\sqrt{\frac{R(d^1)}{J_{\text{old}}E}}\mathbf{1}, \cdots, \sqrt{\frac{R(d^E)}{J_{\text{old}}E}}\mathbf{1}\right\}$, d^e is e-th trajectory, and $\tilde{\mathbf{a}}$ is return weighted action samples, $\tilde{\mathbf{a}} = \mathbf{W}\mathbf{a}$. The lower bound of the expected return (Eq. 2.10) is the lower bound of the return weighted marginal likelihood.

We derive an EM-like policy update law that maximizes the lower bound of expected return J_L by alternately optimizing variational distribution $q(\bar{\mathbf{f}})$, parameters $\boldsymbol{\theta}$ by variational Bayesian inference.

The action selection for new state $\mathbf{s}_*$ is determined by the predicted distribution of the SGP policy model using variational distribution $q(\bar{\mathbf{f}})$ and hyperparameter

$\boldsymbol{\theta}$:

$$p(a_* \mid \mathbf{s}_*) \approx \int p(a_* \mid f_*) p(f_* \mid \bar{\mathbf{f}}, \mathbf{s}_*) q(\bar{\mathbf{f}}) \mathrm{d}f_* \mathrm{d}\bar{\mathbf{f}}. \tag{2.11}$$

2.3. Bayesian Optimization-based Policy Learning

Given the control policy with policy parameter $\mathbf{w}$, the objective of optimization is to find optimal parameter $\mathbf{w}^*$ that maximizes return function $R(\mathbf{w})$:

$$\mathbf{w}^* \leftarrow \arg\max_{\mathbf{w}} R(\mathbf{w}). \tag{2.12}$$

We assume that the return function $R(\mathbf{w})$ is unknown due to the difficulty in modeling, so such optimization is analytically intractable, but sampling the value of $R(\mathbf{w})$ is possible by executing tasks using the policy with the parameter $\mathbf{w}$. However, a brute force approach is infeasible for such problems with high sample cost as waste crane control. Such optimization problems are often called black-box optimization [16].

Bayesian Optimization (BO) is a sequential design strategy for black-box optimization. We employ BO to optimize policy parameter data efficiently. The policy parameter is optimized by repeating the following two processes as shown in Fig. 2.3:

1. Evaluation of policy parameter,

2. Determination of next query parameter.

BO utilizes an acquisition function $\alpha(\mathbf{w})$ and alternatively optimizes it to find next query parameter $\mathbf{w}'$:

$$\mathbf{w}' \leftarrow \arg\max_{\mathbf{w}} \alpha(\mathbf{w}). \tag{2.13}$$

The new query is tested to obtain evaluation value $y = f(\mathbf{w}')$, which is then used to update the acquisition function. These procedures are iteratively and alternatively executed until the process converges to optimal parameter $\mathbf{w}^*$.

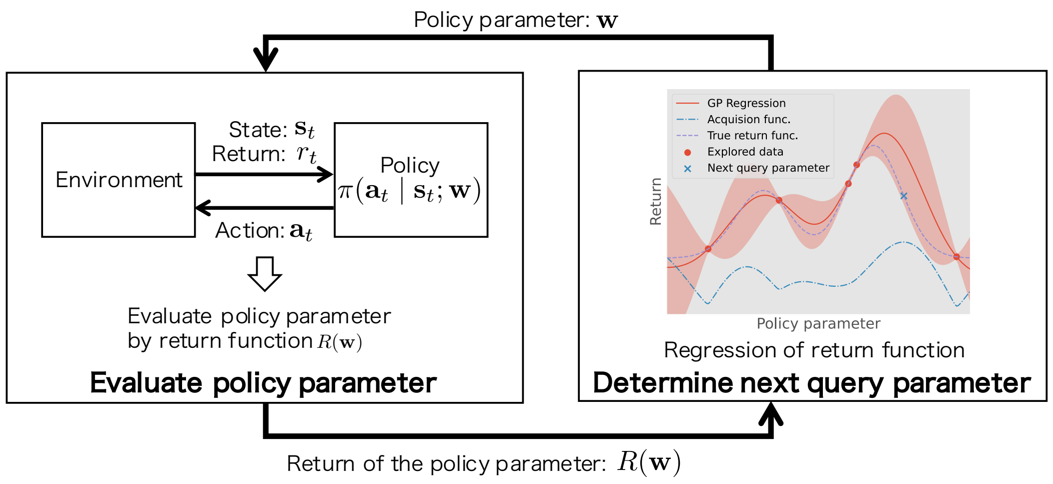

Policy parameter: $\mathbf{w}$
State: $\mathbf{s}_t$
Return: r_t
Environment
Policy
$\pi(\mathbf{a}_t \mid \mathbf{s}_t; \mathbf{w})$
Action: $\mathbf{a}_t$
Evaluate policy parameter by return function $R(\mathbf{w})$
Evaluate policy parameter
GP Regression
Acquision func.
True return func.
Explored data
Next query parameter
Return
Policy parameter
Regression of return function
Determine next query parameter
Return of the policy parameter: $R(\mathbf{w})$

Figure 2.3.: The overview of policy parameter optimization by Bayesian optimization. The policy parameter is explored sequentially by repeating policy parameter evaluation and determining the next query parameter.

To this end, the evaluation function is learned by Gaussian processes (GP) [4]. GP regression learns the relationship between the parameter and the function value is $y_n = f_n + \varepsilon_n$ with Gaussian noise $\varepsilon_n \sim \mathcal{N}(0, \beta)$ where $f_n = f(\mathbf{w}_n)$. GP regresses the evaluation function as a predictive distribution using previously evaluated parameters $\mathbf{w} = [\mathbf{w}_1, \cdots, \mathbf{w}_N]^T$ and evaluation values $\mathbf{Y} = [y_1, \cdots, y_N]^T$:

$$p(f(\mathbf{w}) \mid \mathbf{w}, \mathbf{Y}, \mathbf{w}) = \mathcal{N}(f(\mathbf{w}) \mid \mu(\mathbf{w}), \sigma^2(\mathbf{w})), \tag{2.14}$$

$$\mu(\mathbf{w}) = \mathrm{k}_{\mathbf{w},*}^T(\mathbf{K}_\mathbf{w} + \beta\mathbf{I})^{-1}\mathbf{Y}, \tag{2.15}$$

$$\sigma^2(\mathbf{w}) = \mathrm{k}(\mathbf{w}, \mathbf{w}) - \mathbf{k}_{\mathbf{w},*}^T(\mathbf{K}_\mathbf{w} + \beta\mathbf{I})^{-1}\mathbf{k}_{\mathbf{w},*}, \tag{2.16}$$

where $\mathrm{k}(\cdot, \cdot)$ is a kernel function that can calculate similarity between data with kernel parameter θ_k, $\mathbf{K}_\mathbf{w}$ is a kernel gram matrix as $[\mathbf{K}_\mathbf{w}]_{ij} = \mathrm{k}(\mathbf{w}_i, \mathbf{w}_j)$, $\mathbf{k}_{\mathbf{w},*}$ is a kernel vector as $[\mathbf{k}_{\mathbf{w},*}]_i = \mathrm{k}(\mathbf{w}_i, \mathbf{w})$, $\mathbf{I}$ is an identity matrix. Mean function $\mu(\mathbf{w})$ indicates the mean of the predictive distribution. Variance function $\sigma^2(\mathbf{w})$ is the prediction variance, and the value of $\sigma^2(\mathbf{w})$ tends to increase in the region where the data are insufficient due to the prediction's uncertainty.

The GP predictive distribution can be used to obtain Upper Confidence Bound (UCB) as an acquisition function $a(\mathbf{w})$ [17] defined as:

$$a(\mathbf{w}) = \mu(\mathbf{w}) + \kappa\sigma^2(\mathbf{w}), \tag{2.17}$$

where $\kappa \geq 0$ is a parameter that controls the trade-off between exploration and exploitation.

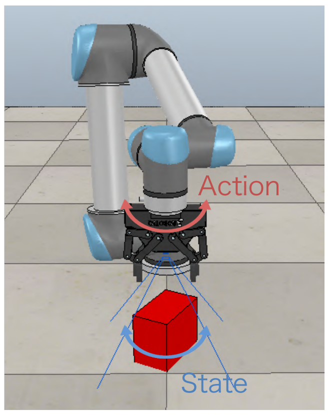
3. Variational Policy Search for Multiple Optimal Actions

3.1. Introduction

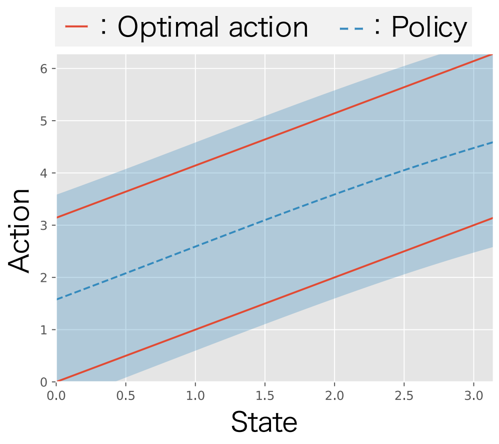
GP-PS methods implicitly assumed that optimal actions become unique for each state. This assumption can severely limit to such practical applications as robot manipulations since designing a reward function that appears only one optimal action for each state is often complicated for complex tasks. A reward function that appears multiple optimal actions may lead to poor performance in the previous methods. Multiple optimal actions may appear at particular states, although typical non-parametric policies cannot be captured due to their unimodality.

As an illustrative example of a robotic task, we consider a hand-posture adjustment task by a robotic arm (Fig. 3.1 (a)). The objective is to grasp the object by rotating the robot's wrist. We simply designed the reward function so that a positive value is given when the robot can grab and lift the object. Such a reward function appears in multiple optimal actions at each state since the robot grabs the object from multiple wrist angles. Due to the multiple optimal actions, previous methods that assume unimodality in their policy model choose incorrect actions (Fig. 3.1 (b)). Of course, for such simple tasks, we could elaborate the reward by adding additional terms. However, designing a reward function that appears only one optimal action is challenging when dealing with more complex tasks.

To alleviate this limitation, we propose novel approaches in a GP-PS with multiple optimal actions and offer two different algorithms: a multimodal sparse Gaussian process policy search (multimodal SGP-PS) and a mode-seeking sparse Gaussian process policy search (mode-seeking SGP-PS). Both methods employ



(a)

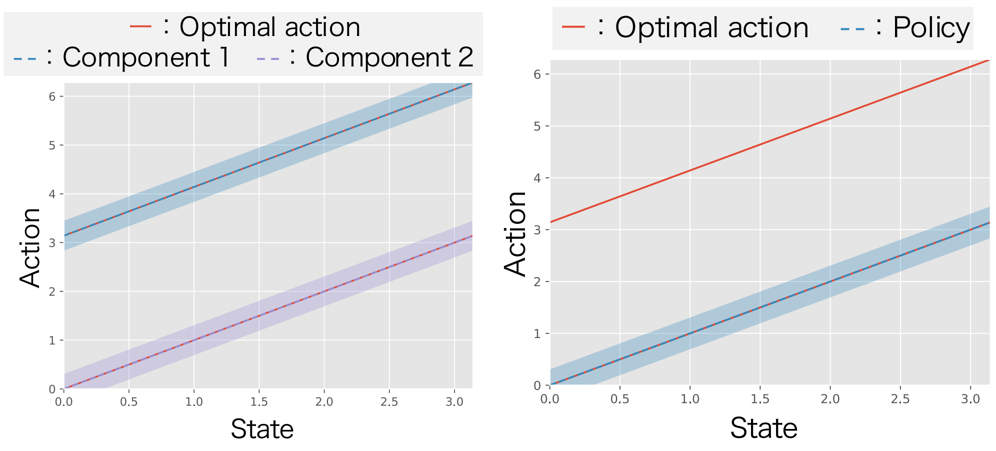


(b)

Figure 3.1.: Illustrative example of robotic tasks with multiple optimal actions: (a) hand-posture adjustment task environment and (b) red solid lines indicate multiple optimal actions. Blue broken line and region indicate mean and standard deviation of learned policy with a unimodal policy model, which cannot capture optimal actions.

sparse Gaussian processes [12,15] as a prior of a policy model to determine a robot action and derive update laws based on variational Bayesian inference [18–20]. We introduce the following key ideas: 1) multimodality for capturing multiple optimal actions (Fig. 3.2 (a)) and 2) mode-seeking for capturing one optimal action by ignoring the others (Fig. 3.2 (b)). Multimodal SGP-PS employs a multimodal policy prior inspired by a recent extension of Gaussian processes [21]. This prior distribution assumes that each component is responsible for a global function over the common input space. Mode-seeking SGP-PS employs a student-t distribution with outlier robustness as a likelihood function, which facilitates the learned policy to capture one of the optimal actions by treating the others as outliers to be ignored. For deriving reasonable policy update schemes, we use scale mixture representation to the student-t distribution [22, 23].

We validated the effectiveness of these algorithms through robotic manipulation tasks with multiple optimal actions in simulations: 1) hand-posture adjustment task using a robot simulator, V-REP [9] and 2) table-sweeping task using MuJoCo simulator [10]. The results of the hand-posture adjustment task demonstrate that our methods can efficiently learn optimal actions even with multiple optimal ac-



(a) (b)

Figure 3.2.: Two proposed methods for multiple optimal actions: (a) multimodal SGP-PS that can capture multimodality in optimal action using multiple models and (b) mode-seeking SGP-PS that can capture one optimal action and ignore others

tions that previous methods cannot. The multimodal SGP-PS captures multiple optimal actions with the multimodal policy model. The mode-seeking SGP-PS learns an effective unimodal policy by seeking an optimal action. In the table-sweeping task, we confirmed the performance of our methods for more challenging situations.

3.2. Related Work

In this section, we describe related works in the following three categories: 1) non-parametric policy search, 2) multimodal reinforcement learning, and 3) robust reinforcement learning.

Non-parametric policy search

Non-parametric policy search uses a non-parametric model as a policy model. Bagnell et al. [24] embedded the function of the policy model in a reproducing kernel Hilbert space (RKHS) and used the policy gradient method to maximize the expected return, making it possible to learn non-parametric policies. However, that approach can only learn a policy with a discrete action. Therefore, a method was proposed for learning continuous policies by assuming a Gaussian distribution in the policy model, embedding the mean function in RKHS, and learning the policy gradient method [25]. For robot applications, the Cost-regularized Kernel Regression [5] method was applied to a ball-hitting task in table tennis. To avoid the difficulty of convergence in a gradient-based policy search, an EM-inspired non-parametric policy search was applied to a door-opening task [26]. A data-efficient non-parametric policy search was proposed [6] that can learn a policy with high-dimensional features.

Although the above studies focused on optimizing such policy models with high representation capability, scant attention was given to the relationship between the reward function design and the limitations of unimodality in the policy. A reward function that generates multiple optimal actions may degrade the performance in those methods. On the other hand, we focused on this relationship and developed algorithms that can alleviate the limitations when the tackled task has multiple optimal actions.

Multimodal reinforcement learning

A multimodal policy search aims to learn a policy that can capture the multimodality of optimal actions. Many previous policy search-based methods employ a hierarchical policy model that consists of low-level policies that determine robot action and a high-level policy that determines which low-level policy is used [27–30]. However, the proposed approaches employ such hierarchical parametric models as 1) a softmax gating function with linear Gaussian sub-policies or 2) Gaussian mixture models with a parametric policy model, both of which require hand-engineered features to cope with a high-dimensional sensor input, unlike non-parametric methods.

Multimodality in optimal action has also been considered in neural network-based RL methods. For example, soft Q-learning learns a value function to acquire diverse behaviors by introducing an entropy term in the Bellman equation that promotes capturing multimodality in policies [31]. Soft actor-critic (SAC) is an extension of soft Q-learning; the learning performance is greatly improved [32,33]. In SAC, the value function captures various behaviors, whereas the policy employs a unimodal model. Thus, it cannot explicitly capture multimodality in optimal actions. SAC-GMM with a GMM policy was also explored as an early version of SAC, but its performance is worse than SAC due to the algorithmic complexity [34]. Kalashnikov et al. proposed another method that selects an action from an action-value function using the cross-entropy method [35].

We propose a multimodal SGP-PS that employs a policy model inspired by the overlapping mixtures of Gaussian processes (OMGP) [21]. Multimodal SGP-PS learns a policy whose components in the mixture are global and overlapping in state space, unlike a typical mixture model that dictates the input space for capturing non-stationarity [36–39]. We incorporate this feature into the policy search. We also employ a sparse Gaussian process as a prior distribution for each low-level policy. Such a simpler structure with a non-parametric prior allows us to derive policy update algorithms that can cope with high-dimensional sensor input without requiring hand-engineered features.

Robust reinforcement learning

Robust reinforcement learning acquires a robust policy for several kinds of noise. Recent research has attracted attention by focusing on the errors between simulations and real environments in sim-to-real domains. Reinforcement learning (robust to state noise [40]) and robust reinforcement learning (robust to disturbances in actual environments) have been proposed [41]. In research that focuses on reward functions, methods have been proposed that learn reward functions from human demonstrations and action evaluations [42, 43] and reinforcement learning robust to noise in rewards [44, 45].

Other methods have been proposed for capturing one optimal action even when multiple optimal actions exist. Trust region policy optimization (TRPO) can seek optimal actions by conservatively updating policies [46]. In this method, a conservative policy update retards policy updates due to a characteristic of learning stabilization. A deep deterministic policy gradient (DDPG) also focuses on choosing one optimal action using a deterministic policy model [47]. These methods are implicit mode-seeking policy search methods. For them, it is commonly necessary to manually design many parameters in such neural network models as network structure and hyperparameters for each task.

We propose a mode-seeking SGP-PS, which effectively solves the issue of the multimodality of optimal actions. Using student-t distribution as a likelihood, a mode-seeking SGP-PS captures one optimal action. A non-parametric-based policy model enables efficient policy learning with few task-specific hyperparameters even with a small number of data samples.

3.3. Proposed Method

In this section, we derive two different GP-PS algorithms: a multimodal SGP-PS and a mode-seeking SGP-PS. Each method is derived by extending a sparse Gaussian process policy search (SGP-PS), which is a policy search that uses SGP as a policy prior.

3.3.1. Multimodal sparse Gaussian process policy search (multimodal SGP-PS)

We derive the multimodal SGP-PS based on SGP-PS. To capture multiple optimal actions, we consider the following product of M different GPs as a prior of function f of the control policy:

$$f \sim \alpha \prod_{m=1}^{M} \mathcal{GP}^{(m)}(\mathbf{0}, \mathrm{k}^{(m)}(\mathbf{s}, \mathbf{s}')). \tag{3.1}$$

We assume a multimodality control policy that uses M GPs and expresses the lower bound of the return weighted likelihood product of the M lower bounds:

$$\log J_L(\boldsymbol{\theta}_M, q) \approx \prod_{m=1}^{M} \int p(\mathbf{f}^{(m)} \mid \bar{\mathbf{f}}^{(m)}) q(\bar{\mathbf{f}}^{(m)}) \log \frac{p(\tilde{\mathbf{a}} \mid \mathbf{f}^{(m)}) p(\bar{\mathbf{f}}^{(m)})}{q(\bar{\mathbf{f}}^{(m)})} \mathrm{d}\mathbf{f}^{(m)} \mathrm{d}\bar{\mathbf{f}}^{(m)}, \tag{3.2}$$

where $\bar{\mathbf{f}}^{(m)}$ and $\mathbf{f}^{(m)}$ indicate the pseudo outputs and the function outputs of the m-th SGP. Optimizing the above lower bound w.r.t. policy parameters is, however, insufficient to search for multimodal behaviors; all the M GPs tend to converge to the same solution since all the training data are commonly shared to train all the GPs. To allow exploration of different solutions for all the GPs, we introduce another latent variable, the so-called binary indicator matrix $\mathbf{Z}$ whose

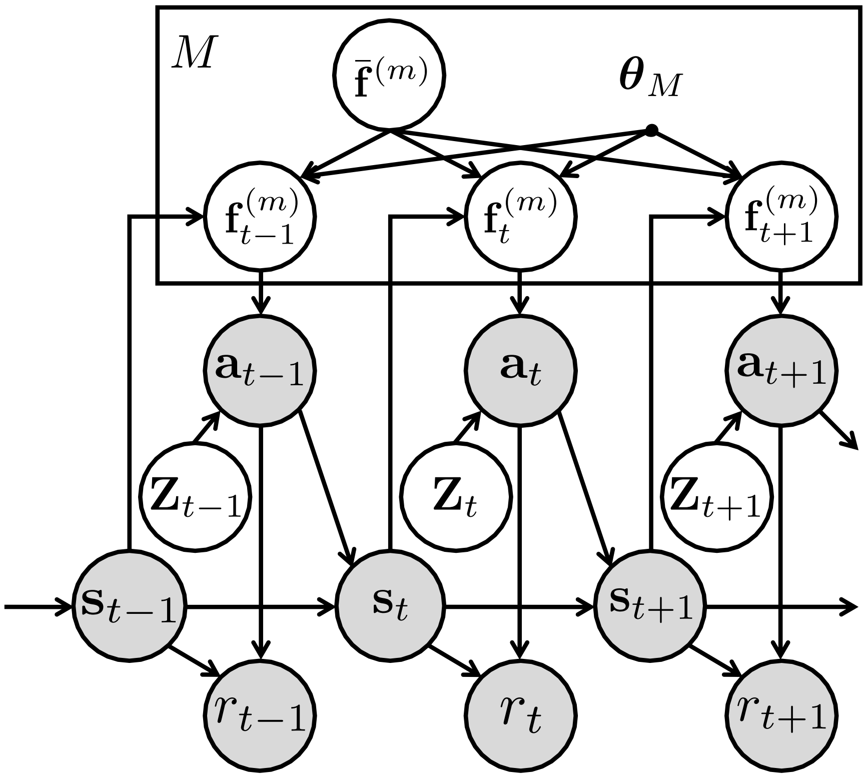


Figure 3.3.: Graphical model of multimodal SGP-PS

nm element is a binary variable that is associated with m-th GP [21]. Each row in $\mathbf{Z}$ has one non-zero entry. We assume the prior on binary indicator matrix $\mathbf{Z}$:

$$p(\mathbf{Z}) = \prod_{n=1,m=1}^{N,M} [\mathbf{\Pi}]_{nm}^{[\mathbf{Z}]_{nm}}, \tag{3.3}$$

where $\mathbf{\Pi}$ is the $N \times M$ probability matrix and $[\mathbf{\Pi}]_{nm}$ indicates the probability that a_n is generated by the m-th GP. Moreover, we extend the return weighted likelihood function:

$$p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}) = \int p(\mathbf{Z}) p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}, \mathbf{Z}) \mathrm{d}\mathbf{Z}, \tag{3.4}$$

$$p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}, \mathbf{Z}) = \prod_{n=1,m=1}^{N,M} \mathcal{N}(\tilde{\mathbf{a}}_n \mid \mathbf{W}_{nn}\mathbf{f}, \sigma^2)^{[\mathbf{Z}]_{nm}}. \tag{3.5}$$

Algorithm 3.1: Multimodal SGP-PS algorithm

```
   Initialize: $\boldsymbol{\theta}_M$, $\bar{\mathrm{s}}$
1  while reward not converged do
2      # Sample $E$ trajectories
3      for $e = 1 : E$ do
4          Sample trajectory $d$ using policy $\pi$
5      end
6      # Policy Improvement
7      while $\log J'_L$ is not converged do
8          # E step
9          while $\log J'_L$ is not converged do
10             Update $q(\{\bar{\mathbf{f}}^{(m)}\})$ with Eq. (A.2)
11             Update $q(\mathbf{Z})$ with Eq. (A.5)
12         end
13         # M step
14         $\boldsymbol{\theta}_M \leftarrow \arg\max_{\boldsymbol{\theta}_M} \log J'_L$
15     end
16 end
```

Due to the analytical intractability of the integral of $\mathbf{Z}$, we apply Jensen's inequality:

$$\log J'_L(\boldsymbol{\theta}_M, q) = \int p(\{\mathbf{f}^{(m)}\} \mid \{\bar{\mathbf{f}}^{(m)}\}) q(\{\bar{\mathbf{f}}^{(m)}\}) q(\mathbf{Z}) \cdot \\ \log \frac{p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}, \mathbf{Z}) p(\{\bar{\mathbf{f}}^{(m)}\}) p(\mathbf{Z})}{q(\{\bar{\mathbf{f}}^{(m)}\}) q(\mathbf{Z})} \mathrm{d}\{\mathbf{f}^{(m)}\} \mathrm{d}\{\bar{\mathbf{f}}^{(m)}\} \mathrm{d}\mathbf{Z}, \tag{3.6}$$

where $q(\{\bar{\mathbf{f}}^{(m)}\}) = \prod_{m=1}^{M} q(\bar{\mathbf{f}}^{(m)})$, $p(\{\bar{\mathbf{f}}^{(m)}\}) = \prod_{m=1}^{M} p(\bar{\mathbf{f}}^{(m)})$ and $p(\{\mathbf{f}^{(m)}\} \mid \{\bar{\mathbf{f}}^{(m)}\}) = \prod_{m=1}^{M} p(\mathbf{f}^{(m)} \mid \bar{\mathbf{f}}^{(m)})$. The graphical model of multimodal SGP-PS is shown in Fig. 3.3.

Finally, we derive an EM-like iterative optimization scheme. We repeat the E-step that finds optimal variational distribution $q(\{\bar{\mathbf{f}}^{(m)}\})$ and $q(\mathbf{Z})$ and the M-step that finds optimal parameter $\boldsymbol{\theta}_M$ by alternatively maximizing $\log J'_L$. We describe the derivation of the analytical update laws of the variational distributions and

Table 3.1.: Computational complexity of each optimization in multimodal SGP-PS: N, M, and L are amounts of training data, components, and pseudo inputs.

	$q(\{\bar{\mathbf{f}}^{(m)}\})$	$q(\mathbf{Z})$	$\log J'_L$
Multimodal SGP-PS	$\mathcal{O}(MNL^2)$	$\mathcal{O}(MNL^2)$	$\mathcal{O}(MNL^2)$

the analytical form of the objective function in appendix A.1. A summary of the multimodal SGP-PS is given in Algorithm 3.1. Table 3.1 shows the computational complexity of each optimization, and the complexity of $q(\{\bar{\mathbf{f}}^{(m)}\})$ and $\log J'_L$ is reduced due to the pseudo inputs. Although the computational complexity of $q(\mathbf{Z})$ is increased, the overall complexity is reduced.

We analytically obtain the M predictive distribution using variational distribution $q(\{\bar{\mathbf{f}}^{(m)}\})$ and parameter $\boldsymbol{\theta}_M$ variational learning. The m-th predictive distribution of action $a_*^{(m)}$ corresponds to new state $\mathbf{s}_*$:

$$p(a_*^{(m)} \mid \mathbf{s}_*) \approx \int p(a_*^{(m)} \mid f_*) p(f_* \mid \bar{\mathbf{f}}^{(m)}, \mathbf{s}_*) q(\bar{\mathbf{f}}^{(m)}) \mathrm{d}f_* \mathrm{d}\bar{\mathbf{f}}^{(m)}. \tag{3.7}$$

The analytical solution of the predictive distribution of the multimodal SGP policy search is described in A.2.

We employ a softmax function that uses the negative variance of the predictive distribution:

$$p(m) = \frac{\exp(-\sigma_*^{(m)}/\beta)}{\sum_{m=1}^{M} \exp(-\sigma_*^{(m)}/\beta)}, \tag{3.8}$$

where $p(m)$ is the probability of using the m-th predictive distribution and β is a temperature parameter that controls the trade-off between exploration and exploitation.

3.3.2. Mode-seeking sparse Gaussian process policy search (mode-seeking SGP-PS)

We derive a mode-seeking sparse Gaussian process policy search based on SGP-PS. Mode-seeking means that the policy can capture an optimal action at each state by ignoring other optimal actions. To capture one optimal action, the mode-seeking SGP-PS employs student-t distribution as a likelihood and estimates the reliability of each data to ignore low-reliability data. The following is the probabilistic density function (PDF) of the student-t distribution:

$$\mathrm{St}(x \mid \mu, a, b) = \frac{b^a}{\Gamma(a)\sqrt{2\pi}} \left[b + \frac{(x-\mu)^2}{2} \right]^{-a-1/2} \Gamma(a + 1/2), \tag{3.9}$$

where $\Gamma(\cdot)$ is a gamma function. Although the PDF is complicated, it can be described more simply by a scale mixture representation that uses Gaussian and gamma distributions:

$$\mathrm{St}(x \mid \mu, a, b) = \int_0^\infty \mathcal{N}(x \mid \mu, \tau^{-1}) \mathrm{Gam}(\tau \mid a, b) \mathrm{d}\tau, \tag{3.10}$$

$$\mathrm{Gam}(\tau \mid a, b) = \frac{b^a}{\Gamma(a)} \tau^{a-1} \exp(-b\tau), \tag{3.11}$$

where τ is the precision of the Gaussian distribution and indicates the data reliability. We use gamma distribution for the precision τ of the Gaussian distribution.

To learn a policy that captures one optimal action, we use the student-t distribution as the likelihood:

$$\begin{aligned} p(\tilde{\mathbf{a}} \mid \mathbf{f}) &= \mathrm{St}(\tilde{\mathbf{a}} \mid \mathbf{f}) \\ &= \int p(\tilde{\mathbf{a}} \mid \mathbf{f}, \mathbf{T}) p(\mathbf{T}) \mathrm{d}\mathbf{T}, \end{aligned} \tag{3.12}$$

$$p(\tilde{\mathbf{a}} \mid \mathbf{f}, \mathbf{T}) = \prod_{n=1}^{N} \mathcal{N}(\tilde{\mathbf{a}}_n \mid \mathbf{f}_n, \tau_n^{-1}), \tag{3.13}$$

$$p(\mathbf{T}) = \prod_{n=1}^{N} \mathrm{Gam}(\tau_n \mid a, b), \tag{3.14}$$

where $\mathbf{T} = \mathrm{diag}\{\tau_1, \cdots, \tau_N\}$. The integration w.r.t $\mathbf{f}$ in Eq. (2.10) cannot be solved analytically since the likelihood is complicated. By introducing a scale mixture representation and variational distribution $q(\mathbf{T})$ of the data reliability, we derive a new lower bound for the mode-seeking SGP-PS:

$$\log J_L'(\boldsymbol{\theta}_R, q) = \int p(\mathbf{f} \mid \bar{\mathbf{f}}) q(\bar{\mathbf{f}}) q(\mathbf{T}) \log \frac{p(\tilde{\mathbf{a}} \mid \mathbf{f}, \mathbf{T}) p(\mathbf{T}) p(\bar{\mathbf{f}})}{q(\bar{\mathbf{f}}) q(\mathbf{T})} \mathrm{d}\mathbf{f} \mathrm{d}\bar{\mathbf{f}} \mathrm{d}\mathbf{T}. \tag{3.15}$$

The graphical model of mode-seeking SGP-PS is shown in Fig. 3.4.

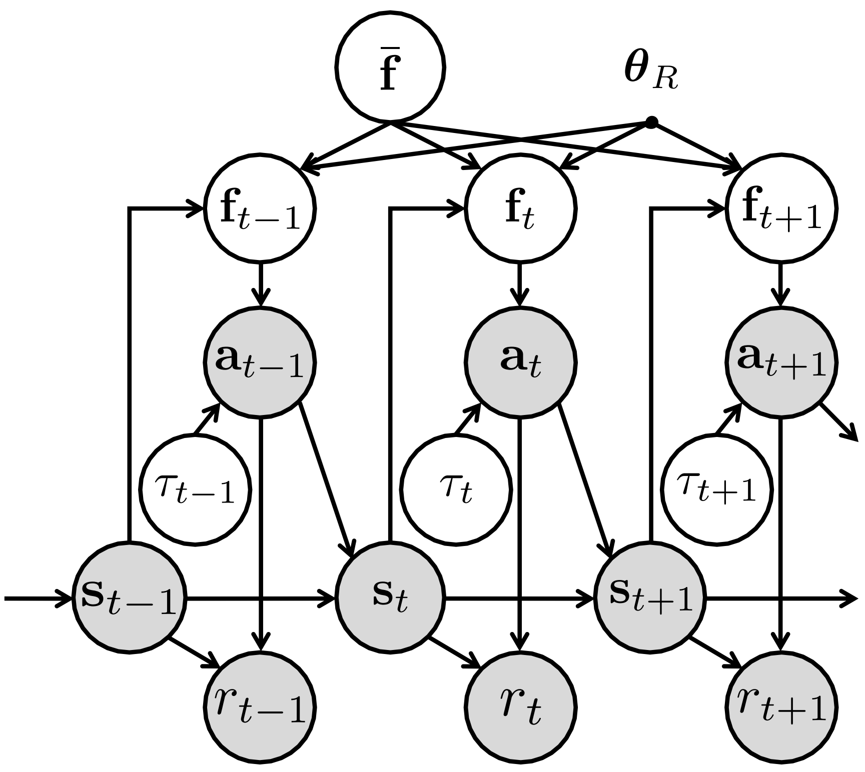


Figure 3.4.: Graphical model of mode-seeking SGP-PS

We assume that $q(\mathbf{T}) = \prod_{n=1}^{N} q(\tau_n)$. We derive the update laws of $q(\bar{\mathbf{f}})$ and $q(\tau_n)$ which maximize the lower bound of the expected return based on variational Bayesian inference. The analytical solution of the update laws and the lower bound are described in appendix B.1. Alternately updating the variational distribution and the hyperparameters alternately, the mode-seeking SGP policy is learned. This method's algorithm is described in Algorithm 3.2. Table 3.2 shows the computational complexity of each optimization, and the complexity of $q(\{\bar{\mathbf{f}}^{(m)}\})$ and $\log J'_L$ is reduced by the pseudo inputs. Although the computation amount of $q(\mathbf{T})$ increased, the overall complexity was reduced.

We analytically obtained the predictive distribution of the mode-seeking SGP policy for new state $\mathbf{s}_*$ using variational distribution $q(\bar{\mathbf{f}})$, $q(\mathbf{T})$, and parameters $\boldsymbol{\theta}_R$, which are obtained in the learning:

$$p(a_* \mid \mathbf{s}_*) \approx \int p(a_* \mid f_*, \tau) p(f_* \mid \bar{\mathbf{f}}, \mathbf{s}_*) q(\bar{\mathbf{f}}) q(\tau) \mathrm{d}f_* \mathrm{d}\bar{\mathbf{f}} \mathrm{d}\tau. \tag{3.16}$$

The analytical solution of the predictive distribution of mode-seeking SGP-PS

Algorithm 3.2: Mode-seeking SGP-PS algorithm

```
  Initialize: $\boldsymbol{\theta}_R$, $\bar{\mathbf{s}}$
1 while reward not converged do
2     # Sample $E$ trajectories
3     for $e = 1 : E$ do
4         Sample trajectory $d$ using policy $\pi$
5     end
6     # Policy Improvement
7     while $\log J'_L$ is not converged do
8         # E step
9         while $\log J'_L$ is not converged do
10            Update $q(\bar{\mathbf{f}})$ with Eq. (B.11)
11            Update $q(\mathbf{T})$ with Eq. (B.10)
12        end
13        # M step
14        $\boldsymbol{\theta}_R \leftarrow \arg\max_{\boldsymbol{\theta}_R} \log J'_L$
15    end
16 end
```

is described in appendix B.2.

3.4. Experiments

We conducted experiments with two robot control tasks in simulations: hand-posture adjustment task and table-sweeping task. We investigated the effectiveness of our proposed methods in the hand-posture adjustment task and demon-

Table 3.2.: Computational complexity of each optimization in mode-seeking SGP-PS: N and L are amounts of training data and pseudo inputs.

	$q(\bar{\mathbf{f}}^{(m)})$	$q(\mathbf{T})$	$\log J'_L$
Mode-seeking SGP-PS	$\mathcal{O}(NL^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$

strated scalability for more challenging situations in the table-sweeping task.

3.4.1. Hand-posture adjustment task

Settings

We experimented with the simple robot task shown in Fig. 3.1 (a) to confirm the performance of our proposed methods on the task with the multiple optimal actions. The task's environment consisted of a UR5 robot arm and a red, $7 \times 12 \times 10$ cm^3 cube. The robot arm has a two-fingered gripper and can rotate its wrist in one revolution. The task aims to grasp the red cube by its longer side; due to its physical constraint, the robot cannot grasp its shorter side. The robot can grasp the object in two ways: by rotating its wrist left or right. So, the task has two optimal actions.

The state is the orientation of the red cube. The action is the rotation angle of the robot wrist between $-\pi$ rad to π rad. After determining the action, the robot moves to grasp the object with its wrist angle decided as an action. Since each episode is terminated in one step, the length of the trajectory is $T = 1$. Return function $R(d)$ is binary. $R(d) = 100$ if the robot grasps the cube, otherwise $R(d) = 0$.

We confirm the performances of our methods to the multiple optimal actions by comparing them with a unimodal SGP-PS that employs a standard SGP as a policy model. We set the number of components of the multimodal SGP-PS to two and three to confirm its performance with different numbers of components to the number of multiple optimal actions. We used an isotropic-squared exponential kernel in each method. In this experiment, each method explored $E = 100$ episodes and learned a policy using the data of 100 episodes and the best 80 episodes as sample reuse after exploration. The number of pseudo-inputs of each method was set: $L = 20$.

Results

Figure 3.5 shows the mean and the standard deviation of the return over ten experiments using multimodal SGP-PS with two and three components, mode-seeking SGP-PS, and unimodal SGP-PS. The policies learned by our proposed

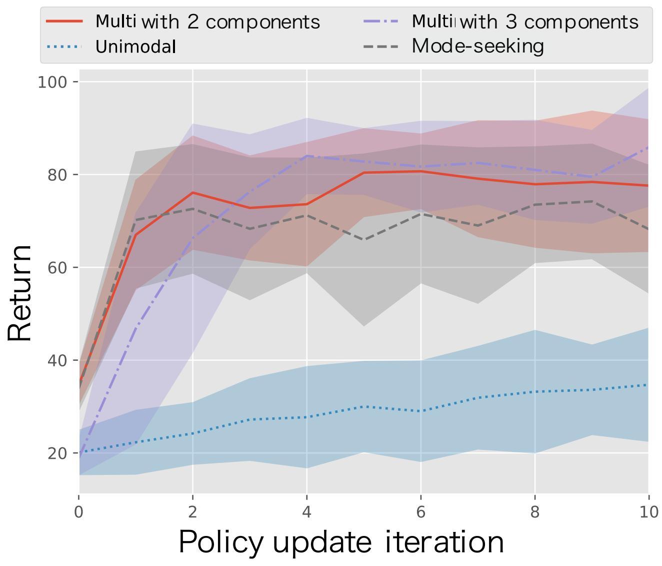


Figure 3.5.: Learning performance of policy search for hand-posture adjustment task over ten experiments: Mean value and standard deviations are shown.

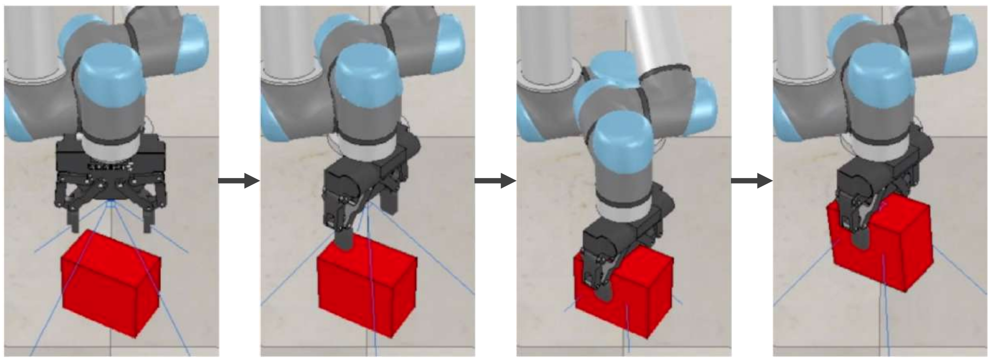


Figure 3.6.: Robot behavior controlled with a policy learned by multimodal SGP-PS in hand-posture adjustment task

methods indicate higher performance than those obtained by unimodal SGP-PS. This result suggests that the expansions of a policy model in our proposed methods are valid for the task with multiple optimal actions. Moreover, multimodal SGP-PS can learn an appropriate policy even when the number of components is redundant. Fig. 3.6 shows the robot behavior using a learned multimodal SGP policy which, can select an appropriate action for the object's angle. Fig. 3.7 (a) shows a learned policy by multimodal SGP-PS with two components. The

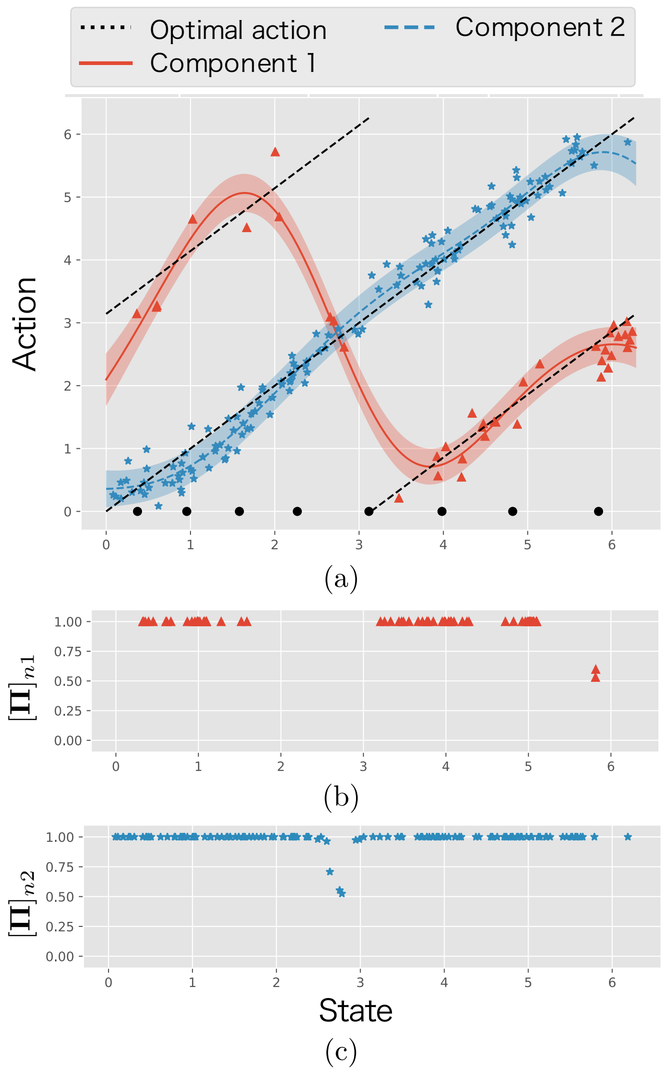


Figure 3.7.: (a) Learned policy for hand-posture adjustment task by multimodal SGP-PS with two components. Black dots indicate pseudo input. (b) and (c) show posterior probability of the data association of each data point shown in (a).

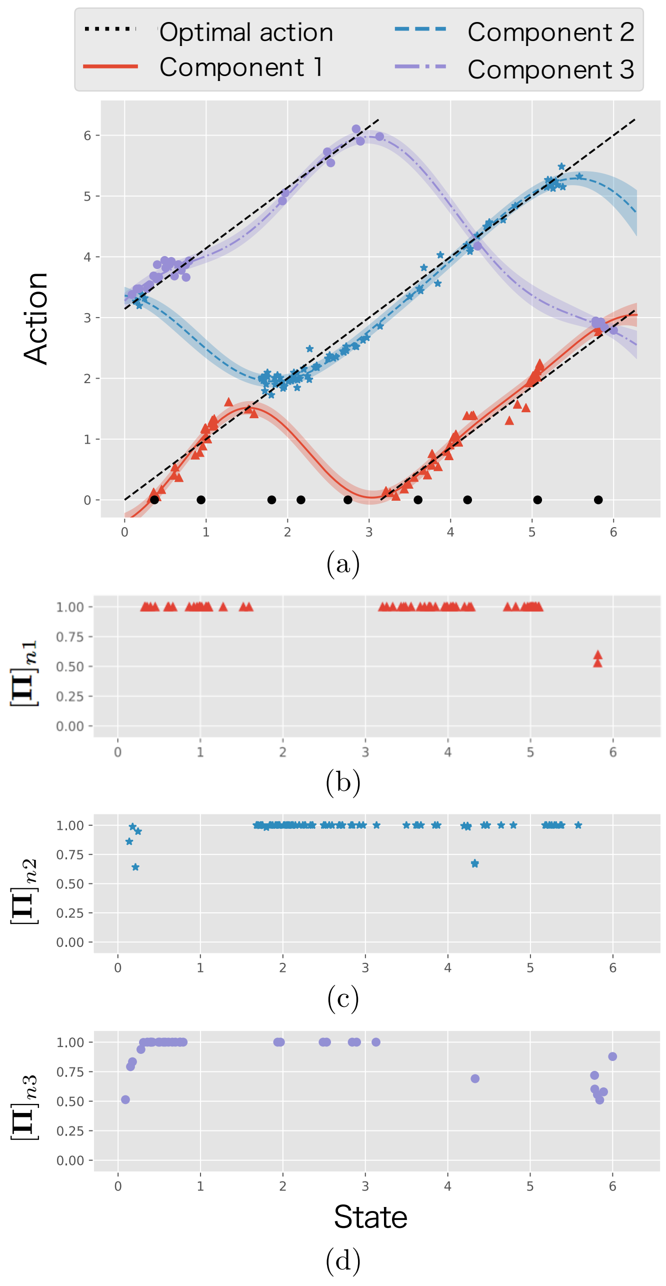
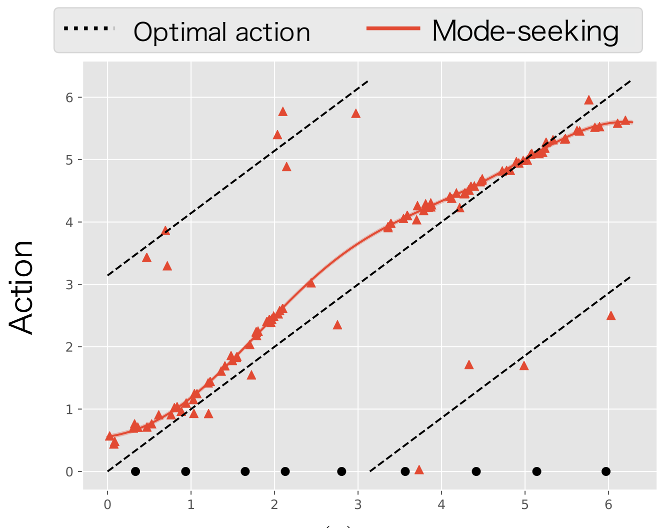
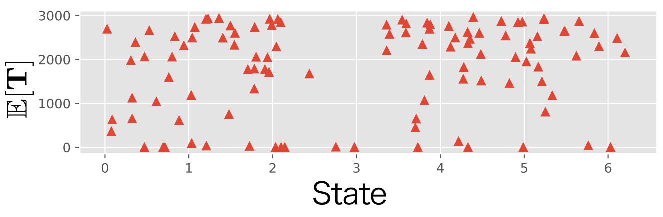


Figure 3.8.: (a) Learned policy for hand-posture adjustment task by multimodal SGP-PS with three components. Black dots indicate pseudo input. (b), (c), and (d) show posterior probability of data association of each data point in (a).



(a)



(b)

Figure 3.9.: (a) Learned policy for hand-posture adjustment task by mode-seeking SGP-PS. Black dots indicate pseudo input. (b) the posterior probability of data association.

learned multimodal SGP policy can capture multiple optimal actions indicated by the black dotted line by properly estimating data associations. To capture two optimal actions, component 1 captures two dotted lines, and component 2 captures the middle line. Figs. 3.7 (b) and 3.7 (c) show the optimized posterior distribution of **Z**, which is the data points assigned to each component. In the crossing point of the two components, the posterior of the data association indicates low probability. Fig. 3.8 (a) shows a learned policy by multimodal SGP-PS with three components. Each component of the learned policy captures one of three dotted lines. Figs. 3.8 (b), 3.8 (c), and 3.8 (d) show the optimized posterior distribution of **Z**. Similar to the case of the multimodal SGP-PS with two components, the posterior of the data association indicates a low probability of the two components' crossing points. Fig. 3.9 (a) shows a learned policy by

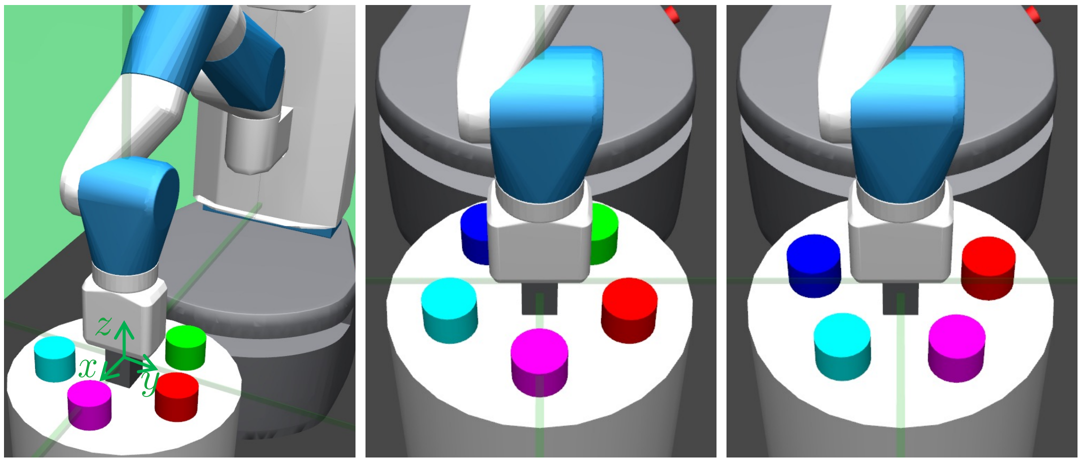


Figure 3.10.: Table-sweeping task environment with five objects. (a) is an overview of the table-sweeping task. Green vectors show axes of the Cartesian coordinate to represent a position of the end-effector and objects. (b) and (c) indicate two different initial positions of five objects.

mode-seeking SGP-PS. The policy acquired by it captured one of the optimal actions at each state, coincidentally in a similar way as with two components in Fig. 3.7 (a). Fig. 3.9 (b) shows the posterior distribution of reliability $\mathbf{T}$. The mode-seeking SGP-PS learned a unimodal policy by estimating the reliability of the data and ignoring low-reliability data.

In summary, the simulation results in the hand-posture adjustment task suggest the effectiveness of our proposed methods for learning a policy in tasks with multiple optimal actions.

3.4.2. Table-sweeping task

Settings

The aim of the table-sweeping task is to have the fetch robot sweep the five objects on the table individually by robot's end-effector in the environment shown in Fig. 3.10. In this task, we learned an individual action policy based on the number of objects by exploiting the fact that this task can be naturally decomposed into subtasks according to the number of objects. Thus, five individual policies are learned. We constructed the table-sweeping task environment based on the fetch environment proposed in [48] and implemented it as a learning environment in OpenAI Gym. The table-sweeping task is more challenging than the hand-posture adjustment task for the following reasons:

1. The number of optimal actions is up to five.
2. Multiple steps are required to accomplish the task.
3. The state space has a higher dimension.

The task environment consists of the fetch robotic arm, a 40-cm diameter table, and five cylindrical objects with diameters of 7 cm, and heights of 2.5 cm. The state consists of the two-dimensional Cartesian position of the end-effector and five objects, and the two-dimensional Cartesian relative position of the five objects related to the position of the end-effector. Each axis of the Cartesian coordinate is shown in Fig. 3.10 (a). Therefore, a state is represented by a 22-dimensional vector consisting of positions of the end-effector and five objects and relative positions of five objects. The definition of a state is relied on [48]. We set that the position and the relative position in a state corresponding to swept objects are assigned 0. An action is a two-dimensional vector and each dimension specifies the desired end-effector movement in each axis of two-dimensional Cartesian coordinates on the table. The absolute value of each dimension of action is limited to 4 cm or less so that even an optimal policy takes multiple steps to achieve the task. Return function $R(d)$ rewards the trajectory data with 10 when an object is swept and punishes $0.1 \times T$ for the length of the episode T.

We set that the end-effector's position is initialized to the center of the table after sweeping an object. The positions of the five objects are initialized by uniformly sampling from two patterns shown in Fig. 3.10. The five objects are arranged at 72-degree intervals on a circle with a diameter of 12 cm around the z-axis, but there is 32-degree difference between the two patterns. The task is terminated when an object is swept or when the length of the trajectory has reached to $T = 20$. After the robot swept an object, the end-effector moves to the center of the table to execute the next task.

To compare our methods, we employed unimodal SGP-PS, SAC [32], and TRPO [46]. SAC and TRPO are state-of-the-art reinforcement learning methods that learn neural-network policies for continuous action space. The hyperparameters of SAC and TRPO are described in Tables C.1 and C.2.

We used a squared exponential kernel with automatic relevance determination in the SGP-based methods. The number of pseudo-inputs of each method is set to $L = 20$. We set the number of objects on the table as the number of components of the multimodal SGP-PS. For a fair comparison, the SGP-based methods and the NN-based methods as TRPO and SAC commonly update their policies using every 1000 steps worth of exploration episodes.

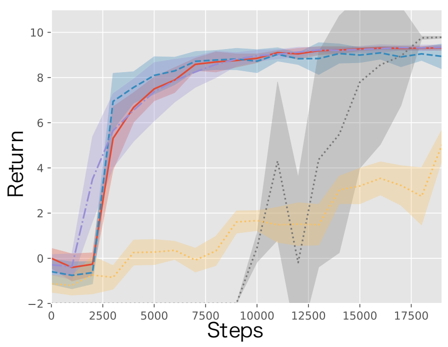
Results

Figure 3.11 shows the mean and standard deviation of the total reward over ten experiments using multimodal SGP-PS, mode-seeking SGP-PS, unimodal SGP-PS, SAC, and TRPO. The learning experiment was conducted individually for each number of objects. In the task with one object, the performance of unimodal SGP-PS is comparable multimodal SGP-PS and mode-seeking SGP-PS. However, as the number of objects increases, multimodal SGP-PS and mode-seeking SGP-PS outperform unimodal SGP-PS by effectively handling multiple optimal actions.

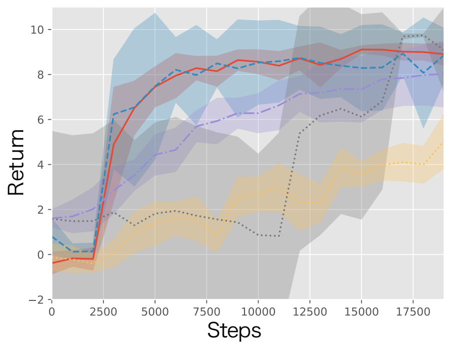
In the SAC algorithm, although the value function captured multiple optimal actions, the policy employs a unimodal model. Then it needs to seek one of the optimal actions among multiple ones. The policy's performance may become unstable because mode-seeking is not stable at the beginning of learning. Even though TRPO improves the policy monotonically; however, the performance is inadequate. It needs more data to learn a suitable policy in tasks with multiple optimal actions may be due to neural-network-based policies with many parameters. This result resembles one that was previously reported [6].

Fig. 3.12 shows the behavior of a policy learned by the multimodal SGP-PS. For each object, our methods can learn a policy to sweep an object by deciding action multiple steps.

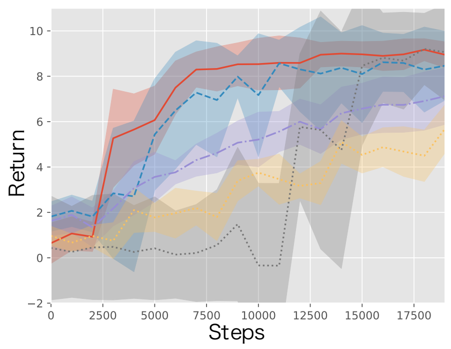
Fig. 3.13 shows five trajectories generated by the policy learned by multimodal SGP-PS, mode-seeking SGP-PS, and unimodal SGP-PS on the task with



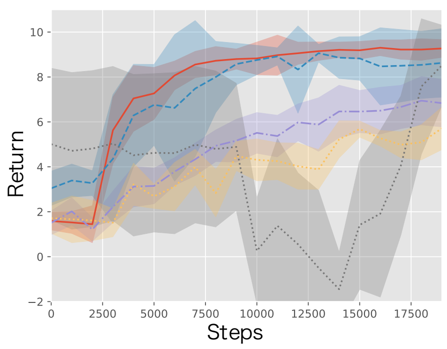
(a) One object



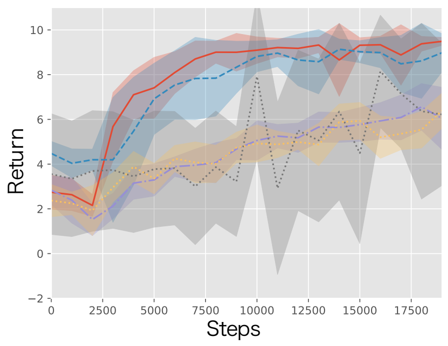
(b) Two objects



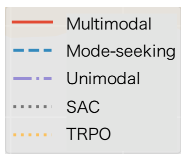
(c) Three objects



(d) Four objects

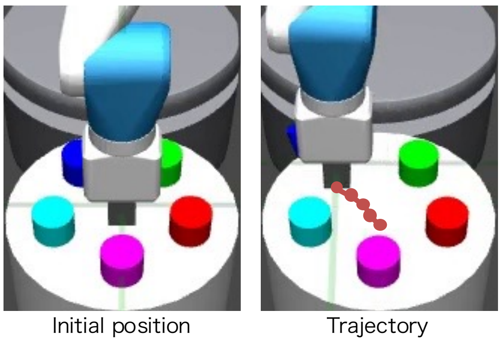


(e) Five objects

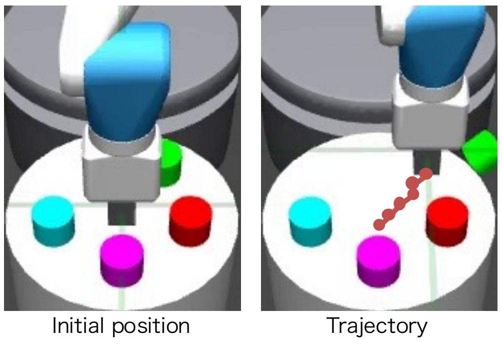


(f) Legends of each figure

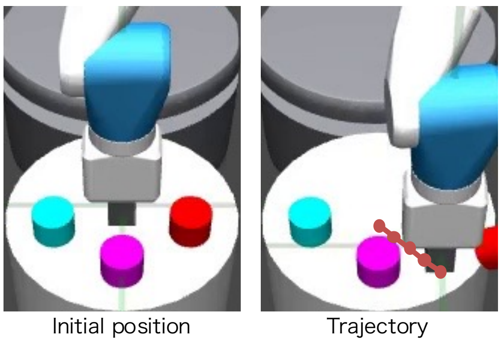
Figure 3.11.: Learning performances of multimodal SGP-PS, mode-seeking SGP-PS, unimodal SGP-PS, SAC, TRPO in table-sweeping task with one to five objects: Each curve is averaged over ten experiments.



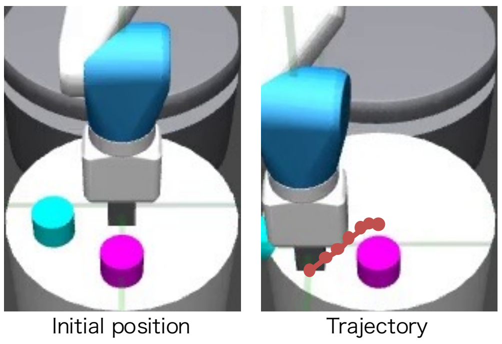
(a) Five objects



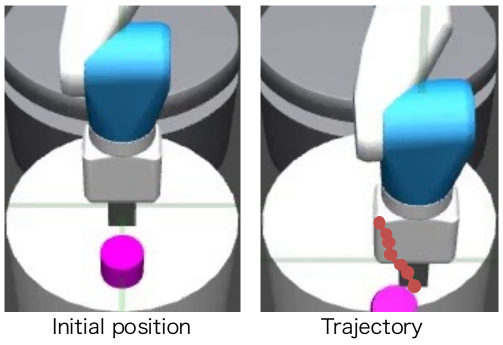
(b) Four objects



(c) Three objects

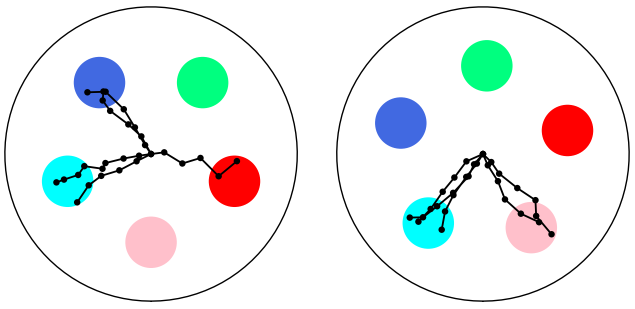


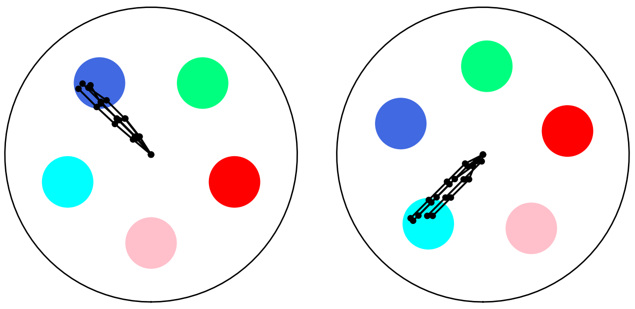
(d) Two objects



(e) One object

Figure 3.12.: Behaviors of policies learned by multimodal SGP-PS in table-sweeping task


(a) Multimodal SGP policy


(b) Mode-seeking SGP policy

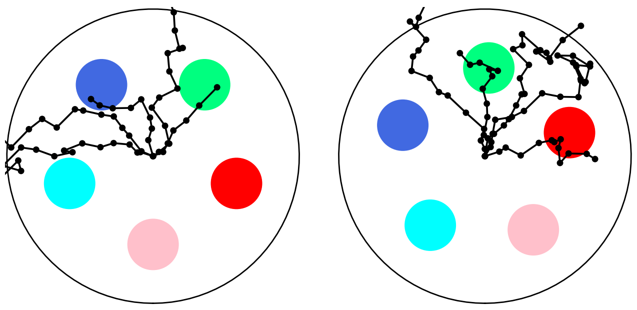

(c) Unimodal SGP policy

Figure 3.13.: Five trajectories of the end-effector by learned policy by Multimodal SGP-PS, Mode-seeking SGP-PS, and Unimodal SGP-PS. The left and right figures have different initial positions of five objects. The colored circles indicate the initial positions of five objects on the table. Black lines indicate trajectories by the learned policy.

five objects. The multimodal SGP policy learned the multiple optimal actions for sweeping an object in each initial state by the nature of multimodality. However, it could learn two or three optimal actions out of five due to limited data explored. The mode-seeking SGP policy learned one of the optimal actions by ignoring others successfully, while the unimodal SGP policy could not learn any optimal actions properly. We confirmed that each learned policy obtained different optimal actions according to the initial positions of objects.

In summary, all of our simulation results in the table-sweep task suggest the effectiveness of our methods for learning policies with multiple optimal actions. We confirmed that our methods more effectively learn a policy than SAC and TRPO in a task environment with a small number of samples and multiple optimal actions.

3.5. Summary of Chapter 3

We proposed a multimodal SGP-PS and a mode-seeking SGP-PS, which are GP-PS methods for a task with multiple optimal actions. The multimodal SGP-PS employs a multimodal policy prior inspired by OMGP to learn a policy that can capture multiple optimal actions. The mode-seeking SGP-PS learns a unimodal policy that captures one optimal action by employing an outlier-robust likelihood function. We derived the updating laws of both methods based on variational Bayesian inference.

To investigate the performance of our methods, we conducted two manipulation tasks: 1) a hand-posture adjustment task and 2) a table-sweeping task. We confirmed that our methods can learn suitable policies in an environment with multiple optimal actions.

4. Gaussian Process Self-triggered Policy Search for Weakly Observable Environment

4.1. Introduction

GP-PS is difficult for automating real-world tasks such as heavy industrial machinery in actual workplaces. Their control systems still have to rely on complicated system-specific models created by humans [49]. In particular, two major challenges must be tackled in GP-PS for industrial machines: 1) obtaining trial and error data by a large and heavy industrial machine is very costly, and 2) since the environments of industrial machines are *weakly observable*, little information about the state is contained in observations due to technical difficulties or maintenance costs.

More specifically, a waste crane, remotely controlled by a skilled operator at a waste incineration plant, does not have sensors that collect information about the garbage's state, although its characteristics have diversity and changing trends that depend on the day of the week and the season (size and hardness of each element, moisture content, viscosity, etc.) [50–53]. The crane is only equipped with a sensor that weighs the garbage grasped by the bucket. Although the operators can roughly see the whole garbage pit from the control room, due to severe occlusion they generally cannot see the garbage around the bucket, especially when it lands the waste and executes a grasping motion.

With such scant information about the environmental state, skilled operators

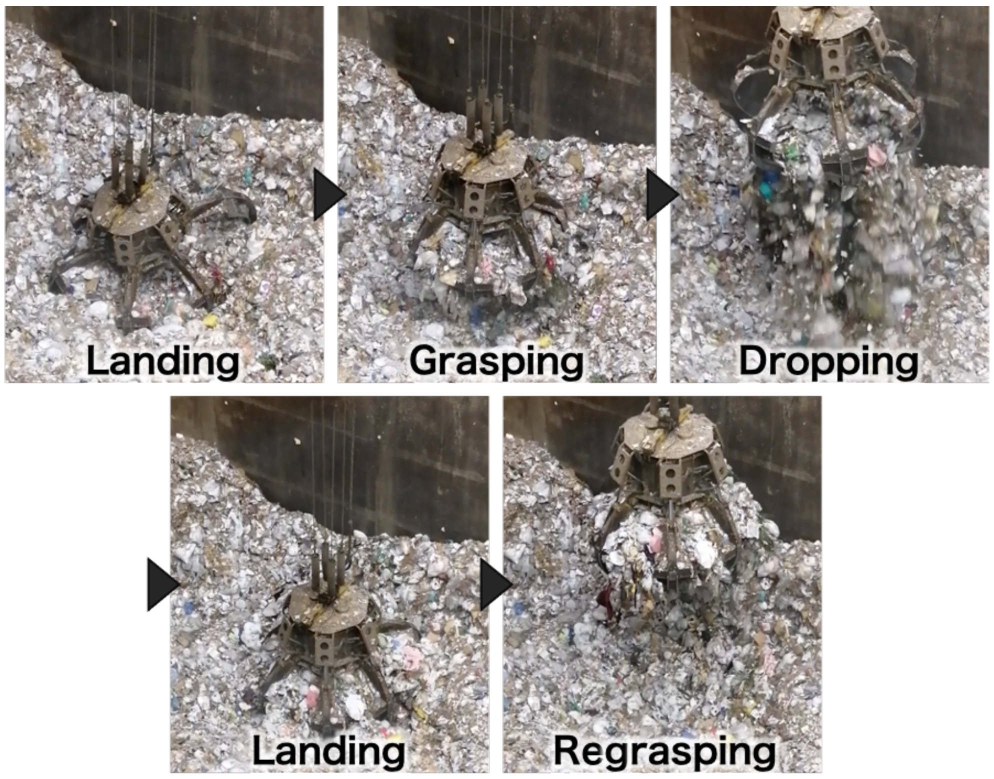


Figure 4.1.: Garbage grasping by an actual waste crane by a human operator

can control such machines and achieve high performance. Their behavior seems different from general sensory feedback control that selects an action based on sensor values at regular time intervals (Fig. 4.1), which is a typical policy model in policy search and reinforcement learning (e.g., [7,11]). This situation can be attributed to the weakness of the observations; if the value of the weight sensor does not change significantly, the operator will not be able to select a different action. Instead, selecting a predetermined control strategy (e.g., grasping or scattering) and a duration based on sensor values creates robust behavior and stabilizes it even with weak observations. Such an approach can reduce the dependency of the policy on sensor values and the frequency of references. However, these policies need to be adjusted according to the garbage's characteristics. Such thoughts motivated us to explore a novel policy search framework with a specific type of policy applicable in weakly observable environments in a sample efficient way.

In this chapter, we propose GP self-triggered policy search (GPSTPS), which

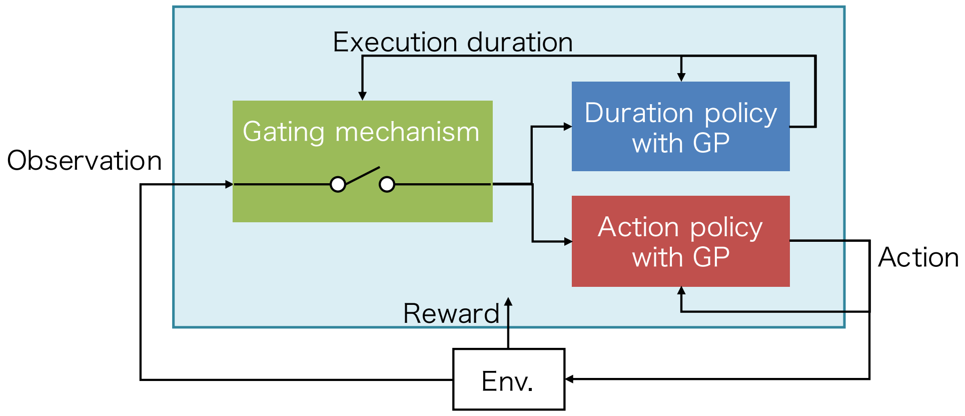


Figure 4.2.: Self-triggered policy search with GP policy model

is a novel GP-PS algorithm (Fig. 4.2). GPSTPS has two types of control policies: action and duration. The gating mechanism either maintains the action selected by the action policy for the duration specified by the duration policy or updates the action and duration by passing new observations to the policy; therefore, it is described as *self-triggered*. GPSTPS simultaneously learns both policies by trial and error based on sparse GP priors [12] and variational learning [7] to maximize the return. We experimentally verified the performance of our proposed method on garbage-grasping-scattering task with a waste crane with weak observations using a simulation and a robotic waste crane system. Our experimental results suggest that our proposed method acquired suitable policies to determine the action and duration based on the garbage's characteristics.

The following are the contributions of this chapter:

1. Our GPSTPS formulation incorporates self-triggered control into a GP-PS.
2. It derives a learning algorithm for GPSTPS based on variational learning.
3. Experimental verification used simulations and a robotic waste crane.

4.2. Related Work

4.2.1. Crane and Bucket Automation

Crane automation is conventionally implemented with a complicated model of a crane system [49]. Robust control over disturbances has also been explored by modeling such as wind [54] and waves [55, 56]. For such difficult-to-model objects as soil, previous work designed robust movements for an excavator [57, 58]. In a data-driven approach, learning methods were proposed that are predictive models for excavation performance [59,60] and imitation learning methods for wheel loaders [61]. Such works designed elaborate models and learned models with much sensor information; however, such approaches may not be suitable in environments that have weak observations like waste cranes. In this paper, we propose a data-driven method for policy learning in weakly observable environments.

4.2.2. Self-triggered Control

Self-triggered control is known as a method of controlling a system while reducing communication costs by determining the duration until the next communication in the networked control system in which controllers, sensors, and actuators are connected by a network [62, 63]. Previous works proposed self-triggered control methods for linear systems [64–66]. Self-triggered control has been applied to a multi-agent system [67] and a quadcopter [68]. A data-driven method that combines model-based reinforcement learning into self-triggered control has been proposed [69]. Self-triggered control has been widely studied based on model-based control. In this chapter, we combine self-triggered control with GP policy search to learn policies in a model-free manner in a problem setting that is different from networked control.

4.2.3. Reinforcement Learning on Semi-Markov Decision Process

The framework of reinforcement learning methods built on the semi-Markov decision process can perform long-term and complex tasks by temporal abstraction called an *option* [70]. In the option framework, a policy model consists of an option transition model, sub-policies, and an option termination model. By selecting an option from the state and executing the sub-policy associated with the option until it is terminated, complex and long-term tasks are divided and simplified. Daniel et al. [71] hypothesized options and their termination as latent variables and derived a learning algorithm based on the EM algorithm. In recent years, the option framework has made it possible to learn complex policies using a large amount of data in combination with deep reinforcement learning [72–74]. A data-efficient method for robot applications was also proposed [75]. The option framework updates options in an event-triggered manner since it assumes a fully observable environment. Our GPSTPS, on the other hand, updates actions in a self-triggered manner to manage weakly observable environments.

4.3. Gaussian Process Self-triggered Policy Search

In this section, we propose self-triggered policy search with GP as a policy model. This policy model consists of the action and duration policies and gating mechanism. The action policy selects an action strategy for the state as an action, and the duration policy determines the execution time. The gating mechanism either maintains the action selected by the action policy for the duration specified by the duration policy or updates the action and duration by passing new observations to the policy. An action and duration selection by the self-triggered policy in task execution is indicated in Alg. 4.1. We derive the policy update low based on variational policy search [7] by formulating the expected return as the return-weighted marginal likelihood.

4.3.1. Problem Formulation

Self-triggered policy search has an execution duration of τ_t, which indicates the time to continue action a_t, and binary gating variable o_t, which indicates when the action and its duration time are updated by the policy at time t. The action

Algorithm 4.1: Task execution by self-triggered policy

```
   Input          : Action policy $\pi_a$, Duration policy $\pi_\tau$
   Initialization: Binary gating variable $o_1 = 1$, Initial state $\mathbf{s}_1 \sim p(\mathbf{s}_1)$
 1 for $t = 1$ to $T$ do
 2     // Action and duration selection
 3     if $o_t == 1$ then
 4         $a_t \sim \pi_a(a_t \mid \mathbf{s}_t)$ // Update action and duration
 5         $\tau_t \sim \pi_\tau(\tau_t \mid \mathbf{s}_t)$
 6     else
 7         Reuse action $a_t = a_{t-1}$
 8         Decrement duration $\tau_t = \tau_{t-1} - 1$
 9     end
10     // Update binary gating variable
11     if $\tau_t == 1$ then
12         $o_{t+1} = 1$
13     else
14         $o_{t+1} = 0$
15     end
16     $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t)$ // Execute action
17 end
```

and duration at each time are modeled:

$$p(a_t \mid \mathbf{s}_t, o_t, a_{t-1}, \pi_a) = \begin{cases} \pi_a(a_t \mid \mathbf{s}_t) & \text{if } \ o_t = 1 \\ \delta(a_t - a_{t-1}) & \text{else} \end{cases}, \tag{4.1}$$

$$p(\tau_t \mid \mathbf{s}_t, o_t, \tau_{t-1}, \pi_\tau) = \begin{cases} \pi_\tau(\tau_t \mid \mathbf{s}_t) & \text{if } \ o_t = 1 \\ \delta(\tau_t - \tau_{t-1} + 1) & \text{else} \end{cases}. \tag{4.2}$$

These two distributions include action policy π_a and duration policy π_τ. Gating variable o_t indicates action update or maintenance, and when $o_t = 1$, the action and the duration are updated by each policy. When $o_t = 0$, the previous action is continued and the duration is decremented. Duration time τ_t indicates the remaining execution duration of action a_t. Gating variable o_t is determined by duration τ_{t-1}:

$$p(o_t \mid \tau_{t-1}) = \begin{cases} \delta(o_t - 1) & \text{if} \quad \tau_{t-1} = 1 \\ \delta(o_t) & \text{else} \end{cases}. \tag{4.3}$$

The gating variable becomes $o_t = 1$ when the duration is $\tau_{t-1} = 1$.

These distributions extend the episode's probabilities:

$$d_s = \{\mathbf{s}_1, a_1, \tau_1, o_1, \cdots, \mathbf{s}_T, a_T, \tau_T, o_T, \mathbf{s}_{T+1}\}, \tag{4.4}$$

$$\begin{aligned} p(d_s \mid \pi_a, \pi_\tau) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(a_t \mid \mathbf{s}_t, o_t, a_{t-1}, \pi_a) \cdot \\ p(\tau_t \mid \mathbf{s}_t, o_t, \tau_{t-1}, \pi_\tau) p(o_t \mid \tau_{t-1}) p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t). \end{aligned} \tag{4.5}$$

Self-triggered policy search uses the episode's probability to calculate the expected return and learns action policy π_a and duration policy π_τ:

$$J(\pi_a, \pi_\tau) = \int R(d_s) p(d_s \mid \pi_a, \pi_\tau) \mathrm{d}d, \tag{4.6}$$

$$\pi_a^*, \pi_\tau^* = \arg\max_{\pi_a, \pi_\tau} J(\pi_a, \pi_\tau). \tag{4.7}$$

4.3.2. Sparse Gaussian Process Policy Models

GPs are employed as a policy model in this chapter. The nonlinear functions of the action and duration policies are defined as f and g, and GPs are set as their priors:

$$f \sim \mathcal{GP}(m_f, \mathrm{k}(\mathbf{s}, \mathbf{s}')), \tag{4.8}$$

$$g \sim \mathcal{GP}(m_g, \mathrm{k}(\mathbf{s}, \mathbf{s}')), \tag{4.9}$$

where $\mathrm{k}(\cdot, \cdot)$ is a kernel function and m_f and m_g are mean of each GP. We assume a Gaussian distribution for the action and duration:

$$p(a_t \mid f_t) = \mathcal{N}(a_t \mid f_t, \sigma_f^2), \tag{4.10}$$

$$p(\tau_t \mid g_t) = \mathcal{N}(\tau_t \mid g_t, \sigma_g^2), \tag{4.11}$$

where $f_t = f(\mathbf{s}_t)$, $g_t = g(\mathbf{s}_t)$, σ_f^2 and σ_g^2 are the variances of each Gaussian distribution.

To reduce the computational complexity of GPs used as the prior distribution of the nonlinear function, pseudo outputs $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$ corresponding to pseudo input $\bar{\mathbf{s}}$ [12] are introduced and the prior distributions are set:

$$p(\bar{\mathbf{f}} \mid \bar{\mathbf{s}}) = \mathcal{N}(\bar{\mathbf{f}} \mid m_f \mathbf{1}, \mathbf{K}_{\bar{\mathbf{s}}}), \tag{4.12}$$

$$p(\bar{\mathbf{g}} \mid \bar{\mathbf{s}}) = \mathcal{N}(\bar{\mathbf{g}} \mid m_g \mathbf{1}, \mathbf{K}_{\bar{\mathbf{s}}}), \tag{4.13}$$

where $\mathbf{K}_{\bar{\mathbf{s}}} = \mathrm{k}(\bar{\mathbf{s}}, \bar{\mathbf{s}})$ is the kernel gram matrix of pseudo input $\bar{\mathbf{s}}$. The distributions of nonlinear function outputs f_t and g_t are represented by the following Gaussian distributions as the GP regression of the distribution of the pseudo outputs:

$$p(f_t \mid \mathbf{s}_t, \bar{\mathbf{f}}) = \mathcal{N}(f_t \mid \mathbf{k}_{\mathbf{s}_t,\bar{\mathbf{s}}} \mathbf{K}_{\bar{\mathbf{s}}}^{-1} (\bar{\mathbf{f}} - m_f) + m_f, \lambda_t), \tag{4.14}$$

$$p(g_t \mid \mathbf{s}_t, \bar{\mathbf{g}}) = \mathcal{N}(g_t \mid \mathbf{k}_{\mathbf{s}_t,\bar{\mathbf{s}}} \mathbf{K}_{\bar{\mathbf{s}}}^{-1} (\bar{\mathbf{g}} - m_g) + m_g, \lambda_t), \tag{4.15}$$

where $\lambda_t = k_{\mathbf{s}_t} - \mathbf{k}_{\mathbf{s}_t,\bar{\mathbf{s}}} \mathbf{K}_{\bar{\mathbf{s}}}^{-1} \mathbf{k}_{\mathbf{s}_t,\bar{\mathbf{s}}}^T$, $k_{\mathbf{s}_t} = \mathrm{k}(\mathbf{s}_t, \mathbf{s}_t)$, and $\mathbf{k}_{\mathbf{s}_t,\bar{\mathbf{s}}} = \mathrm{k}(\mathbf{s}_t, \bar{\mathbf{s}})$.

The probability of an episode of self-triggered policy search using GPs as a policy model is calculated as follows:

$$\begin{aligned} p(d_s, \mathbf{f}, \mathbf{g}, \mid \bar{\mathbf{f}}, \bar{\mathbf{g}}) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(a_t \mid f_t, o_t, a_{t-1}) p(f_t \mid \mathbf{s}_t, \bar{\mathbf{f}}) \cdot \\ p(\tau_t \mid g_t, o_t, \tau_{t-1}) p(g_t \mid \mathbf{s}_t, \bar{\mathbf{g}}) p(o_t \mid \tau_{t-1}) p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t), \end{aligned} \tag{4.16}$$

where $\mathbf{f} = \{f_t\}_{t=1}^T$ and $\mathbf{g} = \{g_t\}_{t=1}^T$. A graphical model of a GPSTPS is shown in Fig. 4.3.

4.3.3. Variational Learning for Policy Improvement

GPSTPS is formulated as a return-weighted marginal likelihood maximization problem. The posterior distribution of the nonlinear functions in the two policy models, kernel parameters θ, and pseudo inputs $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$ are optimized by variational learning.

With the episode's probability using the GP policy model in (4.16), the expected return is calculated:

$$J = \int R(d_s) p(d_s, \mathbf{f}, \mathbf{g} \mid \bar{\mathbf{f}}, \bar{\mathbf{g}}) p(\bar{\mathbf{f}}) p(\bar{\mathbf{g}}) \mathrm{d}d_s \mathrm{d}\mathbf{f} \mathrm{d}\bar{\mathbf{f}} \mathrm{d}\mathbf{g} \mathrm{d}\bar{\mathbf{g}}. \tag{4.17}$$

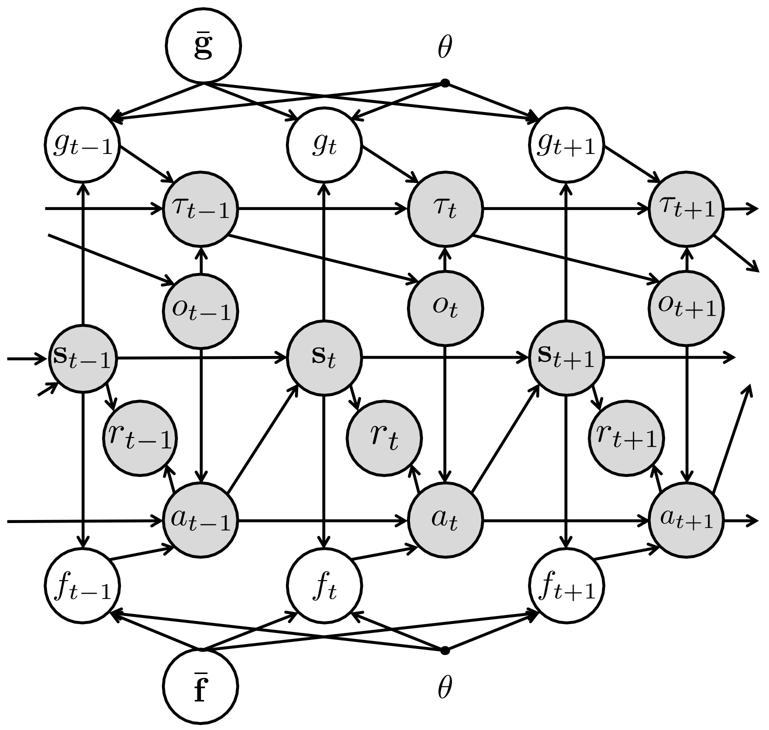


Figure 4.3.: Graphical model of GP self-triggered policy search: $\mathbf{s}_t$ is a state, a_t is an action, τ_t is an execution duration, o_t is a binary gating variable, f_t and g_t are outputs of functions f and g, and $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$ are pseudo outputs of GPs.

This equation cannot be analytically solved due to the complexity of d_s, $\bar{\mathbf{f}}$, and $\bar{\mathbf{g}}$. Therefore, by introducing the distribution of the data sample $p_{\text{old}}(d_s)$, expected return J_{old} by that distribution, and variation distribution $q(d_s, \mathbf{f}, \bar{\mathbf{f}}, \mathbf{g}, \bar{\mathbf{g}}) = p_{\text{old}}(d_s)p(\mathbf{f} \mid \bar{\mathbf{f}})q(\bar{\mathbf{f}})p(\mathbf{g} \mid \bar{\mathbf{g}})q(\bar{\mathbf{g}})$, the lower bound of expected return $\log J_L$ is derived:

$$\begin{aligned}
\log \frac{J(\theta)}{J_{\text{old}}} &\geq \int \frac{R(d_s)}{J_{\text{old}}} q(d_s, \mathbf{f}, \bar{\mathbf{f}}, \mathbf{g}, \bar{\mathbf{g}}) \log \frac{p(d_s \mid \bar{\mathbf{f}}, \bar{\mathbf{g}})p(\bar{\mathbf{f}})p(\bar{\mathbf{g}})}{q(d_s, \mathbf{f}, \bar{\mathbf{f}}, \mathbf{g}, \bar{\mathbf{g}})} \mathrm{d}d_s \mathrm{d}\mathbf{f} \mathrm{d}\mathbf{g} \mathrm{d}\bar{\mathbf{f}} \mathrm{d}\bar{\mathbf{g}} \\
&= \int \frac{R(d_s)}{J_{\text{old}}} p_{\text{old}}(d_s) p(\mathbf{f} \mid \bar{\mathbf{f}}) q(\bar{\mathbf{f}}) \log \frac{p(\mathbf{a} \mid \mathbf{f}) p(\bar{\mathbf{f}})}{q(\bar{\mathbf{f}})} \mathrm{d}d_s \mathrm{d}\mathbf{f} \mathrm{d}\bar{\mathbf{f}} \\
&+ \int \frac{R(d_s)}{J_{\text{old}}} p_{\text{old}}(d_s) p(\mathbf{g} \mid \bar{\mathbf{g}}) q(\bar{\mathbf{g}}) \log \frac{p(\boldsymbol{\tau} \mid \mathbf{g}) p(\bar{\mathbf{g}})}{q(\bar{\mathbf{g}})} \mathrm{d}d_s \mathrm{d}\mathbf{g} \mathrm{d}\bar{\mathbf{g}} \\
&= \log J_L(\theta, q), \qquad (4.18)
\end{aligned}$$

where $\mathbf{a} = \{a_t\}_{t=1}^T$ and $\boldsymbol{\tau} = \{\tau_t\}_{t=1}^T$. The expectation of $p_{\text{old}}(d)$ is solved by Monte

Carlo approximation using trial and error data. The lower bound of the expected return can be expressed:

$$\log J_L(\theta, q) \approx \int p(\mathbf{f} \mid \bar{\mathbf{f}}) q(\bar{\mathbf{f}}) \log \frac{p(\tilde{\mathbf{a}} \mid \mathbf{f}) p(\bar{\mathbf{f}})}{q(\bar{\mathbf{f}})} \mathrm{d}\mathbf{f} \mathrm{d}\bar{\mathbf{f}}$$
$$+ \int p(\mathbf{g} \mid \bar{\mathbf{g}}) q(\bar{\mathbf{g}}) \log \frac{p(\tilde{\boldsymbol{\tau}} \mid \mathbf{g}) p(\bar{\mathbf{g}})}{q(\bar{\mathbf{g}})} \mathrm{d}\mathbf{g} \mathrm{d}\bar{\mathbf{g}} + C, \quad (4.19)$$

where $p(\tilde{\mathbf{a}} \mid \mathbf{f}) = \mathcal{N}(\tilde{\mathbf{a}} \mid \mathbf{W}\mathbf{f}, \sigma_f^2 \mathbf{I})$ and $p(\tilde{\boldsymbol{\tau}} \mid \mathbf{g}) = \mathcal{N}(\tilde{\boldsymbol{\tau}} \mid \mathbf{W}\mathbf{g}, \sigma_g^2 \mathbf{I})$. $\mathbf{W} = \mathrm{diag}\left\{ \sqrt{\frac{R(d_s^{(1)})}{J_{\mathrm{old}} E}} \mathbf{1} \cdots \sqrt{\frac{R(d_s^{(E)})}{J_{\mathrm{old}} E}} \mathbf{1} \right\}$ is weight based on return, $\tilde{\mathbf{a}} = \mathbf{W}\mathbf{a}$, and $\tilde{\boldsymbol{\tau}} = \mathbf{W}\boldsymbol{\tau}$. Eq. (4.19) is the lower bound of the return-weighted marginal likelihood.

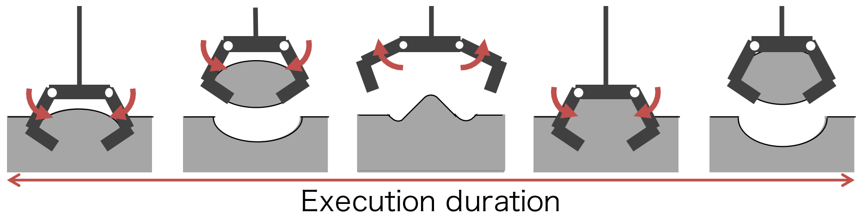
The variational learning update policy repeats the E-step and M-step, like the EM algorithm. In the E-step, the variational distributions of $q(\bar{\mathbf{f}})$ and $q(\bar{\mathbf{g}})$ are optimized alternately. In the M-step, kernel parameter θ and pseudo input $\bar{\mathbf{s}}$ are optimized by a gradient-based method. Optimal variational distributions approximate the true posterior. Using learned variational distribution, the predictive distribution of any action a_* and any duration τ_* according to any state $\mathbf{s}_*$ can be analytically obtained. The update law of the variational distributions and the predictive distribution are derived based on previous research [7]. Each predictive distribution is derived to determine the action and execution durations according to the policy.

4.4. EXPERIMENT

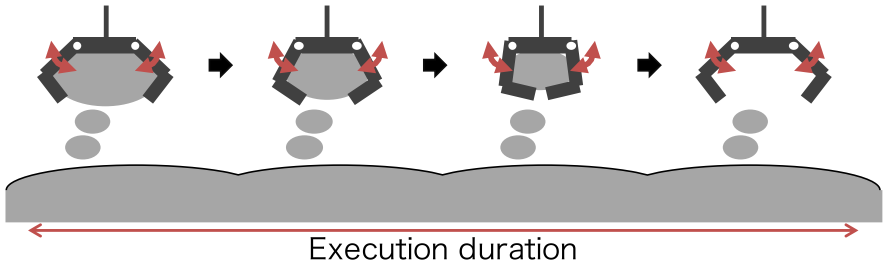
We applied GPSTPS to a garbage-grasping-scattering task by simulation and a robotic waste crane system to investigate its effectiveness. We verified that GPSTPS appropriately learned the action and duration policies based on the garbage's characteristics.

4.4.1. Garbage-grasping-scattering Task

The garbage-grasping-scattering task aims to evenly and widely scatter a large amount of garbage in a short time (Fig. 4.4). The grasping strategy generates a motion that lowers the bucket onto the garbage's surface and closes the claws



(a) Grasping



(b) Scattering

Figure 4.4.: Garbage-grasping-scattering task

while raising the bucket. The scattering strategy generates a motion where some of the garbage in the bucket falls by opening and closing it. To efficiently scatter a large amount of garbage, we need to switch between appropriate grasping and scattering, as well as their execution duration based on the garbage's characteristics.

In this task, state s_t is defined as the weight of the grasped garbage. The action is defined as binary $a_t = \{0, 1\}$, which indicates either grasping or scattering strategies. Execution duration τ_t indicates the number of execution steps for each strategy. The reward function is defined:

$$r_t = \begin{cases} 0 & (a_t = 0) \\ r_a \times r_\tau & (a_t = 1) \end{cases}. \tag{4.20}$$

r_a is the action reward for the scattering performance, and r_τ is the reward related to a scattering's duration. Each episode terminates when the crane finishes scattering the garbage grasped by the bucket.

Table 4.1.: Amount of garbage grasped with respect to grasping durations in simulation experiment: ϵ is sampled by $\mathcal{N}(0, 0.7)$.

Grasping durations	1	2	3	later
Setting 1 (soft)	$3+\epsilon$	$3+\epsilon$	$3+\epsilon$	$3+\epsilon$
Setting 2 (hard)	$2+\epsilon$	$3+\epsilon$	$5+\epsilon$	$5+\epsilon$

4.4.2. Simulation

Experimental settings

We simulated garbage grasping and scattering and verified the GPSTPS performance. The amount of garbage grasped by the waste crane depends on its characteristics. Thus, in this simulation, we prepared the two garbage characteristics shown in Table 4.1. Settings 1 and 2 indicate soft and hard garbage that requires different execution durations. Hard garbage's grasping strategy requires a longer duration since it cannot be loosened by being grasped. We assumed that for one execution duration step, a grasping strategy takes ten seconds and that a scattering strategy takes five seconds.

We defined the action and time rewards:

$$r_a = \min(s_t, \tau_t) - \alpha_{\text{sim}} \|s_t - \tau_t\|, \tag{4.21}$$

$$r_\tau = \exp\{-\beta_{\text{sim}}(u_{\text{act}} - u_{\text{min}})^2\}, \tag{4.22}$$

where $\alpha_{\text{sim}} = 1.5$, $\beta_{\text{sim}} = 0.004$, and $u_{\text{min}} = 30$ are the parameters of the reward function and u_{act} is the seconds required during garbage scattering. The action reward is high when the execution durations are similar to a state and a large amount of garbage is scattered. Time reward r_τ increases as the execution time is shortened.

The task begins without no grasped garbage in the bucket, and an episode terminates when the crane has scattered all of the grasped garbage. The action policy is modeled by a binary sparse GP classification model with $m_f = 0.5$. Since the classification model directly regresses the probability of the action selection, we ignore the uncertainty of the predictive model. The duration policy is sparse GP regression model with $m_g = 0$. The maximum execution duration of GPSTPS is set to six steps. For comparison, we employed GP policy search (GPPS) with a fixed duration from one to six steps. Pseudo inputs of sparse GP in each method are set $M = 5$. Also, we implemented a neural network (NN) based STPS, which employs NNs with three full-connect layers for both action and duration policies. We set multiple numbers of units in a hidden layer in NN as 10, 100, and 300. STPS with NN uses return weight likelihood in (4.19) as the objective function and learns each NN policy by Adam optimizer.

Result

Figs. 4.5 and 4.6 show the experimental result. Fig. 4.5 (a), 4.5 (b), 4.6 (a), and 4.6 (b) compares learning curves of GPSTPS, GPPS with a fixed duration, and GPSTPS with NN policy in each setting. Fig. 4.5 (c) shows that the action policy appropriately selected the grasping and scattering strategies based on the state. In Fig. 4.5 (d), the duration policy selected a short duration (a one- or two-step grasping strategy) since the amount of grasped garbage was not changed by the execution duration. The duration policy selected a similar execution time step as a state for the scattering strategy. The action policy learned by GPSTPS appropriately selected the action based on the state (Fig. 4.6 (c)). The learned duration policy selected a longer two- or three-step execution duration than the duration policy in setting 1 as its grasping strategy (Fig. 4.6 (d)). For the scattering strategy, this duration policy resembles the duration policy learned in setting 1.

In the learning curve of GPPS with a fixed duration in both settings shown in Figs. 4.5 (a) and 4.6 (a), we found different duration needs for each setting. Our GPSTPS outperformed the fixed duration method by learning a suitable duration for each setting. In comparison with the NN policy model, the performance of STPS with the NN policy model largely depends on the number of units in hidden layers. If we could appropriately set it (100 units), it would result in high performance; if we set too many (300 units) or too few (10 units), the performance will be severely degraded. On the other hand, our GPSTPS achieved comparable performance to that by NN policy models with appropriate settings without explicitly setting the number of units in hidden layers due to non-parametric characteristics of GPs.

In summary, these simulation results confirmed that GPSTPS can learn appro-

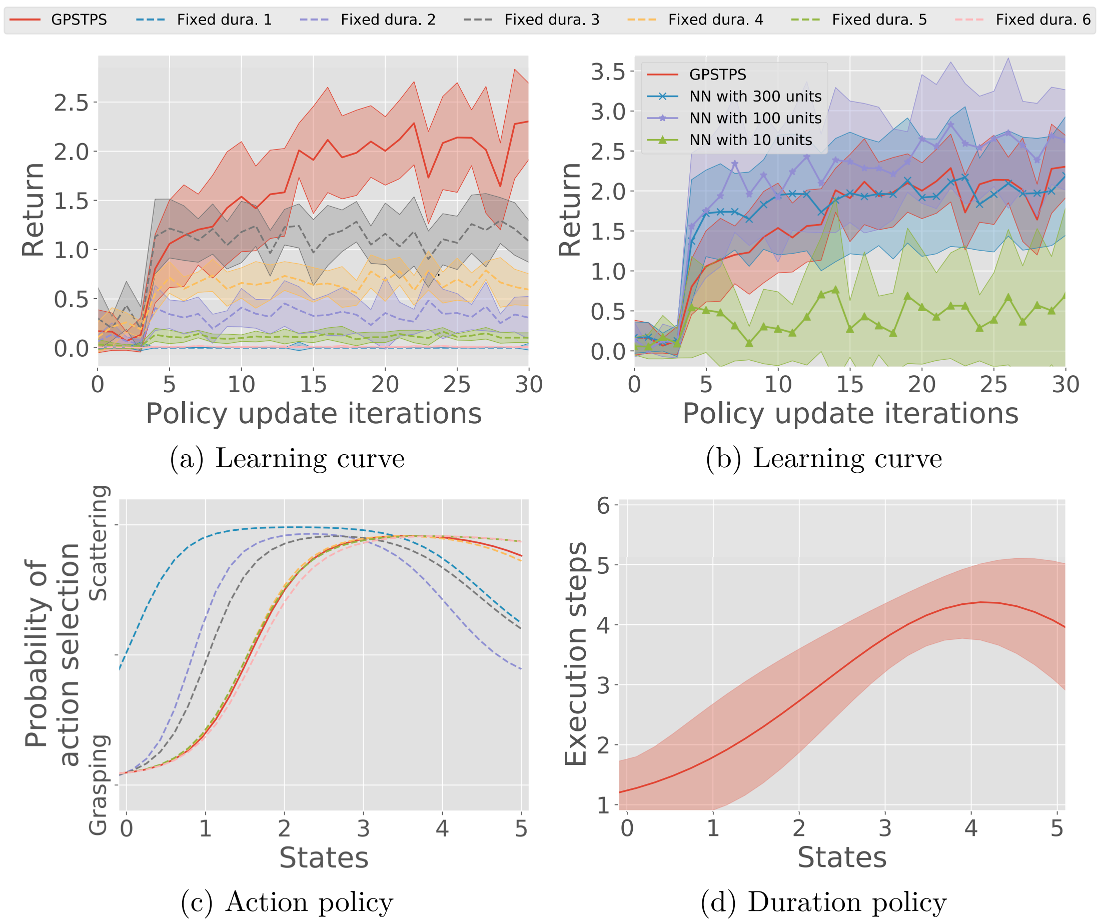
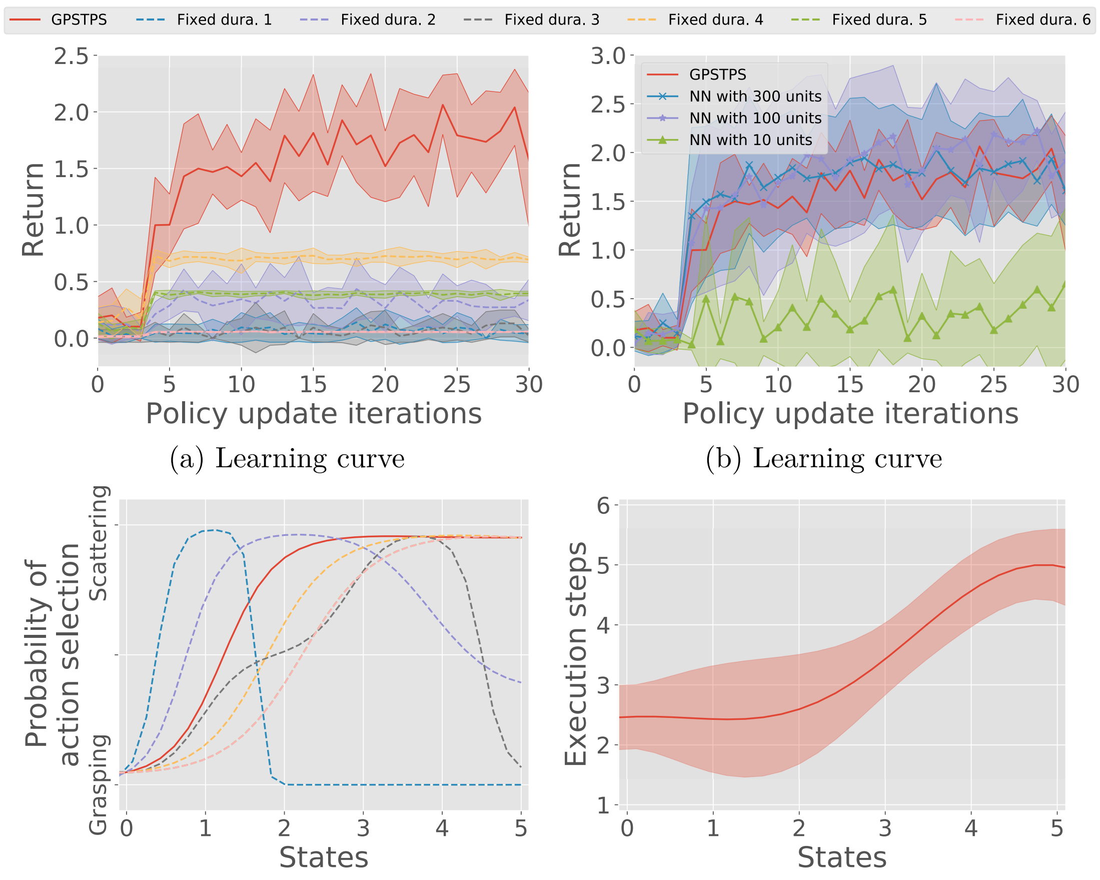


Figure 4.5.: Result of simulation experiments in setting 1. (a) the mean and standard deviation of the learning curve of ten experiments by GPSTPS and GPPS with a fixed duration. (b) mean and standard deviation of the learning curve of ten experiments by GPSTPS and STPS with NN. (c) learned action policies by GPSTPS. (d) learned duration policy by GPSTPS.



(a) Learning curve (b) Learning curve

(c) Action policy (d) Duration policy

Figure 4.6.: Result of simulation experiments in setting 2. (a) the mean and standard deviation of the learning curve of ten experiments by GPSTPS and GPPS with a fixed duration. (b) mean and standard deviation of the learning curve of ten experiments by GPSTPS and STPS with NN. (c) learned action policies by GPSTPS. (d) learned duration policy by GPSTPS.

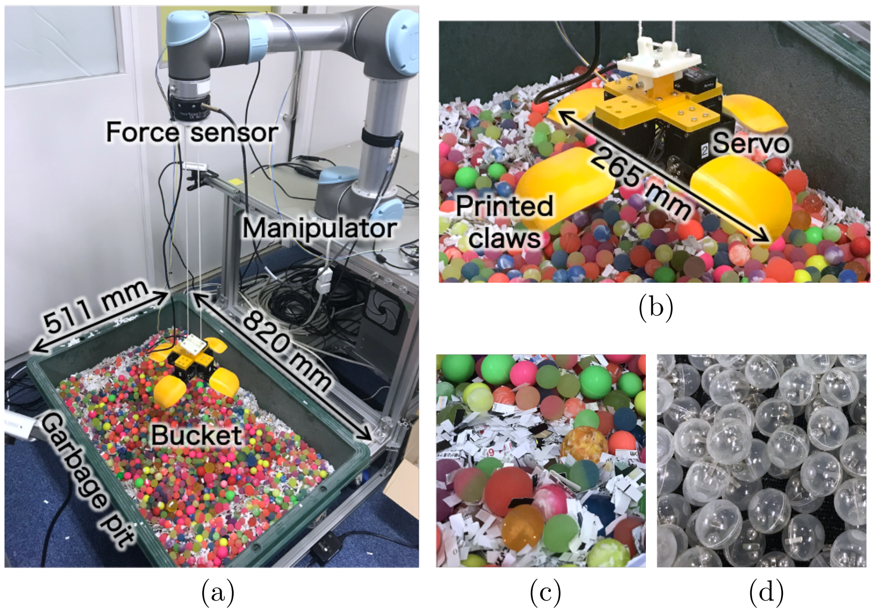


Figure 4.7.: Environment of robot experiment: (a) overview, (b) bucket, (c), and (d) garbage with different characteristics.

priate action and execution duration policies that maximize the return, depending on the garbage's characteristics.

4.4.3. Robot Experiment

Robotic waste crane

We conducted an experiment with a robotic waste crane in an environment that resembles an actual waste crane using a manipulator (Fig. 4.7 (a)). The crane part consists of a robot manipulator Universal Robot 5 and a force sensor Robotiq FT-300. The manipulator moves the bucket and weighs the grasped garbage by a force sensor. The bucket consists of four servo motors (ROBOTIS DYNAMIXEL XM430-W350-T) and four claws made by a 3D printer (Fig. 5.3 (b)). The bucket and the robot arm are connected by string. The four claws are moved by the same movement by servo motors to reproduce the movements of the actual bucket. The waste crane moves in a 511-mm long, 820-mm wide mock garbage pit.

Algorithm 4.2: Grasping strategy

```
  Input: Execution duration: $\tau_t$
1 Bucket is lowered with open claws until it lands on garbage
2 for $i = 1$ to $\tau_t$ do
3     for $n\_close = 1$ to $4$ do
4         Bucket's claws are closed until they stop
5         Bucket is raised for $t_{\text{lift}}$
6     end
7 end
```

Experimental settings

We conducted an experiment with a robotic waste crane to verify GPSTPS's effectiveness. We used two types of garbage (Figs. 4.7 (c) and 4.7 (d)) to verify the performance with different garbage characteristics. Fig. 4.7 (c) shows paper-based garbage that consists of shredded paper and rubber balls with 17 and 30 mm diameters. Since this garbage is soft with a small particle size, it resembles dry and non-sticky garbage like plastic. The bucket can grasp paper-based garbage with a short duration and the grasped garbage falls from a small gap between the claws. The magnet-based garbage in Fig. 4.7 (d) is composed of 27-mm diameter capsules containing magnets and iron balls. This garbage has a large particle size, and the magnets attract each other. It resembles wet and easily aggregated garbage that is collected on rainy days. The bucket needs a longer grasp duration because the garbage falls from the bucket during the aggregation. The initial positions of the robotic crane in the grasping and scattering strategies are randomly selected in the mock garbage pit and automatically moved to the initial position. The crane's moving distance in the scattering strategy is set to 30 cm. Each predetermined strategy is defined in Algorithms 4.2 and 4.3. The parameters in each algorithm are set to $t_{\text{lift}} = 0.8$ s, $w_{\text{fall}} = 7$ g, and $t_{\text{close}} = 2$ s.

In the robot experiment, action reward function r_a and time reward function r_τ are defined:

$$r_a = w_{\max}\exp\{-\gamma \text{RMS}(\mathbf{m} - \mathbf{m}_{\text{I}})\}, \tag{4.23}$$

$$r_\tau = \exp\{-\beta_{\text{robot}}(u_{\text{act}} - u_{\min})^2\}, \tag{4.24}$$

Algorithm 4.3: Scattering strategy

```
   Input: Execution duration: $\tau_t$
1  Move to initial position
2  Set crane speed $v_{\text{crane}}$ using the execution time $\tau_t$
3  for $n\_scatter = 1$ to $\tau_t$ do
4      Bucket's claws are opened until they detect the fall of $w_{\text{fall}}$ of garbage
5      Bucket's claws are closed for $t_{\text{close}}$
6  end
```

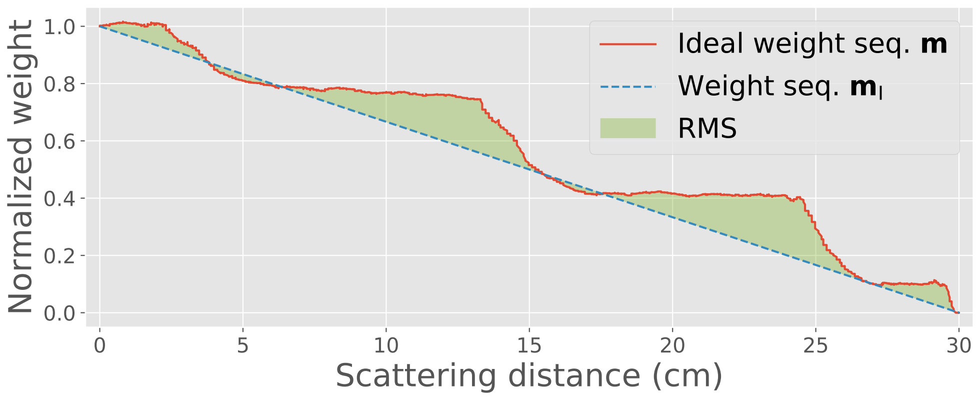


Figure 4.8.: Illustration of RMS between actual and ideal grasped-weight sequences in garbage-scattering

where $\beta_{\text{robot}} = 2.5 \times 10^{-4}$, $\gamma = 7$, $u_{\min} = 30$, and u_{act} is the execution time of an episode. $u_{\min}$ means minimum execution time of grasping and scattering. The purpose of this task is to scatter a lot of garbage in a shorter time evenly. The action reward function evaluates scattering performance using RMS between sequence of grasped weight $\mathbf{m}$ and ideal weight $\mathbf{m}_\mathrm{I}$ that decreases weight linearly. The actual and ideal grasped weight sequences are shown in Fig 5.2. The time reward function evaluates the shortness of scattering.

The policy is learned with the same experimental settings as in the simulation experiment. The initial action policy selects the grasping strategy when the grasped weight is 0 g and the scattering strategy at other times.

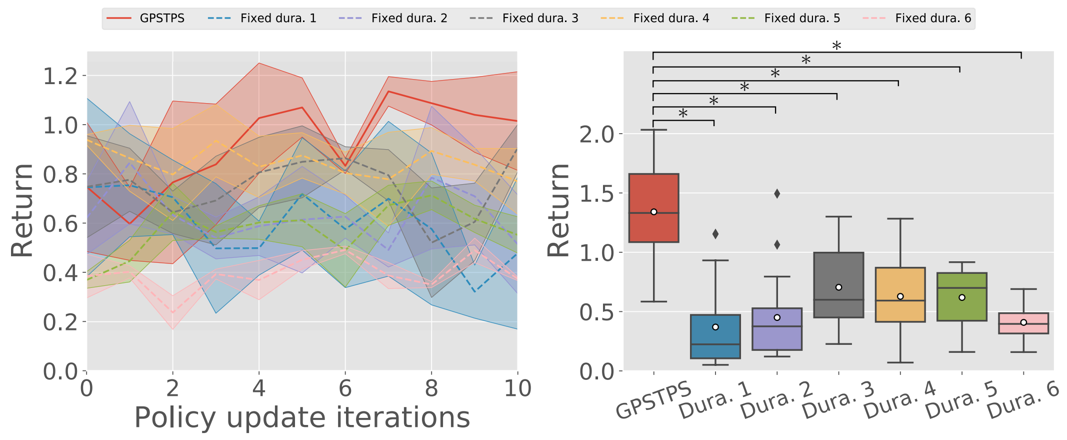
Result

Figs. 4.9 and 4.10 show the experimental result of the robotic experiment. Figs. 4.9 (a) and 4.9 (b) show that GPSTPS outperformed GPPS with each fixed duration. The paired t-test result shows that GPSTPS's return has a significant difference against all the compared methods. Figs. 4.9 (c) and 4.9 (d) indicate the learned policies. Figs. 4.11 (a) show the robotic waste crane trajectory by the policies learned by GPSTPS. The learned action policy appropriately selected a control strategy. The learned duration policy selected two and five steps of execution duration for grasping and scattering. This result indicates that the duration policy captured the paper-based garbage's easy-to-grasp and gradually falling characteristics from a small gap between the claws.

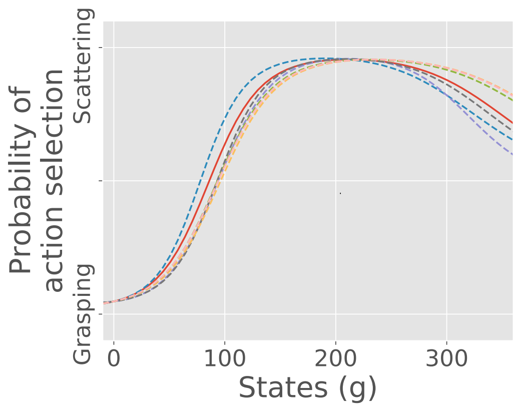
Figs. 4.10 (a) and 4.10 (b) show that GPSTPS outperformed GPPS with each fixed duration. The paired t-test result shows that GPSTPS's return has a significant difference against all the compared methods except GPPS with a fixed four duration. GPSTPS and GPPS with such a duration do not have a significant difference, although GPSTPS obtained higher returns than GPPS with the fixed four duration. Figs. 4.10 (c) and 4.10 (d) show the learned policies. Fig. 4.11 (b) and the robotic waste crane trajectory by the policy learned by GPSTPS. The learned duration policy selected three duration steps for grasping the garbage and four for scattering it by capturing the characteristic of the magnet-based garbage of the difficult-to-grasp and falling together. In the robotics experiment, running each episode took about two to three minutes. One learning experiment took three hours.

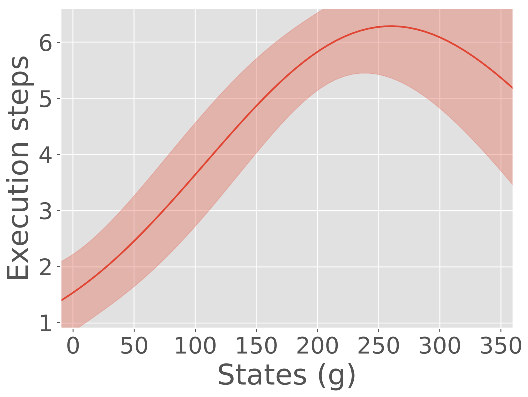
Table 4.2 shows the performance of initial and learned policies by GPSTPS in terms of RMS weight sequence and task execution time. We confirmed that learned policy by GPSTPS significantly improved the evenness of scattering and execution time by t-test.

In summary, we developed a robotic waste crane system that imitates an actual one and conducted experiments on a garbage-grasping-scattering task using two different kinds of mock garbage. We confirmed that GPSTPS properly learned the action and duration policies based on the garbage's characteristics.



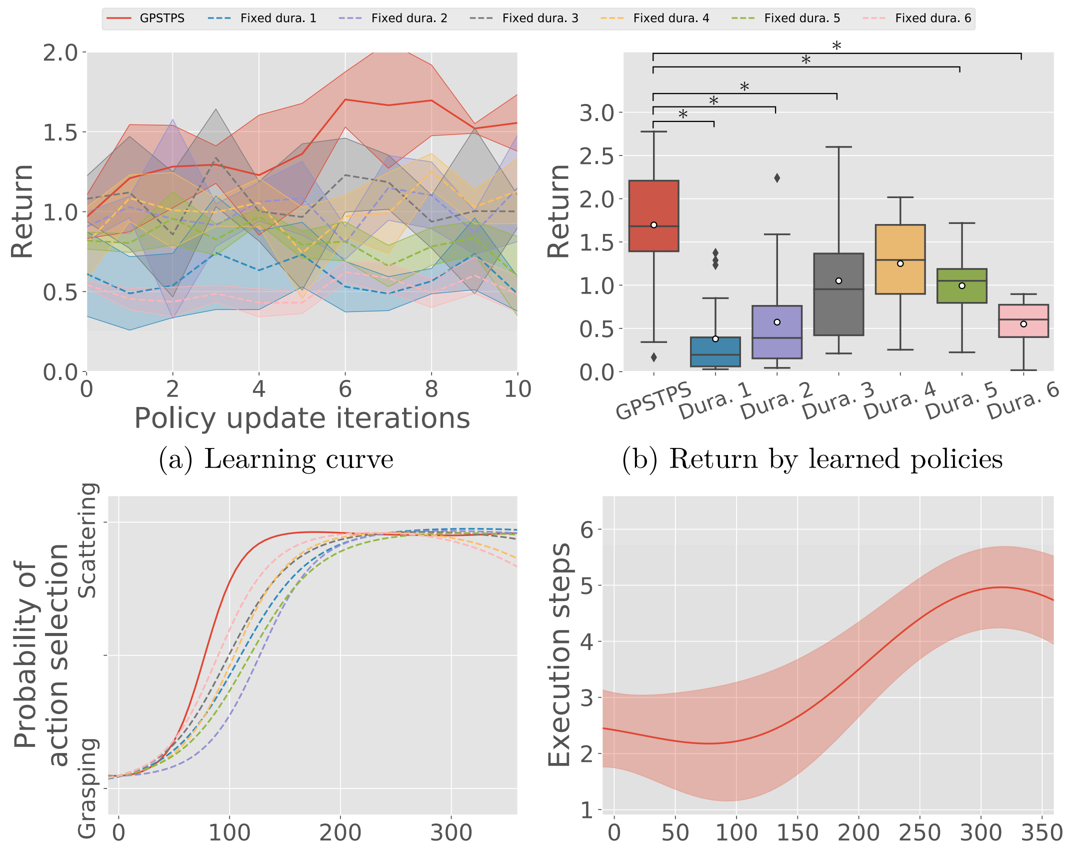
(a) Learning curve (b) Return by learned policies





(c) Action policy (d) Duration policy

Figure 4.9.: Result of robot experiment with paper-based garbage: (a) mean and standard deviation of return of three experiments. (b) test performance of learned policies where * denotes $p < 0.05$ on paired t-test. (c) and (d) action and duration policies learned by each method.



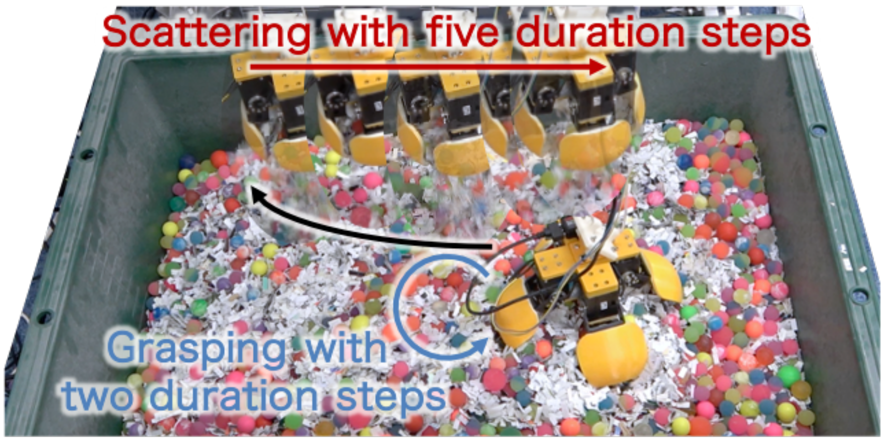
(a) Learning curve

(b) Return by learned policies

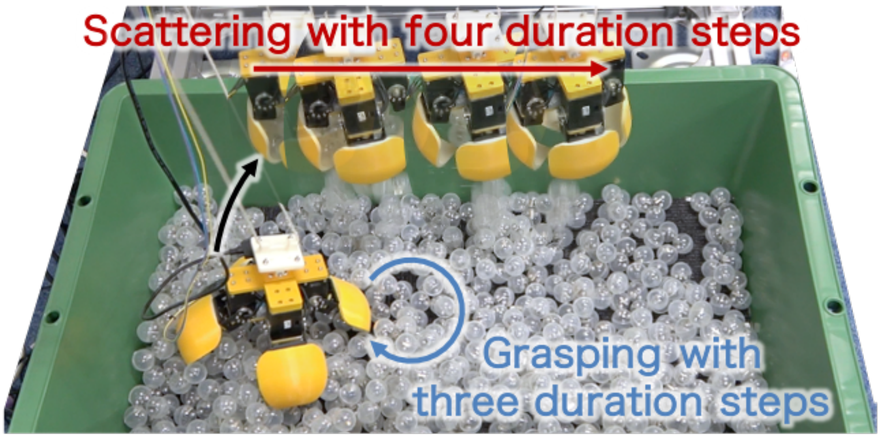
(c) Action policy

(d) Duration policy

Figure 4.10.: Result of robot experiment with magnet-based garbage: (a) mean and standard deviation of return of three experiments. (b) test performance of learned policies where * denotes $p < 0.05$ on paired t-test. (c) and (d) action and duration policies learned by each method.


Scattering with five duration steps
Grasping with two duration steps

(a) Paper-based garbage


Scattering with four duration steps
Grasping with three duration steps

(b) Magnet-based garbage

Figure 4.11.: Robot trajectory by policy learned by GPSTPS

Table 4.2.: Comparison of the performance of initial and learned policy by GP-STPS. Each policy is compared in terms of RMS weight sequence and the execution time of an episode. Values indicate mean and standard deviation of reward by 20 times of execution of the garbage-grasping-scattering task with paper- and magnet-based garbage. * and ** denote $p < 0.05$ and $p < 0.01$ on paired t-test, respectively.

	Policy	Paper garbage	Magnet garbage
RMS of weight seq.	Initial	1.17 ± 0.992*	1.74 ± 1.33**
	Learned	$\mathbf{0.516 \pm 0.354}$*	$\mathbf{0.629 \pm 0.694}$**
Execution time	Initial	64.0 ± 19.5*	59.4 ± 14.2*
	Learned	$\mathbf{50.9 \pm 18.4}$*	$\mathbf{50.8 \pm 8.91}$*

4.5. Summary of Chapter 4

This study aimed to automate machines in weakly observable environments in an industrial work place and proposed GPSTPS that can learn both action and duration policies to repeatedly perform the same action to overcome uncertainty. Its effectiveness was experimentally verified with simulations and a robotic waste crane system.

5. Multi-task Robust Bayesian Optimization for Unpredictable Fluctuation of Performance

5.1. Introduction

Automatic control of large industrial cranes is an attractive alternative to the use of human workers in high repetition and high-risk environments. Manual, repetitive control of heavy-duty construction machinery is a heavy-labor task, so accidents at worksites remain high [76]. Therefore, integrating automatic control policies in such environments as waste incineration plants [50–52] is particularly desirable.

However, automated crane and bucket control remains a challenging task, due to the difficulty in modeling their kinematic and dynamic characteristics, as often requires sophisticated non-linear controllers specific to the system [49]. Even a well-designed model of both the mechanism and controller suffers from the inherent operational problem of inhomogeneous loading material, and as such, traditional load estimation techniques [77] rely on the assumption of static loads that fall within expected shape and weight constraints.

In the context of waste incineration plants, garbage bags have mixed sizes, weights in addition to material properties such as hardness and wetness. These environmental uncertainties are a crucial issue limiting innovation in automated control of cranes and buckets [49]. While external factors such as wind [54], waves [55,56], or soil properties in automated excavation [57,59] have been investigated in different contexts, they often rely on assumptions on the underlying distribution of uncertainty (such as the repetitive nature of waves); such assumptions are not valid for the garbages. Therefore, there is little research on the issue of automated crane and bucket control handling such inhomogeneous garbage in an industrial environment.

As a solution to these uncertainties, an alternative is to use data-driven methods from the field of machine learning to automatically build a model of the task environment or a control policy, solely from data collected during operation in that environment. There is no requirement for a priori description of the deformable material properties or complex analytical models of the plant. Such automated skill acquisition methods have previously been applied to overcoming payload uncertainties in small-scale robotic control systems, such as learning and training policies from user demonstrations [14], using reinforcement learning to manipulate objects in industrial assembly lines [78] or front-end loaders [61], vision-based deep learning with uncertainty in grip pose [79] or with regards to deformable soft objects [11], or policy optimization [80].

For a large-scale industrial waste crane, however, obtaining data samples by executing tasks is very costly, due to the slow-moving system (takes several minutes for executing one task) and limited plant downtime. Therefore, a large amount of trial and error is infeasible, unlike in [13,14]. Another issue is that no sensors are available that can observe the state of the grasped flammable waste composed of various materials with different degrees of hardness and wetness. Therefore, the inhomogeneity of waste causes unpredictable fluctuations in the crane's task performance. Therefore, to our knowledge, such a framework that can optimize control policies of an actual garbage crane handling inhomogeneous waste has not been established so far.

The objective of this study is to develop a framework that can optimize control policies of a waste crane at a waste incineration plant through an autonomous trial and error manner based on a sample-efficient black-box optimization scheme so-called Bayesian optimization (BO) [17,81,82]. Although conventional BO algorithms utilize standard Gaussian process regression to learn a surrogate model of task performance from data samples [81], however, it is sensitive to outliers inevitable in waste cranes, and its performance may deteriorate significantly. Therefore, our framework employs a novel BO algorithm, Multi-Task Robust Bayesian Optimization (MTRBO). The MTRBO has the following characteristics: (1) outlier robustness against garbage inhomogeneity and (2) sample reuse from previously solved tasks to enhance the sample efficiency further. To investigate our framework's effectiveness, we conducted experiments on garbage-scattering tasks with (i) a robot waste crane with pseudo-garbage and (ii) an actual waste crane at a waste incineration plant. Experimental results demonstrate that our MTRBO framework robustly optimized the control policies of the garbage cranes, even with a significantly reduced number of data under the influence of garbage inhomogeneity.

The following are the contributions of this chapter

1. Proposed a control policy optimization framework for a waste crane;
2. Derived the MTRBO algorithm based on MTRGP with variational Bayesian inference;
3. Verified its effectiveness through experiments with both a robot waste crane and an actual waste crane.

5.2. Related Work

Black-box optimization-based policy search has been proposed as a data-efficient policy learning method. It is possible to obtain the optimal policy with a small number of trials by formulating policy parameter optimization as a black-box optimization problem. Especially, Bayesian optimization (BO) is applied to control policy optimization of various robot tasks, for example, the gait of snake robot in different environments [83], pick-up operation of care support robot using user feedback [82], the gait of biped robot operation [17]. Various methods of extending BO to acquire control policy for the robot were proposed as follows, a method of acquiring optimal and safe parameters by focusing on the sensitivity of robot movement parameters [84], optimization of parameter by robot safety [85], a method to efficiently obtain high-dimensional policy parameter by designing the features of motion sequences from simulation data [86, 87]. In this chapter, we propose BO with robustness and multi-task for policy learning of waste cranes worked in a weakly observable environment.

5.3. Multi-task Robust Bayesian Optimization

MTRBO uses Multi-Task Robust Gaussian Processes (MTRGP) to learn the evaluation function instead of standard Gaussian processes. We present a variational Bayesian inference procedure that optimizes the parameters and the posterior distribution of the MTRGP model alternatively in an EM-like scheme with trial and error data. The resulted predictive mean and variance from MTRGP are then used in the acquisition function of BO to form MTRBO.

5.3.1. MTRGP model

The critical ingredient of MTRBO is Multi-Task Robust Gaussian Process regression (MTRGP), which is a novel combination of robust Gaussian process regression [23] with multi-task Gaussian process regression [88, 89].

Outlier Robustness To robustly learn the evaluation functions for the outliers caused by garbage inhomogeneity, MTRGP uses student-t distribution [8], as the likelihood function instead of typical Gaussian distribution. Student-t distribution has heavier tails than Gaussian distribution, and the distribution means are robust to outliers. The student-t likelihood function is represented as follows:

$$\begin{aligned} p(y_n \mid f_n) &= \mathrm{St}(y_n \mid f_n, a, b) \\ &= \frac{b^a}{\Gamma(a)\sqrt{2\pi}} \left(b + \frac{(y_n - f_n)^2}{2} \right)^{-a-1/2} \Gamma(a + 1/2), \end{aligned} \tag{5.1}$$

where $\Gamma(\cdot)$ is gamma function, a and b are parameters of student-t distribution.

Sample Reuse Mechanism MTRGP incorporates a multi-task GP (MTGP) model [88] that enables efficient learning of the evaluation function of the current task by reusing the data of previously optimized $M-1$ tasks. MTRGP treats previously optimized task data by associating latent task labels t^m. t^1 indicates the current task label, and the others $t^2, \cdots, t^M$ indicate reused task labels. MTRGP learns evaluation functions which are modeled as $y_n^m = f(\mathbf{w}_n^m, t^m) + \varepsilon_n$ where y_n^m is m-th task's evaluation value that corresponds to policy parameter $\mathbf{w}_n^m$ and t^m is task label of m-th task. We assume the following GP prior to the latent function $\mathbf{f}_{\text{multi}} = [f_1^1, \cdots, f_n^m, \cdots, f_N^M]^T$, $f_n^m = f(\mathbf{w}_n^m, t^m)$ using the evaluated parameters $\mathbf{w}_{\text{multi}} = [\mathbf{w}_1^1, \cdots, \mathbf{w}_n^m, \cdots, \mathbf{w}_N^M]^T$ and the task labels $\mathbf{l} = [t^1, \cdots, t^M]^T$:

$$p(\mathbf{f}_{\text{multi}} \mid \mathbf{w}_{\text{multi}}, \mathbf{l}) = \mathcal{N}(\mathbf{f}_{\text{multi}} \mid \mathbf{0}, \mathbf{K}_{\text{multi}}), \tag{5.2}$$

where $\mathbf{K}_{\text{multi}}$ is multi-task kernel gram matrix as $[\mathbf{K}_{\text{multi}}]_{ij} = \mathrm{k}_{\text{multi}}((\mathbf{w}_i^m, t^m), (\mathbf{w}_j^{m'}, t^{m'}))$, $\mathrm{k}_{\text{multi}}(\cdot, \cdot)$ is the multi-task kernel function. The multi-task kernel function is defined as follow [88]:

$$\mathrm{k}_{\text{multi}}((\mathbf{w}_i^m, t^m), (\mathbf{w}_j^{m'}, t^{m'})) = \mathrm{k}(\mathbf{w}_i^m, \mathbf{w}_j^{m'}) \ \mathrm{k}_t(t^m, t^{m'}), \tag{5.3}$$

where $\mathrm{k}_t(\cdot, \cdot)$ is a task kernel function that calculates the similarity between each task defined as: $\mathrm{k}_t(t^m, t^{m'}) = [\mathbf{K}_t]_{mm'}$, where, $\mathbf{K}_t$ is a kernel gram matrix between tasks. $\mathbf{K}_t$ must be a positive semi-definite matrix for the task kernel function $\mathrm{k}_t(\cdot, \cdot)$ to be a proper kernel function that calculates inner product of associated features based on task labels implicitly. A matrix decomposed $\mathbf{K}_t$ by Cholesky decomposition is defined as a task kernel parameter θ_t [88].

5.3.2. Model Training Procedure

To learn the MTRGP model from the data, we apply variational Bayesian inference. To make it tractable, we utilize the scale-mixture representation of student-t distribution [23]:

$$\mathrm{St}(y_n^m \mid f_n^m) = \int p(y_n^m \mid f_n^m, \tau_n) p(\tau_n) \, \mathrm{d}\tau_n, \tag{5.4}$$

$$p(y_n^m \mid f_n^m, \tau_n) = \mathcal{N}(y_n^m \mid f_n, \tau_n^{-1}), \tag{5.5}$$

$$p(\tau_n) = \mathrm{Gam}(\tau_n \mid a, b), \tag{5.6}$$

where τ_n is the precision of the Gaussian distribution for the n-th data, indicating the reliability of the data, and a and b are the parameters of the gamma distribution.

Then, we follow the standard variational Bayesian inference procedure [8]: a lower bound of marginal likelihood F_v is derived by applying the Jensen's inequality as follows:

$$\begin{aligned}
&\log p(\mathbf{Y}_{\text{multi}} \mid \mathbf{w}_{\text{multi}}, \mathbf{l}) \\
&= \log \int p(\mathbf{Y}_{\text{multi}} \mid \mathbf{f}_{\text{multi}}, \mathbf{T}) p(\mathbf{f}_{\text{multi}} \mid \mathbf{w}_{\text{multi}}, \mathbf{l}) p(\mathbf{T}) \, d\mathbf{f}_{\text{multi}} d\mathbf{T} \\
&\geq \int q(\mathbf{f}_{\text{multi}}, \mathbf{T}) \log \frac{p(\mathbf{Y}_{\text{multi}}, \mathbf{f}_{\text{multi}}, \mathbf{T} \mid \mathbf{w}_{\text{multi}}, \mathbf{l})}{q(\mathbf{f}_{\text{multi}}, \mathbf{T})} \, d\mathbf{f}_{\text{multi}} d\mathbf{T} \\
&= F_v,
\end{aligned} \tag{5.7}$$

where evaluation values $\mathbf{Y}_{\text{multi}} = [y_1^1, \cdots, y_n^m, \cdots, y_N^M]^T$, $\mathbf{T} = \text{diag}\{\tau_n\}_{n=1}^N$ is a diagonal matrix whose elements are τ_n, $p(\mathbf{Y}_{\text{multi}} \mid \mathbf{f}_{\text{multi}}, \mathbf{T}) = \prod_{n=1}^N p(y_n^m \mid f_n^m, \tau_n)$ is a likelihood, and $p(\mathbf{T}) = \prod_{n=1}^N p(\tau_n)$ is a prior distribution of τ_n, $q(\mathbf{f}_{\text{multi}}, \mathbf{T})$ is a variational distribution. We assume the independence of $\mathbf{f}_{\text{multi}}$ and $\mathbf{T}$ and decompose the variational distribution $q(\mathbf{f}_{\text{multi}}, \mathbf{T}) = q(\mathbf{f}_{\text{multi}}) \prod_{n=1}^N q(\tau_n)$. The variational distributions that maximize the lower bound F_v approximate the true posterior distribution [8].

At the end, our learning procedure for MTRGP follows an EM-like scheme. In E-step, the variational distributions are updated alternately. The update formula for variational distribution $q(\mathbf{f}_{\text{multi}}) = \mathcal{N}(\mathbf{f}_{\text{multi}} \mid \mu_{\mathbf{f}}, \Sigma_{\mathbf{f}})$ and $q(\tau_n) = \text{Gam}(\tau_n \mid a_n, b_n)$ are described in (D.13) and (D.17) in Appendix. In M-step, input kernel parameter θ_k, task kernel parameter θ_t, and the parameters of student-t distribution, a and b, are optimized by gradient-based optimization with F_v, which is described in (D.20), as the objective function.

5.3.3. Find next query parameter by MTRBO

The mean and variance functions of predictive distribution of MTRGP are calculated using the posterior distribution:

$$\begin{aligned}
p(f(\mathbf{w}, t^1) \mid \mathbf{w}, t^1, \mathbf{Y}_{\text{multi}}, \mathbf{w}_{\text{multi}}, \mathbf{l}) & \\
= \mathcal{N}(f(\mathbf{w}, t^1) \mid \mu(\mathbf{w}, t^1), \sigma^2(\mathbf{w}, t^1)), & \quad (5.8) \\
\mu(\mathbf{w}, t^1) = \mathbf{k}_{\mathbf{w}_{\text{multi}},*}^T \mathbf{K}_{\text{multi}}^{-1} \mu_{\mathbf{f}}, & \quad (5.9) \\
\sigma^2(\mathbf{w}, t^1) = k_{\text{multi}} - \mathbf{k}_{\mathbf{w}_{\text{m}},*}^T (\mathbf{K}_{\text{multi}} + \hat{\mathbf{T}}^{-1})^{-1} \mathbf{k}_{\mathbf{w}_{\text{m}},*}, & \quad (5.10)
\end{aligned}$$

Algorithm 5.1: MTRBO

```
Input  : Previously evaluated data: $\mathbf{D} = \{\mathbf{w}_{\text{multi}}, \mathbf{Y}_{\text{multi}}, \mathbf{l}\}$
Output: Next query parameter $\mathbf{w}'$
1 initialization $\theta_k$, $\theta_t$, $a$, $b$
2 while $F_v$ is not converged do
3     E-step: Optimize variational distribution $q(\mathbf{f})$ and $q(\mathbf{T})$ alternatively
       (Appendix D.1)
4     M-step: Optimize model parameters $\boldsymbol{\theta_k^*, \theta_t^*, a^*, b^*} \leftarrow \mathop{\mathbf{arg\,max}}_{\boldsymbol{\theta_k,\theta_t,a,b}} \boldsymbol{F_v}$
       (Appendix D.2)
5 end
6 Find next query parameter by $\mathbf{w}' \leftarrow \arg\max_{\mathbf{w}} a(\mathbf{w})$
```

where $k_{\text{multi}} = \mathrm{k}_{\text{multi}}((\mathbf{w}, t^1), (\mathbf{w}, t^1))$ is a kernel value of the input of predictive distribution, $\mathbf{k}_{\mathbf{w}_{\text{m}},*}$ is a kernel vector as $[\mathbf{k}_{\mathbf{w}_{\text{m}},*}]_i = \mathrm{k}_{\text{multi}}((\mathbf{w}_i^m, t^m)(\mathbf{w}, t^1))$, $\hat{\mathbf{T}} = E_q[\mathbf{T}]$ is the expected value by variational distribution. We determine the next parameter to evaluate by UCB (2.17) using mean function $\mu(\mathbf{w}, t^1)$ and variance function $\sigma^2(\mathbf{w}, t^1)$. The algorithm 5.1summarizes the detailed process of how to find the next query parameter by MTRBO.

5.4. Experiments

To investigate the effectiveness of our framework, we applied our framework to a garbage-scattering task with a robot waste crane and an actual waste crane. The experiments with an actual crane are limited to system downtime of the plant at maintenance, so evaluation experiments cannot be performed multiple times. To evaluate the effectiveness of the proposed method with a sufficient amount of trials prior to that with the actual crane, we built a robot crane that mimics an actual waste crane and conducted experiments each to verify the reproducibility of the results.

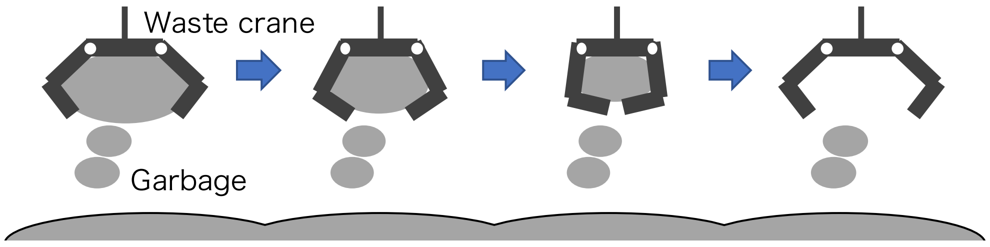


Figure 5.1.: Garbage-scattering task

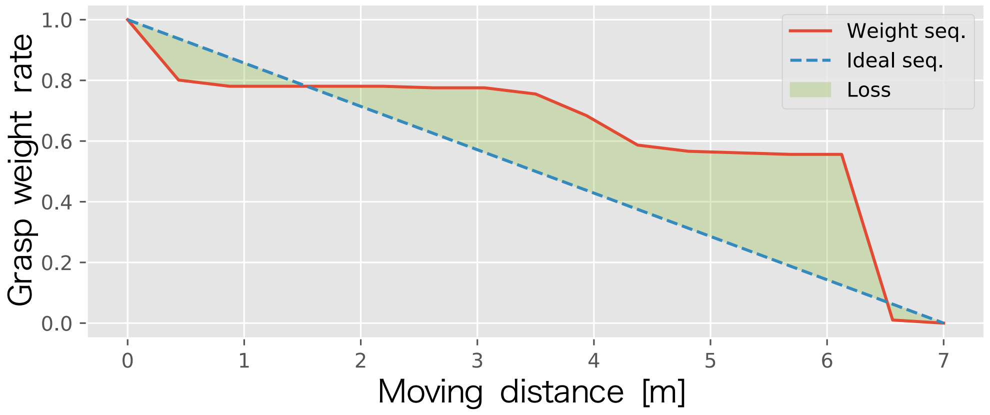


Figure 5.2.: Illustration of RMS between actual and ideal grasped-weight sequences in scattering task

5.4.1. Garbage-Scattering Task

The daily work of human operators at waste incineration plants is garbage-scattering tasks to homogenize garbage and stabilize the incineration process. The waste crane homogenizes garbage in the pit by first grasping a sufficient amount and dropping it at a constant pace (Fig. 5.1). We designed a finite state machine policy as a control policy model for the garbage-scattering task (Algorithm 5.2). Note that the policy's behavior strongly depends on the moving distance. Therefore, each task is conditioned by the targeted moving distance. The policy parameter is evaluated using the transition of the grasped weight of the bucket during the garbage-scattering task. We evaluated the policy parameter with a normalized sequence of grasped weight $\mathbf{m}(\mathbf{w})$ observed by a weight sensor in one scattering of garbage. Evaluation function f is designed using root mean square (RMS) between a sequence of grasped weight $\mathbf{m}(\mathbf{w})$ and an ideal

Algorithm 5.2: State-machine policy for garbage-scattering task: t_{close} indicates the time that policy is in a close state and w_{open} indicates the weight of fallen waste when the policy is in an open state. v is the rating velocity of the bucket's actuator.

```
Input: Policy parameter: $\mathbf{w} = \{w_1, w_2\}$
state = "open"
while now scattering do
    if state == "open" then
        Set bucket's actuator velocity $v$ to continue opening up to the limit
        if $w_1 < w_{\text{open}}$ then
            state = "close"
        end
    else if state == "close" then
        Set bucket's actuator velocity $-v$ to continue closing up to the limit
        if $w_2 < t_{\text{close}}$ then
            state = "open"
        end
    end
end
```

sequence of grasped weight $\mathbf{m}_{\text{I}}$ (Fig. 5.2):

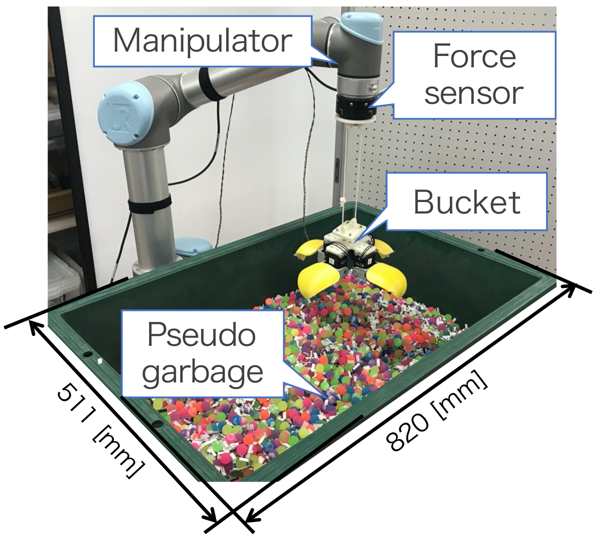
$$f(\mathbf{w}) = p - q \times \text{RMS}(\mathbf{m}(\mathbf{w}) - \mathbf{m}_{\text{I}}), \tag{5.11}$$

where we set $p = 5$ and $q = 10$ in subsequent experiments. Due to garbage's inhomogeneity, the sequence of grasped weight $\mathbf{m}(\mathbf{w})$ may vary considerably even with identical parameters and conditions (i.e., scattering distance and garbage weight), causing outliers in evaluation values.

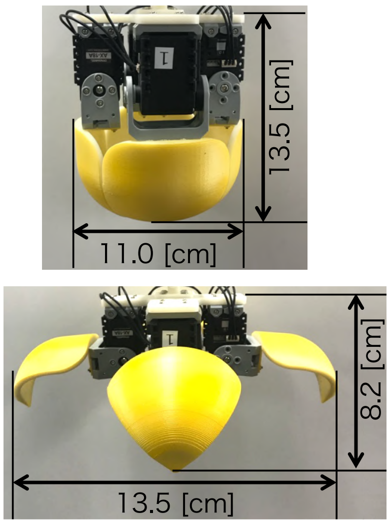
5.4.2. Policy Optimization with Robot Waste Crane

Robot Waste Crane

We developed a robot waste crane system (Fig. 5.3 (a)) to verify our proposed framework. The crane moves using a robot manipulator (UR5), measures weight



(a) Overview of robot waste crane



(b) Bucket of robot waste crane

Figure 5.3.: Robot waste crane

with a force sensor (Robotiq FT-300), and grasps garbage using a self-made bucket with four servo motors (ROBOTIS AX-18A) and 3D printed pawls (Fig. 5.3 (b)). The bucket is suspended from the robot manipulator by a wire. Instead of actual garbage, we used a mixture of shredded paper and rubber balls with an 18 mm diameter as pseudo-garbage with inhomogeneity.

Experimental Settings

We applied our framework to three garbage-scattering tasks with different distances: 20, 30, and 40 cm. The optimal parameter $\mathbf{w}^*$ is acquired when the next query parameter $\mathbf{w}'$ converges to a point. Human operators executed the task initialization, and we set the amount of pseudo-garbage in each grasp to around 120 to 300 g based on existing garbage-scattering systems for the actual waste crane. First, we experimented with a 30 cm scattering task and applied BO and our framework without reuse to confirm the effectiveness of the robustness. Second, we experimented with a 40 cm scattering task and applied our framework to confirm the effectiveness of the data reuse by reusing the data from the 30 cm task. Finally, we conducted a 20 cm task by reusing the data from the previous two tasks. Each experiment was executed three times. We evaluate

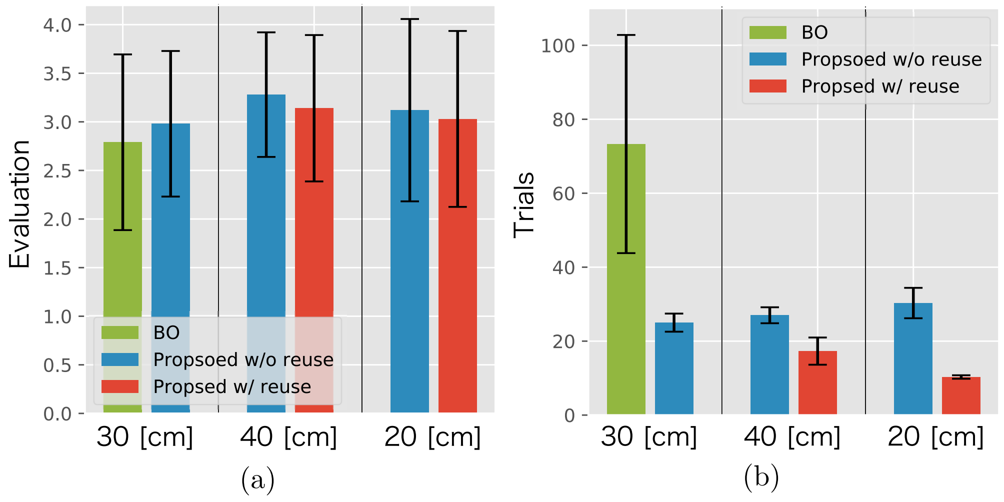


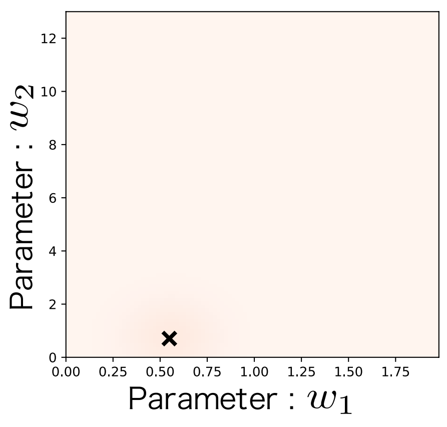
Figure 5.4.: Results of policy optimization with the robot waste crane. (a) Mean and standard deviation of evaluation of ten executions by the optimal policy learned by each method. (b) Mean and standard deviation of the number of trials to optimize policy parameters by each method in three experiments.

the performance of MTRBO by comparing the evaluation value of the optimized parameters and the number of trials for the optimization with BO and MTRBO without reuse. Moreover, we analyze the optimization process and the learned kernel gram matrix of task kernel in detail through visualization to show the effectiveness of robustness and sample-reuse of MTRBO.

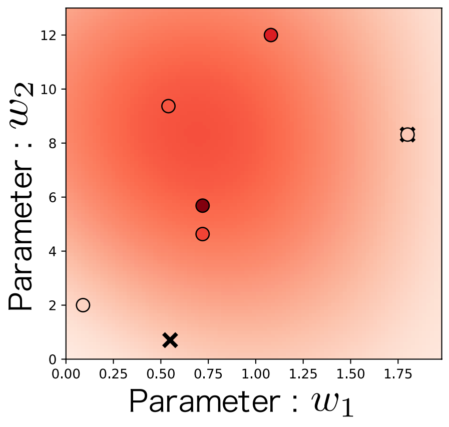
Results

Our experimental results are summarized in Fig. 5.4. In the 30 cm scattering task, BO and our framework obtained comparable high-evaluation policies. However, it optimized the policy with fewer trials than BO. In the 40 cm task, our framework further reduced the trials without degrading the performance and optimized the policy by reusing the previous task data. With the 20 cm task, our framework further reduced the trials without degrading the performance by reusing more previous task data.

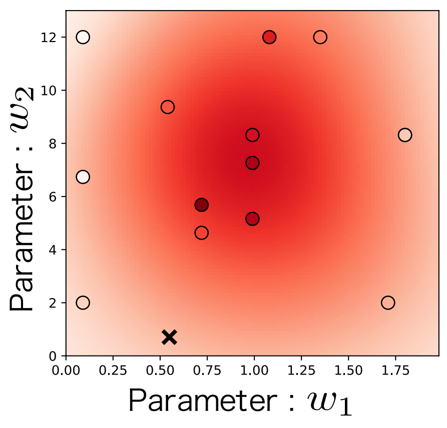
Figs. 5.5 and 5.6show the transition of the mean function in the optimization with our framework without reuse and BO. By inferring the data's reliability, our



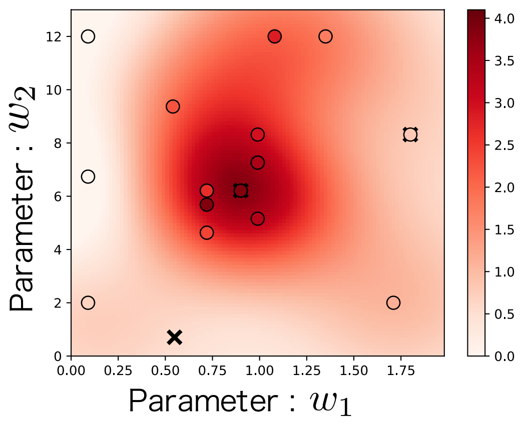
(a) 1 trial



(b) 8 trials



(c) 15 trials



(d) 22 trials

Figure 5.5.: Transition of mean function of our framework during 30 cm, garbage-scattering. Marker $\times$ indicates detected outlier in data.

framework treats data with greatly different trends as outliers and optimizes the parameter with fewer trials. Markers $\times$ in Fig. 5.5 indicate the outliers detected by our framework when we considered the data with reliability $E_q[\tau_n] < 0.2$ as outliers. Our framework detected four outliers out of 22 points of sample data. BO requires more trials for optimization since the target function becomes steeper due to the effect of such outliers.

Fig. 5.7 shows the task kernel. In Fig. 5.7 (a), the non-diagonal elements have high values, which suggest that the 40 cm task effectively reused the data collected in the 30 cm task. This is consistent with the results in Fig. 5.4. A similar trend was also observed in Fig. 5.7 (b).

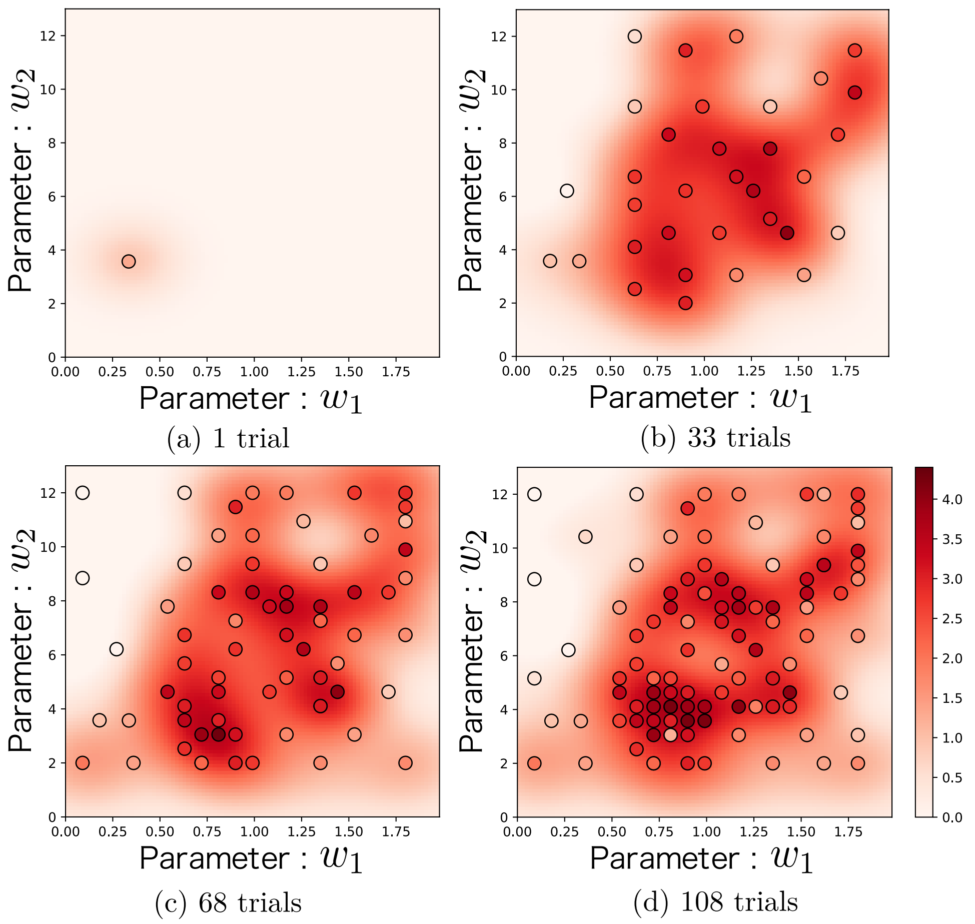
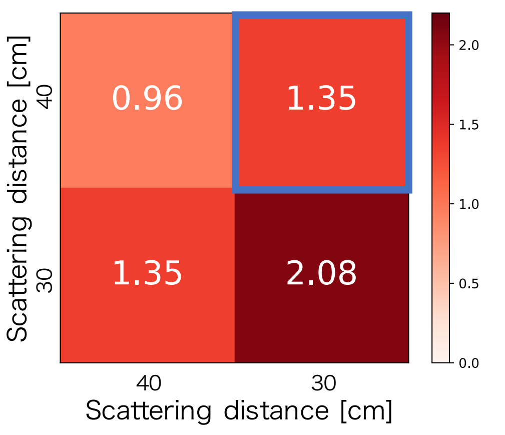


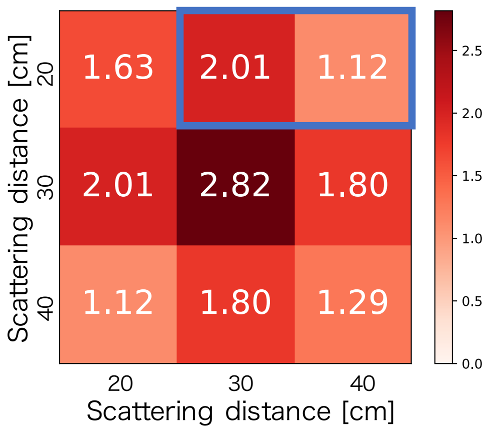
Figure 5.6.: Transition of mean function of GP in BO during experiment of 30 cm, garbage-scattering task.

Fig. 5.8 indicates garbage-scattering behaviors and sequences of the grasped garbage weight using the initial and optimized policy for the 30 cm task by the robot waste crane. Garbage scattering with the initial policy dropped almost all the garbage at the beginning. On the other hand, the behaviors with the optimized policy obtained a high evaluation by dropping the garbage three times.

In summary, all the experimental results with the robot waste crane demonstrated the effectiveness of the outlier robustness and the sample-reuse of our framework for waste crane policy optimization. Our framework efficiently acquired a high-performance policy sample.

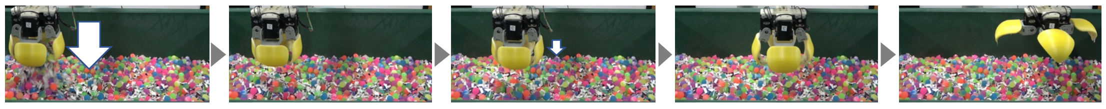


(a)

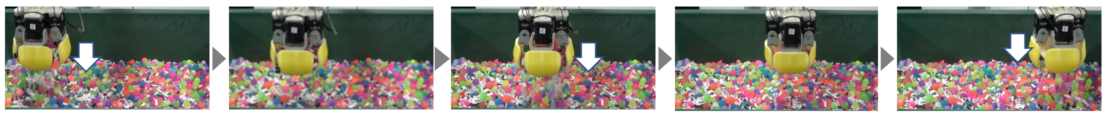


(b)

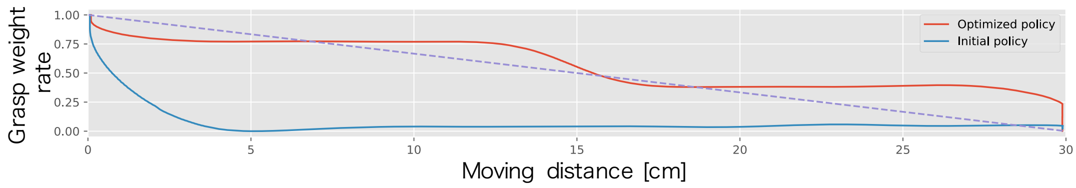
Figure 5.7.: Learned task kernel function with robot waste crane: (a) Optimization of 40 cm scattering task by reusing 30 cm task data. (b) Optimization of 20 cm scattering task by reusing 30 and 40 cm task data.



(a) With initial policy



(b) With optimized policy



(c) Weight sequences in garbage-scattering task

Figure 5.8.: Garbage-scattering behavior for 30 cm task with robot waste crane

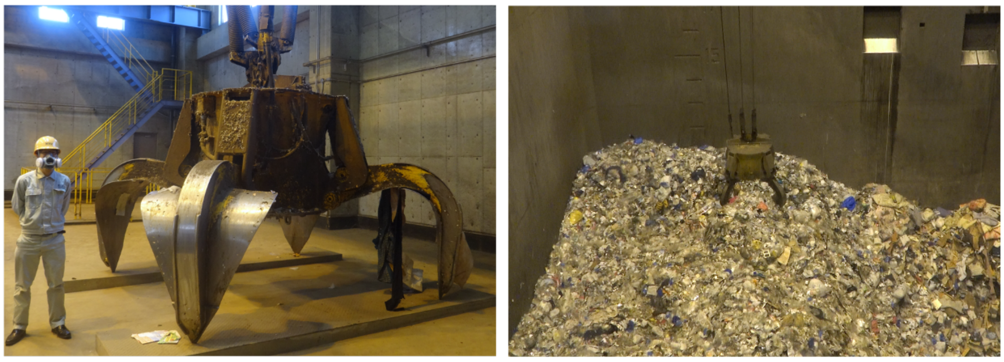

(a) A bucket (b) Garbage pit

Figure 5.9.: Actual waste crane

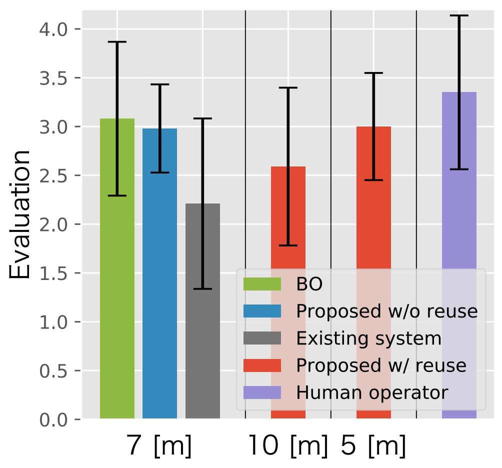
5.4.3. Policy Optimization with Actual Waste Crane

Actual Waste Crane

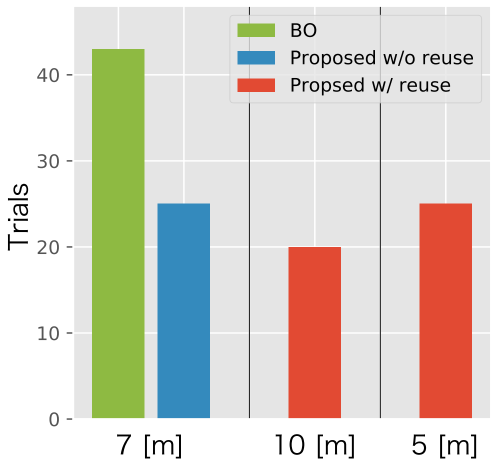
We experimented with a waste crane from an operational waste incineration plant in a city in Japan. Fig. 5.9 shows the plant's pit and the bucket. The crane's rated load is 2.45 t. The bucket has four hydraulically driven pawls that can be opened/closed or stopped. The operator can observe the hoist's position, the bucket's height, and the weight of the garbage grabbed by the bucket.

Experimental Settings

We applied our framework to three garbage-scattering tasks with different distances: 5, 7, and 10 m. An automatic control system executed the task initialization and grasps actual garbage in an amount to around 1.05 to 2.35 t. First, we experimented with a 5 m scattering task and applied BO and our framework without reuse to confirm the effectiveness of its robustness. Second, we experimented with a 10 m task and applied our framework to confirm the effectiveness of the data reuse by reusing the 7 m data. Finally, we applied our framework to the 5 m task by reusing the previous data of the two tasks. In the actual waste crane experiments, each experiment was executed one time due to limitation to system downtime of the plant at maintenance. We evaluate the performance of MTRBO as in the experiment with the robot waste crane. We compare the performance of the optimized policy with the performance of the operator by t-test.



(a)

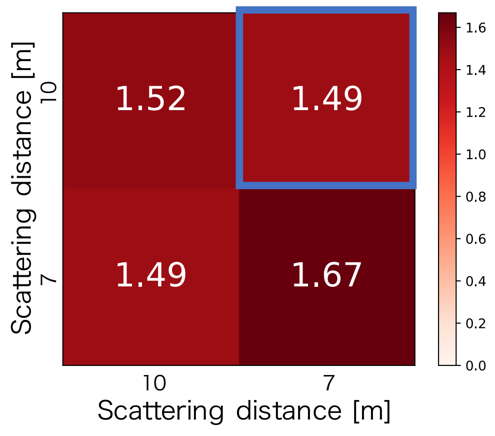


(b)

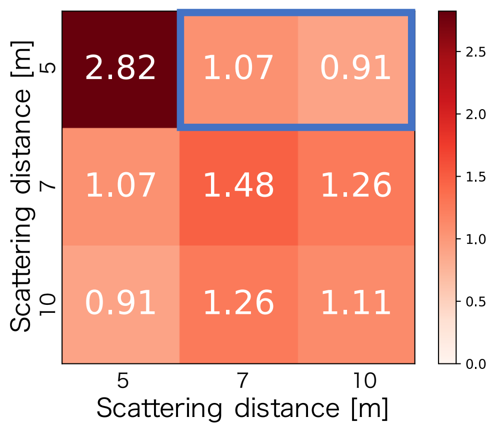
Figure 5.10.: Results of policy optimization with the actual waste crane. (a) Mean and standard deviation of evaluation of the optimal policy learned by each method. Learned policy optimized by BO-based method executed the task 10 times for evaluation. The existing system and human operator executed the task 38 and 25 times, respectively. (b) Mean and standard deviation of the number of trials to optimize policy parameters by each method in one experiment.

Results

Fig. 5.10 summarizes all the experiments with moving distances of 5, 7, and 10 m using the actual waste crane by BO and our framework without reuse. Even in experiments using the actual waste crane, our framework optimized the policies with fewer trials than BO. Our framework without reuse detected two outliers out of 25 points of 7 m sample data when we considered the data with reliability $E_q[\tau_n] < 0.2$ as outliers. In addition, the optimized policy acquired by our framework obtained a higher evaluation value than the existing systems used in the waste incineration plant. By verifying the evaluation value of the garbage-scattering behavior by the operator and the optimized policy acquired by our framework by the paired t-test, no significant difference was found ($p = 0.174 > 0.05$). As a result of verifying the evaluation value of the garbage-scattering behavior by the existing system and the policy acquired by our framework by the



(a)



(b)

Figure 5.11.: Learned task kernel function with actual waste crane: (a) Optimization of 10 m scattering task by reusing 7 m task data. (b) Optimization of 5 m scattering task by reusing 7 and 10 m task data.

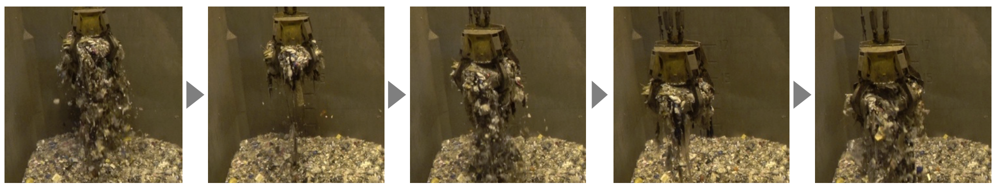
paired t-test, a significant difference was recognized ($p = 0.0212 < 0.05$).

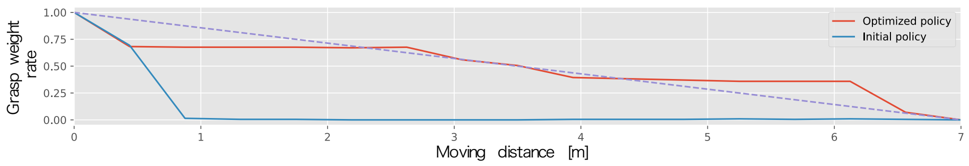
We conducted two experiments that optimized a policy by reusing data with different moving distances and compared the effectiveness of reuse, although the moving distances of the garbage-scattering task were different. As a result of the garbage-scattering task with a 10 m moving distance, the number of trials was reduced, as was the optimized policy's performance. As a result of the garbage-scattering task with a 5 m moving distance, the number of trials was not reduced, but the optimized policy's performance was high. Fig. 5.11 shows a task kernel that indicates the similarity of each task learned by our framework with reuse, which is also consistent with the trends in the number of trials (Fig. 5.10).

Fig. 5.12 indicates the garbage-scattering behaviors and sequences of the grasped garbage weight using the initial and optimized policy for the 7 m task by the actual waste crane. Almost all the garbage was dropped from the crane's grasp at the start of the task. On the other hand, the optimized policy obtained a high evaluation by effectively dropping the garbage three times.

Our experimental results with an actual waste crane demonstrated that our framework sample efficiently and robustly acquired a human-level policy for garbage-scattering tasks by an actual waste crane task.

(a) With initial policy


(b) With optimized policy



(c) Weight sequences in garbage-scattering task

Figure 5.12.: Garbage-scattering behavior for 7 m task with actual waste crane

5.5. Summary of Chapter 5

We proposed a framework of policy optimization for a waste crane from trial and error. Our framework employed MTRBO, whose characteristics include outlier robustness and sample reuse. Our framework's effectiveness was demonstrated by applying its proposed framework to a garbage-scattering task by a robot waste crane. Our experiments optimized the policy for garbage-scattering tasks with an actual waste crane and confirmed that our proposed framework learned a policy with the same performance as a human operator.

6. Discussion

This dissertation proposed GP-PS methods that are introduced latent variables to capture the complex mapping between states and optimal actions caused by environmental uncertainties in real-world tasks. Each proposed method is introduced latent variables to the policy model and derives the learning law of policies based on variable inference. The effectiveness of the proposed methods has been verified in simulation and robot experiments. In this chapter, we will discuss the proposed methods and their limitations.

6.1. Variational Policy Search for Multiple Optimal Actions

In chapter 3, multimodal SGP-PS and mode-seeking SGP-PS are derived. A learned policy by multimodal SGP-PS earned a higher return in simulations. However, it needs to decide the number of components affecting the performance before learning. Therefore, we could select a suitable one from the multimodal SGP-PS and the mode-seeking SGP-PS depending on whether the number of multimodalities of the function is known. The multimodal SGP-PS needs to decide the number of components since we use overlapping mixtures of GPs inspired by previous work [21]. We can also infer the number of components by incorporating the stick-breaking Dirichlet process model into the policy model [90].

In a mode-seeking SGP-PS, if there is a clear difference in the size of data among multiple optimal actions in the training data, learning tends to proceed faster since it fits the model. If the multiple optimal actions are observed almost evenly, the mode-seeking SGP-PS may struggle to capture one among them; in that case, the multimodal SGP-PS tends to be faster and more stable than the mode-seeking SGP-PS. Our experimental results may support the above discussion: For the table-sweeping task with more than three objects, the multimodal SGP-PS converged learning faster than the mode-seeking SGP-PS. For the table-sweeping task with one or two objects, the mode-seeking SGP-PS outperformed the multimodal SGP-PS.

6.2. Gaussian Process Self-triggered Policy Search for Weakly Observable Environment

In chapter 4, GPSTPS is derived and applied to a garbage-grasping-scattering task by simulation and a robotic waste crane and confirmed that it effectively incorporated the action execution duration. The limitation of GPSTPS is that such control strategies as grasping and scattering must be designed in advance. Applying STPGPS to more practical tasks may require multiple control strategies. The action policy should employ a multi-class GP classification model [91]. If the applied task has multimodal state transitions, its policy model can be extended by multimodality or robustness [7].

A future task is indispensable that experimentally verifies the possibility of applications with an actual waste crane. We must also verify the effectiveness of actual cranes/buckets in other weakly observable environments.

6.3. Multi-task Robust Bayesian Optimization for Unpredictable Fluctuation of Performance

In chapter 5, MTRBO is proposed to optimize control policy robustly and efficiently against unpredictable return fluctuation. The limitation of MTRBO is pre-design a parametric and task-specific policy model. The design of the policy model also affects the safety of the task execution by the policy. It may be essential to guarantee the safety of task execution by adding constraints when determining the next query parameter [92, 93].

In this dissertation, verification is performed only with two-dimensional policy parameters, but optimization efficiency may decrease when the policy parameters become higher-dimensional. In order to apply this method to more challenging tasks, it is necessary to extend it for optimizing high-dimensional parameters [94, 95].

6.4. Bias in Domain Knowledge

This dissertation has proposed a policy learning method that introduces latent variables as domain knowledge for the uncertainty of the environment of real-world tasks. Generally, domain knowledge is often biased and can adversely affect the learning of control policies. Since the proposed method assumes the prior distribution of latent variables and calculates the posterior distribution in the framework of Bayesian estimation, If the prior distribution is inappropriate for the task, the influence of the prior distribution decreases as the training data increases. Also, If the latent variable is improperly designed, posterior distributions that have little effect on learning are learned. However, the introduction of latent variables has the disadvantage of increasing the computational complexity of learning and prediction.

6.5. Open Issues

6.5.1. GP-based Policy Model

Our methods can compute the predictive distribution to determine action at any state since they employ GP-based policy models. Our methods explore state-action space by sampling the action using the predictive distribution, i.e., on-policy learning. In the multimodal SGP-PS, which has multiple components, a softmax function with the uncertainties of all the predictive distributions computes the probability of the component selection.

Although GP provides several advantages for our method, it has limitations in computational complexity. For tasks that require handling huge amounts of data, our methods require much computational complexity of learning and predictive distribution. Perhaps our methods are unsuitable for long-horizon tasks.

6.5.2. Kernel Function

In this dissertation, we have used only the Gaussian kernel function, one of the popular kernel functions. This kernel function is used widely domain, however, the Gaussian kernel function cannot handle raw images often used as input in reinforcement learning. In order to handle various data such as images as input of GP policies, it is necessary to consider the use of kernel functions other than the Gaussian kernel. As one of the solutions, a kernel function that incorporates the structure of convolutional neural networks and deep learning technology specializing in image processing was proposed [96]. Also, a method using deep learning as a kernel function has also been proposed [97, 98].

6.5.3. Designing safe control strategies and parametric policy model

Our proposed methods, GPST-PS and MTRBO, need task-specific and pre-designed control strategies or parametric policy models. Machines operated by humans daily, such as garbage cranes, can easily collect demonstration data. Since human operators can operate machines and robots safely, control strategies and parametric policy models can be designed to guarantee safety into consideration by using operation data. Imitation learning method with option proposed in [71] and motion segmentation methods [99, 100] is helpful to learn control strategies using demonstration data. In addition, it may be possible to design a parametric policy model suitable for the task by extracting the features from the demonstration data. They may be helpful for a wide range of applications.

7. Conclusion

In this dissertation, we aimed to automate real-world tasks by applying a non-parametric policy search method and proposed a novel policy search method that extends the policy model with latent variables to solve the problems of the conventional policy search method. Especially, we proposed methods focusing on 1) a reward function that appears multiple optimal actions for real-world tasks and 2) observations from an environment that contain little information about the state.

For the problem of multiple optimal actions, we proposed a multimodal SGP-PS and a mode-seeking SGP-PS that have different algorithm design concepts. The multimodal SGP-PS employs a multimodal policy prior inspired by OMGP to learn a policy that can capture multiple optimal actions. The mode-seeking SGP-PS learns a unimodal policy that captures one optimal action by employing an outlier-robust likelihood function. We derived the updating laws of both methods based on variational Bayesian inference.

To investigate the performance of multimodal SGP-PS and mode-seeking SGP-PS, we conducted two manipulation tasks: 1) a hand-posture adjustment task and 2) a table-sweeping task. We confirmed that our methods could learn suitable policies in an environment with multiple optimal actions.

For a little information of observation, we aimed to automate machines in weakly observable environments in an industrial workplace and proposed GP-STPS that can learn both action and duration policies to repeatedly perform the same action to overcome uncertainty. Its effectiveness was experimentally verified with simulations and a robotic waste crane system.

Finally, we proposed a framework of policy learning for an actual waste crane from trial and error. Our framework employed MTRBO, whose characteristics include outlier robustness and sample reuse. Our framework's effectiveness was demonstrated by applying its proposed framework to a garbage-scattering task by a robot waste crane. Our experiments optimized the policy for garbage-scattering tasks with an actual waste crane. They confirmed that our proposed framework learned a policy with the same performance as a human operator. Future work will investigate the effectiveness of our framework in other tasks with an actual waste crane.

Appendix

A. Multimodal SGP-PS

A.1. Derivation of analytical solutions

The analytical solution of variational distribution $q(\bar{\mathbf{f}}^{(m)})$ is obtained by solving the following equation:

$$\begin{aligned}\log q(\mathbf{f}^{(m)}) = \int \Big\{ &\log p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}, \mathbf{Z}) p(\{\mathbf{f}^{(m)} \mid \{\bar{\mathbf{f}}^{(m)}\}) p(\mathbf{Z}) \cdot \\ &p(\{\bar{\mathbf{f}}^{(m)}\}) \Big\} q(\{\bar{\mathbf{f}}^{(i \backslash m)}\}) q(\mathbf{Z}) \mathrm{d}\{\bar{\mathbf{f}}^{(i \backslash m)}\} \mathrm{d}\mathbf{Z} + C \end{aligned} \tag{A.1}$$

Here $\{\bar{\mathbf{f}}^{(i \backslash m)}\}$ indicates all the pseudo outputs without $\bar{\mathbf{f}}^{(m)}$. The following is the analytical solution of $q(\bar{\mathbf{f}}^{(m)})$:

$$\begin{aligned} q(\bar{\mathbf{f}}^{(m)}) &= \mathcal{N}(\bar{\mathbf{f}}^{(m)} \mid \boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)}), \\ \boldsymbol{\mu}^{(m)} &= \mathbf{K}_{\bar{\mathbf{s}}}^{(m)} \mathbf{Q}_L^{(m)^{-1}} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}}^{(m)} \mathbf{W} \mathbf{B}^{(m)} \mathbf{W} \mathbf{a}, \\ \boldsymbol{\Sigma}^{(m)} &= \mathbf{K}_{\bar{\mathbf{s}}}^{(m)} \mathbf{Q}_L^{(m)^{-1}} \mathbf{K}_{\bar{\mathbf{s}}}^{(m)}, \\ \mathbf{Q}_L^{(m)} &= \mathbf{K}_{\bar{\mathbf{s}}}^{(m)} + \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}}^{(m)} \mathbf{W} \mathbf{B}^{(m)} \mathbf{W} \mathbf{K}_{\mathbf{s},\bar{\mathbf{s}}}^{(m)}, \qquad (A.2) \\ \mathbf{B}^{(m)} &= \mathrm{diag}\left\{ \frac{\hat{\boldsymbol{\Pi}}_{nm}}{\lambda_n^{(m)} + \sigma^2} \right\}. \qquad (A.3) \end{aligned}$$

The analytical solution of variational distribution $q(\mathbf{Z})$ is obtained by solving the following equation:

$$\begin{aligned}\log q(\mathbf{Z}) = \int \Big\{ &\log p(\tilde{\mathbf{a}} \mid \{\mathbf{f}^{(m)}\}, \mathbf{Z}) p(\{\bar{\mathbf{f}}^{(m)} \mid \mathbf{f}^{(m)}) p(\mathbf{Z}) \cdot \\ &p(\{\bar{\mathbf{f}}^{(m)}\}) \Big\} q(\{\bar{\mathbf{f}}^{(m)}\}) \mathrm{d}\{\bar{\mathbf{f}}^{(m)}\} + C. \end{aligned} \tag{A.4}$$

The following is the analytical solution of $q(\mathbf{Z})$:

$$\begin{aligned}
q(\mathbf{Z}) &= \prod_{n=1,m=1}^{N,M} \hat{\mathbf{\Pi}}_{nm}^{\mathbf{Z}_{nm}}, \qquad\qquad \text{(A.5)}\\
\hat{\mathbf{\Pi}}_{nm} &= \mathbf{\Pi}_{nm} \exp(b_{nm}),\\
b_{nm} &= -\frac{1}{2}\log 2\pi(\lambda_n^{(m)} + \sigma^2) - \frac{\left(\mathbf{W}_{nn}a_n - \mathbf{W}_{nn}\mathbf{K}_{\mathbf{s}_n,\bar{\mathbf{s}}}^{(m)}\mathbf{K}_{\bar{\mathbf{s}}}^{(m)^{-1}}\boldsymbol{\mu}^{(m)}\right)^2}{2(\lambda_n^{(m)} + \sigma^2)}\\
&\quad - \frac{\left(\mathbf{W}_{nn}\mathbf{K}_{\mathbf{s}_n,\bar{\mathbf{s}}}^{(m)}\mathbf{Q}_L^{(m)^{-1}}\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_n}^{(m)}\mathbf{W}_{nn}\right)}{2(\lambda_n^{(m)} + \sigma^2)}.
\end{aligned}$$

The lower bound of the marginal likelihood is also obtained analytically:

$$\begin{aligned}
\log J_L' &= \sum_{m=1}^{M} -\frac{1}{2}\mathbf{a}^T\mathbf{W}\left(\mathbf{B}^{(m)^{-1}} + \mathbf{W}\mathbf{K}_{\mathbf{s},\bar{\mathbf{s}}}^{(m)}\mathbf{K}_{\bar{\mathbf{s}}}^{(m)^{-1}}\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}}^{(m)}\mathbf{W}\right)^{-1}\mathbf{W}\mathbf{a}\\
&\quad + \sum_{n=1,m=1}^{N,M} \log[\mathbf{R}^{(m)}]_{nn} - \mathrm{KL}(q(\mathbf{Z}) \mid\mid p(\mathbf{Z}))\\
&\quad - \frac{1}{2}\sum_{n=1,m=1}^{N,M} [\hat{\mathbf{\Pi}}]_{nm} \log 2\pi(\lambda_n^{(m)} + \sigma^2), \qquad\qquad \text{(A.6)}\\
\mathbf{R}^{(m)} &= \mathrm{chol}\left(\mathbf{I} + \mathbf{B}^{(m)^{-1/2}}\mathbf{W}\mathbf{K}_{\mathbf{s},\bar{\mathbf{s}}}^{(m)}\mathbf{K}_{\bar{\mathbf{s}}}^{(m)^{-1}}\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}}^{(m)}\mathbf{W}\mathbf{B}^{(m)^{-1/2}}\right). \qquad \text{(A.7)}
\end{aligned}$$

A.2. Predictive distribution

Compute the predictive distribution using the variational distribution instead of the true posterior distribution:

$$\begin{aligned}
p(a_*^{(m)} \mid \mathbf{s}_*) &\approx \int p(a_*^{(m)} \mid \bar{\mathbf{f}}^{(m)}\mathbf{s}_*)q(\bar{\mathbf{f}}^{(m)})\mathrm{d}\bar{\mathbf{f}}^{(m)}\\
&= \mathcal{N}\left(a_*^{(m)} \mid \mu_*^{(m)}, \sigma_*^{(m)}\right), \qquad\qquad \text{(A.8)}\\
\mu_*^{(m)} &= \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}}^{(m)}\mathbf{Q}_L^{(m)^{-1}}\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}}^{(m)}\mathbf{W}\mathbf{B}^{(m)}\mathbf{W}\mathbf{a},\\
\sigma_*^{(m)} &= \mathbf{K}_{\mathbf{s}_*}^{(m)} - \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}}^{(m)}(\mathbf{K}_{\bar{\mathbf{s}}}^{(m)^{-1}} - \mathbf{Q}_L^{(m)^{-1}})\mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_*}^{(m)} + \sigma^2.
\end{aligned}$$

B. Mode-seeking SGP-PS

B.1. Derivation of analytical solutions

The analytical solution of variational distribution $q(\tau_n)$ is obtained by solving the following equation:

$$\begin{aligned}\log q(\tau_n) = \int \Big\{ & \log p(\tilde{\mathbf{a}} \mid \mathbf{f}, \mathbf{T}) p(\mathbf{f} \mid \bar{\mathbf{f}}) p(\mathbf{T}) p(\bar{\mathbf{f}}) \Big\} \cdot \\ & q(\{\tau_{i\backslash n}\}) q(\bar{\mathbf{f}}) \mathrm{d}\{\tau_{i\backslash n}\} \mathrm{d}\bar{\mathbf{f}} + C, \end{aligned} \tag{B.9}$$

where $\{\tau_{i\backslash n}\}$ indicates all precision variables without τ_n. The following is the analytical solution of $q(\tau_n)$:

$$\begin{aligned} q(\tau_n) &= \mathrm{Gam}(\tau_n \mid a_n, b_n), \\ a_n &= \frac{\nu+1}{2}, \\ b_n &= \frac{\mathbf{W}_{nn}^2(\mathbf{a}_n - \mathbf{A}_n\boldsymbol{\mu})^2}{2} + \frac{\mathrm{tr}(\mathbf{A}_n^T \mathbf{W}_{nn}^2 \mathbf{A}_n \mathbf{C})}{2} \\ &+ \frac{\mathbf{W}_{nn}^2 \lambda_n + \nu\sigma^2}{2}, \\ \mathbf{A} &= \mathbf{K}_{\mathbf{s},\bar{\mathbf{s}}} \mathbf{K}_{\bar{\mathbf{s}}}^{-1}. \end{aligned} \tag{B.10}$$

The analytical solution of variational distribution $q(\bar{\mathbf{f}})$ is obtained by solving the following equation:

$$\begin{aligned} q(\bar{\mathbf{f}}) &= \mathcal{N}(\bar{\mathbf{f}} \mid \boldsymbol{\mu}, \mathbf{C}), \\ \boldsymbol{\mu} &= \mathbf{K}_{\bar{\mathbf{s}}} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}} \mathbf{W} \hat{\mathbf{T}} \mathbf{W} \mathbf{a}, \\ \mathbf{C} &= \mathbf{K}_{\bar{\mathbf{s}}} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{\bar{\mathbf{s}}}, \\ \boldsymbol{\Sigma} &= \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}} \mathbf{W} \hat{\mathbf{T}} \mathbf{W} \mathbf{K}_{\mathbf{s},\bar{\mathbf{s}}} + \mathbf{K}_{\bar{\mathbf{s}}}, \\ \hat{\mathbf{T}} &= \mathrm{diag}\{a_n/b_n\}. \end{aligned} \tag{B.11}$$

The lower bound of the marginal likelihood is also obtained analytically:

$$\begin{aligned} \log J'_L = &- \frac{N}{2} \log 2\pi - \frac{1}{2} \log |\hat{\mathbf{T}}^{-1}| \\ &- \frac{1}{2} (\mathbf{a} - \mathbf{A}\boldsymbol{\mu})^T \mathbf{W} \hat{\mathbf{T}} \mathbf{W} (\mathbf{a} - \mathbf{A}\boldsymbol{\mu}) - \frac{1}{2} \mathrm{tr}(\mathbf{A}^T \mathbf{W} \hat{\mathbf{T}} \mathbf{W} \mathbf{A} \mathbf{C}) \\ &- \mathrm{KL}(q(\bar{\mathbf{f}}) \mid\mid p(\bar{\mathbf{f}})) - \mathrm{KL}(q(\mathbf{T}) \mid\mid p(\mathbf{T})) - \frac{1}{2} \mathrm{tr}(\mathbf{W} \hat{\mathbf{T}} \mathbf{W} \boldsymbol{\Lambda}). \end{aligned}$$

B.2. Predictive distribution

Compute the predictive distribution using the variational distribution instead of the true posterior distribution:

$$
\begin{aligned}
p(a_* \mid \mathbf{s}_*) &\approx \int p(a_* \mid \mathbf{f}, \tau) p(\mathbf{f} \mid \bar{\mathbf{f}}) q(\bar{\mathbf{f}}) q(\tau) \mathrm{d}\mathbf{f} \mathrm{d}\bar{\mathbf{f}} \mathrm{d}\tau \\
&= \mathcal{N}(y_* \mid \mu_*, \sigma_*^2), \qquad \text{(B.12)} \\
\mu_* &= \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}} \mathbf{W} \hat{\mathbf{T}} \mathbf{W} \mathbf{a}, \\
\sigma_*^2 &= \mathbf{K}_* - \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} \mathbf{K}_{\bar{\mathbf{s}}}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_*} + \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} \boldsymbol{\Sigma}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}*} + \sigma^2.
\end{aligned}
$$

C. Hyperparameters of SAC and TRPO in the table-sweeping task

This section describes the hyperparameters of SAC and TRPO.

Table C.1.: SAC hyperparameters

Parameter	Value
optimizer	Adam
learning rate	3×10^{-4}
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers	2
number of hidden units per layer	128
number of sample per minibatch	256
nonlinearity	ReLU

Table C.2.: TRPO hyperparameters

Parameter	Value
optimizer	Adam
learning rate	1×10^{-3}
discount (γ)	0.995
replay buffer size	10^6
number of hidden layers	2
number of hidden units per layer	64
number of sample per minibatch	256
nonlinearity	Tanh
Stepsize (KL)	0.01

D. Multi-task Robust Bayesian Optimization

D.1. E-step: Optimize Variational Distributions

E-step optimizes variational distributions $q(\mathbf{f}_{\text{multi}})$ and $q(\tau_n)$ which maximize the lower bound F_v by updating the variational distribution alternatively. Update formula of variational distribution $q(\mathbf{f}_{\text{multi}})$ is follow:

$$q(\mathbf{f}_{\text{multi}}) = \mathcal{N}(\mathbf{f}_{\text{multi}} \mid \boldsymbol{\mu}_{\mathbf{f}}, \boldsymbol{\Sigma}_{\mathbf{f}}), \tag{D.13}$$

$$\boldsymbol{\mu}_{\mathbf{f}} = \boldsymbol{\Sigma}_{\mathbf{f}} \hat{\mathbf{T}} \mathbf{Y}_{\text{multi}}, \tag{D.14}$$

$$\boldsymbol{\Sigma}_{\mathbf{f}} = (\mathbf{K}_{\text{multi}}^{-1} + \hat{\mathbf{T}})^{-1}, \tag{D.15}$$

$$\hat{\mathbf{T}} = \text{diag}\{a_1/b_1, \cdots, a_n/b_n\}. \tag{D.16}$$

Update formula of variational distribution $q(\tau_n)$ is follow:

$$q(\tau_n) = \text{Gam}(\tau_n \mid a_n, b_n), \tag{D.17}$$

$$a_n = a + 1/2, \tag{D.18}$$

$$b_n = b + \frac{1}{2}\left((y_n - [\boldsymbol{\mu}_{\mathbf{f}}]_n)^2 - [\boldsymbol{\Sigma}_{\mathbf{f}}]_{nn}\right). \tag{D.19}$$

D.2. M-step: Optimize Model Parameters

M-step optimizes model distributions which maximize the lower bound F_v. The lower bound of the marginal likelihood is obtained analytically:

$$\begin{aligned} F_v = & -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\left|\hat{\mathbf{T}}^{-1}\right| - \frac{1}{2}(\mathbf{Y}_{\text{multi}} - \boldsymbol{\mu}_{\mathbf{f}})^T \hat{\mathbf{T}} (\mathbf{Y}_{\text{multi}} - \boldsymbol{\mu}_{\mathbf{f}}) \\ & - \frac{1}{2}\text{Tr}(\hat{\mathbf{T}}\boldsymbol{\Sigma}_{\mathbf{f}}) - \sum_{n=1}^{N} \text{KL}(q(\tau_n) \mid\mid p(\tau_n)) - \text{KL}(q(\mathbf{f}_{\text{multi}}) \mid\mid p(\mathbf{f}_{\text{multi}} \mid \mathbf{w}_{\text{multi}}, \mathbf{l})), \end{aligned} \tag{D.20}$$

where $\text{KL}(\cdot \mid\mid \cdot)$ is a Kullback-Leibler divergence.

E. Gaussian Process Self-triggered Policy Search

The predictive distribution of action policy and duration policy from any state $\mathbf{s}_*$ by the GPSTPS is shown as:

$$\begin{aligned}
p(a_* \mid \mathbf{s}_*) &= \int p(a_* \mid \mathbf{f}_*) p(\mathbf{f}_* \mid \mathbf{s}_*, \bar{\mathbf{f}}) p(\bar{\mathbf{f}} \mid \tilde{\mathbf{a}}) \, \mathrm{d}\mathbf{f}_* \mathrm{d}\bar{\mathbf{f}} \\
&\approx \int p(a_* \mid \mathbf{f}_*) p(\mathbf{f}_* \mid \mathbf{s}_*, \bar{\mathbf{f}}) q(\bar{\mathbf{f}}) \, \mathrm{d}\mathbf{f}_* \mathrm{d}\bar{\mathbf{f}} \\
&= \mathcal{N}(a_* \mid \hat{\mu}_f, \hat{\sigma}_f^2), \qquad \text{(E.21)} \\
\hat{\mu}_f &= \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} \mathbf{Q}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}} \mathbf{W} \sigma_f^2 \mathbf{I} \mathbf{W} (\mathbf{a} - m_f \mathbf{1}) + m_f, \qquad \text{(E.22)} \\
\hat{\sigma}_f^2 &= \mathbf{K}_{\mathbf{s}_*} - \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} (\mathbf{K}_{\bar{\mathbf{s}}}^{-1} - \mathbf{Q}_f^{-1}) \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_*} + \sigma_g^2, \qquad \text{(E.23)} \\
\mathbf{Q}_f &= \mathbf{K}_{\bar{\mathbf{S}},\mathbf{S}} \mathbf{W} \sigma_f^2 \mathbf{I} \mathbf{W} \mathbf{K}_{\mathbf{S},\bar{\mathbf{S}}} + \mathbf{K}_{\bar{\mathbf{S}}}, \qquad \text{(E.24)}
\end{aligned}$$

$$\begin{aligned}
p(\tau_* \mid s_*) &= \int p(\tau_* \mid g_*) p(g_* \mid \bar{\mathbf{g}}) p(\bar{\mathbf{g}} \mid \tilde{\boldsymbol{\tau}}) \, \mathrm{d}g_* \mathrm{d}\bar{\mathbf{g}} \\
&\approx \int p(\tau_* \mid g_*) p(g_* \mid \bar{\mathbf{g}}) q(\bar{\mathbf{g}}) \, \mathrm{d}g_* \mathrm{d}\bar{\mathbf{g}} \\
&= \mathcal{N}(\tau_* \mid \hat{\mu}_g, \hat{\sigma}_g^2), \qquad \text{(E.25)} \\
\hat{\mu}_g &= \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} \mathbf{Q}^{-1} \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}} \mathbf{W} \sigma_g^2 \mathbf{I} \mathbf{W} (\tau - m_g \mathbf{1}) + m_g, \qquad \text{(E.26)} \\
\hat{\sigma}_g^2 &= \mathbf{K}_{\mathbf{s}_*} - \mathbf{K}_{\mathbf{s}_*,\bar{\mathbf{s}}} (\mathbf{K}_{\bar{\mathbf{s}}}^{-1} - \mathbf{Q}_g^{-1}) \mathbf{K}_{\bar{\mathbf{s}},\mathbf{s}_*} + \sigma_g^2, \qquad \text{(E.27)} \\
\mathbf{Q}_g &= \mathbf{K}_{\bar{\mathbf{S}},\mathbf{S}} \mathbf{W} \sigma_g^2 \mathbf{I} \mathbf{W} \mathbf{K}_{\mathbf{S},\bar{\mathbf{S}}} + \mathbf{K}_{\bar{\mathbf{S}}}. \qquad \text{(E.28)}
\end{aligned}$$

Acknowledgements

I would like to express my sincere gratitude to Associate Professor Takamitsu Matsubara for his continued support and constructive advice. I learned about research activities, such as how to approach research, technical writing, and how to proceed with research projects from him. He also gave me the opportunity to learn from many projects in the laboratory, such as joint research with companies.

I would like to express my sincere gratitude to Professor Kenji Sugimoto for his advice and support for my research.

I also would like to thank Professor Tsukasa Ogasawara, Professor Takahiro Wada, and Assistant Professor Taisuke Kobayashi for their valuable advice in meetings at NAIST.

I would like to thank Assistant Professor Yoshihisa Tsurumine for my research and personal life support since I entered NAIST. In particular, He gave me beneficial advice on program implementation.

I am grateful to secretary Hideko Hayashi and Hiromi Nishimura for helping with the paperwork.

I appreciate Kaoru Kawabata, Terushi Hirabayashi, and Yukio Onuki of Hitachi Zosen Corporation for their support in joint research and experiments in the waste incinerator.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[2] B. Bischoff, D. Nguyen-Tuong, H. van Hoof, A. McHutchon, C. E. Rasmussen, A. Knoll, J. Peters, and M. Deisenroth, "Policy search for learning robot control using sparse data," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 3882–3887.

[3] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J. Mouret, "A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 328–347, 2020.

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2005.

[5] J. Kober, E. Öztop, and J. Peters, "Reinforcement Learning to Adjust Robot Movements to New Situations," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011, pp. 2650–2655.

[6] H. van Hoof, G. Neumann, and J. Peters, "Non-parametric Policy Search with Limited Information Loss," *Journal of Machine Learning Research*, vol. 18, no. 73, pp. 1–46, 2017.

[7] H. Sasaki and T. Matsubara, "Variational policy search using sparse Gaussian process priors for learning multimodal optimal actions," *Neural Networks*, vol. 143, pp. 291–302, 2021.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[9] E. Rohmer, S. P. N. Singh, and M. Freese, “V-rep: A versatile and scalable robot simulation framework,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1321–1326.

[10] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[11] Y. Tsurumine, Y. Cui, K. Yamazaki, and T. Matsubara, “Generative Adversarial Imitation Learning with Deep P-Network for Robotic Cloth Manipulation,” in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 274–280.

[12] M. Titsias, “Variational Learning of Inducing Variables in Sparse Gaussian Processes,” in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[13] X. Zhu and N. Wang, “Cuckoo search algorithm with membrane communication mechanism for modeling overhead crane systems using RBF neural networks,” *Applied Soft Computing*, vol. 56, pp. 458–471, 2017.

[14] H. Maske, E. Kieson, G. Chowdhary, and C. Abramson, “Learning Task-Based Instructional Policy for Excavator-Like Robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1962–1969.

[15] E. Snelson and Z. Ghahramani, “Sparse Gaussian processes using pseudo-inputs,” in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 1257–1264.

[16] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient Global Optimization of Expensive Black-Box Functions,” *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[17] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, “Bayesian Optimization for Learning Gaits Under Uncertainty,” *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 5–23, 2016.

[18] G. Neumann, "Variational inference for policy search in changing situations," in *International Conference on Machine Learning (ICML)*, 2011, pp. 817–824.

[19] S. Levine and V. Koltun, "Variational policy search via trajectory optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 207–215.

[20] K. S. Luck, J. Pajarinen, E. Berger, V. Kyrki, and H. B. Amor, "Sparse latent space policy search," in *Conference on Artificial Intelligence (AAAI)*, 2016, p. 1911–1918.

[21] M. Lázaro-Gredilla, S. V. Vaerenbergh, and N. D. Lawrence, "Overlapping mixtures of Gaussian processes for the data association problem," *Pattern Recognition*, vol. 45, no. 4, pp. 1386–1395, 2012.

[22] J. Vanhatalo, P. Jylänki, and A. Vehtari, "Gaussian process regression with student-t likelihood," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1910–1918.

[23] P. Jylänki, J. Vanhatalo, and A. Vehtari, "Robust Gaussian Process Regression with a Student-t Likelihood," *Journal of Machine Learning Research*, vol. 12, no. Nov, pp. 3227–3257, 2011.

[24] J. A. D. Bagnell and J. Schneider, "Policy search in reproducing kernel hilbert space," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-45, November 2003.

[25] G. Lever and R. Stafford, "Modelling Policies in MDPs in Reproducing Kernel Hilbert Space," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015, pp. 590–598.

[26] N. A. Vien, P. Englert, and M. Toussaint, "Policy search in reproducing kernel hilbert space," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016, pp. 2089–2096.

[27] S. Calinon, P. Kormushev, and D. G. Caldwell, “Compliant skills acquisition and multi-optima policy search with EM-based reinforcement learning,” *Robotics and Autonomous Systems*, vol. 61, no. 4, pp. 369–379, 2013.

[28] C. Daniel, G. Neumann, O. Kroemer, and J. Peters, “Hierarchical relative entropy policy search,” *Journal of Machine Learning Research*, vol. 17, no. 93, pp. 1–50, 2016.

[29] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, and G. Neumann, “Model-based contextual policy search for data-efficient generalization of robot skills,” *Artificial Intelligence*, vol. 247, pp. 415 – 439, 2017.

[30] T. Osa and M. Sugiyama, “Hierarchical policy search via return-weighted density estimation,” in *Conference on Artificial Intelligence (AAAI)*, 2018, pp. 3860–3867.

[31] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in *International Conference on Machine Learning (ICML)*, 2017, pp. 1352–1361.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[33] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft Actor-Critic Algorithms and Applications,” *arXiv:1812.05905 [cs, stat]*, 2019.

[34] T. Haarnoja, “Acquiring diverse robot skillsvia maximum entropy deep reinforcement learning,” Ph.D. dissertation, University of California, Berkeley, 2018.

[35] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, “Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning (CoRL)*, 2018, pp. 651–673.

[36] V. Tresp, "A Bayesian committee machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[37] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS)*, 2001, pp. 881–888.

[38] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 883–890.

[39] C. Yuan and C. Neubauer, "Variational mixture of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 1897–1904.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30.

[41] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2017, pp. 2817–2826.

[42] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2018.

[43] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4299–4307.

[44] M. Sugiyama, H. Hachiya, H. Kashima, and T. Morimura, "Least absolute policy iteration for robust value function approximation," in *IEEE/RSJ International Conference on Robotics and Automation (IROS)*, 2009, pp. 2904–2909.

[45] J. Wang, Y. Liu, and B. Li, “Reinforcement Learning with Perturbed Rewards,” *arXiv:1810.01032 [cs, stat]*, 2018.

[46] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust Region Policy Optimization,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897.

[47] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv:1509.02971 [cs, stat]*, 2015.

[48] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, “Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research,” *arXiv:1802.09464 [cs]*, 2018.

[49] L. Ramli, Z. Mohamed, A. M. Abdullahi, H. I. Jaafar, and I. M. Lazim, “Control strategies for crane systems: A comprehensive review,” *Mechanical Systems and Signal Processing*, vol. C, no. 95, pp. 1–23, 2017.

[50] R. el Asri and D. Baxter, “Process Control in Municipal Solid Waste Incinerators: Survey and Assessment,” *Waste Management & Research*, vol. 22, no. 3, pp. 177–185, 2004.

[51] K. J. Mackin and M. Fujiyoshi, “Intelligent Waste Crane Scheduling using Evolutionary Computation,” in *International Conference on Soft Computing and Intelligent Systems and International Symposium on Advanced Intelligent Systems*, 2018, pp. 689–692.

[52] T. Kaneko, Y. Tsurumine, J. Poon, Y. Onuki, Y. Dai, K. Kawabata, and T. Matsubara, “Learning Deep Dynamical Models of a Waste Incineration Plant from In-Furnace Images and Process Data,” in *IEEE International Conference on Automation Science and Engineering*, 2019, pp. 873–878.

[53] H. Sasaki, T. Hirabayashi, K. Kawabata, Y. Onuki, and T. Matsubara, “Bayesian Policy Optimization for Waste Crane With Garbage Inhomogeneity,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4533–4540, 2020.

[54] A. M. Abdullahi, Z. Mohamed, H. Selamat, H. R. Pota, M. S. Zainal Abidin, F. S. Ismail, and A. Haruna, “Adaptive output-based command shaping for sway control of a 3D overhead crane with payload hoisting and wind disturbance,” *Mechanical Systems and Signal Processing*, vol. 98, pp. 157–172, 2018.

[55] N. Ku, J. Cha, M. Roh, and K. Lee, “A tagline proportional–derivative control method for the anti-swing motion of a heavy load suspended by a floating crane in waves,” *Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, pp. 357–366, 2012.

[56] Y. Qian, Y. Fang, and B. Lu, “Adaptive repetitive learning control for an offshore boom crane,” *Automatica*, vol. 82, pp. 21–28, 2017.

[57] D. Jud, G. Hottiger, P. Leemann, and M. Hutter, “Planning and Control for Autonomous Excavation,” *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2151–2158, 2017.

[58] F. E. Sotiropoulos and H. H. Asada, “A Model-Free Extremum-Seeking Approach to Autonomous Excavator Control Based on Output Power Maximization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1005–1012, 2019.

[59] R. J. Sandzimier and H. H. Asada, “A Data-Driven Approach to Prediction and Optimal Bucket-Filling Control for Autonomous Excavators,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2682–2689, 2020.

[60] Q. Lu and L. Zhang, “Excavation Learning for Rigid Objects in Clutter,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7373–7380, 2021.

[61] S. Dadhich, F. Sandin, U. Bodin, U. Andersson, and T. Martinsson, “Field test of neural-network based automatic bucket-filling algorithm for wheel-loaders,” *Automation in Construction*, vol. 97, pp. 1–12, 2019.

[62] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, “An introduction to event-triggered and self-triggered control,” in *IEEE IEEE Conference on Decision and Control (CDC)*, 2012, pp. 3270–3285.

[63] C. Peng and F. Li, “A survey on recent advances in event-triggered communication and control,” *Information Sciences*, vol. 457-458, pp. 113–125, 2018.

[64] J. D. J. B. Berglind, T. M. P. Gommans, and W. P. M. H. Heemels, “Self-triggered MPC for constrained linear systems and quadratic costs,” *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 342–348, 2012.

[65] F. D. Brunner, M. Heemels, and F. Allgöwer, “Robust self-triggered MPC for constrained linear systems: A tube-based approach,” *Automatica*, vol. 72, pp. 73–83, 2016.

[66] M. Kishida, “Event-triggered Control With Self-triggered Sampling for Discrete-time Uncertain Systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1273–1279, 2019.

[67] X. Yi, K. Liu, D. V. Dimarogonas, and K. H. Johansson, “Dynamic Event-Triggered and Self-Triggered Control for Multi-agent Systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3300–3307, 2019.

[68] J. Wang, X. Ma, H. Li, and B. Tian, “Self-triggered sliding mode control for distributed formation of multiple quadrotors,” *Journal of the Franklin Institute*, vol. 357, no. 17, pp. 12 223–12 240, 2020.

[69] K. Hashimoto, Y. Yoshimura, and T. Ushio, “Learning Self-Triggered Controllers With Gaussian Processes,” *IEEE Transactions on Cybernetics*, pp. 1–11, 2020.

[70] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[71] C. Daniel, H. van Hoof, J. Peters, and G. Neumann, “Probabilistic inference for determining options in reinforcement learning,” *Machine Learning*, vol. 104, no. 2, pp. 337–357, 2016.

[72] P. Bacon, J. Harb, and D. Precup, “The option-critic architecture,” in *AAAI Conference on Artificial Intelligence*, 2017, pp. 1726–1734.

[73] S. Zhang and S. Whiteson, “DAC: The Double Actor-Critic Architecture for Learning Options,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 2012–2022, 2019.

[74] A. Harutyunyan, W. Dabney, D. Borsa, N. Heess, R. Munos, and D. Precup, “The Termination Critic,” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019, pp. 2231–2240.

[75] M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, and M. Riedmiller, “Data-efficient Hindsight Off-policy Option Learning,” in *International Conference on Machine Learning (ICML)*, 2021, pp. 11 340–11 350.

[76] R. L. Neitzel, N. S. Seixas, and K. K. Ren, “A Review of Crane Safety in the Construction Industry,” *Applied Occupational and Environmental Hygiene*, vol. 16, no. 12, pp. 1106–1117, 2001.

[77] A. Walawalkar, S. Heep, C. Schindler, R. Leifeld, and M. Frank, “Validation of an analytical method for payload estimation in excavators,” in *Commercial Vehicle Technology (CVT)*, 2018, pp. 3–16.

[78] F. Li, Q. Jiang, S. Zhang, M. Wei, and R. Song, “Robot skill acquisition in assembly process using deep reinforcement learning,” *Neurocomputing*, vol. 345, pp. 92–102, 2019.

[79] E. Johns, S. Leutenegger, and A. J. Davison, “Deep learning a grasp function for grasping under gripper pose uncertainty,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4461–4468.

[80] B. J. Hodel, “Learning to Operate an Excavator via Policy Optimization,” *Procedia Computer Science*, vol. 140, pp. 376–382, 2018.

[81] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2951–2959.

[82] T. Matsubara, Y. Funaki, M. Ding, T. Ogasawara, and K. Sugimoto, "Data-efficient human training of a care motion controller for human transfer assistant robots using Bayesian optimization," in *International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2016, pp. 606–611.

[83] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in *International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 1069–1074.

[84] J. Nogueira, R. Martinez-Cantin, A. Bernardino, and L. Jamone, "Unscented Bayesian optimization for safe robot grasping," in *International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1967–1972.

[85] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics," *arXiv:1602.04450 [cs]*, 2018.

[86] A. Rai, R. Antonova, F. Meier, and C. G. Atkeson, "Using Simulation to Improve Sample-Efficiency of Bayesian Optimization for Bipedal Robots," *Journal of Machine Learning Research*, vol. 20, no. 49, pp. 1–24, 2019.

[87] R. Antonova, A. Rai, T. Li, and D. Kragic, "Bayesian Optimization in Variational Latent Spaces with Dynamic Compression," in *3rd Conference on Robot Learning (CoRL)*, 2019, pp. 3G–11.

[88] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task Gaussian Process Prediction," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 153–160.

[89] K. Swersky, J. Snoek, and R. P. Adams, "Multi-Task Bayesian Optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2004–2012.

[90] J. C. Ross and J. G. Dy, "Nonparametric mixture of Gaussian processes with constraints," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1346–1354.

[91] T. Galy-Fajou, F. Wenzel, C. Donner, and M. Opper, "Multi-Class Gaussian Process Classification Made Conjugate: Efficient Inference via Data Augmentation," in *Uncertainty in Artificial Intelligence*, 2020, pp. 755–765.

[92] L. P. Fröhlich, M. N. Zeilinger, and E. D. Klenske, "Cautious Bayesian Optimization for Efficient and Scalable Policy Search," in *Learning for Dynamics and Control*, 2021, pp. 227–240.

[93] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, Jun. 2021.

[94] M. Mutný and A. Krause, "Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 9019–9030.

[95] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka, "Batched Large-scale Bayesian Optimization in High-dimensional Spaces," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018, pp. 745–754.

[96] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, "Deep convolutional networks as shallow Gaussian processes," in *International Conference on Learning Representations (ICLR)*, 2019.

[97] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing, "Stochastic Variational Deep Kernel Learning," in *Advances in Neural Information Processing Systems*, vol. 29, 2016, pp. 2586–2594.

[98] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 51, 2016, pp. 370–378.

[99] T. Nakamura, T. Nagai, D. Mochihashi, I. Kobayashi, H. Asoh, and M. Kaneko, "Segmenting Continuous Motions with Hidden Semi-markov Models and Gaussian Processes," *Frontiers in Neurorobotics*, vol. 11, p. 67, 2017.

[100] M. Nagano, T. Nakamura, T. Nagai, D. Mochihashi, I. Kobayashi, and W. Takano, "HVGH: Unsupervised Segmentation for High-Dimensional Time Series Using Deep Neural Compression and Statistical Generative Model," *Frontiers in Robotics and AI*, vol. 6, 2019.

Publication List

Journal

[1] Hikaru Sasaki, Terushi Hirabayashi, Kaoru Kawabata, Yukio Onuki, and Takamitsu Matsubara, “Bayesian Policy Optimization for Waste Crane with Garbage Inhomogeneity,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4533-4540, July 2020.
[Correspond to Chapter 3]

[2] Hikaru Sasaki and Takamitsu Matsubara, “Variational Policy Search using Sparse Gaussian Process Priors for Learning Multimodal Optimal Actions,” *Neural Networks*, vol. 143, pp. 291-302, November 2021.
[Correspond to Chapter 5]

[3] Hikaru Sasaki, Terushi Hirabayashi, Kaoru Kawabata, and Takamitsu Matsubara, “Gaussian Process Self-triggered Policy Search in Weakly Observable Environments,” *IEEE Robotics and Automation Letters*, (Submitted).
[Correspond to Chapter 4]

International Conference (Reviewed)

[1] Hikaru Sasaki and Takamitsu Matsubara, “Multimodal Policy Search using Overlapping Mixtures of Sparse Gaussian Process Prior,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp.2433-2439, May 2019.

[2] Hikaru Sasaki, Terushi Hirabayashi, Kaoru Kawabata, Yukio Onuki, and Takamitsu Matsubara, “Bayesian Policy Optimization for Waste Crane with Garbage Inhomogeneity,” *IEEE International Conference on Automation Science and Engineering (CASE)*, August 2020.
(as presentation option of Robotics and Automation Letters)

Domestic Conference

[1] 佐々木光，松原崇充，“ConvNet カーネルを用いた多峰性ガウス過程方策探索による生画像からの行動学習，” 第37回日本ロボット学会学術講演会（RSJ2019），3A1-05，2019.

[2] 佐々木光，平林照司，川端馨，小貫由樹雄，松原崇充，“ロバストベイズ最適化によるゴミクレーンの動作最適化，” 第 7 回 計測自動制御学会 制御部門マルチシンポジウム（MCSC2020），3C2-4，2020.

[3] 佐々木光，松原崇充，“ガウス過程に基づく自己駆動型方策による方策探索，” ロボティクス・メカトロニクス講演会 2021 （ROBOMECH2021），1P1-I17, 2021.

Award

[1] 佐々木光, 日本機械学会若手優秀講演フェロー賞，2019 年 6 月 6 日.

Patent

[1] 松原崇充，佐々木光，川端馨，平林照司，小貫由樹雄，戴英達,【発明名称】情報処理装置，制御システム，制御変数決定方法，および制御変数決定プログラム,【出願番号】2020-33904,【出願日】2020.2.28.

Others

Journal

[1] Tomoya Miyamoto, Hikaru Sasaki, Takamitsu Matsubara, “Exploiting Visual-outer Shape for Tactile-inner Shape Estimation of Objects Covered with Soft Materials,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6278-6285, October 2020.

International Conference (Reviewed)

[1] Yuhwan Kwon, Takumi Kaneko, Yoshihisa Tsurumine, Hikaru Sasaki, Kimiko Motonaka, Seiji Miyoshi and Takamitsu Matsubara, “Combining Model Predictive Path Integral with Kalman Variational Auto-Encoder for Robot Control from Raw Images,” *IEEE/SICE International Symposium on System Integration (SII)*, pp. 271-276, August 2020.

[2] Tomoya Miyamoto, Hikaru Sasaki, Takamitsu Matsubara, “Exploiting Visual-outer Shape for Tactile-inner Shape Estimation of Objects Covered with Soft

Materials," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , October 2020.
(as presentation option of Robotics and Automation Letters)

[3] Hanbit Oh, Hikaru Sasaki, Brendan Michael, Takamitsu Matsubara, "Bayesian Disturbance Injection: Robust Imitation Learning of Flexible Policies," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8629–8635, October 2021.

Domestic Conference

[1] 宮本知弥，佐々木光，松原崇充，ガウス過程陰関数曲面に基づく不確実な 3 次元形状情報を用いた滑り動作計画，ロボティクス・メカトロニクス 講演会 (ROBOMECH2019)，2A2-C06，2019.

[2] 宮本知弥，佐々木光，松原崇充，柔軟素材で覆われた剛体の効率的な形状推定，第 20 回計測自動制御学会システムインテグレーション部門講演会 (SI2019)，3E2-07，2019.

[3] 山之口智也，鶴峯義久，佐々木光，内部英治，森本淳，松原崇充，潜在動的モデルを持つ real-to-sim 画像変換の学習，ロボティクス・メカトロニクス 講演会 (ROBOMECH2020)，1P1-B07，2020.

[4] 莫亜強，佐々木光，松原崇充，山崎公俊，個々の動作と動作切り替えを同時に考慮した方策学習に基づく手順あり作業の実行能力獲得，第 38 回日本ロボット学会学術講演会 (RSJ2020)，3D2-05，2020.

[5] Hanbit Oh，佐々木光，松原崇充，無限重複混合ガウス過程に基づく頑健・柔軟な模倣学習，第 38 回日本ロボット学会学術講演会 (RSJ2020)，1C2-02，2020.

[6] 田原熙昂，Oh Hanbit，佐々木光，松原崇充 (奈良先端大)，タスク達成度を考慮した教示者に摂動を加えるロバスト模倣学習，第 21 回計測自動制御学会システムインテグレーション部門講演会 (SI2020)，3D3-13，2020.

[7] 莫亜強，佐々木光，松原崇充，山崎公俊，"人の実演教示に基づく個々の動作と動作の切り替えを考慮した手順あり作業の実行能力獲得，" 第 39 回日本ロボット学会学術講演会（RSJ2021），3I1-04，2021.

[8] Hanbit Oh, Hikaru Sasaki, Brendan Michael, and Takamitsu Matsubara, "Bayesian Disturbance Injections: Safely learning robust and flexible policies，" 第 39

回日本ロボット学会学術講演会（RSJ2021），2A3-04，2021.