NORTHWESTERN UNIVERSITY

Optimal Experimental Learning and Infinite Linear Embeddings

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

Ian Abraham

EVANSTON, ILLINOIS

September 2020

Abstract

Optimal Experimental Learning and Infinite Linear Embeddings

Ian Abraham

In the current state of robotics, the systems we create are heavily reliant on our consistent guidance, programming of tasks, and oracle information that allow them to operate in the world that we inhabit. What happens to our robotic systems when we are unable to perform as an oracle, creating the absence of information about what is known and unknown to our robots? Can we expect our robotic systems to operate in our world? And what are the necessary requirements for them to explore and navigate the increasing complexities that we face in an unknown world? In this thesis, I argue that robotic systems are required to have the ability to intentionally learn, model, and explore what is unknown about the world for them to be less reliant on the oracle information that we as the developers provide.

This thesis establishes the stated problems as one of active robot learning and decision making. Through the use of existing methods and tools from hybrid control theory, this thesis first looks to

enhance the capabilities of the current approaches for robot learning and lays the groundwork for subsequent theoretical advancements for learning and control. Active learning through automated experimental design is then motivated as an approach that enables robotic systems to develop actions that intentionally seek out informative measurements for learning what is unknown. Coupled with methods from ergodicity and ergodic exploration, active learning is shown to be a promising approach for modeling complex and spatially sparse environments using only rudimentary contact sensing. These results are extended to the case of safe exploration and active learning in dynamic state-spaces where robot safety and the quality of informative measurements are provably balanced through Lyapunov attractiveness and hybrid control theoretic analysis. Last, I argue that we should not only care about active learning, but also how we model and represent what robotic systems are learning. The class of infinite linear embeddings is presented as a candidate model that simplifies and improves the control and active learning capabilities of robotic systems. Through simulated and experimental application, I illustrate the potential of the presented approaches for pushing the boundaries of robotic systems towards being more capable, self-sufficient, and curious systems that intentionally learns the unknown and complex nature of interacting in our world.

Acknowledgements

I would first and foremost like to thank my advisor, Todd Murphey for his advising and research freedom he has provided me all these years. Our (often heated) discussions where we argue completely unrelated topics has taught me to constantly question and analyze my presentation which allowed me to become more articulate and purposeful in my own research. It is within these discussions that I have learned to control my hubris and grow not only as a researcher but as a person. To my labmates old and new, I would like to thank them for their ability to sit there and listen to me talk about problems that I probably, and abruptly, realized the solution to mid conversation and leave. Without their patience and mild interest, many problems in this thesis would be left unsolved. I would also like to thank my closets collaborators, Alex, Ahalya, Barrett, and Ankur. Working with them has been a genuine pleasure. If it wasn't for their insightful input, vision, and mentorship, my approach to research and the quality of my work would not be the way it is today. I am proud of all the work that we have accomplished and I look forward to the prospect of (and planned) future work that we will achieve. I would also like to that those that have been with me through thick and thin. Through ups and downs, long evenings shop-talking, late night burritos

runs, blindly arguing random topics, upside-down hooks, and the occasional log derivative that continually slips my mind. Thank you for the support, trust, and guidance.

To my closest friends, my brothers from uniquely, and technically unrelated mothers. Roche, Pablo, I want to thank them for their support throughout my entire life and for reminding me that life is meant to be enjoyed. All of our adventures and fond memories are what keep me grounded to this day.

Last, I want to thank my family for all that they done. I would not be here today if it wasn't for their support and the difficult choice to leave everything behind and start over in a place where I have opportunities. Gracias por todo los sacrificios y el apoyo para ser lo mejor que puedo ser. Esta tesis esta dedicada a ustedes.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

What hidden assumptions do we make when we develop and analyze systems that operate autonomously in the world? Typically, we assume that these systems have a way of interacting in the world, a way of observing the world through various sensing modalities, an understanding of the world through predictive models, some knowledge of uncertainty, and a signal associated with good and bad behavior while interacting in the world. In the case for robotic systems, these assumptions are synonymous with having a model of robot dynamics and the interaction with the physical world, a way to measure the state of the robot, a guess at what might not be known to the robot, and a reward or cost function. But what happens when these assumptions are not present or missing? Can the autonomous systems properly function and interact with the world to discover and learn the necessary information to achieve success at an intended task?

For robotic systems that interact with the physical world, this problem is one which involves exploring and learning with respect to the constraints of the sensors and physical interactions.

In order for robots to construct and learn what is necessary to operate in the world, we cannot simply tell the robot "go and learn X from observing Y". The robot needs to take action and move through the physical world to observe what is needed to learn. However, choosing what needs to be observed is often arbitrary and a passive process. In addition, choosing what to observe given time and physical restrictions adds constraints that exasperates the problem. This problem is underscored by the fact that observing redundant measurements yields redundant models. Thus action, sensing, and measurements are intertwined and must be chosen and optimized with intent to improve the quality and efficacy of what is being learned with the consideration of the robot's safety in mind. Moreover, how we choose to model and structure what is being learned comes with its own limitations. These are questions that are fundamental to robot learning and control that are constantly appearing as robotic systems start to become more autonomous and independent of human operators.

1.1 Contributions and related work

This thesis addresses the problem of intentional learning in robotics where actions must be chosen such that they result in information-rich measurements that improve the efficiency of learning tasks and respect the physical constraints of robotic systems. This is accomplished through the use of tools and concepts from hybrid control theory, optimal experimental design, ergodic exploration, and Koopman operator theory that improves and optimizes the learning capabilities of robotic systems into an active and intentional process grounded by principled mathematics. These principled approaches provide theoretical guarantees for active learning and safety that extends to the physical domain which is presented through simulated and experimental robot examples throughout this thesis.

Thus, the contribution of this thesis is the development of fast and efficient methods for online active learning that are readily deployed on robotics systems that operate subject to safety constraints.

This thesis resides at the intersection of optimal control, reinforcement learning, sensing, active learning, and optimal experimental design. As a result, there is a plethora of topics which relate to each of these fields of study individually. Conveniently, much of the contributions in this thesis are related to the problem of robot learning which is commonly modeled as a robotic decision process (more formally defined in general terms as a Markov process [1]). Rather than doing an large expansive review of each of the fields of intersection and machine learning as a whole, I focus on how the contributions presented in this thesis relate and expand upon our knowledge of robot learning and control.

1.1.1 Learning robot motor skills

One of the most common and general methods for describing and solving the problem of robot learning and control is with the Markov decision processes and reinforcement learning [2, 3]. Most reinforcement learning methods fall between one of two categories: model-based or model-free reinforcement learning.

Model-based learning methods, as the name implies, uses models to solve reinforcement learning tasks in robotics. The primary idea is that learned models can be used to predict how a robot will interact with the world given an input action to the robot. The predicted feedback from the environment (usually through a reward function) is used to estimate whether the action was desirable and forwarded to the robot. Model-based learning methods possess the desirable trait of being "sample-efficient" [4–6] when solving robot learning tasks. That is, model-based methods

require significantly less data to learn a control response to achieve the desired robot learning task. However, a downside to model-based learning is that the models are often highly complex and require special structure to model the necessary intricacies [6–9].

Model-free methods, on the other hand, approach the problem of robot learning through a unique interpretation of the reinforcement learning objective that avoids the need to have predictive models and instead learns a mapping (a policy) that returns an action based on prior experience [10, 11]. Despite having better performance than model-based methods, model-free approaches require a significant amount of data and diverse experience to work properly [5]. How is it then, that humans are capable of rapidly learning tasks with a limited amount of experience? And is there a way to enable robotic systems to achieve similar performance?

Some recent work tries to address these questions by exploring "how" models of environments are structured by combining probabilistic models with deterministic components [5]. Other work has explored using latent-space representations to condense the complexities [8,9]. Related methods use high fidelity Gaussian Processes to create models, but are limited by the amount of data that can be collected [12]. Finally, some researchers try to improve model-free methods by adding exploration as part of the objective [13]. However, these approaches often do not formally combine the use of model-based planning with model-free learning.

Those that do combine model-based planning and experience-based learning tend to do so in stages [7, 14]. First, model-based learning is used to collect data to warm-start a policy for model-free learning. Supervised learning is used to update a policy [14,15] and model-free learning is used to continue the learning from that stage [7]. While novel, this approach heuristically combines the two robot learning methods (which may require hand-tuning by a human assistant). Moreover, the model is often used as an oracle, which provides labels to the supervised learning. As a result, the model-based approach is held fixed and does not get improved upon, and the resulting models are

under-utilized. This thesis presents an approach that algorithmically combines model-based and model-free learning by using the learned models as a gauge for how well a model-free policy will behave. The approach then optimally updates the resulting actions that is forwarded to the robotic system. Using tools from hybrid control theory, I derive a controller that optimally uses model-based actions when the policy is uncertain, and allows the robot to fall back on the model-free policy (which is based on experience) when there exists high confidence actions that will result in a favorable outcome. As a result, the approach does not rely on improving the model (but can easily integrate high fidelity models), but instead optimally combines the policy generated from model-free methods with predictive models to achieve improved task performance. Improving reinforcement learning methods through existing formal tools in control theory provides us with insight that opens up new research avenues which are used in later contributions of this thesis. Specifically, within the field of reinforcement learning, the contributions presented in this thesis are as follows:

(i) a hybrid control theoretic perspective on robot learning

(ii) deterministic and stochastic variations for robot learning using hybrid control

(iii) a measure for determining the agreement between planning with learned predictive models and policies

(iv) improved sample-efficiency and robot learning performance

(v) diverse implementation using standard off-policy reinforcement learning [11] and behavior cloning [16]

1.1.2 Active learning

Active learning is a field that is focused on acquiring measurements that have high information which improves modeling accuracy. The field of active learning has existing for some time, but has

taken a variety of meanings depending on the underlying learning task and how sampling is done (with or without robotic systems involved). In its essence, active learning reduces model uncertainty through sequential active sampling and estimation that yields the most information about what is being learned.

Active Exploration: One of the earliest approaches for active learning in robotics focusing on how robotic systems explore an environment. Existing work formulates this problem of active exploration as information maximization with respect to a known parameterized model [17, 18]. These approaches often have an abundance of local optima [18, 19] resulting in insufficient data collection. Other approaches have sought to solve this problem by viewing information maximization as an area coverage problem [19, 20]. Ergodic exploration, in particular, has remedied the issue of local optima by using the ergodic metric to minimize the Sobelov distance [21] from the time-averaged statistics of the robot's trajectory to the expected acquired information from exploring a region. This enables both exploration (quickly in low information regions) and exploitation (spending more time in highly informative regions) in order to avoid local optima and harvest informative measurements. In this thesis, I explore the concept of ergodicity to improve how robotic systems explore and learn models from measurements in sparse environments.

Ergodicity and Ergodic Exploration: A downside with the current methods for generating ergodic exploration for robotic systems is that there is the assumption that the model of the robot dynamics is fully known. Moreover, there is little guarantee that the robot will not destabilize during exploration. This becomes an issue when the robot must explore its dynamics (i.e., velocity and acceleration space) in order to explore for informative measurements. Another challenge with current ergodic exploration methods is scalability with the dimensionality of the search space. This thesis contributes an approach that overcomes these issues using a sample-based KL-divergence measure [20] as a replacement for the ergodic metric. This form of measure has been used pre-

viously; however, it relies on motion primitives in order to synthesize controls for the robot [20]. This thesis shows that it is possible to generate a continuous control signal that minimizes this ergodic measure using tools from hybrid control theory. The same approach is shown to be readily amenable to existing equilibrium policies which provide stability for robotic systems. As a result, approximate models of dynamical systems can be used instead of complete dynamic reconstructions while ensuring safety while actively learning through a notion of Lyapunov attractiveness.

Information Maximization: Many active learning problems in robotics are commonly posed as an information maximizing problems [22–24]. These methods operate through direct maximization of an information measure (which assigns a utility to what measurements are most informative) [25]. Generally, this approach suffers from multiple local minima due to the non-convex nature of information objectives [23]. Moreover, the existing methods do not scale well with number of model parameters used in a learning task. In this thesis, I present examples of direct information maximization using approximations of information measures constructed with optimal experimental design [24] in mind. These approximations allow for learning of complex neural-network models with many parameters. In addition, this thesis introduces the idea of emergent learning as a direct consequence of optimizing information objectives that does not require exploration heuristics which is common in many deterministic robot learning examples [26]. These ideas are later expanded upon in the thesis to address the local minima problem using ergodic exploration [19, 27]. Ergodic exploration provides robotic systems with a method for sampling from information objectives indirectly. Thus, it is possible to avoid the multiple local minima problem found with non-convex information objectives. As a contribution of this thesis, I show that using ergodic exploration expands robot learning to modeling sparse environments which require robots to actively seek out informative measurements. This thesis then builds upon earlier work on hybrid learning to develop safe active learning strategies for robotic systems using the ergodic exploration strategy.

Bayesian Optimization: Bayesian optimization (which I go into more detail in Chapter 2)[28–30] is a Bayesian approach to active learning. In Bayesian optimization, the goal is to find the maximum of an objective function which is unknown or can only be sampled a finite number of times. At each iteration of Bayesian optimization, the unknown objective is sampled and a probabilistic model is generated. An acquisition function is then used as a metric for an "active learner" to find the next best sample. This loop repeats until a maximum is found. In this thesis, the "active learner" becomes the robotic system which must abide by the physical constraints that govern its motion in the world. As a result, the assumption that the active learner has the ability to sample anywhere in the search space becomes invalid. In this thesis, Bayesian optimization is viewed as an ergodic sampling problem [19, 27]. Thus, the active learner is able to sample from regions which have lower probability densities quickly and spend time in regions which are likely to produce an optima. A useful benefit of solving Bayesian optimization with ergodic exploration is that the robot is able to avoid local optima.

Safe learning: Safety-based robot controllers are controllers that enforce desirable safety properties onto robotic systems. Control Lyaponov functions (CLFs) [31, 32] is a class of safety-based controllers that enforce stability properties of a robotic system through a feedback stabilizing control law that monotonically drives a positive definite differentiable function towards zero and ultimately drives the robot to an equilibrium state or set (e.g., LaSalle's invariance principle). Another, related, approach is known as control barrier functions (CBFs) [33–35]. CBFs controllers require the specification of a "safe" set of states through a differentiable barrier function and knowledge of the dynamics. CBF controllers then maintain a robotic system within the specified safe set where the robot is free to move about. Note that the CBF approach requires construction of a function that meets the criteria of a CBF and knowledge of the safe set to begin with. Other, reachability-based approaches, directly solve for the safe set, but require the computation of the

Hamilton-Jacobi-Isaacs partial differential equation which can be computationally expensive [36,37]. Furthermore, CLFs and CBFs can be used directly with additional objective functions but requires solving Quadratic Programs (QPs) in order to obtain safety controllers that impose the safety constraints [32, 34, 38, 39]. In this thesis, I develop methods for active learning that directly inherits and utilizes CLF-based safety controllers and safety properties that filters unsafe robot exploration and enables safe exploration[1] so long as the conditions for the CLF controller can still be met. This is accomplished by combining pre-defined CLF controllers with an exploration strategy using tools from hybrid control theory instead of directly solving QP problems which would require strictly imposing the safety constraints (i.e., monotonically decreasing Lyaponov function). The result is the synthesis of a single active learning controller that inherits the properties of a safety controller that allows robot exploration within a derived safety set where the CLF constraints are relaxed and can be used with approximate robot dynamics with locally known equilibrium states without the need to pre-specify a safe set of states. In summary, within the field of active learning for robotics this thesis contributes the following:

(i) approximations of information measures in optimal experimental design for complex models

(ii) introduction of ergodic exploration as a method for robot learning

(iii) extension of the ergodic metric to active learning in high-dimensional dynamics exploration spaces

(iv) safe active learning with ergodic measure using tools from hybrid control theory

(v) various examples of active learning for robotic systems using the proposed approaches

[1] Safe exploration is defined by exploration of states that meet the conditions of CLFs when the equilibrium policies are applied.

1.1.3 Modeling dynamics with latent embeddings

How we represent and model robotic systems is always a fundamental question in robot learning. Typically the complexity of the model will vary depending on the system in question; however, the resulting models are often nonlinear to start. If the state is large, it is common to embed the state into a compressed latent representation [40] which makes it computationally easier to use for planning and prediction. Unfortunately, the latent representations can often be more complex than the original system which makes it difficult to use with gradient-based optimal control methods. Is it possible to construct simple representations that can embed the nonlinearities of robotic systems to improve control synthesis without the added structural complexities? How much improvement can be expected from modeling robotic systems in such a way and what learning trade-offs are to be expected?

In this thesis, I examine the class of *infinite linear* embeddings known as Koopman operators as a candidate model for model-based robot learning and control. As the name of the class suggests, Koopman operators embed nonlinear dynamical systems into a larger (often infinite) embedded space that evolves linearly in time. Koopman operators were first proposed in 1931 in work by B.O. Koopman [41]. At the time, approximating the Koopman operator was computationally infeasible; the onset of computational methods enabled data-driven approximations to the Koopman operator [42–44]. The typical research into the use of Koopman operator looks to use the linear structure in search of eigenfunctions and invariant subspaces that determine the nonlinear dynamic characteristics and the overall dimensionality of the Koopman operator formulated dynamical system [45–47]. However, some of the most recent research has shown the capabilities of the Koopman operator model for control [46, 47].

Koopman operators for model-based control research has suggested that such a use is promising

avenue for many fields including robotics [46–54]. Koopman operators are closely related to latent variable (embedded) dynamic models [55]. In embedded dynamic models, an autoencoder [40, 55] is used to compress the original state-space into a lower-dimensional representation. The embedded dynamics model then only evolves the latent representation that can predict the overall dynamical system behavior. In contrast, Koopman operators represent the state of dynamical systems in an expanded dimensional representation where the evolution of the embedding is defined as a linear dynamical system. Thus, Koopman operators latent states are able to represent the nonlinear behavior of the original dynamical system as a linear differential equation that can be used for linear model-based control. In this thesis, I present the Koopman operator for enhancing robot control and learning. I show that this slight change in representation of the robot dynamics improves the learning and control performance compared to existing nonlinear approaches for robotic systems. In keeping with the theme of this thesis, I present an active learning approach that exploits the linear structure of the Koopman operator that allows robotics systems to learn within a single roll-out (i.e., execution of the robot in the world). Within the field of latent embeddings for robot control the contributions of this thesis are as follows:

(i) robot control in infinite linear embedded space

(ii) active learning in infinite embedded spaces

(iii) flexible automated learning of lifted linear embeddings

1.2 Thesis layout and published content

This thesis is organized in a way which is designed for each chapter to build off one another. Therefore, each new chapter brings with it either an advancement of a previous chapter or uses the content as the foundational work. As described below, the work in Chapters 3,5,6, and 7 is partially

published in [56–60], and the work in Chapter 4 along with Chapter 6.3.4 will be submitted for publication.

For the reader following along, I provide an initial chapter on background and preliminary mathematics that are used throughout the thesis in Chapter 2. In this chapter, I discuss the general problem setup for robot learning and decision making. I then introduce variations to the problem of learning and decision making as well as the common solutions which I will be comparing to in each chapter that contributes an advancement to the field. Some of the tools that are used throughout the thesis will also be briefly introduced to provide the reader with some information and context for motivation. These tools are overviewed in more detail in the main text of the thesis.

Chapter 3 introduces the first set of contributions which are used throughout this thesis. Specifically, this chapter presents hybrid learning as a method for enhancing motor skill learning in robotic systems through the use of tools from hybrid control theory. The main contribution of this chapter is the formulation of robot learning as a hybrid mode scheduling problem. Deterministic and stochastic variations of hybrid learning are presented and are used as the foundation for the subsequent chapters. The proposed approaches are then tested against a variety of robot learning tasks and evaluated against existing methods. The theoretical, simulated and experimental work presented in this chapter are published in [56].

Chapter 4 deviates momentarily from learning motor skills through hybrid control and instead introduces the problem of active learning for robotic systems through examples of direct optimization of information measures. In this chapter, I first motivate the problem of how robotic systems should acquire data for learning as a problem of generating a control signal that gives robotic systems the ability to experiment in the world. This is accomplished through the maximization of measures that label informative data in a maximum likelihood estimation problem. As a contribution, this chapter presents a derivation for these information measures for more general, complex

network-like models. Examples of active learning are then presented for learning robot dynamics and query-based interactive imitation learning. The results in this chapter are discussed and used as motivation for the subsequent chapters which present solutions to some of the limitations with direct optimization of information measures. The contents in this chapter will be submitted for publication.

Chapter 5 looks at the problem of active learning in sparse environments where measurements change infrequently through space. This chapter is where the thesis first introduces ergodicity and ergodic control for active learning. The problem of active learning is motivated for robotic systems that need to explore and model environments where sensory signals (like touch or contact) are spatially sparse, requiring the robot to take action in order to measure the world. Information measures for non-parametric models are introduced and are indirectly optimized through the use of ergodic control. Examples for modeling and localization using contact-based robotic systems are presented. The theoretical, simulated, and experimental work presented in this chapter is published in [57].

This next chapter builds upon each of the previous chapters to develop a method for safe active learning for robotic systems. Specifically, Chapter 6 presents KL-E^3, a method for safe ergodic exploration in high dimensional dynamic spaces that uses tools and methods from hybrid control theory and hybrid learning. In this chapter, I first present the problem with using the ergodic controller shown in Chapter 5 for active learning. In addition, I motivate the need for active learning methods that are bounded to safe equilibrium robotic states but do not hinder the acquisition of informative measurements for learning. This is accomplished through the use of ideas stemming from Chapter 3 and a notion of Lyaponov attractiveness[2] is derived for the proposed approach. This chapter then present various use-cases for KL-E^3 for active learning in robotic systems. The

[2]Not to be confused with Lyaponov stability.

theoretical contributions and parts of the examples in this chapter are published in [58, 59]. The results and examples in Chapter 6.3.4 are to be submitted for publication.

In the following Chapter 7, the thesis takes a drastic turn and reevaluates some of the last chapters through the lenses of infinite linear embeddings. The thesis first introduces and justifies the use of infinite linear embeddings for modeling and control of robotic systems in Chapter 7. The justifications are used to present infinite linear embeddings for active learning. Specifically, I show that the structure of modeling the robot dynamic through infinite linear embeddings results in not only simpler mathematical structure, but also improved learning and control performance of robotic systems. Through a variety of simulated and experimental results, I show that active learning with infinite linear embedding structure give robotic systems the ability to learn in a single roll-out of the robotic system. In addition, the special modeling structure allows equilibrium policies to be quickly synthesized for control which makes use of Chapters 3 6. The results in these chapters are published in [52, 60].

Last, I conclude this thesis in Chapter 8. In this last chapter, I take this opportunity to restate the contributions of this thesis and provide some take-away messages that I have learned throughout each chapter. I provide an outlook for future work and what I envision to be the end goal of a long and arduous journey for true robot autonomy that starts with the work in this thesis.

1.3 Collaboration acknowledgments and extended published content

In this section, I would like to acknowledge the collaborative work presented in this thesis and the published/ongoing work that is not a part of this thesis, but stems directly from the ideas

presented. To start, the work published in Chapter 3 is work that is a result of a collaborative effort with the Computer Science department at Northwestern University and the Shirley Ryan Rehabilitation Institute. The ideas from this work has led to a collaborative effort with the University of Washington and the NVIDIA Seattle robotics lab on the development of methods that exploit complex simulations with large parameter uncertainty for robot control. The contents of this research is published in [6] are are continually being developed and stem directly from the theoretical work developed in Chapter 3.1.2. The work on ergodic exploration presented in Chapter 5 has also been expended for decentralized control of multi-agent systems and published in [61]. The ideas and theoretical contributions from this work has largely been a part of an effort that further develops human-swarm capabilities that has recently been published in [62] and whose work is currently on-going. The theoretical work in Chapter 6 is currently being adapted for on going research for developing human-robot rehabilitation techniques and on learning structured representations of complex sensory systems. In addition, part of the work developed in Chapter 7 has also influenced collaborative work done with the Computer Science department at Northwestern University and the Shirley Ryan Rehabilitation Institute on shared-control that is published in [63]. Last, the work in Chapter 7 has inspired a series of research that further develops Koopman operator theory for robot control [48, 54].

Chapter 2

Background

This chapter presents an overview of some mathematical concepts that are often discussed in robot learning and modeling decision making. Specifically, the topics are directly related to the context of this thesis and the contributions that the thesis provides. Thus, I encourage anyone reading to use this chapter as reference for the following chapters and as a source of motivating future work. I will first introduce a common approach to modeling decision making in robotics and relate it to optimal control. I will then introduce methods for solving robot decision making problems and additional variations to the optimal control problem which will be used in the following chapters as motivation. Section 2.2 introduces and motivate a reformulation of modeling robot decisions for learning, also known as active learning. In particular, this section will present various approaches for active learning and motivate active learning in the context of physical systems whose actions have consequences to what is measured and learned. Last, Section 2.3 introduces the idea of modeling physical dynamics as infinite, linear dynamic embeddings known as the Koopman operator.

2.1 Markov decision processes and optimal control

Modeling how autonomous systems make decisions is a first step towards achieving true autonomy. There needs to be a way to model and understand the complexities of decisions, and their consequences, particularly in the field of robotics where physical phenomena plays a large role in how we control these systems. Markov decision processes (MDP's) enable modeling of decision making in not just robotic systems, but many complex systems with different interaction dynamics, actions, and objectives. Robot decision making is modeled as an MDP through the definition of a tuple $\{\mathcal{X}, \mathcal{U}, \mathcal{L}, \mathcal{P}\}$ which represents a set of n dimensional accessible states $x_t \in \mathcal{X}^n$, $u_t \in \mathcal{U}^m$ defines the set of admissible m dimensional actions (or control inputs) that can be taken, $\mathcal{L} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ defines the space of external values associated with a state-action pair[1], and $\mathcal{P}(x_{t+1} \mid x_t, u_t)$ is the state transition probability associated with transitioning from a state x_t and action u_t to the state at the next time step x_{t+1}.

Given an MDP formulation, the goal is to find a mapping from state to action that maximizes the expected external value over a finite time. This can be written as the following optimization problem:

$$\pi^\star = \arg\max_{\pi} \mathbb{E}_{u \sim \pi(\cdot \mid x)} \left[\sum_{t=0}^{T-1} r_t \right] \qquad (2.1)$$
$$\text{subject to } \mathcal{P}(x_{t+1} \mid x_t, u_t)$$

where $\mathbb{E}$ is the expectation operator, T is the discrete time horizon[2] $r_t = r(x_t, u_t) \in \mathcal{L}$ is an external reward value[3] at time t, $u \sim \pi(\cdot \mid x)$ is a stochastic policy as a function of state, and

[1] Often the external value can be defined as a cost ℓ (i.e., penalty) or reward r for the state-action pair.
[2] The optimization problem can also be formulated in continuous time.
[3] The problem can be posed as minimization if a cost function $\ell_t = \ell(x_t, u_t)$ is used.

$\pi^\star$ is the optimal policy that returns sampled actions that maximize the expected reward. If the transition model and the reward function are known, the MDP formulation becomes a stochastic optimal control problem. [4]. Given a deterministic transition model, $\mathcal{P} = 1$, then the stochastic optimal control problem is converted into a deterministic optimal control problem subject to the deterministic transition model often defined as

$$x_{t+1} = f(x_t, u_t) \tag{2.2}$$

where $f(x, u) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the general transition model (in robotics the transition model often defines the dynamics of the robot itself).

2.1.1 Learning-based approaches to MDP

If we are given the conditions of an optimal control problem one can choose from a long list of methods such as iLQG, MPPI, and Projection-Based optimization [64–66]. However, if the transition model or the reward function are unknown, the robotic system would be required to learn models of the missing functions from sensor measurements and then solve the MDP objective using what is learned. These methods often fall under two categories of reinforcement Learning: model-based and model-free reinforcement learning.

Model-based reinforcement learning operators by first learning a model of the reward function and transition model from measurement data. This is done through optimizing some loss function (usually a least-squares variation) between the model predictions and observed measurements. The models are then used to maximize the reward either through directly learning the state-action

[4] Often optimal control problems are specified with a cost (penalty) function; however, the definition remains the same with a minimization instead of a maximization

policy (gradient based optimization of a parameterized policy) or through model-based planning[5]. In model-based planning, the models are used to predict the state and reward outcomes given a sequence of predicted actions over a time horizon. The sequence of actions are optimized and the first action is applied to the robot and the subsequent actions are recycled following a new sample of the state x_t. Model-based methods have the benefit of being "sample-efficient", meaning that significantly less measurements and interactions with the environment are required to solve the MDP than the counter-part model-free learning approaches. The downside is that the model-based methods often require significantly more complex models and yield poor performance when compared against model-free methods.

In contrast, model-free reinforcement learning methods approach the problem of solving the MDP through a unique interpretation of the MDP objective stated in Eq. 2.1 which avoids the need of a transition model unused [2]. Therefore, only experienced measure data and a value function which indicates whether a sequence of actions was improves the objective is used to update the policy. Thus, only a policy and value function are learned to solve the MDP using model-free methods (where in model-based methods, a policy, value, and transition model is learned). As a consequence, model-free methods require less complex models and are able to achieve better performance than model-based methods. This is at a cost of requiring significantly more data and interaction with the environment to obtain the models and maximize the expected reward.

Note that both methods can be formulated into a variation of optimal control with the constraint that the models are learned from data. That being said, this thesis attempts to bridge that gap between optimal control and reinforcement learning using methods from hybrid control theory. In doing so, I present the problem of robot learning and decision making as a problem of switching between different modes of learning (see Chapter 3).

[5]Another term for a similar approach is model predictive control

2.1.2 Hybrid control theory for mode scheduling

Hybrid control theory is an approach to solving a specific kind of optimal control problem. Specifically, the goal is to solve a variation of the MDP problem where actions are selected between a set of "modes" or fixed control strategies. This shifts the optimization from learning the policy to learning a schedule of switching times for the policies.

To describe the hybrid control problem, it is often easier to think of the MDP formulation as a deterministic continuous time MDP. First, define the transition model as the continuous process

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.3}$$

where $\dot{x}(t) : \mathbb{R} \to \dot{\mathcal{X}}^n$ denotes the infinitesimal change in state at the continuous time t, $f(x, u) : \mathcal{S} \times \mathcal{A} \to \dot{\mathcal{X}}$ represents the continuous transition model in the tangent space $\dot{\mathcal{X}}$, and $x(t), u(t)$ denote the continuous time state and action which now become a function of continuous time. Note that we overload the notation to avoid complicating the background discussion and introducing unnecessary notations. A common variation to $f(x, u)$ is the control affine transition dynamics

$$\dot{x}(t) = f(x(t), u(t)) = g(x(t)) + h(x(t))u(t) \tag{2.4}$$

where $g(x) : \mathcal{X} \to \dot{\mathcal{X}}$ is the free dynamics and $h(x) : \mathcal{X} \to \dot{\mathcal{X}} \times \mathcal{U}$ is a mapping from state to control.

The hybrid control problem (as used in this thesis) is then specified as

$$\arg\max_{\tau,\lambda} \mathcal{J} = \int_{t=0}^{t_H} r(x(t), u(t))dt \tag{2.5}$$

$$\text{subject to } \dot{x}(t) = f(x(t), u(t)),\ x(0) = x_0$$

where

$$u(t) = \begin{cases} \hat{u}(t), & \text{if } t \in [\tau, \tau + \lambda] \\ u_{\text{def}}(t) & \text{otherwise} \end{cases}, \tag{2.6}$$

t_H is the time horizon (in seconds), and $x(t)$ is generated from some initial condition $x(0)$ using the model (2.3) and action sequence (2.6), τ is then the switching time and λ is the time duration for switching between a default action $u_{\text{def}}(t)$ and an arbitrary action $\hat{u}(t)$. These problems are interesting because they occur so often in engineering problems. As an example, consider the vertical take-off-landing vehicle (VTOL) which has two modes: flight and hover. The problem of flight is well studied as is hovering, but switching is problematic particularly when the optimal behaviors differ significantly between modes. Thus, finding when to optimally switch without having an adverse reaction to the performance of the VTOL can be constructed as an hybrid control problem. I make similar analogies for robot learning which is discussed in Chapter 3 as a contribution of this thesis.

2.2 Active learning

While the MDP formulation models decision making in robotics, it does not directly address the problem of learning the possibly unknown models (like the transition dynamics or external reward function) in an optimal manner. In most common approaches for solving MDP's within the context of robot learning, an initial guess at the unknown models is used and updated as new measurements of the state and reward are obtained. As a result, there is this trade-off of exploration and exploitation that occurs in most robot learning tasks. How much of the environment interaction time is spent exploring for novel measurements versus how much time is spent exploiting the learned models to solve the MDP problem? With robots that physically interact in the world, blindly ex-

ploring will lead to permanent damage and harm. In contrast, not exploring enough can result in the inability to solve the MDP due to poor modeling of the world. If we consider that robots can only have limited interactions with the environment, then how informative each measurements is must be taking into account.

In this thesis, I address this problem of "optimally" learning as an active learning problem. Active learning address the problem of selective learning where an autonomous system has to make decisions on what measurements are collected to improve the conditions of the learning problem as best as possible. In other words, seek out the most informative measurements that improves learning efficiency. The idea behind active learning is that measurements are chosen such that they provide high information about what is being learned (resulting in fast learning to achieve the desired outcome). The high information measurements can be interpreted as measurements that reduce model uncertainty and prediction error. Active learning is particularly a powerful concept in robotics as actions taken by the robot incur a physical consequence that has to be managed and dealt with efficiency. This can include actively learning the transition model (or dynamics), the reward function, or interactively learning from expert demonstrations. In this thesis, I introduce a set of existing methods for active learning and extend the problem statement to robotic systems with physical constraints while proposing methods that are capable of handling the added restrictions.

2.2.1 Bayesian optimization

A common example of active learning is known as Bayesian optimization. In Bayesian optimization, the goal is to find the maximum of an objective function which is unknown or can only be evaluated

a handful of times. Formally, the problem is specified as

$$x^\star = \arg\max_x \psi(x) \tag{2.7}$$

where $x \in \mathcal{S}^d$ is a d dimensional measurement in the search space $\mathcal{S}$ of the function $\psi : \mathcal{S}^d \to \mathbb{R}$. In many cases, ψ can not be evaluated directly, thus the approach in Bayesian optimization is to construct a stochastic surrogate model from measured evaluations of ψ. The idea is that the posterior of the surrogate model can be used to infer regions in $\mathcal{S}$ where measurements are more likely to provide information about the maximum of ψ through what is defined as a utility function. This utility function $\mathcal{U}(x) : \mathcal{S} \to \mathbb{R}$ is constructed from the surrogate model posterior given the already sampled measurements $x_i, \psi(x_i)$. The surrogate function is often modeled using a Gaussian Process $\mathcal{GP}(\mu(x), \sigma^2(x))$ where $\mu(x), \sigma^2(x)$ are the predictive mean and variance posterior of the $\mathcal{GP}$ obtained from a previously sampled measurement data set $\mathcal{D} = \{x_i, \psi(x_i)\}_{i=1}^N$ with N being the total number of queries to the function ψ.

At each iteration of Bayesian optimization, the acquisition function is generated from the posterior to inform an "active learner" where samples are most informative. The active learner then samples the search space based on the acquisition function and updates the posterior of the $\mathcal{GP}$. This loop repeats until a maximum is found. An example algorithmic pseudo-code is provided in Algorithm 1.

In this thesis, I consider the problem of when the "active learner" inhabits the physical world, resulting in dynamic constraints that govern the motion of the robot and its ability to sample at desired informative locations (specifically when $\mathcal{S}$ intersect the dynamic space of the robot i.e., velocity, acceleration). Thus, the ability of the active learner to query particular measurements is constrained and will involve planning.

Algorithm 1 Bayesian Optimization

init: Gaussian prior on objective ψ, data set $\mathcal{D}$, $i = 0$, $i_{\max}$
while $i < i_{\max}$ **do**
    $x_{\text{sample}} = \arg\max \mathcal{U}(x)$
    active learner samples $y_i = \psi(x_{\text{sample}})$
    $\mathcal{D} \leftarrow \{x_{\text{sample}}, y_i\}$
    update $\mathcal{GP}$ posterior distribution on ψ using $\mathcal{D}$
    update $\mathcal{U}(x)$ from $\mathcal{GP}$
    $i \leftarrow i + 1$
end while
return max y_i of $\mathcal{GP}$ surrogate at x_{sample}

2.2.2 Optimal experimental design

Another approach to active learning is known as optimal experimental design (OED) (there is also a Bayesian optimal experimental design, but I do not go over that method in this thesis). The idea behind OED is to use the structure of the learning problem itself (i.e., the model) to decide how to improve the conditions and experiments for learning (e.g., what to measure, where to measure, how many measurements). The ultimate goal being to improve the learning by selecting measurements that infers the most about the underlying model and minimize parameter uncertainty.

I motivate the use of optimal experimental design through the following linear regression example and then generalize the motivation to a larger class of maximum likelihood learning problems. Consider a linear model $y = \phi(x)^\top \theta + \epsilon$. where $x \in \mathbb{R}^d$ is a measurement point, $y \in \mathbb{R}$ is the predicted output, $\theta \in \mathbb{R}^p$ is a vector of learnable parameters, $\phi : \mathbb{R}^d \to \mathbb{R}^p$ is a vector of basis functions that are evaluated at the measurement point x, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is i.i.d noise with variance σ^2. For a data set of N measurement points $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ obtained by querying the

unknown function $y_i = \psi(x_i)$, the linear regression problem is defined as

$$\theta^\star = \arg\min_\beta \left(Y - \Phi\theta\right)^\top \left(Y - \Phi\theta\right)$$

where $Y = [y_1, \ldots, y_N]^\top$, $\Phi = [\phi(x_1), \ldots, \phi(x_N)]^\top \in \mathbb{R}^{N \times p}$, and $\theta^\star$ is the optimal parameter which minimize the least squares regression. Since the regression problem is convex in θ, the optimal solution is

$$\theta^\star = \left(\Phi^\top \Phi\right)^{-1} \Phi^\top Y$$

where $\theta^\star$ is known as the maximum likelihood estimator. If we calculate the variance of $\theta^\star$ we obtain the following relationship

$$\begin{aligned} \text{var}\left[\theta^\star\right] &= \left(\left(\Phi^\top \Phi\right)^{-1} \Phi^\top\right) \text{var}\left[Y\right] \left(\Phi \left(\Phi^\top \Phi\right)^{-1}\right) \\ &= \sigma^2 \left(\Phi^\top \Phi\right)^{-1} \end{aligned}$$

where $\Phi^\top \Phi$ is defined as the Fisher information matrix [24]. Note that the Fisher information is only a function of the measurement query points x_i and is directly related to the variance of the maximum likelihood estimator without needing to query the unknown function we are trying to learn. We can use this knowledge to *optimize* over query points x_i to improve our estimate of the maximum likelihood estimator through maximizing the Fisher information matrix! This is a direct consequence of what is known as the Cramér-Rao lower bound [67, 68] defined as

$$\text{var}\left[\theta^\star\right] \geq \mathcal{I}(\theta)^{-1}$$

where $\mathcal{I}(\theta)$ is the Fisher information matrix.

We can define the Fisher information for the more general maximum likelihood estimation problem. Let $p(y \mid x, \theta)$ be a family of θ parameterized probability densities that model observations $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$. Then the likelihood that parameter θ models the data is given by $\Pi_{n=1}^N p(y_i \mid x_i, \theta)$. Maximizing the log likelihood with respect to the parameters θ gives the maximum likelihood estimator

$$\theta^\star = \arg\max_\theta \log \Pi_{n=1}^N p(y_i \mid x_i, \theta).$$

The Fisher information for the more general maximum likelihood problem is defined through variance of the "score" function and the Hessian of the likelihood problem:

$$\begin{aligned}
\mathcal{I}(\theta) &= \operatorname{var}\left[\nabla_\theta \log p\right] \\
&= \mathbb{E}\left[\left(\nabla_\theta \log p\right)\left(\nabla_\theta \log p\right)^\top\right] - \underbrace{\mathbb{E}\left[\nabla_\theta \log p\right]^2}_{=0}. \\
&= \mathbb{E}\left[\left(\nabla_\theta \log p\right)\left(\nabla_\theta \log p\right)^\top\right] = -\mathbb{E}\left[\nabla_\theta^2 \log p\right]
\end{aligned}$$

where $\nabla_\theta \log p = \frac{\partial}{\partial \theta} \log p(y \mid x, \theta)$ is the score function, $\nabla_\theta^2 = \frac{\partial^2}{\partial \theta^2}$ is the Hessian, and $\mathbb{E}$ indicates an expectation over measurements drawn from the underlying data distribution. Here, the last equatilty is obtained by $\nabla_\theta^2 \log p$ and relating it to $\left(\nabla_\theta \log p\right)\left(\nabla_\theta \log p\right)^\top$ at expectation. If we use the model $p(y \mid x, \theta) = \mathcal{N}(\phi(x)^\top \theta, \sigma^2)$, it is easy to show that we recover the original definition of the Fisher information for the linear regression problem.

Thus, we can infer and improve upon the minimum obtainable variance of a model's parameters through maximizing the Fisher information matrix and effectively planning what measurements to collect. In Chapters 4, I introduce various forms of optimizing the Fisher information through optimality conditions and expand upon applications of Fisher information maximization for more general, complex modeling problems. This thesis also presents methods for active learning through

direct and indirect optimization of the Fisher information for fast and efficient robot learning.

2.2.3 Ergodicity and ergodic exploration

I will now overview a more recent strategy for exploration and sampling known as ergodic exploration. Consider the problem of exploration versus exploitation where the robot must decide whether to allocate time to exploit what it already knows (maximum of some objective using learned models) or explore for more information (improve models sampling measurements in unexplored areas).

Definition 1. *Ergodicity, in robotics, is defined when a robot whose time-averaged statistics over its states is equivalent to the spatial statistics of an arbitrarily defined target distribution that intersects those states.*

Rather than choosing to explore or exploit, ergodic exploration reduces the exploration versus exploitation trade-off as proportionally matching the time-averaged trajectory distribution to a spatially determined statistic.[6] Thus, a robot that explores ergodicity will spend more of its time in regions directly proportion to high statistics (positive outcomes in exploitation) and quickly explores the search space where there are low statistics (low probability of positive outcome exploration).

The exact specifications of a spatial statistic varies depending on the underlying task and is defined differently depending on the learning task and a utility function similar to Bayesian optimization. For now, let us define the time-averaged trajectory distribution of a robot by considering its continuous time trajectory $x(t) : \mathbb{R} \to \mathcal{X}^n \ \forall t \in [t_0, t_f]$ generated through an arbitrary control sequence $u(t) : \mathbb{R} \to \mathcal{U}^m$.

[6] The spatial statistics are often defined through a utility function that indicates where informative measurements reside.

Definition 2. *Given a search domain $\mathcal{S}^v \subset \mathcal{X}^n$ where $v \leq n$, the time-averaged statistics (i.e., the time the robot spends in regions of the search domain $\mathcal{S}^v$) of a trajectory $x(t)$ is defined as*

$$c(s \mid x(t)) = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \delta\left[s - \bar{x}(t)\right] dt, \tag{2.8}$$

where $s \in \mathcal{S}^v$ is a point in the search domain, $\bar{x}(t)$ is the component of the robot's state $x(t)$ that intersects the search domain, and $\delta[\cdot]$ is the Dirac delta function.

In general, the target spatial statistics are defined through its probability density function $p(s)$ where $p(s) > 0$, and $\int_{\mathcal{S}^v} p(s)ds = 1$. Given a target spatial distribution $p(s)$, we can calculate an ergodic metric $\mathcal{E}(x(t))$as the distance between the Fourier decomposition of $p(s)$ and $c(s \mid x(t))$: [7]

$$\mathcal{E}(x(t)) = \sum_{k \in \mathbb{N}^v} \Lambda_k \left(c_k - p_k\right)^2$$

where Λ_k is a weight on the harmonics defined in [27],

$$c_k = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} F_k(\bar{x}(t))dt, \qquad p_k = \int_{\mathcal{S}^v} p(s)F_k(s)ds,$$

and $F_k(s)$ is the k^{th} cosine Fourier basis function. Minimizing the ergodic metric results in the time-averaged statistics of $x(t)$ matching the defined target spatial distribution $p(x)$. In addition, on a finite horizon, an ergodically exploring robotic system samples measurements in high utility regions more often while still exploring at low utility regions.

Exploring in low utility regions has the added benefit of acquiring novel measurement and information that often improves the learning capabilities of robots. In addition, this approach side-steps the problem of gradient-based optimization of non-convex utility functions by instead,

[7] This distance is known as the Sobelov distance [69].

solving a simpler optimization problem through the Fourier decomposition. As in [19, 70], one can calculate a controller that optimizes the ergodic metric such that the trajectory of the robot is approximately ergodic with respect to the distribution $p(x)$. Unfortunately, this approach scales $\mathcal{O}(|k|^n)$ where $|k|$ is the maximum integer–valued Fourier term. As a result, this approach is ill-suited for high-dimensional search spaces ($n > 3$). Furthermore, the resulting time-averaged distribution reconstruction will often have residual artifacts which require additional conditioning to remove. This motivates part of the thesis work on defining a variation to the ergodic metric that can be expanded to high dimensional search spaces.

2.3 Infinite linear embeddings

In this last section, I introduce a class of infinite linear representations of dynamical systems (often referred to as Koopman Operators). The Koopman operator is an important part of this thesis and will be discussed more completely within the context of robotics in the later chapters.

2.3.1 The Koopman operator

The Koopman operator is a way to represent nonlinear systems as linear systems in the infinite vector space. Thus, the Koopman operator is a class of infinite linear embedding where the non-linearity of a dynamical system can be represent by an infinite set of features that are related linearly in time. Consider a set of features $z(x) : \mathcal{X} \to \mathcal{F}$ where $\mathcal{F}$ is the lifted functional space of dimension.[8] Then the Koopman operator dynamical system is defined as

$$z(x_{t+1}) = \mathcal{K}z(x_t) = z(f(x_t)) \tag{2.9}$$

[8] As before, we can define the dimensionality of $\mathcal{X}$ and $\mathcal{F}$ with a superscript.

where $\mathcal{K} : \mathcal{F} \rightarrow \mathcal{F}$ is the linear Koopman operator matrix that predicts the value of the functions $z(x_t) \rightarrow z(x_{t+1})$ in time, $f(x)$ is the transition model (excluding control). Adding control actuation u to the Koopman operator is as simple as expanding the definition of the features to include the control space $\mathcal{U}$: [9]

$$z(x_{t+1}, u_{t+1}) = \mathcal{K} z(x_t.u_t). \tag{2.10}$$

The intuition behind the infinite linear embedding is that there exists a set of observations of the state whose evolution in time approximates a linear dynamical system. Most nonlinear systems can be represented as linear systems as the dimensionality of z approaches infinity (resulting in the infinite Koopman operator matrix). While there does exist finite representations (finite Koopman subspaces), these are often difficult to find and have only been discovered for specific kinds of dynamical systems [46]. In this thesis, I focus on generating approximate Koopman dynamical systems in a subset of the functional space from data.

Let $z(x, u)$ be defined by a finite set of real-valued functions. Then, given a data set of N sequential measurements $\mathcal{D} = \{x_i, u_i, x_{i+1}, u_{i+1}\}_{i=1}^{N}$ the Koopman operator can be approximated through the linear regression

$$\min_{\mathcal{K}} \frac{1}{2} \sum_{i=1}^{N} \|z(x_{i+1}, u_{i+1}) - \mathcal{K} z(x_i, u_i)\|^2. \tag{2.11}$$

The solution defines the approximate Koopman dynamical system which can be used for prediction in the lifted latent space. In this thesis, I explore the linear dynamics of the Koopman operator for improving learning and control of nonlinear robotic systems. In addition, I introduce an approach for automatically learning the functions z and present examples of active learning using the Koopman operator model.

[9] I will later show different variations to choosing z with control than one stated here.

Chapter 3

Learning as a hybrid mode scheduling problem

In this first chapter, I present the problem of robot learning and decision making from the perspective of hybrid control theory. Specifically, the goal of this chapter is to introduce hybrid learning as viable method for robot learning through optimally combining model-based and model-free learning (which in this chapter I refer to as experience-based learning). The predictive models in model-based learning provide an understanding of the task and the physics (which improves sample-efficiency), while experience-based policy mappings in model-free learning are treated as "muscle memory" that encode favorable actions as experiences that overrides planned actions. Hybrid control tools are used to create an algorithmic approach for combining learned predictive models with experience-based learning. In the following sections, hybrid learning is presented as a method for robot learning that efficiently learns motor skills by systematically combining and improving the performance of predictive models and experience-based policies. I derive a deterministic variation of hybrid learning and extend the approach into a stochastic implementation that relaxes some of the key assumptions in

the original derivation. Each variation is tested on various tasks (where the robot interacts with the environment to gain experience) as well as scenarios found in imitation learning (where experience is provided through demonstrations and tested in the environment). The results show that this approach is capable of improving the performance and sample-efficiency of learning motor skills in a variety of experimental domains. The foundations of this chapter are then utilized throughout the remainder of this thesis.

3.1 Hybrid Learning

The goal of this section is to introduce hybrid learning as a method for optimally utilizing model-based predictions and model-free policy learning. I start with the deterministic variation of the algorithm that provides theoretical proofs, which describe the foundations of the approach. The stochastic variation is then derived as a method for relaxing the assumptions made in the deterministic variation. The main theme in both the deterministic and stochastic derivations is that the learning problem is solved *indirectly*. That is, we solve the (often harder) learning problem by instead solving sub-problems whose solutions imply that the harder problem is solved.

3.1.1 Deterministic

Consider the continuous time formulation of the objective and dynamics in (2.5) and (2.3) with the MDP formation where f and r are learned using arbitrary regression methods (e.g., neural network least squares, Gaussian processes), and the policy π in (2.1) is learned through a model-free approach (e.g., policy gradient [2]). In addition, let us assume that in the default action in (2.6) is defined as the mean of the policy π where we ignore uncertainty for the time being[1]. That

[1]We will add the uncertainty into the hybrid problem in the stochastic derivation of our approach for hybrid learning

is, $u_{\text{def}}(t) = \mu(x(t))$ where we assume that the policy has the form $\pi(u \mid x) = \mathcal{N}(\mu(x), \Sigma(x))$, where $\mathcal{N}$ is a normal distribution and $\mu(x)$, $\Sigma(x)$ are the mean and variance of the policy as a function of state. For now we assume that we leave $\hat{u}$ as a free variable, let us first calculate how sensitive (2.5) is at any τ to switching from $\mu(x) \to \hat{u}$ for an infinitely small λ[2].

Lemma 1. *Assume that f, r, and μ are first order differentiable and continuous in time. The sensitivity of (2.5) (also known as the mode insertion gradient [71]) with respect to the duration time λ from switching between $\mu(x)$ to $\hat{u}$ and any time $\tau \in [0, t_H]$ is defined as*

$$\frac{\partial}{\partial \lambda} \mathcal{J}(\tau) = \rho(\tau)^\top (f_2 - f_1)|_\tau \tag{3.1}$$

where $f_1 = f(x(t), \mu(x(t)))$ and $f_2 = f(x(t), \hat{u}(t))$, and $\rho(t) \in \mathcal{X}^n$ is the adjoint variable which is the the solution to the the differential equation

$$\dot{\rho}(t) = -\frac{\partial r}{\partial x} - \left(\frac{\partial f}{\partial x} + \frac{\partial \mu}{\partial x}^\top \frac{\partial f}{\partial u} \right)^\top \rho(t) \tag{3.2}$$

with terminal condition $\rho(t_H) = \mathbf{0}$.

Proof. See Appendix A for proof. □

Lemma 1 gives us the proof and definition of the mode insertion gradient (3.1), that tells us the infinitesimal change in the objective function when switching from the default policy behavior to some other arbitrarily defined control $\hat{u}$ for a small time duration λ. The mode insertion gradient is directly used to see how an arbitrary action changes the performance of the task from the policy that is being learned. However, in this chapter the mode insertion gradient is used as a method for

[2]We avoid the problem of instability of the robotic system from switching control strategies as later we develop and use the best action for all $\tau \in [0, t_H]$ instead of searching for a particular time when to switch.

obtaining the best action the robot can take given the learned predictive models of the dynamics and the task rewards. We can be more direct in our approach and ask the following question. **Given a suboptimal policy π, what is the best action the robot can take to maximize (2.5), at any time $t \in [0, t_H]$, subject to the uncertainty (or certainty) of the policy defined by $\Sigma(x)$?**

We approach this new sub-problem by specifying the auxiliary optimization problem:

$$u^\star(t) = \underset{\hat{u}(t) \ \forall t \in [0,t_H]}{\arg\max} \int_0^{t_H} \frac{\partial}{\partial \lambda} \mathcal{J}(t) + \log \pi \left(\hat{u}(t) \mid x(t) \right) dt \tag{3.3}$$

where the idea is to maximize the mode insertion gradient (i.e., find the action that has the most impact in changing the objective) subject to the $\log \pi$ term that ensures the generated action $u^\star$ is penalized for deviating from the policy, when there is high confidence that was based on prior experience.

Theorem 1. *Assuming control affine dynamics (2.4) and f, r, and π are continuous and differentiable in x, u and t, the best possible action that improves the performance of (2.5) and is a solution to (3.3) for any time $t \in [0, t_H]$ is*

$$u^\star(t) = \Sigma(x(t)) h(x(t))^\top \rho(t) + \mu(x(t)) \tag{3.4}$$

where $\rho(t)$ is defined by (3.2) and $h(s) : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the affine mapping from actions to the dynamics.

Proof. Inserting the definition of the mode insertion gradient (3.1) and taking the derivative of

(3.3) with respect to the point-wise $\hat{u}$ and setting it to zero gives

$$\rho^\top h(x)\left(\hat{u}-\mu(x)\right)-\Sigma(x)^{-1}\left(\hat{u}-\mu(x)\right)=0$$

where we drop the dependency on time for clarity. Solving for $\hat{u}$ gives the best actions $u^\star$

$$u^\star(t)=\Sigma(x(t))h(x(t))^\top\rho(t)+\mu(x(t))$$

which is the action that maximizes the mode insertion gradient subject to the certainty of the policy π for all $t\in[0,t_H]$. □

The proof in Theorem 1 provides the best action that a robotic system can take given a default experience-learned policy. Each action generated uses the sensitivity of changing the objective based on the predictive models while relying on the experience-based policy to regulate when the predicted information will be useful. We convert the result in Theorem 1 into our first (deterministic) hybrid learning learning algorithm (see Alg. 2).

The benefit of the proposed approach is that we are able to make (numerically based) statements about the generated action and the contribution of the learned predictive models towards improving the task. Furthermore, it is possible to make the claim that (3.4) provides the best possible action given the current belief of the dynamics f and the task reward r.

Corollary 1. *Assuming that* $\frac{\partial}{\partial u}\mathcal{H}\neq 0$ *where* $\mathcal{H}=r(x)+\log\pi(u\mid x)+\rho^\top f(x,u)$ *is the control Hamiltonian for (2.5), then* $\frac{\partial}{\partial\lambda}\mathcal{J}=\|h(x)^\top\rho\|_{\Sigma(x)}>0$ *and is zero when the policy satisfies the control Hamiltonian condition* $\frac{\partial}{\partial u}\mathcal{H}=0$.

Algorithm 2 Hybrid Learning (deterministic)

Randomly initialize continuous differentiable models f, r with parameters ψ and policy π with parameter θ. Initialize data buffer $\mathcal{D}$, prediction horizon parameter t_H, exploration noise ε.
while task not done **do**
    reset environment and initialize exploration noise ε
    for $i = 1, \ldots, T$ **do**
        observe state $x(t_i)$ from environment
        ▷ simulation loop
        for $\tau_i \in [t_i, \ldots, t_i + t_H]$ **do**
            ▷ forward predict states using any integration method (Euler shown)
            $x(\tau_{i+1}), r(\tau_i) = x(\tau_i) + f(x(\tau_i), \mu(x(\tau_i)))dt,\ r(x(\tau_i), \mu(x(\tau_i)))$
        end for
        ▷ backwards integrate using $\dot{\rho}(t)$ defined in (3.2)
        $\rho(t_i + t_H) = \mathbf{0}$
        for $\tau_i \in [t_H + t_i, \ldots, t_i]$ **do**
            $\rho(\tau_{i-1}) = \rho(\tau_i) - \dot{\rho}(\tau_i)dt$
        end for
        $u^\star(t_i) = \Sigma(x(t_i))h(u(t_i))^\top \rho(t_i) + \mu(x(t_i)) + \varepsilon(t)$
        apply $u^\star(t_i)$ to robot
        append data $\mathcal{D} \leftarrow \{x(t_i), u^\star(t_i), r_t, x(t_{i+1})\}$
    end for
    Update f, r by sampling N data points from $\mathcal{D}$ using any learning method
    Update π using any model-free method
end while

Proof. Inserting (3.4) into (3.1) using the control affine dynamics 2.4 yields

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \lambda} &= \rho^\top \left(g(x) + h(x)\left(\Sigma(x))h(x)^\top \rho + \mu(x)\right) - g(x) - h(x)\mu(x)\right) \\ &= \rho^\top h(x)\Sigma(x)h(x)^\top \rho = \|h(x)^\top \rho\|_{\Sigma(x)} > 0.\end{aligned}$$

From Pontryagin's Maximum principle, a solution that is a local optima of the objective function satisfies the following

$$\frac{\partial}{\partial u}\mathcal{H} = -\Sigma(x)^{-1}\left(u - \mu(x)\right) + h(x)^\top \rho = 0$$

when $u = \Sigma(x)h(x)^\top \rho + \mu(x)$ or $\rho = 0$. Therefore, if the policy π is a solution, then it must be that

the adjoint $\rho = 0$ and π a solution to the optimal control problem (2.5). $\square$

Corollary 1 tells us that the action defined in (3.4) generates the best action that will improve the performance of the robot given valid learned models. In addition, Corollary 1 also states that if the policy is already a solution, then our approach for hybrid learning does not impede on the solution and returns the optimal policy's action.

Taking note of each proof, we can see that there is the strict requirement of continuity and differentiability of the learned models and policy. As this restrictions are not always possible to obtain, and often learned models from complex function approximators have noisy derivatives, the goal is to try to reformulate (2.5) into an equivalent problem that can be solved without the need for the assumptions. One way is to formulate the problem in discrete time (as an expectation), which we will do in the following section.

3.1.2 Stochastic

We relax the continuity, differentiability, and continuous-time restrictions specified (2.5) by first restating the objective as an expectation:

$$\max \mathbb{E}_{v \sim \pi(\cdot \mid x)} \left[\mathcal{J}(v) \right] \tag{3.5}$$

where $\mathcal{J}(v) = \sum_{t=0}^{T-1} r(x_t)$ subject to $x_{t+1} = f(x_t, v_t)$, and $v = [v_0, \dots v_{T-1}]$ is a sequence of T randomly generated actions from the policy π. Rather than trying to find the best time τ and discrete duration λ, we approach the problem from an hybrid information theoretic view and instead find the best augmented actions to π that improve the objective. This is accomplished by defining two

distributions $\mathbb{P}$ and $\mathbb{Q}$ which are the uncontrolled system response distribution[3] and the open loop control distribution (augmented action distribution) described by their probability density functions $p(v) = \prod_{t=0}^{T-1} \pi\left(v_t \mid x_t\right)$ and $q(v) = \prod_{t=0}^{T-1} \frac{1}{\sqrt{(2\pi)^m |\Sigma(x_t)|}} \exp\left(-\frac{1}{2}(v_t - u_t)^\top \Sigma(s_t)^{-1}(v_t - u_t)\right)$ respectively. Here, $\pi(u \mid x) = \mathcal{N}(\mu(x), \Sigma(x))$ and $q(v)$ use the same variance $\Sigma(x)$ as the policy. The uncontrolled distribution $\mathbb{P}$ represents the default predicted behavior of the robotic system under the learned policy π. Furthermore, the augmented open-loop control distribution $\mathbb{Q}$ allows us to define a probability over arbitrary augmented actions, but more importantly, a free variable for which to optimize over given the learned predictive models. Following the work in [4], we use Jensen's inequality and importance sampling on the free-energy [72] definition of a stochastic control system using the open loop augmented distribution $\mathbb{Q}$:

$$\mathcal{F}(v) = -\lambda \log\left(\mathbb{E}_{\mathbb{P}}\left[\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)\right]\right) \leq -\lambda \mathbb{E}_{\mathbb{Q}}\left[\log\left(\frac{p(v)}{q(v)}\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)\right)\right] \tag{3.6}$$

where $\lambda \in \mathbb{R}^+$ here is what is known as the temperature parameter (and not the time duration as used prior). Note that in (3.6) if $\frac{p(v)}{q(v)} \propto 1/\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)$ then the inequality becomes a constant. Further reducing the free-energy gives the following:

$$\mathcal{F}(v) \leq -\lambda \mathbb{E}_{\mathbb{Q}}\left[\log\left(\frac{p(v)}{q(v)}\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)\right)\right] \leq -\mathbb{E}_{\mathbb{Q}}\left[\mathcal{J}(v) - \lambda \log\left(\frac{p(v)}{q(v)}\right)\right]$$

which is the optimal control problem we seek to solve plus a bounding term which penalizes actions that are far from the policy. In other words, the free-energy formulation can be used as an indirect approach to solve for the hybrid optimal control problem by asserting that the likelihood ratio pushes the dynamic system towards an equilibrium of the free-energy (implying a solution to the

[3] We refer to uncontrolled as the unaugmented control response of the robotic agent subject to a stochastic policy π

optimal control problem). Specifically, if we can make $\frac{p(v)}{q(v)} \propto 1/\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)$, then the free-energy bound reduces to a constant (implying the likelihood ratio results in an equilibrium state of the free-energy). Using this knowledge, we can define an optimal distribution $\mathbb{Q}^\star$ through its density function

$$q^\star(v) = \frac{1}{\eta}\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v), \quad \eta = \int_\Omega \exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v)dv$$

where Ω is the sample space.[4] Letting the ratio be defined as $\frac{p(v)}{q^\star(v)}$ gives us the proportionality that we seek to make the free-energy a constant. However, because we can not directly sample from $\mathbb{Q}^\star$, and we want to generate a separate set of actions a_t defined in $q(v)$ that augments the policy given the learned models, our goal is to push $q(v)$ towards $q^\star(v)$. As done in [4, 73] this corresponds to the following optimization:

$$u^\star = \arg\min_u D_{\mathrm{KL}}\left(\mathbb{Q}^\star \mid \mathbb{Q}\right) \tag{3.7}$$

which minimizes the Kullback-Leibler divergence of the optimal distribution $\mathbb{Q}^\star$ and the open-loop distribution $\mathbb{Q}$. In other words, we want to construct a separate distribution that augments the policy distribution $p(v)$ (based on the optimal density function) such that the objective is improved.

Theorem 2. *The recursive, sample-based, solution to (3.7) is*

$$u_t^\star = u_t + \sum_k \omega(v_t^k)\delta u_t^k \quad \textit{where} \quad \omega(v) = \frac{\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v)}{\sum_n \exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v)} \tag{3.8}$$

where k denotes the sample index, $v_t = u_t + \delta u_t$, and $v_t \sim \pi(\cdot \mid x_t)$.

Proof. See Appendix A for proof. □

The idea behind the stochastic variation is to generate samples from the stochastic policy and

[4] The motivation being to use the optimal density function to gauge how well the policy π performs.

evaluate its utility based on the current estimate of the dynamics and the reward function. Since samples directly depend on the likelihood of the policy, any actions that steers too far from the policy will be penalized depending on the confidence of the policy at that state. Conversely, when the policy has low confidence (high variance) the action sample span increases and the penalty decreases, this more of the model-based information is utilized. Note that we do not have to worry about continuity and differentiability conditions on the learned models and can utilize arbitrarily complex models for use of this algorithm (albeit the optimal control has only been defined for normally distributed policies). We outline the stochastic algorithm for hybrid learning Alg. 3.

3.2 Robot learning examples

In this section, I apply hybrid learning (both deterministic and stochastic variations) on various robot learning examples. In these examples, I show that hybrid learning improves the overall performance and sample-efficiency of robot learning when compared against the current state of the art in model-free and model-based learning. In addition, I show that through this approach the individual components of model-based and model-free methods that are used in the joint hybrid learning are improved. The proposed hybrid learning is also posed as an imitation learning method where expert demonstrations are used to generate the experience-based policy (through behavior cloning) and use the learned predictive models to adapt to the uncertainty in the policy. All implementation details are provided in Appendix B.

3.2.1 Model-based and model-free learning

We evaluate hybrid learning in the deterministic and stochastic settings and compare against the standard in model-based and model-free learning. In addition, we illustrate the ability to evaluate

Algorithm 3 Hybrid Learning (stochastic)

Randomly initialize continuous differentiable models f, r with parameters ψ and policy π with parameter θ. Initialize data buffer $\mathcal{D}$, prediction horizon parameter T, environment interaction time H.
while task not done **do**
    reset environment
    for $t = 0, \ldots, H-1$ **do**
        observe state $\bar{x}_t$ from environment
        ▷ simulation loop
        for $k \in \{0, \ldots, K-1\}$ **do**
            set sim state x_τ^k
            for $\tau \in \{0, \ldots, T-1\}$ **do**
                $v_\tau^k \sim \pi(\cdot \mid x_\tau^k)$
                ▷ forward predict state and reward
                $x_{\tau+1}^k, r_\tau^k = f(x_\tau^k, v_\tau^k), r(x_\tau^k, v_\tau^k)$
            end for
        end for
        ▷ update actions
        for $\tau \in \{0, \ldots, T-1\}$ **do**
            $\mathcal{J}(v_\tau^k) \leftarrow \sum_{t=\tau}^{T-1} r_t^k$
            $\delta u_\tau^k \leftarrow v_\tau^k - u_\tau$
            $u_\tau \leftarrow u_\tau + \sum_{k=0}^{K-1} \omega(v_\tau^k) \delta u_\tau^k$
        end for
        apply u_0 to robot and observe $\bar{x}_{t+1}$
        append data $\mathcal{D} \leftarrow \{\bar{x}_t, u_0, r_t, \bar{x}_{t+1}\}$
        shift actions $u_{0:T-2} = u_{1:T-1}$, init $u_{T-1} = 0$
    end for
    Update f, r by sampling N data points from $\mathcal{D}$ using any learning method
    Update π using any model-free method
end while

our method's agreement of the learned models through the mode insertion gradient. Experimental results validate hybrid learning for real robot tasks. For each example, we use Soft Actor Critic (SAC) [11] as our model-free method and a neural-network based implementation of model-predictive path integral for reinforcement learning [4] as a benchmark standard method. The parameters for SAC are held as default across all experiments to remove any impact of hyperparameter tuning.

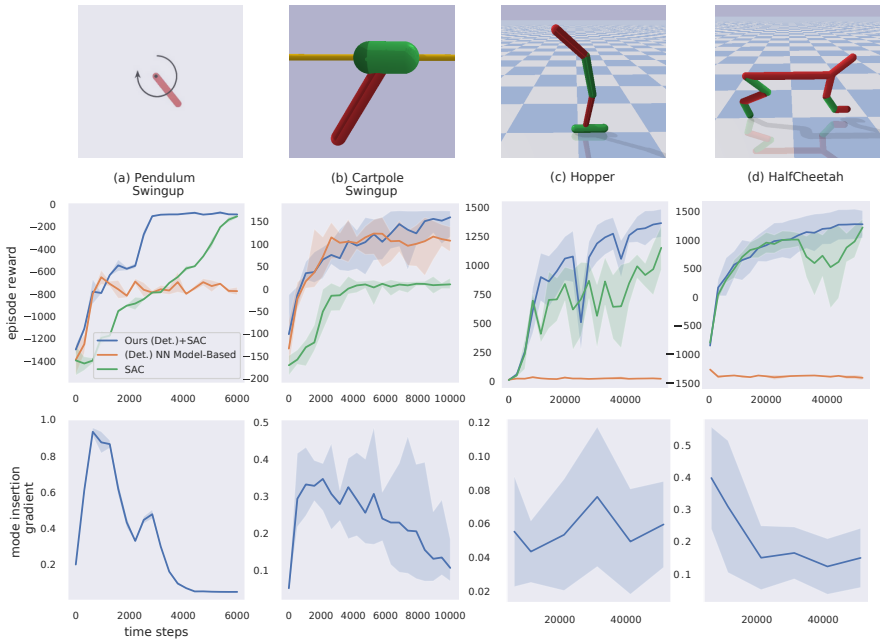


Figure 3-1: Performance curves of our proposed deterministic hybrid learning algorithm on multiple environments (averaged over 5 random seeds). All methods use the same structured learning models. Our method is shown to improve the model-based benchmark results (due to the use of experience-based methods) while maintaining significant improvements on the number of interactions necessary with the environment to obtain those results. The mode insertion gradient is also shown for each example which illustrates the model-policy agreement over time and the improvement over time.

Hybrid learning is tested in four simulated environments: pendulum swingup, cartpole swingup, the hopper environment, and the half-cheetah environment (Fig. 3-1) using the Pybullet simulator [74]. In addition, we compare against state-of-the-art approaches for model-based and experience-based learning. We first illustrate the results using the deterministic variation of hybrid learning in Fig. 3-1 (compared against SAC and a deterministic model-predictive controller [75]). Our approach uses the confidence bounds generated by the stochastic policy to infer when best to rely on the policy or predictive models. As a result, hybrid learning allows for performance comparable to experience-based methods with the sample-efficiency of model-based learning approaches.

Furthermore, the hybrid control approach allows us to generate a measure for calculating the agreement between the policy and the learned models (bottom plots in Fig. 3-1), as well as when (and how much) the models were assisting the policy. The commonality between each example is the eventual reduction in the assistance of the learned model and policy. This allows us to better monitor the learning process and dictate how well the policy is performing compared to understanding the underlying task.

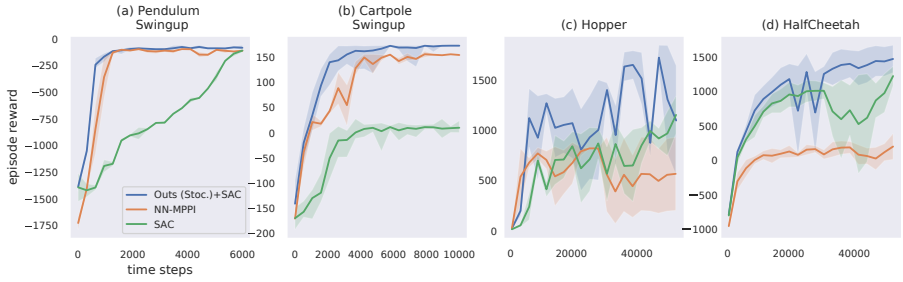


Figure 3-2: Performance curves of our proposed stochastic hybrid learning algorithm on multiple environments (averaged over 5 random seeds). As shown before, our approach improves both the sample-efficiency but also the highest expected reward. In addition, the stochastic variation of the hybrid learning algorithm generates smoother learning curves as a result of not requiring derivatives of learned models.

We next evaluate the stochastic variation of hybrid learning, where we compare against a stochastic neural-network model-based controller [4] and SAC. As shown in Fig. 3-2, the stochastic variation still maintains the improved performance and sample-efficiency across all examples while also having smoother learning curves. This is a direct result of the derivation of the algorithm where continuity and differentiability of the learned models are not necessary. In addition, exploration is naturally encoded into the algorithm through the policy, which results in more stable learning when there is uncertainty in the task. In contrast, the deterministic approach required added exploration noise to induce exploring other regions of state-space. A similar result is found when comparing the model-based performance of the deterministic and stochastic approaches, where the deterministic

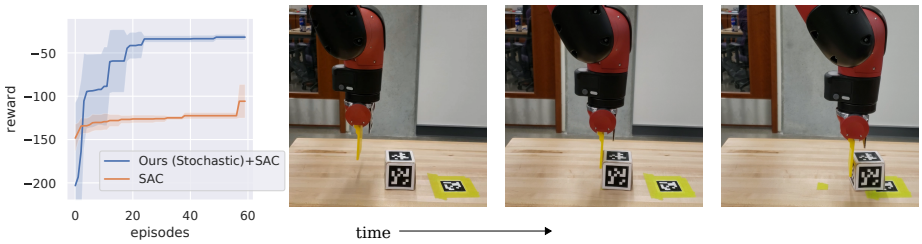


Figure 3-3: Hybrid learning compared with model-free policy results on the Sawyer robot (averaged over 5 trials). The task is to push a block to a designated target through environment interactions (see time-series results above). Our method is able achieve the task within 3 minutes (each episode takes 10 seconds) through effectively using both predictive models and experience-based methods. The same amount of interaction with SAC was unable to successfully push the block to the target. For videos visit `https://sites.google.com/view/hybrid-learning-theory`.

variation suffers from modeling discontinuous dynamics.

We can analyze the individual learned model and policy in Fig. 3-2 obtained from hybrid learning. Specifically, we look at the cartpole swingup task for the stochastic variation of hybrid learning in Fig. 3-4 and compare against benchmark model-based learning (NN-MPPI [4]) and experience-based learning (SAC [11]) approaches. Hybrid learning is shown to improve the learning capabilities of both the learned predictive model and the policy through the hybrid control approach. In other words, the policy is "filtered" through the learned model and augmented, allowing the robotic system to be guided by both the prediction and experience. Thus, the predictive model and the policy are benefited, ultimately performing better as a standalone approach using hybrid learning.

Next, we apply hybrid learning on real robot experiments to illustrate the sample-efficiency and performance our approach can obtain (see Fig. 3-3 for task illustration). [5] We use a Sawyer robot whose goal is to push a block on a table to a specified marker. The position of the marker and the block are known to the robot. The robot is rewarded for pushing the block to the marker. What makes this task difficult is the contact between the arm and the block that the robot needs to

[5]The same default parameters for SAC are used tor this experiment.

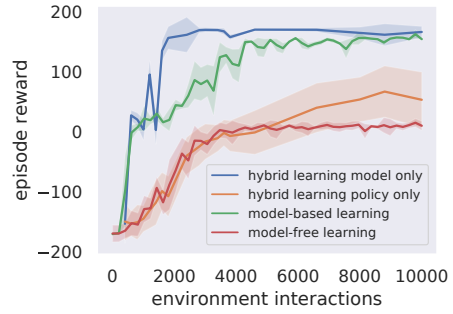


Figure 3-4: Performance curves for the individual learned model and policy on the cartpole swingup environment during hybrid learning (averaged over 10 trials). Our method is shown to improve the capabilities of the model-based and experience-based components through mutual guidance defined by hybrid control theory. Reference model-based learning (NN-MPPI) and experience-based learning (SAC) approaches are shown for comparison.

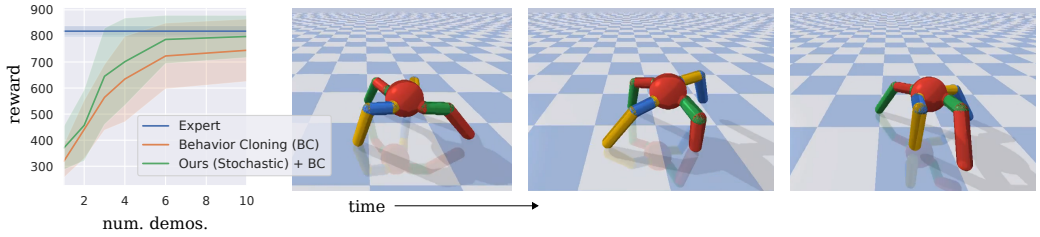


Figure 3-5: Results for hybrid stochastic control with behavior cloned policies (averaged over 10 trials) using the Ant Pybullet environment (shown in a time-lapsed running sequence). Expert demonstrations (actions executed by an expert policy on the ant robot) are used as experience to boot-strap a learned stochastic policy (behavior cloning) in addition to predictive models which encode the dynamics and the underlying task of the ant. Our method is able to adapt the expert experience to the predictive models, improving the performance of behavior cloning and performing as well as the expert. For videos visit `https://sites.google.com/view/hybrid-learning-theory` .

discover in order to complete the pushing task. Shown in Fig. 3-3 our hybrid learning approach is able to learn the task within 20 episodes (total time is 3 minutes, 10 seconds for each episode). Since our method naturally relies on the predictive models when the policy is uncertain, the robot is able to plan through the contact to achieve the task whereas SAC takes significantly longer to discover the pushing dynamics. As a result, we are able to achieve the task with minimal environment interaction.

3.2.2 Learning from demonstrations

We extend our method to use expert demonstrations as experience (also known as imitation learning [76,77]). Imitation learning focuses on using expert demonstrations to either mimic a task or use as initialization for learning complex data-intensive tasks. We use imitation learning, specifically behavior cloning, as an initialization for how a robot should accomplish a task. Hybrid learning as described in Section 3.1 is then used as a method to embed model-based information to compensate for the uncertainty in the learned policy, improving the overall performance through planning. The specific algorithmic implementation of hybrid imitation learning is provided in Appendix B.

Hybrid imitation learning is tested on the Pybullet Ant environment. The goal is for the four legged ant to run as far as it can to the right (from the viewer's perspective) within the allotted time. At each iteration, we provide the agent with an expert demonstration generated from a PPO [10] solution. Each demonstration is used to construct a predictive model as well as a policy (through behavior cloning). The stochastic hybrid learning approach is used to plan and test the robot's performance in the environment. Environment experience is then used to update the predictive models while the expert demonstrations are solely used to update the policy. In Fig. 3-5, we compare hybrid learning against behavior cloning. Our method is able to achieve the task at the level of the expert within 6 (200 step) demonstrations, where the behavior cloned policy is unable to achieve the expert performance. Interestingly, the ant environment is less susceptible to the covariate shift problem (where the state distribution generated by the expert policy does not match the distribution of states generated by the imitated policy [77]), which is common in behavior cloning. This suggests that the ant experiences a significantly large distribution of states during the expert demonstration. However, the resulting performance for the behavior cloning is worse than that of the expert. Our approach is able to achieve similar performance as behavior cloning

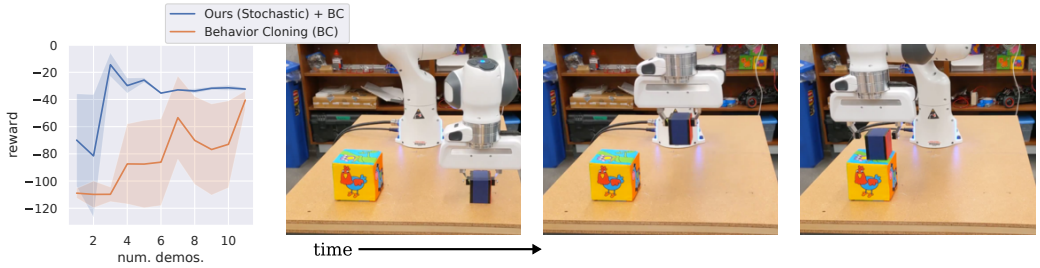


Figure 3-6: Hybrid learning compared with behavior cloning results on the Franka panda robot (averaged over 5 trials). The task is to stack a block on top of another using expert demonstrations. Our method is able to learn the block stacking task within three expert demonstrations and provides solutions that are more repeatable than with behavior cloning. For videos of experiment visit `https://sites.google.com/view/hybrid-learning-theory` .

with roughly 2 fewer demonstrations and performs just as well as the expert demonstrations.

The hybrid imitation learning approach is tested on a robot experiment with the Franka Panda robot (which is more likely to have the covariate shift problem). The goal for the robot is to learn how to stack a block on top of another block using demonstrations (see Fig. 3-6). As with the ant simulated example in Fig. 3-5, a demonstration is provided at each attempt at the task and is used to update the learned models. Experience obtained in the environment is solely used to update the predictive models. We use a total of ten precollected demonstrations of the block stacking example (given one at a time to the behavior cloning algorithm before testing). At each testing time, the robot arm is initialized at the same spot over the initial block. Since the demonstrations vary around the arm's initial position, any state drift is a result of the generated imitated actions and will result in the covariate shift problem leading to poor performance. As shown in Fig. 3-6, hybrid learning is capable of learning the task in as little as two demonstrations where behavior cloning suffers from poor performance. Since our approach synthesizes actions when the policy is uncertain, the robot is able to interpolate between regions where the expert demonstration was lacking, enabling the robot to achieve the task.

3.3 Discussion and summary

This chapter presents hybrid learning, a method that formally introducing robot learning as a hybrid mode switching problem between model-based and model-free learning. The proposed approach derives the best action a robotic agent can take given the learned models acquired from model-based and model-free methods. Hybrid learning is shown to improve both the sample-efficiency of the learning process as well as the overall performance of both the model and policy combined and individually. Each of these claims are tested in various simulated and real-world environments and compared against existing methods for robot learning.

Within the context of this thesis, this chapter serves as a foundation for the following chapters. More specifically, the ideas of robot learning while subject to different safety and exploration restrictions are explored in the following chapters using the analysis and derivations provided in this chapter. I encourage the reader to refer back to this chapter and compare the theoretical analysis to later proofs in search of analogous ideas that may bring new insights in robot learning.

Chapter 4

Direct active learning through optimal experimental design

One of the fundamental assumptions in learning theory is that data is readily available and accessible. This, unfortunately, does not always hold true in most examples of robot learning (as demonstrated in the previous chapter). In many cases the robot needs to infer its own dynamics and how they interact with the world which requires actively moving and measuring the interaction in the physical domain. Approaching the problem through heuristic exploration methods (e.g., injecting noise into the control input [78]) generally requires additional tuning and multiple testing trials which is counter-productive to the ultimate goal of self-sufficient robotic systems. Thus, this chapter motivates the question: how do we make robot learning a more natural an emergent consequence of solving optimal control problems. That is, can we embed experimentation and curiosity about physical interactions (akin to what scientists and toddlers do when they try to understand the world) into a formal optimization that avoids the need for exploration heuristics. The goal for this chapter is to illustrate the necessity of active learning for intentional robot learning. In

doing so, I aim to present the underlying issues that one might encounter when setting up an active learning problem, the limitations that require approximations, and variations to the robot learning problem which are a consequence of dealing with physical robotic systems.

In more explicit detail, I present the problem of active learning in robotics where the goal is to directly optimize for informative measurements that improve the underlying learning task. I first present optimal experimental design (see Chapter 2.2) as a method for active learning where exploration is a emergent consequence optimization with respect to planned actions. I then illustrate approximations which make directly optimizing active learning objectives tractable. Examples are presented for learning robot dynamics and active (interactive) imitation learning.

4.1 Direct information maximization and emergent exploration

Optimal experimental design [24] is an approach to active learning that considers the structure of the learning problem and how a set of anticipated independent measurements (or experiments) change the resulting model accuracy. Through choosing an appropriate experiment, it is possible to predict and reduce the modeling uncertainty in a learning task. In this section, I introduce optimal experimental design as an approach for active learning in robotics. Specifically, I consider the problem of actively learning the dynamics of a robotic system through direct optimization of the Fisher information matrix which is commonly used in optimal experimental design. The following equations and derivations were first presented in [79].

Consider a trajectory generated from a sequence of T discrete actions $u = \{u_0, \ldots, u_{T-1}\}$ which yields a data set $\mathcal{D} = \{x_i, u_i, x_{i+1}\}_{i=0}^{T-1}$ of state x_i, action u_i, and the subsequent measured state

x_{i+1}. Let the task be to learn the parameter $\theta \in \mathbb{R}^p$ that parameterizes a transition dynamics model $p(x_{t+1} \mid x_t, u_t, \theta)$. What is the best possible sequence of actions u such that we minimize the uncertainty in the learned parameter $\theta^\star$? To solve this problem, we first define the log likelihood loss function of model learning problem as

$$\theta^\star = \arg\max_\theta \ \log \Pi_{i=0}^{T-1} p(x_{i+1} \mid x_i, u_i, \theta).$$

While the ultimate goal is to maximize the log likelihood, instead, let us first try to minimize the variance of the maximum likelihood estimator $\theta^\star$. As discussed in Chapter 2.2, we can relate the variance of the maximum likelihood estimator through the the Cramér-Rao bound

$$\text{var}(\theta^\star) \geq \mathcal{I}(\theta)^{-1}$$

where

$$\mathcal{I}(\theta) = -\mathbb{E}\left[\nabla_\theta^2 \log p(x_{t+1} \mid x_t, u_t, \theta)\right]$$

is the Fisher information. Using this relationship, it is possible to optimize over the sequence of actions that yields a predicted maximum of the Fisher information given the current belief of θ. Assuming that $p(x_{t+1} \mid x_t, u_t, \theta) = \mathcal{N}(f(x_t, u_t, \theta), \Sigma)$ where $f(x, u, \theta) : \mathcal{X}^n \times \mathcal{U}^m \to \mathcal{X}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$ parameterizes the mean and variance of the multivariate normal distribution, we can use the reparametrization property of the Fisher information [24] to obtain

$$\mathcal{I}(\theta) = \frac{\partial f_t}{\partial \theta}^\top \mathcal{I}(f) \frac{\partial f_t}{\partial \theta}$$

where $\mathcal{I}(f) = \Sigma^{-1}$ and $f_t = f(x_t, u_t, \theta)$. Since the state measurements are time-dependent and

occur sequentially starting from an initial condition x_0, we can expand the derivative term $\frac{\partial f_t}{\partial \theta}$ and obtain the following [79]:

$$\begin{aligned} \frac{\partial f_t}{\partial \theta} &= \frac{\partial f_t}{\partial x_t}\frac{\partial x_t}{\partial \theta} + \frac{\partial f_t}{\partial \theta} \\ &= \frac{\partial f_t}{\partial x_t}\frac{\partial f_{t-1}}{\partial \theta} + \frac{\partial f_t}{\partial \theta} \end{aligned} \tag{4.1}$$

where we use the fact that $x_t = f(x_{t-1}, u_{t-1}, \theta)$ to replace $\frac{\partial x_t}{\partial \theta}$ with $\frac{\partial f_{t-1}}{\partial \theta}$. Given that we can measure the initial condition x_0, and noting that the (4.1) is defined recursively, we can redefine (4.1) as the following dynamical system

$$\begin{aligned} \psi_{t+1} &= \frac{\partial f_t}{\partial x_t}\psi_t + \frac{\partial f_t}{\partial \theta} \\ \text{such that } \psi_0 &= \{0\}^{n \times p}. \end{aligned} \tag{4.2}$$

This equation become the parameter dynamics under the maximum likelihood problem statement [1]. Using the parameter dynamics (4.2), we can define optimality conditions [80] on the information matrix which maps the information matrix to a scalar value for use in directly optimization subject to a sequence of planned actions. These results was first introduced in [79,81] for a finite number of parameters and lays the groundwork for the following contributions which expands the derivation to more general and more complex models with larger parameters.

Computing the information matrix, and any optimality conditions, becomes more intractable as the model complexity increases. This is particularly true when modeling complex dynamics with general function approximation (e.g., a neural network) where $p \gg n$. Instead, let us reduce the model complexity of the dynamics as a diagonalized normal distribution of the form

[1] A similar version is obtained through the least-squares problem statement as well.

$p(x_{t+1} \mid x_t, u_t, \theta) = \mathcal{N}(f(x_t, u_t, \theta), \text{diag}(\sigma^2))$ where $f(x, u, \theta) : \mathcal{X}^n \times \mathcal{U}^m \to \mathcal{X}^n$ and $\sigma^2 \in \mathbb{R}^n$ parameterizes the mean and variance. Next, rather than calculating the full information matrix, we instead compute an approximate diagonal Fisher information matrix

$$\begin{aligned} \mathcal{I}(\theta) &= \frac{\partial f}{\partial \theta}^\top \mathcal{I}(f) \frac{\partial f}{\partial \theta} \\ &\approx \text{diag}\left((\sigma^{-2})^\top \left(\frac{\partial f}{\partial \theta} \right)^2 \right) \end{aligned} \tag{4.3}$$

where

$$\text{diag}\left(\frac{\partial f}{\partial \theta}^\top \mathcal{I}(f) \frac{\partial f}{\partial \theta} \right) = (\sigma^{-2})^\top \left(\frac{\partial f}{\partial \theta} \right)^2 \in \mathbb{R}^p$$

and $\mathcal{I}(f) = \text{diag}(\sigma^{-2})$. Since the Fisher information is related to the maximum likelihood estimator variance, optimzing the diagonal of the Fisher information matrix is equivalent to minimizing the diagonal variance of the maximum likelihood estimator. Although the Fisher information matrix is being approximated as a diagonal matrix, we can now tractably compute its inverse when the parameter space is large as the inverse of a diagonal matrix is simply the inverse of each element in the diagonal. Note that in this approximation of the Fisher information, we are only required to compute the first order derivatives of the model mean with respect to the parameters. Using the A-optimality condition (or trace) of the approximate Fisher information matrix and given some auxiliary cost function $\ell(x, u)$, we can formulate the active learning objective in the following

manner:

$$u^\star = \arg\min_u \sum_{t=0}^{T-1} \ell(x_t, u_t) + \text{tr}\left(\mathcal{I}_t^{-1}\right) \tag{4.4}$$

$$\text{subject to } x_{t+1} = f(x_t, u_t, \theta),$$

$$\mathcal{I}_t = \text{diag}\left((\sigma^{-2})^\top \psi_t^2\right),$$

$$\text{and } \psi_{t+1} = \frac{\partial f_t}{\partial x_t}\psi_t + \frac{\partial f_t}{\partial \theta}$$

with initial conditions x_0 and $\psi_0 = \{0\}^{n \times p}$. The optimal solution $u^\star$ in (4.4) gives the predicted information maximizing actions that improve the model accuracy. At this point, it is straight forward to apply optimization techniques for planning and control for robotic systems. Here, Eq. 4.4 is solved in a receding horizon model-predictive control formulation (see Algorithm 4) where the first action applied and the following sequences are replanned whenever a new state is sampled or the learning parameters are updated. The following sections illustrates toy examples for active learning using information maximization.

4.1.1 Dynamic experimentation

This first example illustrates the benefits of active learning through direct optimization of Eq 4.4 to learn the dynamics of a cart pole (see [79] for dynamics). Here, $\ell(x, u) = -\cos(x_1) + 0.1x_2^8 + 0.01u^2$ which encourages the swing-up behavior for the cart pole system with $x \in \mathcal{X}^4$, where x_1, x_2 are the pole angle and cart position respectively and the remaining two states are their time derivatives. The goal is to actively learn the dynamics model as quickly as possible which allows the cart pole to achieve the swing-up task. The dynamics model we are learning is a two-layer network with 16 nodes each layer using the sin function as a nonlinearity ($p = 436$) making the dimensionality

Algorithm 4 Direct Active Learning via Fisher Information Maximization

Randomly initialize dynamics models f with parameter θ, cost function ℓ. Initialize planned actions $u_{0:T-1}$, learning rate γ, planning iterations N, data buffer $\mathcal{D}$, and prediction horizon parameter T
while task not done **do**
  sample robot state $\bar{x}$
  set sim state $x_0 \leftarrow \bar{x}$, $\psi_0 = \{0\}^{n \times p}$
  for $t \in \{0, \ldots, T-1\}$ **do**
    $x_{t+1}, \ell_t = f(x_t, u_t, \theta), \ell(x_t, u_t)$
    $\psi_{t+1} = \frac{\partial f_t}{\partial x_t}\psi_t + \frac{\partial f_t}{\partial \theta}$
    $\mathcal{I}_t = \text{diag}\left((\sigma^{-2})^\top \psi_t^2\right)$
  end for
  ▷ update actions
  $\mathcal{J}(u_{0:T-1}) \leftarrow \sum_{t=0}^{T-1} \ell_t + \text{tr}\left(\mathcal{I}_t^{-1}\right)$
  for N iterations **do**
    $u_{0:T-1} \leftarrow u_{0:T-1} - \gamma \frac{\partial}{\partial u_{0:T-1}} \mathcal{J}(u_{0:T-1})$
  end for
  apply u_0 to robot and observe $\bar{x}'$
  append data $\mathcal{D} \leftarrow \{\bar{x}, u_0, \bar{x}'\}$
  shift actions $u_{0:T-2} = u_{1:T-1}$, init $u_{T-1} = 0$
  update θ with max. log likelihood using M sampled data points from $\mathcal{D}$
end while

of the parameter dynamics 4×436. We solve (4.4) using automatic differentiation JAX [82] for gradient-based optimization with respect to u in a model-based planning approach as shown in Alg. 4. The active learning approach is compared against 10% and 20% control saturation noise injected control exploration [78] which added exploration noise to the model-based control signal during swingup attempts. Each approach is subjected to the same random seed. Model updates are done at each instance a new measurement is obtained using the Adam optimizer [83] with a learning rate of 0.003 iterated over 5 epochs.

Figure. 4-1 illustrates the results of directly optimizing over the active learning objective for the cart pole. Because the active learning approach is deterministic, exploration is a naturally occurring by-product of optimization of the active learning objective. Thus, how to explore is automatically

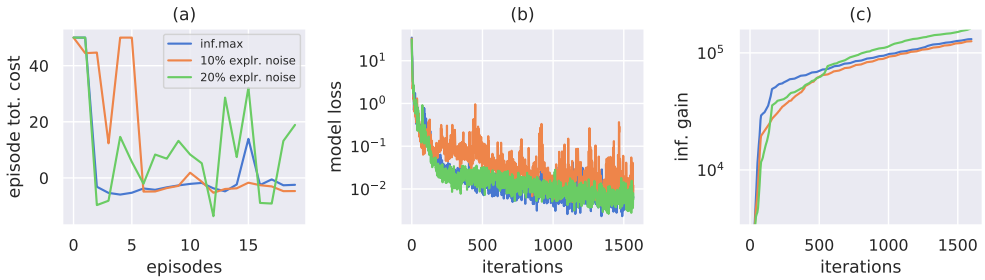


Figure 4-1: Results for active learning of the cart pole dynamics through Fisher information maximization during swing-up task compared against control noise injection of 10% and 20%. (a) Illustrates the trajectory cost for the swing-up task for each episode. (b) Model loss over iterations of Adam [83] during robot execution. (c) Information gain as defined by Eq.(6.28). Information maximization naturally explores the dynamic space earlier in training yielding stable learning curves and improved model-based control performance.

generated with no need to heuristically introduce exploration noise which often destabilizes the robotic system. The destabilization effect is illustrated in the episode costs over each learning episode in Fig. 4-1 (a) for the noise injected exploration methods. Furthermore, optimally planning for information maximizing actions generally results in more stable and consistent learning as the most informative measurements are acquired earlier in the learning as shown by Fig. 4-1(b) and (c). While one could search for the optimal control noise injection (and even an optimal decay rate), this would require multiple runs and "resets" of the robotic system in order to optimize over the additional exploration hyper parameters. Unfortunately doing so is counter-productive to the ultimate goal of achieving true autonomous robotic systems that function in the world.

Exploring for informative measurements should be an emergent consequence of optimizing active learning objectives without the need to consider heuristic additive exploration. That being said, it is often the case that a robot is required to learn a task from a teacher or from demonstrations. In general, a robot will not have a defined objective for every single task that it can experience. This is generally true when the underlying task is difficult to interpret or explain as a cost or reward

function, but instead must but presented through demonstrations. The following section presents this problem from the perspective of active learning where the robot is allowed to ask (or query) the teacher/demonstrator what to do in specific states to accomplish a task.

4.2 Active query imitation learning

In this section I pose the problem of imitation learning, specifically focusing on behavior cloning [77], from the perspective of active learning. I illustrate how giving robotic systems the ability to query enables them to learn a task more quickly when posed as an active learning problem.

The concept behind imitation learning, specifically, the case of behavior cloning, centers around the idea of robots learning how to do a task without having access to a reward (or cost) function and instead learning from a set of demonstrations provided by a teacher (or expert policy). The robot does not have access to the expert policy, but can only obtain state-action pairs from the expert policy. Here, the ideas of active learning are explored to the most fundamental of imitation learning strategies, behavior cloning (see Chapter 3 for behavior cloning examples).

In behavior cloning, the goal is to learn a policy $u = \pi(x, \theta)$ such that it matches a set of demonstrations $\mathcal{D} = \{x_i, \pi^\star(x_i)\}_{i=1}^N$ provided by a an expert policy $u^\star = \pi^\star(x)$ [2] without knowing the underlying task or the expert policy. Thus, the behavior-cloned policy is obtained through the optimization over demonstration

$$\theta^\star = \arg\max_\theta \sum_{\mathcal{D}} \|\pi^\star(x) - \pi(x, \theta)\|^2$$

where $u^\star = \pi^\star(x)$ is the expert demonstrator (which is not accessible and treated as constant).

[2] I abuse and overload the notation for a deterministic and stochastic policy. Here, the $=$ indicates deterministic where $\sim$ indicates stochastic sampling.

Naively solving this optimization results in what is known as the *covariate shift* where the behavior cloned policy drifts due to training on the demonstration data set. The cause is due to the learned policy drifting the robot's state away from the training data distribution. As a result, the policy returns suboptimal actions that compound errors over time, ultimately causing the robot to drift away from the optimal behavior. Existing methods that propose solutions to the covariate shift problem use an on-policy approach, that is, the current estimate of the behavior cloned policy is deployed on the robot during learning and the expert relabels the collected data with the optimal ones [77]. Other approaches try to minimize the covariate shift using an off-policy approach through robust learning with noise injection of the expert demonstrations [77]. Many other variations exists which use reinforcement learning in an on-policy manner to improve the learning and reduce covariate shift. However, what if the robot had the ability to interact and query the expert policy for informative demonstrations?

Given the ability to query the expert, we can pose the problem of behavior cloning as an active learning problem where the robot wants to optimize over the most informative queries to the expert that maximize the behavior cloned policy accuracy. An candidate approach is to seek queries that maximize the Fisher information matrix with respect to the behavior cloned policy. We can quickly justify this approach through the Cramér-Rao bound

$$\text{var}\left[\theta^{\star}\right] \geq \mathcal{I}(\theta)^{-1}$$

and the fact that there is no additional external signal which guides the quality of the behavior cloned policy. Therefore, the only solution is to use the Cramér-Rao bound and reduce the maximum likelihood estimator variance as much as possible by choosing query points which maximize the Fisher information. In the case of on-policy learning, this corresponds to the following optimization

$$x^{\star} = \arg\min_{x} \operatorname{tr}\left(\mathcal{I}(\theta)^{-1}\right) \tag{4.5}$$

where

$$\begin{aligned} \mathcal{I}(\theta) &= \frac{\partial \pi}{\partial \theta}^{\top} \mathcal{I}(\pi) \frac{\partial \pi}{\partial \theta} \\ &\approx \operatorname{diag}\left((\sigma^{-2})^{\top} \left(\frac{\partial \pi}{\partial \theta} \right)^{2} \right) \end{aligned} \tag{4.6}$$

is approximated in the same form as Eq.6.28, and the variance σ is assumed known.

Since the optimization of the Fisher information has many local minima, we seed the optimization (4.5) using the current measured state of the robot during an on-policy roll-out of the current behavior cloned policy parameters. At each step, the point $x^{\star}$ is used to query the expert policy $\pi^{\star}$ which is aggregated into a data-set $\mathcal{D}$. Policy updates occur at each step using sampled batches from the aggregated data-set. Note that the query points $x^{\star}$ may be dynamically infeasible to achieve; however, like any good student, often the best questions may not make any sense to the teacher, but mean a lot to the student.

Active query imitation learning is defined in an sequential learning algorithmic approach similar to the DAgger on-policy algorithm in [77] which we present in Algorithm 5 as Direct Active Imitation Learning (DAIL). The original DAgger algorithm is presented as a batched learning approach where single full trajectory demonstrations are provided and used for learning. For comparison, the original DAgger algorithm is adapted for sequential learning. In addition, a variation to the DAgger algorithm, Random Query DAgger (RQ-Dagger), is presented that randomly queries the expert policy without verifying if the query improves the information matrix. The goal with the comparisons is to illustrate that even random queries that are infeasible can often provide more

information than passively obtaining demonstrations from an expert policy. Querying with respect to an active learning objective (such as the Fisher information) improves the quality of the queries and ultimately yields better learning. Thus, the active learning approach should result is a set of queried demonstrations that learns a behavior-cloned policy that matches the expert policy with less data than a method that passively acquires expert demonstrations.

Algorithm 5 Direct Active Imitation Learning (DAIL) via Fisher Information Maximization

1: Randomly initialize policy model π with parameter θ, data buffer $\mathcal{D}$, max demonstrations N and roll-out horizon parameter T
2: initialize $\mathcal{D}$ with single T trajectory demonstration from expert
3: **for** $i \in \{1, \ldots, N\}$ **do**
4: **for** $t \in \{0, \ldots, T-1\}$ **do**
5: sample state x_t
6: search inf. max. query with x_t as seed
7: $x^\star = \arg\min_x \operatorname{tr}\left(\mathcal{I}(\theta)^{-1}\right)$
8: $\triangleright$ sequential DAgger step
9: apply $u_t = \pi(x_t, \theta)$ to robot
10: query teacher $u^\star = \pi^\star(x^\star)$
11: add label to data $\mathcal{D} \leftarrow \{x^\star, u^\star\}$
12: gradient step θ log likelihood by sampling M data points from $\mathcal{D}$
13: **end for**
14: **end for**

4.2.1 Interactively learning to walk

Algorithm 5 is applied to the ant bullet environment [74] where the task is for simulated ant to learn how to walk solely from expert demonstrations. Here, the expert policy is obtained using a model-free reinforcement method [10]. The ant robotic system has the ability to query a single state point $x^\star$ to get an expert labeled control demonstration $u^\star$. The DAIL is compared against DAgger and the RQ-DAgger. In Algorithm 5 line 7 is solved by sampling 10 normally distributed states with the mean at the current state x with a standard deviation of 0.2. Each sample is used to evaluate

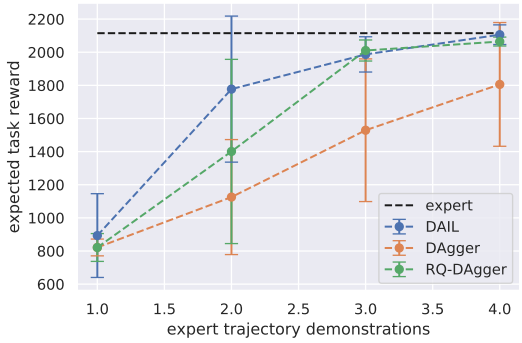


Figure 4-2: Behavior-cloned policy evaluation curves (evaluated over 1000 time steps) for the ant bullet environment using Algorithm 5 (DAIL), DAgger, and RQ-DAgger methods versus number of episodic expert demonstrations (each totaling 1000 queries). The expert policy is obtained using a model-free reinforcement method [10]. The error bars indicate 95% confidence bounds over 5 random seeds. The proposed active imitation learning algorithm is able to learn a policy that is comparable to the expert within a single trajectory demonstration.

the Fisher information matrix where the sample with the highest information value is chosen as a query point. The random query-based version of DAgger uses the same random sampling strategy, but does not post filter for the most informative sample. The random query version is used as benchmark to illustrate the improvements of arbitrarily querying the expert versus doing so with respect to an active learning objective. All methods use a two layer network with 32 nodes each with the rectifying linear unit (ReLU) as the policy model.

In Figure 4-2 we can see that the learned policy through the proposed active query-based method performs comparable to the expert in a couple of episodic demonstration of the task. Furthermore, Fig. 4-2 illustrates that having the ability to arbitrarily query the expert policy improves the behavior cloned policy performance when compared to passively acquiring demonstrations. Thus, having the ability to query in imitation learning generally improves the capabilities of robotic systems for learning tasks from demonstrations alone.

4.3 Discussion and summary

While the results in this chapter illustrate that active learning improves the quality of models that robotic systems are required to learn, there are still a few limitations with the proposed active learning approach. The first issue is the assumption that our measurements are dense and the robot is able to measure data at any time. If the measurements are sparse, either in time or spatially[3], then acquiring informative measurements becomes much more challenging, requiring more of an effort on the robot's part to explore. Given the nature of the nonlinear Fisher information optimality conditions, it is likely that the robot will run into local optima which prevent it from exploring in challenging environments.

The next issue is with regards to dimensionality and robot safety. If you are a keen reader that is familiar with the network function approximators, you would have noticed that the models are small compared to the normal multi-layer networks with at least over a hundred nodes per layer. These modeling choices were made as a consequence of direct gradient-based optimization of active learning objectives which are generally computationally expensive. Thus, the increased computation time, and the non-linearity of the active learning objective make robot safety a primary concern when exploring in highly dynamic and volatile exploration spaces. Unfortunately, avoiding these unstable regions is often undesirable as its generally the case that these regions hold dynamic information that is necessary for learning.

The last limitation is with regards to active imitation learning. In order for active imitation learning to be possible, there needs to be a system in place for which the robot can query the expert through a joint state interface. In the example in this chapter, the expert policy is directly used to return an action at any state. This is not always the possible, and in general not a situation

[3]That is, the measurement value change infrequently depending on time or place

that will occur with robots in the real-world. In addition, the queried states are infeasible, thus any physical interface for queries will not be possible without some projections into a physically realizable state. If the states are dynamic, then the expert may have difficultly providing feedback.

These limitations should not detract from the fact that actively acquiring measurements still provides significant improvements to robot learning and the limitations are simply challenges for future work. The following chapters address each of the limitations mentioned above through the use of indirectly solving active learning objectives through thoughtful structure that embeds exploration, provides safety and stability as part of active learning, and models the physical world through infinite linear abstractions that makes active learning a simpler and more realizable problem to solve in robotics.

Chapter 5

Modeling sparse environments through ergodic exploration

It is common to see a combination of movement and sensing in many biological systems, particularly with tactile sensing [84–87]). In a similar manner, humans use their hands to grasp objects and actively move their fingers across objects. These sensing behaviors are a consequence of spatially sparse environments where learning about elements in an environment can only be done through intentional behaviors that enable informative sensor information. If we expect robotic systems to operate under these conditions, we need to develop methods that are capable of generating intentional exploratory behaviors that overcome sparse environments.

As mentioned in Chapter 2, ergodic control has been used previously to enable exploration for spatially distributed information [19, 88, 89]. In this chapter, I present a method for actively exploring and learning models of sparse environments that uses ergodic exploration at its core. By using ergodic exploration, in combination with information measures on learned models, I present a specific application of active learning where a robot becomes capable of exploring and building

models of environments where sensor feedback is sparse and nonlinear. Furthermore, I show that the same approach can be used for localized the environment through an active learning approach based on spatial transforms and illustrate results on robotic systems. Note that some of the ideas in this chapter were partially presented in a prior masters thesis; however, all the results, the text, and theoretical results do not overlap and are new contributions that were later published in [57].

5.1 Ergodic control for exploring sparse worlds

Consider a robotic agent whose dynamics are governed by the control-affine dynamical system of the form

$$\dot{x} = f(x(t), u(t)) = g(x(t)) + h(x(t))u(t)$$

as mentioned in Chapter 2 Equation 2.4 where x is the state of the robot. Next, define a bounded domain $\mathcal{S}^v \subset \mathcal{X}^n$ with $v \leq n$ such that $y \sim p(\cdot|s)$ is a sample (or measurement) at $s \in \mathcal{S}^v$ with probability $p(y|s)$. Here, $p(y|s)$ is unknown and is *spatially sparse*, that is, the measurements in the search domain take on only a few values that change infrequently over movement in $\mathcal{S}^v$. The goal in this chapter is to learn $p(y|s)$ (often referred to as a measurement model) by collecting measurements y_i, s_i as the robot moves around the world. The difficult part is that the measurements y_i are spatially sparse. Thus, in order to acquire novel measurements y_i, the robot will need to explore the search domain. We approach this problem through ergodic control.

Ergodic control allows us to synthesize a control law that allows a robot to take measurement samples in a space proportional to some spatial statistical distribution $\phi(s)$. We set this problem for sparse environments by specifying a unique $\phi(s)$ such that the robot samples from regions which yield informative measurements that allow the robot to model $p(y|s)$ from sampled data. Ergodic

control is first calculated through the definition of the ergodic metric (see Chapter 2 for more detail) presented again below:[1]

$$\begin{aligned}\mathcal{E}(x(t)) &= \sum_{k\in\mathbb{N}^v} \Lambda_k \left(c_k - \phi_k\right)^2 \\ &= \sum_{k\in\mathbb{N}^v} \Lambda_k \left(\frac{1}{T}\int_{t_i}^{t_i+T} F_k(\bar{x}(t))dt - \phi_k\right)^2\end{aligned}$$

where

$$\phi_k = \int_{\mathcal{S}_v} \phi(s)F_k(s)ds, \quad c_k = \frac{1}{T}\int_{t_i}^{t_i+T} F_k(\bar{x}(t))dt,$$

$T \in \mathbb{R}^+$ is the time horizon, $t_i \in \mathbb{R}^+$ is the i^{th} sampling time, $s \in \mathbb{R}^v$ is the search domain such that $\bar{x}(t) : \mathbb{R}^+ \to \mathcal{S}^v$ is the state of the robot that intersects $\mathcal{S}^v$,

$$F_k(x) = \frac{1}{h_k}\prod_{i=1}^{v} \cos\left(\frac{k_i \pi x_i}{L_i}\right)$$

being the cosine basis function for a given coefficient $k \in \mathbb{N}^v$, h_k is the normalization factor defined in [27], and $\Lambda_k = (1 + \|k\|^2)^{-\frac{v+1}{2}}$ is a weight on the frequency coefficients. Using a variation of the Maximum principle [90] on the augmented objective

$$\mathcal{J} = \mathcal{E}(x(t)) + \int_{t_i}^{t_i+T} u(t)^\top R u(t)dt$$

we can obtain an ergodic controller through the adjoint (or co-state) equation

$$\dot{\rho} = -\frac{2}{T}\sum_{k\in\mathbb{N}^v} \Lambda_k(c_k - \phi_k)\frac{\partial F_k}{\partial x} - \frac{\partial f}{\partial x}^T \rho$$

[1] A slight change in notation for the target distribution is done to distinguish the model $p(y|s)$ from the target distribution $\phi(s)$.

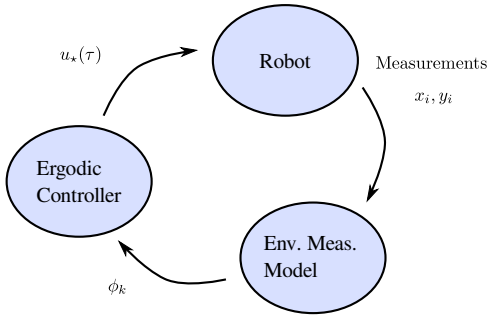


Figure 5-1: Block diagram for receding horizon ergodic control with online model construction. Measurements are processed and used to update the model as well as the target distribution $\phi(s)$. This information is then passed onto the ergodic controller which returns an action for the robot to take.

where $\rho(t_i + T) = \mathbf{0} \in \mathbb{R}^n$. The control is then

$$u^\star(t) = R^{-1} h(x)^T \rho(t)$$

where $R \in \mathbb{R}^{m \times m}$ is a positive definite weight on the control that is obtained though the augmented objective function. Controls are calculated in a receding horizon manner to incorporate new information into the model for $p(y|s)$ which subsequently updates $\phi(s)$ as shown in Fig 5-1.

5.2 Non-parametric model information and spatial transforms

Using the probabilistic nature of the measurement model $p(y|s)$, we can derive a target distribution $\phi(s)$ to which we pass to the ergodic controller. This is done through the reparametrization trick that is often used with the Fisher information matrix (see Chapter 4). Consider that we already have a measurement model $p(y|s)$ and we want to use the model to localize the robot in a new space $\bar{s} = g(\theta)^{-1} s$ where $g(\theta) \in \text{SE}(v)$ where $\text{SE}(v)$ is the v dimensional group transform of the space $s \to \bar{s}$ and θ is a parameterization of $\text{SE}(v)$. Furthermore, let us consider a prior probability

distribution $p(\theta)$ over the parameter θ.

Lemma 2. *The expected information of an environment measurement model with respect to a transformation $g(\theta)$ is given by*

$$\phi(s) = \det\left[\int_{\theta} \frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}^{\top} \mathcal{I}(s) \frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta} p(\theta) d\theta\right] \tag{5.1}$$

where

$$\mathcal{I}(s) \approx \frac{\partial\mu}{\partial s}^{\top} \Sigma^{-1} \frac{\partial\mu}{\partial s}$$

is the Fisher information with respect to the search space of the robot, $p(\theta)$ is the belief of the parameter θ, $p(y|x) \approx \mathcal{N}(\mu(s), \Sigma)$ is approximated locally as a normal distribution, and D-optimality [91,92] (determinant) is chosen a measure of information.

Proof. We first calculate the Fisher information matrix of the locally approximated measurement model $p(y|\bar{s}) \approx \mathcal{N}(\mu(\bar{s}), \Sigma))$ with respect to the transformation:

$$\mathcal{I}(s, \theta) = \frac{\partial\mu}{\partial\theta}^{\top} \Sigma^{-1} \frac{\partial\mu}{\partial\theta}. \tag{5.2}$$

Applying chain rule to the derivative terms in (5.2) gives us the expanded derivative terms

$$\frac{\partial\mu}{\partial\theta} = \frac{\partial\mu(\overbrace{g(\theta)^{-1}s}^{\bar{s}})}{\partial\bar{s}} \frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}. \tag{5.3}$$

Replacing (5.3) into (5.2) gives

$$\begin{aligned}\mathcal{I}(s,\theta) &= \left(\frac{\partial \mu}{\partial \bar{s}}\frac{\partial g(\theta)^{-1}s}{\partial \theta}\right)^{\top}\Sigma^{-1}\left(\frac{\partial \mu}{\partial \bar{s}}\frac{\partial g(\theta)^{-1}s}{\partial \theta}\right) \\ &= \frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}^{\top}\underbrace{\left(\frac{\partial \mu}{\partial \bar{s}}^{\top}\Sigma^{-1}\frac{\partial \mu}{\partial \bar{s}}\right)}_{\mathcal{I}(s)}\frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}.\end{aligned} \tag{5.4}$$

As $\mathcal{I}(s,\theta)$ is still uncertain in θ, we take the expectation of (5.4) and map the information matrix to a scalar value using the D-optimality measure [91,92] giving us the expected information density of a measurement model

$$\phi(s) = \det\left[\int_{\theta}\frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}^{\top}\mathcal{I}(s)\frac{\partial\left(g(\theta)^{-1}s\right)}{\partial\theta}p(\theta)d\theta\right].$$

□

Notice that in Lemma 2, we never specified a parametrization for $p(y|s)$ which gives us the capability to utilize non-parametric methods for acquiring the measurement model from solely data. This is beneficial as non-parametric modeling techniques such as Gaussian processes or support vector machines (SVM) allow us to model arbitrarily complex environments. That being said, it is still possible to parametrize $p(y|s)$ and carry out the analysis for $\mathcal{I}(s)$ if necessary, but for sparse environments, non-parametric methods are ideal as few key measurements are required to model the environment.

In the following sections, we illustrate the use of ergodic control with sampling proportional to the expected information $\phi(s)$ derived through the Fisher information matrix $\det[\mathcal{I}(s)]$ which is obtained through the identity transform in Eq. 5.2. As shown in [93], $\mathcal{I}(s)$ can be approximated using the transition surface dictated by $p(y=1|s)$. Sampling with respect to Fisher information

matrix allows the robot to sample environment measurements that reduce the overall uncertainty of the model based on the measurements themselves (due to the non-parametric nature of the model we will be using). We then illustrate the use of the proposed method for localization in sparse environments through the same information density described in Eq. 5.2.

5.3 Modeling an environment through contact

We consider the case where we have a second order point-mass robot[2] that experiences the world through point-based contact. In the environment, there are fixed objects that are unknown to the robot that we want to model and use to localize the robot at a later time. This is accomplished through the use of ergodic control while constructing the measurement model $p(y|s)$. Here, the measurement model is defined using a SVM and the target spatial statistics $\phi(s)$ is defined through the identity transform, making

$$\phi(s) = \det\left[\mathcal{I}(s)\right]$$

which we approximate with the transition surface of the sparse model $p(y = 1|s)$ as done in [93]. [3] An example ergodic trajectory is presented in Fig. 5-2 for learning a contact-based measurement model of the object in the environment. Here, we can see that the robot is able to collect measurements which fully captures each of the objects in the measurement model. In addition, the trajectory is shown exploring regions where there are no objects, which is a direct consequence of minimizing the ergodic exploration. More specifically, a trajectory that minimizes the ergodic metric is one where the robot spends a proportional amount of time with respect to the statistics in the target distribution. Thus, the non-zero probability that a contact measurement can be encountered

[2] States are the position in the environment and velocity.

[3] This approximation is valid as the gradient of the mean of the sparse measurement model is zero everywhere but at the transition between the sparse measurements.

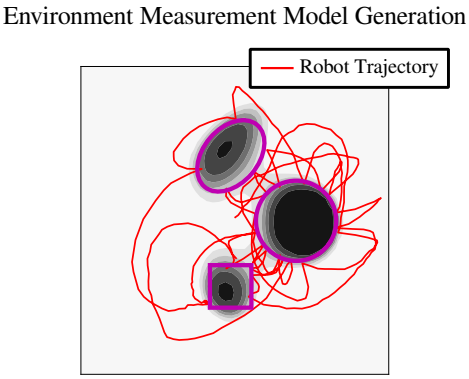


Figure 5-2: Estimating the measurement model of the environment with multiple objects in a 2-dimensional search space using contact measurements. Objects are shown with the purple outline. The background contours indicating where there is high likelihood of a contact in darker regions. The learned measurement model is used to guide the exploration through the model information as data is collected. The resulting robot trajectory is shown as the red line. Note that the robot still visits low probability region in search for new information. Thus, the robot is able to sufficiently explore around objects and the environment.

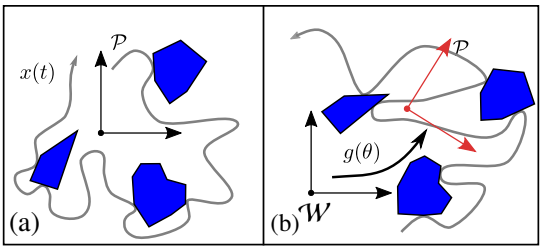


Figure 5-3: Diagram of localization with sparse models using ergodic control. Objects in the environment are defined by the blue shapes. (a) A robot with an exploratory trajectory $x(t)$ is shown as the gray line estimating the measurement model of an environment in the fixed frame $\mathcal{P}$. Contact measurements are used to construct the measurement model. (b) The same environment originally shown in $\mathcal{P}$ is transformed into search space $\mathcal{W}$ under transform $g(\theta)$. The robot exploratory trajectory used to generate an estimate of the transform $g(\theta)$ is created using the learned measurement model in (a).

ensures us that the robot will eventually sample and explore these regions as time goes to infinity.

The model that is learned can now be recycled for localization with a sparse environment (see Fig. 5-3 for illustration). The robot is now in a transformed environment which we can define a measurement point as $\bar{s} = g(\theta)^{-1}s$ where θ is the unknown environment transformation with probability $p(\theta)$. Since we now have a transform $g(\theta)$ that is not the identity transform, we can use the expression for $\phi(s)$ in Eq. 5.1 for ergodic exploration.

Normally, the parameter distribution $p(\theta)$ is updated through some variation of a Bayesian filter based off a measurement model $p(y|s)$. However, because the measurement model is sparse, and

as a consequence discontinuous, the continuity assumptions needed by the Bayesian filter are not available. Hence, we use a manifold particle filter [94] which is generally used with measurement models that are discontinuous.Figure 5-4 illustrates the procedure for localization given a model of the sparse contact environment. In this example, the initial target distribution is initialized as uniform target distribution until the first contact. A belief over how the environment is transformed is provided by the particle filter which is subsequently used to calculate $\phi(s)$ shown in the background of Fig. 5-4. The robot trajectory is shown reacting to the expected information distribution as measurements are acquired. Interestingly, the behavior of the robot is shown to utilize the lack of contact measurements just as much as contact measurements to reduce the uncertainty in where the environment is located. As a result, exploration, regardless of sparsity, is encouraged with ergodic control and provides us with a method to avoid local minima as a result of non-convex information measures which guide active learning. We validate this claim in the following section with a comparison using a common active learning method known as expected entropy reduction (EER) [19, 88, 95].

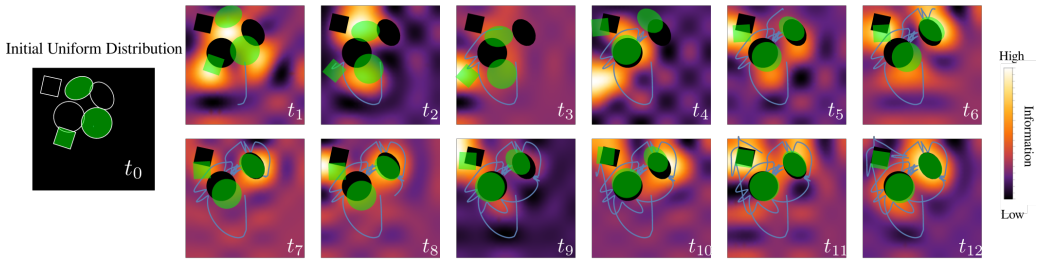


Figure 5-4: Localization of objects using a given measurement model is illustrated for an environment containing three objects. The expected information of the measurement model is shown as the distribution in the background of each time frame t_i space 1.25 seconds apart. The transparent green objects are the depiction of the current belief of their location $p(\theta)$. The ground truth object location is shown in black and the robot trajectory in shown as the blue line. The robot estimates the transformation using the information of the environment measurement model as a target for active sensing.

5.3.1 Comparison to entropy reduction

We compare our method against expected entropy reduction (EER) [19, 88, 95] which is a method similar to that used in [96–99] where a location in the search space is chosen based on where the expected change in entropy is maximized. The location is chosen from a set of 200 uniformly distributed samples in the search space. The trajectory that maximizes the entropy reduction is then chosen and an LQR controller is used to control the robot to the target location.

Here, the reduction in entropy is calculated as

$$H_{\text{reduction}} = H(\theta) - \mathbb{E}\left[H(\theta) \mid y(t)^{+}\right]$$

where

$$H(\theta) = -\int p(\theta) \log p(\theta) d\theta$$

is the entropy for a random variable θ, and $y(t)^{+}$ is the expected measurement given the current model along the sampled trajectory. In the case of measurement model construction with contacts using non-parametric models, θ is replaced with s whereas in environment localization, the random variable is the transform parameters θ for $g(\theta) \in \text{SE}(2)$.

We compare a 20 second simulation of environment construction and localization in Fig. 5-5. EER prioritizes immediate reduction in entropy, thus only the most recent and known measurements are used rather than considering where the robot has or has not been. In situations where measurements are sparse, this causes the EER algorithm to become stuck as shown in Fig. 5-5(d). Similarly, in Fig. 5-5(b), EER is only able to acquire measurements with two out of the three objects in the environment, and only one side of each object. In contrast, ergodic control as shown in Fig. 5-5 (a) and (c) illustrates the benefit for seeking out informative measurements in sparse and

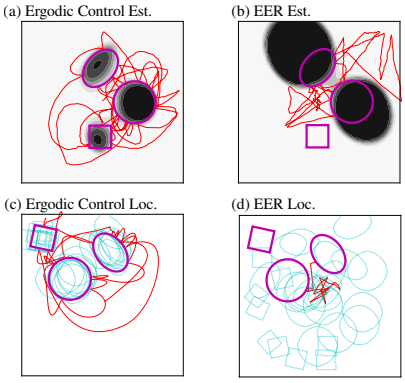


Figure 5-5: Comparison of Ergodic control versus entropy reduction for environment measurement estimation (a-b) and localization using the environment measurement model that is known (c-d). Contact likelihood shown as the dark regions in (a-b). Particle filter beliefs are shown as the cyan colored estimates of the shapes in (c-d). Using ergodic control enables a robot to consider low probability regions which are beneficial in acquiring measurements that are sparse in the environment. Entropy reduction results in search patterns that immediately reduce the entropy which results in a lack of exploration.

in unknown environments. Low probability regions are not omitted and are important to exploration and learning. As time goes to infinity, the ergodic controller would drive the robot to explore annd learn about the whole domain such that the time spent is proportional to the statistics in the region [19, 88, 89, 95].

5.4 Hamster ball robot exploration

In this section, we present experimental validation of our method for active learning in sparse environments. A hamster ball robot (ball with an internal differential drive robot) known as the Sphero SPRK is used for collision-based sensing. The SPRK robot has an internal contact sensor which uses its inertial measurement unit to detect a collision with an external object. Position and velocity of the robot is acquired using image tracking from above. Once a measurement model is generated in the first stage, we use this model to localize the transformed environment. Note that the only assumption that is made is that the robot moves subject to planar point-mass dynamics.

Experimental results for learning the measurement model from contact with the SPRK robot are shown in Fig. 5-6. Within the first few measurements, the environment measurements are not

useful as the robot has not explored the entire domain. However, due to the low probability value of unexplored regions, the robot under ergodic control must still visit these regions. In doing so, it becomes more likely that the robot encounters more informative measurements that are used to learn the difficult to model sparse contact environment. By the termination time ($t_f = 120$ seconds), the robot has constructed a contact-based measurement model that is sufficient for subsequent localization.

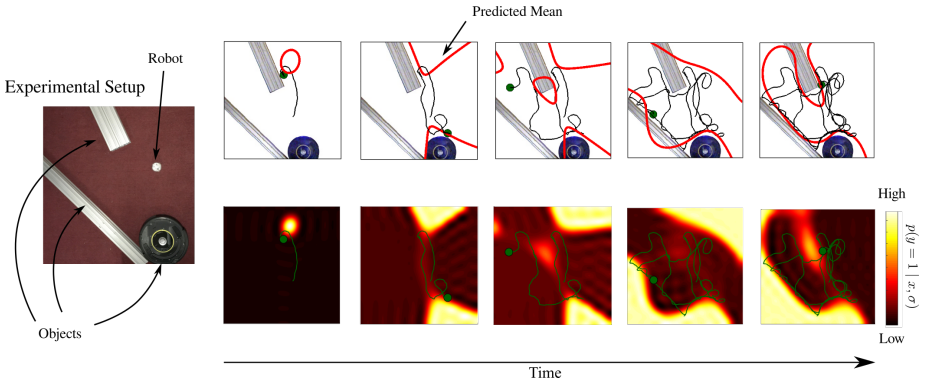


Figure 5-6: Left: Photo of the experiment setup of the environment. The robot is initialized with a uniform target distribution. Top: Time series of the environment measurement model construction process from left to right. The red boundaries indicate the mean transition value of the measurement model. Sphero robot is shown as a green circle. The robot trajectory up to the current time is also shown. Bottom: $p(y|x)$ is shown being built as the robot collides with the objects in the environment.

It is worth noting that the collision sensor of the robot is sensitive to quick movements which will result in false-positive contact measurements. While in a engineered or physics-based approach, the noisy behavior of the robot's sensors are modeled, our approach automatically learns the measurement model, including the uncertainty based on measurements that disagree with one another. Furthermore, this uncertainty is used by the ergodic controller to return to regions that is disagreement due to the information gained by disambiguating the measurements.

We can use the same approach with the learned measurement model to localize a transformed

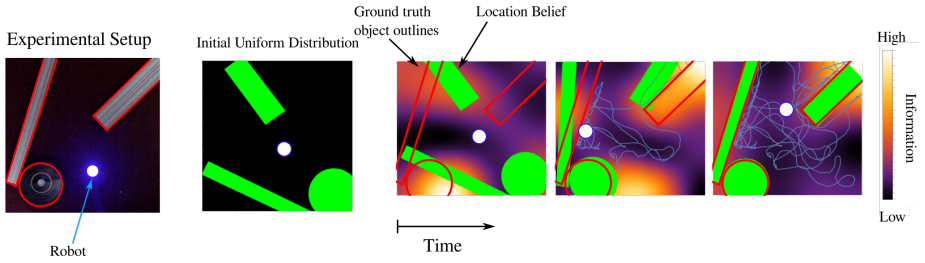


Figure 5-7: The robot estimates the transformation of the environment state using the measurement model extracted from Fig. 5-6 using contact measurements. The global position of the objects is shown as the red outlines. The ergodic controller generates a robot trajectory with a sequence of contact measurements that localizes the environment based on the acquired measurement information defined in Eq. 5.1.

environment with similar contact features. The objects presented in Fig. 5-6, are rotated and translated as shown in Fig. 5-7. The initial prior belief $p(\theta)$ is defined as a uniform distribution over the domain $\theta \in [0, 1.2] \times [0, 1] \times [-2\pi, 2\pi]$ which correspond to the positional and rotational transformation. Here, the experiment is run for $t_f = 130$ seconds. We can see that as the robot begins to react to the measurements with ergodic exploration, the localization of the environment starts to approach the ground truth shown as the red outlined shapes.

While the sparse contact measurements are important in this experiment, the absense of the sparse measurements are found to be equally, if not more important for localizing. We can see an example of this in the final frame of Fig. 5-7 where the robot explores the regions where a sparse contact is less likely to occur. This kind of concept is anecdotally common when we think of how humans or animals explore. Specifically, the absence of an object is generally more informative than the chance to encounter the object. Ergodic exploration allows us to place this anecdote into concrete mathematics that we can use to improve the capabilities of robotic systems. In the end of the experiment, the robot is successful in estimating the current configuration ($\theta_{\text{actual}} = [0.5, 0.6, -1.1]$, $\theta_{\text{estimated}} = [0.521, 0.609, -1.103]$) of the environment through the use of the learned measurement model of the sparse environment.

5.5 Discussion and summary

In this chapter, I presented a method for actively learning and utilizing models of sparse environments with the aid of ergodic exploration. The ergodic controller enables robotic systems to actively seek out informative measurements in sparse environments through Fisher information-based active learning measures in search for new information. The proposed approach illustrates the ability to enact active learning in robotic systems when passive learning is not an option. Through simulated and experimental examples, I show examples of constructing models of sparse environments in simulated and real-robot examples where the robot is required to put additional work into collecting measurements and doing so in an intentional manner. The following chapter looks at the case where we not only want the robot to learn with intent, but we want the robot to do so in a safe manner. This is a problem not mentioned in this chapter as contacts and collisions, in the current state of some robotic systems, are generally undesirable. However, what if we can allow robotic systems to explore and collective information rich measurements in dynamic spaces where safety is of the upmost importance?

Chapter 6

Safe ergodic active learning in high dimensional dynamic search spaces

This chapter reflects upon previously presented work as inspirational building blocks for the development of a safe active learning strategy for robotic systems in high dimensional dynamic exploration spaces. I first present a variation to the ergodic metric that allows robotic systems to expand ergodic active learning to larger search spaces. The ideas from hybrid learning (Chapter 3) are then utilized for embedding safe equilibrium policies in a manner that does not impede on the inherently unstable behavior that is active learning. I prove that the approach not only guarantees that the robot will continue to take actions that provide informative measurements, but also maintains the robot in dynamic regions where it can safely return to an equilibrium state. Last, I evaluate the proposed method on a number of active learning scenarios related to robot learning and show that safe learning is a possibility in robotics when one jointly formulates the problem of learning with stability and safety in mind.

6.1 KL-ergodic measure for exploration

As in discussed in Chapter 5 and in [19,100], one can calculate a controller that optimizes the ergodic metric such that the trajectory of the robot is ergodic with respect to a distribution $p(s)$. However, this approach scales $\mathcal{O}(|k|^n)$ where $|k|$ is the maximum integer–valued Fourier term. As a result, this method is ill-suited for high-dimensional learning tasks whose exploration states are often the full state-space of the robot (often $n > 3$ for most mobile robots). Furthermore, the resulting time-averaged distribution reconstruction will often have residual artifacts from the Fourier transform which require additional conditioning to remove. This motivates the following section which defines an ergodic measure for active learning. [1]

As an alternative to computing the ergodic metric, we present an ergodic measure which circumvents the scalability issues. To do this, we utilize the Kullback-Leibler divergence [20, 101, 102] (KL–divergence) as a measure for ergodicity. Let us first define the approximation to the time-averaged statistics of a trajectory $x(t)$:

Definition 3. *Given a search domain $\mathcal{S}^v \subset \mathcal{X}^n$ the Σ-approximated time-averaged statistics of the robot trajectory from time t_0 to t_f is defined by*

$$q(s \mid x(t)) = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \mu(s \mid \bar{x}(t)) dt \tag{6.1}$$

where $\mu(s \mid \bar{x}(t)) = \mathcal{N}(\bar{x}(t), \Sigma)$, $\Sigma \in \mathbb{R}^{v \times v}$ is a positive definite matrix parameter that specifies the width of the Gaussian, and $\bar{x}$ is the state that intersect the search space.

We call this an approximation because the true time-averaged statistics, as described in [19],

[1] We make note of the use of the work "measure" as opposed to metric as we allude to using the KL-divergence which itself is not a metric, but a measure.

is a collection of delta functions parameterized by time. We approximate the delta function as a Gaussian distribution with variance Σ, converging as $\|\Sigma\| \to 0$. As an aside, one can treat Σ as a function of $\bar{x}(t)$ if there is uncertainty in the position of the robot.

With this approximation, the ergodic objective in [19] is relaxed using the following KL-divergence objective [20]:

$$\begin{aligned} D_{\mathrm{KL}}(p\|q) &= \int_{\mathcal{S}^v} p(s) \log \frac{p(s)}{q(s)} ds \\ &= \int_{\mathcal{S}^v} p(s) \log p(s) ds - \int_{\mathcal{S}^v} p(s) \log q(s) ds, \\ &= -\int_{\mathcal{S}^v} p(s) \log q(s) ds \\ &= -\mathbb{E}_{p(s)} \left[\log q(s)\right] \end{aligned}$$

where $\mathbb{E}$ is the expectation operator, $q(s) = q(s \mid x(t))$, and $p(s)$ is an arbitrary spatial distribution. Note that we drop the first term in the expanded KL-divergence because it does not depend on the trajectory of the robot $x(t)$. Rather than computing the integral over the exploration space $\mathcal{S}^v$ (as this can be intractable), we approximate the expectation operator as

$$\begin{aligned} D_{\mathrm{KL}}(p\|q) &= -\mathbb{E}_{p(s)} \left[\log q(s)\right] \\ &\approx -\sum_{i=1}^{N} p(s_i) \log q(s_i) \\ &\propto -\sum_{i=1}^{N} p(s_i) \log \int_{t_0}^{t_f} \mu(s_i \mid \bar{x}(t)) dt, \end{aligned} \tag{6.2}$$

where N is the number of samples in the search domain drawn from $p(s)$.[2] Through this formulation, we still obtain the benefits of indirectly sampling from the spatial distribution $p(s)$ without having

[2] We can always use importance sampling to interchange which distribution we sample from.

to directly compute derivatives of the commonly non-convex $p(s)$ to generate an optimal control signal for the robot.

6.1.1 KL-ergodic control synthesis

The KL-ergodic measure is used to construct exploration strategies for robotic systems in a similar manner as with ergodic control. Constructing a model-based ergodic controller using the KL-divergence is done in a similar manner as presented in Chapter 5 with an augmented objective function:

$$\begin{aligned}\mathcal{J} &= D_{\mathrm{KL}}(p\|q) + \int_{t_0}^{t_t} u(t)^\top R u(t) dt \\ &= -\sum_{i=1}^{N} p(s_i) \log\left(\int_{t_0}^{t_f} \mu(s_i \mid \bar{x}(t)) dt\right) + \int_{t_0}^{t_t} u(t)^\top R u(t) dt \end{aligned} \tag{6.3}$$

where $R \in \mathbb{R}^{m\times m}$ is a positive definite weight matrix on the control input, and $x(t)$ is obtained from simulating Eq. 2.4. Solving 6.3 with respect to a sequence of planned controls $u(t)$ can be done either through direct optimization or using the Maximum principle [90] to define conditions for optimality through a shooting-based approach. An example trajectory generated by optimizing Eq. 6.3 with respect to a bi-modal distribution using a robot with point mass dynamics is presented in Fig. 6-1. Samples are drawn uniformly within the search space at each optimization iteration of Eq. 6.3.

We can compare the KL-ergodic control with the Fourier-based ergodic control in Fig. 6-2. In particular, we can see that KL-ergodic measure yields time-averaged trajectory statistical reconstructions that do not have additional artifacts (compared with the Fourier-based ergodic control). Furthermore, the complexity of calculating the KL-ergodic measure does not increase with the

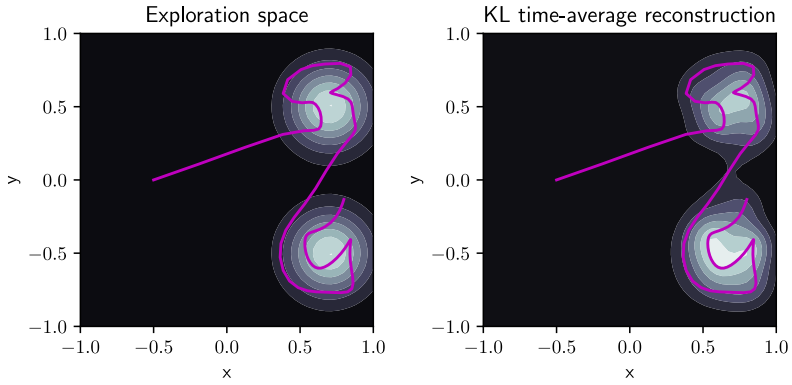


Figure 6-1: Illustration of a KL-ergodic trajectory and the time-averaged trajectory statistics reconstruction. The target distribution (left) is a bi-modal distribution. The robot trajectory (magenta) is shown ergodicaly sampling through optimizing Eq. 6.3. The time-averaged statistics of the robot trajectory (right) is constructed using Definition 3.

exploration state which allows us to compute ergodic exploration controllers for much larger dimensional exploration spaces.

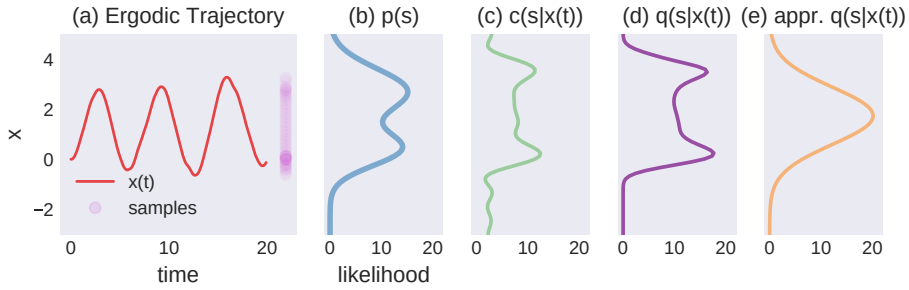


Figure 6-2: (a) Illustration of an ergodic trajectory $x(t)$ with respect to (b) target distribution $p(s)$. Time-averaged distribution reconstructions of $x(t)$ are shown using Definitions 2, 3, and Eq. 6.17. The Fourier decomposition approach often has residual artifacts due to the cosine approximation. 20 basis functions are used to approximate the time-averaged distribution. Σ-approximation to the time-averaged statistics minimizes residual artifacts. (e) Moment matching of the Σ-approximation provides a simplification for computing the time-averaged statistics.

The following section uses the KL-ergodic measure to derive a controller which optimizes (6.2) while directly incorporating learned models and policies. The ideas and theoretical analysis pre-

sented are a direct extension of the work in Chapter 3.

6.2 KL-E^3: KL-Ergodic Exploration from Equilibrium

In this section, KL-Ergodic Exploration from Equilibrium (KL-E^3) is derived which locally optimizes and improves (6.2) subject to safety constraints. Here, stability is imposed through an feedback control equilibrium policies that defined by control Lyaponov functions (CLFs) [31, 32] and approximate transition model of the robot's dynamics on the algorithm. By synthesizing KL-E^3 with existing CLF policies that allow the robot to return to an equilibrium state (e.g., local linear quadratic regulator (LQR) controllers), we can take advantage of approximate transition models for planning the robot's motion while providing a safety guarantees that the robot will be maintained within the safety region of attraction defined by the CLF. I then show how this method is Lyapunov attractive [103, 104] which is related to control Barrier Functions (CBFs) safety definition [35, 38], allowing the robot to freely move about the CLF region of attraction and ensuring that the robot will eventually return to an equilibrium.

6.2.1 Assumptions for equilibrium

We assume that we have a robot whose approximate dynamics can be modeled using the continuous time transition model (2.4) restated below:

$$\begin{aligned}\dot{x}(t) &= f(x(t), \pi(x(t))) \\ &= g(x(t)) + h(x(t))\pi(x(t))\end{aligned}$$

In in our modeling assumption, we consider a feedback stabilizing equilibrium policy $\pi(x) : \mathcal{X}^n \rightarrow \mathcal{U}^m$ which provides the control signal to the robotic system such that there exists a continuous Lyapunov function $V(x)$ [31, 104, 105] which has the following conditions:

$$\begin{aligned} & V(0) = 0 \\ \forall x \in \mathcal{B} \backslash \{0\} \quad & V(x) > 0 \\ \forall x \in \mathcal{B} \backslash \{0\} \quad & \nabla V \cdot f(x, \pi(x)) < 0 \end{aligned} \tag{6.4a}$$

where $\mathcal{B} \subset \mathbb{R}^n$ is a compact and connected set, and

$$\dot{V}(x(t)) = \frac{\partial}{\partial x} V(x) \cdot f(x, u) = \nabla V \cdot f(x, u) \tag{6.5}$$

which implies a monotonically decreasing Lyaponov function. This is also referred to as a control Lyaponov function (CLF) [31, 38, 104] . Thus, a trajectory $x(t)$ with initial condition at time $t = t_0$ subject to the dynamics and $\pi(x)$ is defined as

$$x(t) = x(t_0) + \int_{t_0}^{t} f(x(t), \pi(x(t)))dt. \tag{6.6}$$

For the rest of the paper, we will refer to $\pi(x)$ as an equilibrium policy which is constructed from a CLF which returns the robot to an equilibrium state. As an aside, CLFs are closely related to the value function in optimal control [106] which can be used to define a equilibrium policy for a task other than stabilization.

6.2.2 Synthesizing a schedule of exploratory actions

Given the assumptions of a known approximate model with a valid equilibrium policy, the goal is to generate a control signal that augments $\pi(x)$ and minimizes (6.2) while ensuring the state x remains within the compact set $\mathcal{B}$ which will allow the robot to return to an equilibrium state within the time $t \in [t_0, t_f]$ once the active learning objective is minimized.

The approach starts by using the same approach as done in hybrid learning (Chapter 3) by quantifying how sensitive (6.2) is to switching from the policy $\pi(x(t))$ to an arbitrary control vector $\hat{u}(t)$ at any time $\tau \in [t_0, t_f]$ for an infinitesimally small duration of time λ. We will later use this sensitivity to calculate a closed-form solution to the most influential control signal $u^\star(t)$.

Proposition 1. *The sensitivity of (6.2) with respect to the duration time λ, of switching from the policy $\pi(x)$ to an arbitrary control signal $\bar{u}(t)$ at time τ is*

$$\frac{\partial D_{KL}}{\partial \lambda} = \rho(\tau)^\top (f_2 - f_1) \tag{6.7}$$

where $f_2 = f(x(\tau), \hat{u}(\tau))$ and $f_1 = f(x(\tau), \pi(x(\tau)))$, $\rho(t) \in \mathbb{R}^n$ is the adjoint, or co-state variable which is the solution of the following differential equation

$$\dot{\rho}(t) = \sum_i \frac{p(s_i)}{q(s_i)} \frac{\partial \mu}{\partial x} - \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} \frac{\partial \pi}{\partial x} \right)^\top \rho(t) \tag{6.8}$$

subject to the terminal constraint $\rho(t_f) = \mathbf{0}$, and $\frac{\partial \mu}{\partial x}$ is evaluated at each sample s_i.

Proof. See Appendix A □

The sensitivity $\frac{\partial}{\partial \lambda} D_{\mathrm{KL}}$ is known as the mode insertion gradient [107]. Note that the second term in (6.8) encodes how the dynamics will change subject to the policy $\pi(x)$. We can directly

compute the mode insertion gradient for any control $\hat{u}(t)$ that we choose. However, our goal is to find a schedule of $\hat{u}(t)$ which minimizes (6.2) while still bounded by the equilibrium policy $\pi(x)$. We solve for this augmented control signal by formulating the following optimization problem:

$$u^\star(t) = \underset{\hat{u}(t)\forall t\in[t_0,t_f]}{\arg\min} \int_{t_0}^{t_f} \frac{\partial}{\partial\lambda} D_{\mathrm{KL}}\Big|_{\tau=t} + \frac{1}{2}\|\hat{u}(t) - \pi(x(t))\|_R^2 dt \tag{6.9}$$

where $R \in \mathbb{R}^{m\times m}$ is a positive definite matrix that penalizes the deviation from the policy $\pi(x)$ and $\frac{\partial}{\partial\lambda} D_{\mathrm{KL}}|_{\tau=t}$ is (3.1) evaluated at time t.

Proposition 2. *The augmented control signal $u^\star(t)$ that minimizes (6.9) is given by*

$$u^\star(t) = -R^{-1}h(x(t))^\top \rho(t) + \pi(x(t)). \tag{6.10}$$

Proof. Taking the derivative of (6.9) with respect to $\hat{u}(t)$ at each instance in time $t \in [t_0, t_f]$ gives

$$\begin{aligned} &\int_{t_0}^{t_f} \frac{\partial}{\partial\hat{u}} \left(\frac{\partial}{\partial\lambda} D_{\mathrm{KL}}\Big|_{\tau=t} + \frac{1}{2}\|\hat{u}(t) - \pi(x(t))\|_R^2 \right) dt \\ &\qquad = \int_{t_0}^{t_f} h(x(t))^\top \rho(t) + R(\hat{u}(t) - \pi(x(t)))dt \end{aligned} \tag{6.11}$$

where we expand $f(x,u)$ using (2.4). Since the expression under the integral in (6.9) is convex in $\hat{u}(t)$ and is at an optimizer when (7.6) is equal to 0 $\forall t \in [t_0, t_f]$, we set the expression in (7.6) to zero and solve for $\hat{u}(t)$ at each instant in time giving us

$$u^\star(t) = -R^{-1}h(x(t))^\top \rho(t) + \pi(x(t))$$

which is the schedule of exploratory actions that reduces the objective for time $t \in [t_0, t_f]$ and is

bounded by $\pi(x)$. $\square$

In practice, the first term in (6.10) is calculated and applied to the robot using a true measurement of the state $\hat{x}(t)$ for the policy $\mu(x)$. We refer to this first term as $\delta u^\star(t) = -R^{-1}h(x(t))^\top \rho(t)$ yielding $u^\star(t) = \delta u^\star(t) + \pi(\hat{x}(t))$.

Given the derivation of the augmented control signal that can generate ergodic exploratory motions, we verify the following through theoretical analysis in the next section:

(i) that (6.10) does in fact reduce (6.2)

(ii) that (6.10) imposes a bound on the conditions in (6.4)

(iii) and that a robotic system subject to (6.10) has a notion of Lyapunov attractiveness

6.2.3 Theoretical analysis

We first illustrate that our approach for computing (6.10) does reduce (6.2).

Corollary 2. *Let us assume that* $\frac{\partial}{\partial \mu}\mathcal{H} \neq 0 \; \forall t \in [t_0, t_f]$*, where* $\mathcal{H}$ *is the control Hamiltonian. Then*

$$\frac{\partial}{\partial \lambda} D_{KL} = -\|h(x(t))^\top \rho(t)\|^2_{R^{-1}} < 0 \tag{6.12}$$

$\forall t \in [t_0, t_f]$ *subject to* $u^\star(t)$.

Proof. Inserting (6.10) into (3.1) and dropping the dependency of time for clarity gives

$$\begin{aligned}\frac{\partial}{\partial\lambda}D_{\mathrm{KL}} &= \rho(t)^{\top}\left(f_2 - f_1\right)\\ &= \rho^{\top}\left(g(x) + h(x)u^{\star} - g(x) - h(x)\pi(x)\right)\\ &= \rho^{\top}(-h(x)R^{-1}h(x)^{\top}\rho + h(x)\pi(x) - h(x)\pi(x))\\ &= -\rho^{\top}h(x)R^{-1}h(x)^{\top}\rho\\ &= -\|h(x(t))^{\top}\rho(t)\|^2_{R^{-1}} \leq 0. \end{aligned} \tag{6.13}$$

Thus, $\frac{\partial}{\partial\lambda}D_{\mathrm{KL}}$ is always negative subject to (6.10). □

For $\lambda > 0$ we can approximate the reduction in D_{KL} as $\Delta D_{\mathrm{KL}} \approx \frac{\partial}{\partial\lambda}D_{\mathrm{KL}}\lambda \leq 0$. Thus, by applying (6.10), we are generating exploratory actions that minimize the ergodic measure defined by (6.2) with respect to some active learning utility distribution $p(s)$.

Our next set of analysis involves searching for a bound on the conditions in (6.4) when (6.10) is applied at any time $\tau \in [0, t-\lambda]$ for a duration $\lambda \leq t$.

Theorem 3. *Given the conditions in (6.4) for a policy $\pi(x)$, then $V(x^{\tau}_{\lambda}(t)) - V(x(t)) \leq \lambda\beta$, where $x^{\tau}_{\lambda}(t)$ is given by*

$$x^{\tau}_{\lambda}(t) = x(t_0) + \int_{t_0}^{\tau} f(x, \pi(x))dt + \int_{\tau}^{\tau+\lambda} f(x, u^{\star}(t))dt + \int_{\tau+\lambda}^{t_f} f(x, \pi(x))dt \tag{6.14}$$

for $\tau \in [0, t-\lambda]$, $\lambda \leq t$, and

$$\beta = \sup_{t\in[\tau,\tau+\lambda]} -\nabla V \cdot h(x(t))R^{-1}h(x(t))^{\top}\rho(t). \tag{6.15}$$

Proof. See Appendix A. □

We can choose any time $\tau \in [0, t-\lambda]$ to apply $u^\star(t)$ and provide an upper bound quantifying the change of the Lyaponov function described in (6.4) by fixing the maximum value of λ during active exploration. In addition, we can tune $u^\star(t)$ using the regularization value R such that as $\|R\| \to \infty$, $\beta \to 0$ and $u^\star(t) \to \pi(x(t))$.

Given this bound, we can guarantee Lyapunov attractiveness [108], where the system is allowed to freely move within a bounded set where the system can still be driven to an equilibrium state (or set) defined by a CLF feedback equilibrium policy.

Definition 4. *A robotic system defined by (2.4) is Lyapunov attractive if at some time t, the trajectory of the system $x(t) \in \mathcal{C} \subset \mathcal{B}$ where $\mathcal{C} = \{x(t) | V(x) \leq \beta^\star, \nabla V \cdot f(x(t), \pi(x(t))) < 0\}$, $\beta^\star > 0$ is the maximum level set of $V(x)$ where $\nabla V \cdot f(x, \pi(x)) < 0$, and $\lim_{t\to\infty} x(t) \to x_0$ such that x_0 is an equilibrium state.*

Theorem 4. *Given the schedule of exploratory actions (6.10) $\forall t \in [\tau, \tau+\lambda]$, a robotic system governed by (2.4) is Lyapunov attractive such that $\lim_{t\to\infty} x_\lambda^\tau(t) \to x_0$.*

Proof. Using Theorem 3, the integral form of the Lyapunov function (A.18), and the identity (A.19), we can write

$$\begin{aligned} V(x_\lambda^\tau(t)) &= V(x(0)) + \int_0^t \nabla V \cdot f(x(s), \pi(x(s)))ds \\ &\quad - \int_\tau^{\tau+\lambda} \nabla V \cdot h(x(s))R^{-1}h(x(s))^\top \rho(s)ds \\ &\leq V(x(0)) - \gamma t + \beta\lambda < \beta^\star, \end{aligned}$$

where

$$-\gamma = \sup_{s \in [0,t]} \nabla V \cdot f(x(s), \pi(x(s))) < 0. \tag{6.16}$$

Since λ is fixed and β can be tuned by the matrix weight R, we can choose a t such that $\gamma t \gg \beta\lambda$. Thus, $\lim_{t\to\infty} V(x_\lambda^\tau(t)) \to V(x_0)$ and $\lim_{t\to\infty} x_\lambda^\tau(t) \to x_0$, implies Lyapunov attractiveness, where $V(x_0)$ is the minimum of the Lyapunov function at the equilibrium state x_0. □

The Lyapunov attractiveness property defines a theoretical safe set $\mathcal{C}$ for which the exploration controller can be applied to the robotic system for a specific time t and duration λ such that $\forall x \in \mathcal{C}$, $V(x) \leq \beta^\star$ where $\beta^\star$ is the level set with which $\nabla V \cdot f(x, \pi(x)) < 0$. This is a direct consequence of formulating (6.10) using the hybrid control formulation where the safety properties of the CLF equilibrium policy are inherited without overriding the controls synthesized for ergodic exploration. As a result, the restrictive CLF condition that $\dot{V}(x) < 0 \; \forall x \in \mathcal{B}$ is not enforced and instead is used to filter through exploration-based controllers that will often have $\dot{V}(x) > 0$ so long as $x \in \mathcal{C}$ where it is still possible to enforce the CLF equilibrium policy with $\nabla V \cdot f(x, \pi(x)) < 0$. Note that a similar inheritance-based result can be obtained using closed-form policies from control barrier functions (CBFs) [34, 35] with exploration controllers. CBFs define safety through enforcing invariance of a "safe" set (i.e., a predefined set of known safe states) that prevents a robotic system from leaving the safe set. This would require a knowledge of the safe set and the system dynamics to construct a corresponding closed-form CBF policy for the robotic system. For many robot learning tasks, the safe set and the dynamics are not always known and often only approximate dynamics and a singular safe (equilibrium) state are known. The more restrictive CLF equilibrium policies are used in this thesis as they can be defined for approximate local dynamics and only require knowledge of an equilibrium state[3].

[3] Although one can define a set of states using LaSalle's invariance principle and the local dynamics.

In the following section, we extend this work [109] by further approximating the time averaged-statistics so that computing the adjoint variable can be done more efficiently.

6.2.4 Efficient planning and exploration

We extend our initial work by providing a further approximation to computing the time-averaged statistics which improves the computation time of our implementation. Taking note of (6.2), we can see that we have to evaluate $q(s_i)$ at each sample point s_i where $q(s) = q(s|x(t))$ has to evaluate the stored trajectory at each time. In real robot experiments, often the underlying spatial statistics $p(s)$ change significantly over time. In addition, most robot experiments require replanning which results in lost information over repeated iterations. Thus, rather than trying to compute the whole time averaged trajectory in Definition 3, we opt to approximate the distribution by applying Jensen's inequality to the definition of $\mu(s|\bar{x}(t))$:

$$\begin{aligned} q(s|x(t)) &\propto \int_{t_0}^{t_f} \exp\left[-\frac{1}{2}\|s-\bar{x}(t)\|^2_{\Sigma^{-1}}\right] dt \\ &\geq \exp\left(-\frac{1}{2}\int_{t_0}^{t_f} \|s-\bar{x}(t)\|^2_{\Sigma^{-1}} dt\right). \end{aligned} \tag{6.17}$$

Using this expression in (6.2), we can write

$$\begin{aligned} D_{\mathrm{KL}} &\propto -\int_{\mathcal{S}^v} p(s)\log q(s) ds \\ &= -\int_{\mathcal{S}^v} p(s)\log\left(\exp\left(-\frac{1}{2}\int_{t_0}^{t_f} \|s-\bar{x}(t)\|^2_{\Sigma^{-1}} dt\right)\right) ds \\ &\propto \int_{\mathcal{S}^v} p(s)\left(\int_{t_0}^{t_f} \|s-\bar{x}(t)\|^2_{\Sigma^{-1}} dt\right) ds \\ &\approx \sum_i^N p(s_i)\int_{t_0}^{t_f} \|s_i-\bar{x}(t)\|^2_{\Sigma^{-1}} dt. \end{aligned} \tag{6.18}$$

Following the results to compute (6.7), we can show that (6.10) remains the same where the only modification is in the adjoint differential equation where

$$\dot{\rho}(t) = -\sum_i p(s_i)\frac{\partial \ell}{\partial x} - \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right)^\top \rho(t) \tag{6.19}$$

such that $\ell = \ell(s, x) = \|s - x\|^2_{\Sigma^{-1}}$. This formulation has no need to compute $q(s)$ and instead only evaluate $p(s)$ at sampled points. We reserve using this implementation for robotic systems that are high dimensional or when calculating the derivatives can be costly due to high-parameterization (i.e., multi-layer network models). Note that all the theoretical analysis still holds as the fundamental theory relies on the construction through hybrid systems theory rather than the KL-divergence itself. The downside to this approach is that one now loses the ability to generate finer ergodic movements. The effect can be seen in Figure 6-2(e) where the trajectory is approximated by a wide Gaussian—rather than the bi-model distribution found in Fig. 6-2(d). However, for non-stationary $p(s)$, having exact ergodic behavior is not necessary and such approximations at the time-averaged distribution level are sufficient.

In the following section, we provide a base algorithm and implementation details for KL-E^3 and present variations based on active learning goals one can often find in many robotic scenarios.

6.2.5 Algorithm implementation

In this section, we provide an outline of a base implementation of KL-E^3 in Algorithm 6. We also define some variables which were not previously mentioned in the derivation of KL-E^3 and provide further implementation detail.

There exist many ways one could use (6.10). For instance, it is possible to simulate a dynamical system for some time horizon t_H and apply (6.10) in a trajectory optimization setting. Another way

is to repeatedly generate trajectory plans at each instance and apply the first action in a model-based predictive control (MPC) manner. We found that the choice of τ and λ can determine how one will apply (6.10). That is, given some time t_i, if $\tau = t_i$ and $\lambda = t_H$, we recover the trajectory optimization formulation whereas when $\tau = t_i$ and $\lambda = dt$, where dt is the time step, then the MPC formulation is recovered.[4] Rather than focusing on incremental variations, we focus on the general structure of the underlying algorithm when combined with a learning task.

We first assume that we have an approximate transition model $f(x, u)$ and an equilibrium policy $\mu(x)$. A simulation time horizon t_H and a time step dt is specified where the true robot measurements of state are given by $\hat{x}(t)$ and the simulated states are $x(t)$. Last, a spatial distribution $p(s)$ is initialized (usually uniform to start), and an empty data set $\mathcal{D} = \{\hat{x}(t_j), y(t_j)\}_j$ is initialized where $y(t)$ are measurements.

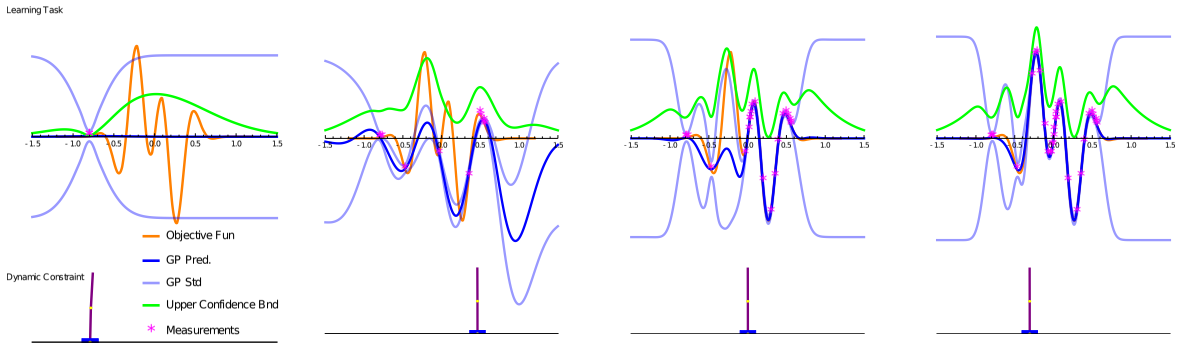
Constructing $p(s)$ will vary depending on the learning task itself. The only criteria that is necessary for $p(s)$ is that it depends on the measurement data that is collected and used in the learning task. Furthermore, $p(s)$ should represent a utility function that indicates where informative measurements are in the search space to improve the learning task. In the following sections, we provide examples for constructing and updating $p(s)$ given various learning tasks.

Provided the initial items, the algorithm first samples the robot's state $\hat{x}(t_i)$ at the current time t_i. Using the transition model and policy, the next states $x(t) \forall t \in [t_i, t_i + t_H]$ are simulated. A set of N, samples $(s_1, s_2, \ldots, s_N)$ are generated and used to compute $p(s), q(s)$. The adjoint variable is then backwards simulated from $t = t_i + t_H \rightarrow t_i$ and is used to compute $\delta\mu_\star(t)$. We ensure robot safety by applying $\delta\mu_\star(t) + \mu(\hat{x}(t))$ with real measurements of the robot's state. Data is then collected and appended to $\mathcal{D}$ and used to update $p(s)$. Any additional steps are specified by the learning task. The pseudo-code for this description is provided in Algorithm 6 .

[4] One can also automate choosing τ and λ using a line search [110].

Algorithm 6 KL-E^3 Base Algorithm

init: approximate transition model $f(x, u)$, initial true state $\hat{x}(0)$, equilibrium policy $\pi(x)$, spatial information distribution $p(s)$, simulation time horizon t_H, time step dt. data set $\mathcal{D}$, $i = 0$
while task not done **do**
  set $x(t_i) = \hat{x}(t_i)$
  ▷ simulation loop
  for $\tau_i \in [t_i, \ldots, t_i + t_H]$ **do**
    ▷ forward predict states using any integration method (Euler shown)
    $x(\tau_{i+1}) = x(\tau_i) + f(x(\tau_i), \pi(x(\tau_i)))dt$
  end for
  ▷ backwards integrate choosing $\dot{\rho}(t)$ set the terminal condition
  generate N samples of s_i uniformly within $\mathcal{S}^v$
  $\rho(t_i + t_H) = \mathbf{0}$
  for $\tau_i \in [t_H + t_i, \ldots, t_i]$ **do**
    $\rho(\tau_{i-1}) = \rho(\tau_i) - \dot{\rho}(\tau_i)dt$
    ▷ since $x(t)$ is simulated, we return
    ▷ just the first term of (6.10)
    ▷ and calculate $\pi(x)$ online
    $\delta u^\star(\tau_{i-1}) = -R^{-1}h(x(\tau_{i-1}))^\top \rho(\tau_{i-1})$
  end for
  ▷ apply to real robot
  chose $\tau \in [t_i, t_i + t_H]$ and $\lambda \leq t_H$ or use line search [110]
  for $t \in [t_i, t_{i+1}]$ **do**
    if $t \in [\tau, \tau + \lambda]$ **then**
      apply $u^\star(t) = \delta u^\star(t) + \pi(\hat{x}(t))$
    else
      apply $\pi(x(t))$
    end if
    if time to sample **then**
      measure true state $\hat{x}(t)$ and measurements $y(t)$
      append to data set $\mathcal{D} \leftarrow \{\hat{x}(t), y(t)\}$
    end if
  end for
  update $p(s)$ given $\mathcal{D}$ ▷ task specific
  update $f(x, u), \pi(x)$ ▷ if needed
  update learning task
  $i \leftarrow i + 1$
end while



(a) $t = 0$ **(b)** $t = 10$ **(c)** $t = 20$ **(d)** $t = 30$

Figure 6-3: Time series snap-shots of cart double pendulum actively sampling and estimating the objective function (orange). The uncertainty (light blue) calculated from the collected data set drives the exploratory motion of the cart double pendulum while our method ensures that the cart double pendulum is maintained in its upright equilibrium state. For videos of this example visit `https://sites.google.com/view/kle3`

6.3 Example active learning tasks

In this section, our goal is to use KL-E^3 for improving example methods for learning. In particular, we seek to show that our method can improve Bayesian optimization, transition model learning (also known as dynamics model learning or system identification), and off-policy robot skill learning. In each subsection, we provide an overview of the learning goal and define the spatial distribution $p(s)$ used in our method. In addition, we show the following:

(i) that our method is capable of improving the learning process through exploration
(ii) that our method does not violate equilibrium policies and destabilize the robot
(iii) and that our method efficiently explores through exploiting the dynamics of a robot and the underlying spatial distribution.

For each example, we provide implementation, including parameters used, in the appendix.

6.3.1 Bayesian optimization with dynamic constraints

In this first example, KL-E^3 is explored for Bayesian optimization using a cart double pendulum system [111] that needs to maintain itself at the upright equilibrium. As discussed in Chapter 2, Bayesian optimization is a probabilistic approach for optimizing objective functions $\phi(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ that are either expensive to evaluate or are highly nonlinear. Pseudo-code for Bayesian optimization is provided in Alg 1.

In many examples of Bayesian optimization, the assumption is that the learning algorithm can freely sample anywhere in the sample space $x \in \mathbb{R}^n$; however, this is not always true. Consider an example where a robot must collect a sample from a Bayesian optimization step where the search space of this sample intersects the state-space of the robot itself. The robot is constrained by

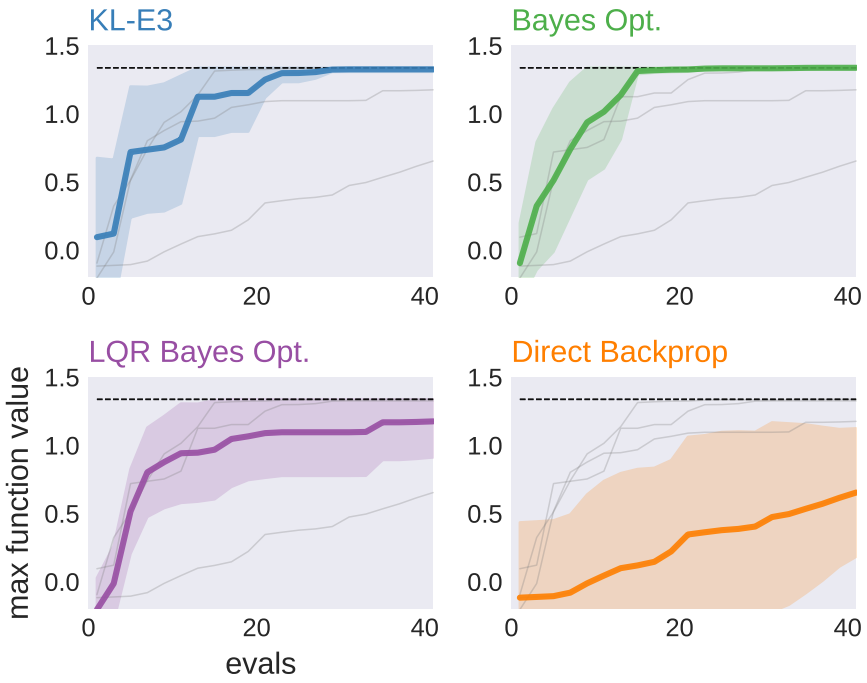


Figure 6-4: Comparison of KL-E^3 against Bayesian Optimization without dynamic constraint, LQR-Bayesian optimization, and direct maximization of the acquisition function through gradient propagation of the cart double pendulum approximate dynamics in determining the maximum value of the objective function through exploration. Our method is able to perform as well as Bayesian optimization directly sampling the exploration space while performing better than the naive LQR-Bayesian optimization. Dashed black line indicates the maximum value of the function.

its dynamics in terms of how it can sample the objective. Thus, the Bayesian optimization step becomes a constrained optimization problem where the goal is to reach the optimal value of the acquisition function subject to the dynamic constraints of the robot. Furthermore, assume that the motion of the robot is restricted to maintain the robot at an equilibrium (such as maintaining the inverted equilibrium of the cart double pendulum). The problem statement is then to enable a robot to execute a sample step of Bayesian optimization by taking into account the constraints of the robot. We use this example to emphasize the effectiveness of our method for exploiting the local dynamic information using a cart double pendulum where the equilibrium state is at the upright inverted state and a policy maintains the double pendulum upright.

Here, we use a Gaussian process with the radial basis function (RBF) to build a model of the objective function shown in Fig. 6-3. Using Gaussian process predictive posterior mean $\bar{\mu}(x)$ and variance $\sigma(x)$, the upper confidence bound (UCB) [29] acquisition function is defined as

$$\text{UCB}(x) = \bar{\mu}(x) + \kappa\sigma(x) \tag{6.20}$$

where $\kappa > 0$. We augment Alg 6 for Bayesian optimization by setting the UCB acquisition function as the target distribution which we define through the Boltzmann softmax function, a common method of converting functions that indicate regions of high-value into distributions [112, 113]: [5]

$$p(s) = \frac{\exp(c\text{UCB}(s))}{\int_{\mathcal{S}^v} \exp(c\text{UCB}(\bar{s}))d\bar{s}} \tag{6.21}$$

where $c > 0$ is a scaling constant. Note that the denominator is approximated as a sum over the samples that our method generates. An approximate linear model of the cart double pendulum

[5] Other distribution forms are possible, but the analysis of their effects is left for future work and we choose the Boltzmann softmax formulation for consistency throughout each example.

dynamics centered around the unstable equilibrium is used along with an LQR policy that maintains the double pendulum upright. We provide brief pseudo-code of the base Algorithm 6 in the Appendix Alg. 9.

We first illustrate that our method generates ergodic exploration through an execution of our method for Bayesian optimization in Figure 6-3. Here, the time-series evolution of KL-E^3 is shown to sample proportional to the acquisition function. As a result, our method generates samples near each of the peaks of the objective function. Furthermore, we can see that our method is exploiting the dynamics as well as the equilibrium policy, maintaining Lyapunov attractiveness with respect to the inverted equilibrium (we will later discuss numerical results in Fig 6-5).

Next, our method is compared against three variants of Bayesian optimization: the first is Bayesian optimization with no dynamics constraint (i.e., no robot is used); second, a linear quadratic regulator (LQR) variation of Bayesian optimization where the maximum of the acquisition function is used as a target for an LQR controller; and last a direct maximization of the acquisition using the stabilizing equilibrium policy (see [60] for detail) is used. A total of 10 trials for each method are collected with the agent starting at the same location uniformly sampled between -0.8 and 0.8 throughout the sample space. In Fig. 6-4 we show that our method not only performs comparably to Bayesian optimization without dynamic constraints[6], but outperforms both LQR and direct maximization variants of Bayesian optimization. Because LQR-Bayes method does not take into account dynamic coverage, and instead focuses on reaching the next sample point, the dynamics of the robot often do not have sufficient time to stabilize which leads to higher variance of the learning objective. We can see this in Fig. 6-5 where we plot a Lyapunov function for the cart double pendulum [111] at the upright unstable equilibrium. Specifically, our method is roughly

[6]This may change if the dynamics of the robot are slower or the exploration space is sufficiently larger. Note that the other methods would also be equally affected.

6 times more stable at the start of the exploration compared to the LQR variant of Bayesian optimization. Lyapunov attractiveness is further illustrated in Fig. 6-5 as time progresses and each exploratory motion is closer to the equilibrium. Last, directly optimizing the highly nonlinear acquisition function often leads to local optima, yielding poor performance in the learning goal. This can be seen with the performance of directly optimizing UCB using the cart double pendulum approximate dynamics in Fig. 6-4) where the cart double pendulum would often find and settle at a local optima.

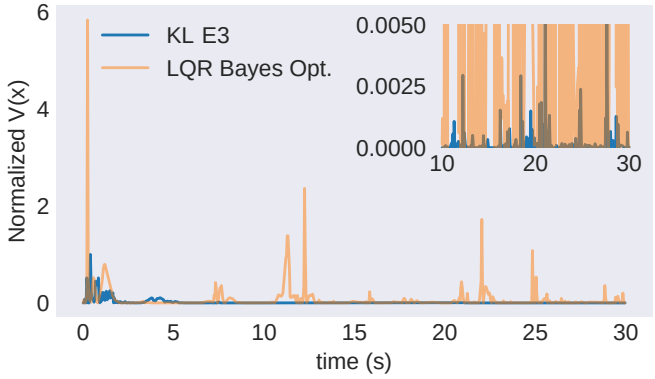


Figure 6-5: Normalized Lyapunov function for the cart double pendulum with upright equilibrium. Our method (shown in blue) is roughly 6 times more stable than LQR-Bayesian optimization. The large initial values indicate the sweeping shown in Fig. 6-3(a) when the cart double pendulum moves across the search space. Subsequent application of the exploratory motion refine the exploration process. The Lyapunov attractiveness property is enforced through automatic switching of the exploration process.

In this example, the cart double pendulum only needed to explore the cart position domain to find the maximum of the objective function. The following example illustrates a more dynamic learning goal where the robot needs to generate a stochastic model of its own dynamics through exploration within the state-space.

6.3.2 Stochastic transition model learning mid-flight

In this next example KL-E^3 is used to collect data for learning a stochastic transition model of a quadcopter [114] dynamical system by exploring the state-space of the quadcopter while remaining at a stable hover. Our goal is to show that our method can efficiently and effectively explore the state-space of the quadcopter (including body linear and angular velocities) in order to generate data for learning a transition model of the quadcopter for model-based control. In addition, we show that the exploratory motions improve the quality of data generated for learning while exploiting and respecting the stable hover equilibrium in a single execution of the robotic system [60].

An LQR policy is used to keep the vehicle hovering while a local linear model (centered around the hover) is used for planning exploratory motions. The stochastic model of the quadcopter is of the form [115]

$$dx \sim \mathcal{N}(f_\theta(x,u), \sigma_\theta(x)) \tag{6.22}$$

where $\mathcal{N}$ is a normal distribution with mean f_θ, and variance $\sigma_\theta(x)$, and the change in the state is given by $dx \in \mathbb{R}^n$. Here, $f(x,u;\theta) = f_\theta(x,u) : \mathbb{R}^{n\times m} \to \mathbb{R}^n$ specifies a neural-network model of the dynamics and $\sigma(x;\theta) = \sigma_\theta(x)$ is a diagonal Gaussian $\sigma_\theta(x) : \mathbb{R}^n \to \mathbb{R}^n$ which defines the uncertainty of the transition model at state x all parameterized by the parameters θ.

We use KL-E^3 to enable the quadcopter to explore with respect to the variance of the model (that is, exploration in the state-space is generated based on how uncertain the transition model is at that state). In a similar manner as done in the previous subsection, we use a Boltzmann softmax function to create the distribution

$$p(s) = \frac{\exp(c\sigma_\theta(s))}{\int_{\mathcal{S}^v} \exp(c\sigma_\theta(\bar{s}))d\bar{s}}. \tag{6.23}$$

Method	Average Power Loss	Average $\|u\|$
KL-E^3	0.16 +- 0.0130	0.68 +- 0.0043
Inf. Max*	0.59 +- 0.0463	1.32 +- 0.3075
Normal 0.1*	1.41 +- 0.0121	0.72 +- 0.0016
OU 0.3	2.73 +- 0.0228	1.17 +- 0.0152
OU 0.1	0.97 +- 0.0096	0.73 +- 0.0033
OU 0.01*	0.10 +- 0.0007	0.67 +- 0.0002
Uniform 0.1*	0.84 +- 0.0090	0.69 +- 0.0004

Table 6.1: Comparison of our method against various methods for state-space exploration using a quadcopter. Each method uses the same base stabilization policy which maintains hover height and is instantiated and run once for 1200 time steps. Data from a single execution is used to generate a neural network dynamics model. This is repeated 20 times to estimate the performance of each method. Methods with (*) were unable to generate a dynamics model that completed the tracking objective.

A more complex target distribution can be built (see [26,60]), however; due to the high-parameterization of the neural-network model, using such methods would require significant computation.

The stochastic model is optimized by maximizing the log likelihood of the model using the likelihood function

$$\mathcal{L} = \mathcal{N}(dx \mid f_\theta(x,u), \sigma_\theta(x)) \tag{6.24}$$

where updates to the parameters θ are defined through the gradient of the log likelihood function:

$$\theta \leftarrow \theta + \alpha \sum_k \nabla_\theta \log \mathcal{L}. \tag{6.25}$$

Here, a batch of K measurements $\{\hat{x}_k, d\hat{x}_k, u_k\}_{k=1}^K$ are uniformly sampled from the data buffer $\mathcal{D}$ where the subscript k denotes the k^{th} time. A variation of Alg. 6 for model learning is provided in the Appendix in Algorithm 10.

We compare our method against time-correlated Ornstein-Uhlenbeck (OU) noise [116], uniform and normally distributed random noise at different noise levels, and using a nonlinear dynamics

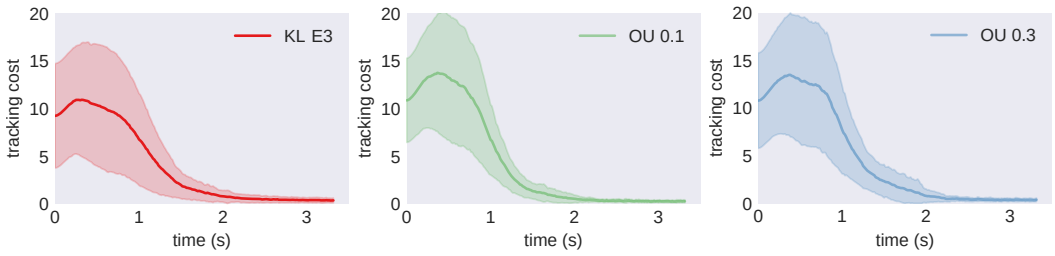


Figure 6-6: Learned quadcopter model evaluations on a model-based tracking objective. Our method is able to generate a model that performs comparably to OU noise at 0.1 and 0.3 noise levels while using less energy through dynamic exploration.

variant of the information maximizing method in [26, 60] which directly maximizes the variance of the model subject to the equilibrium policy. Each simulation is run using the LQR controller as a equilibrium policy for a single execution of the robot (no episodic resets) for 1200 time steps. During this time, data is collected and stored in the buffer $\mathcal{D}$. Our method and the information maximizing method use the data in the stored buffer to update the variance $\sigma_\theta(x)$, guiding the exploration process. However, for evaluation of the transition model, we separately learn a model using the data that has been collected as a gauge for the utility of the collected data for each method. A stochastic model is learned by sampling a batch of 200 measurements offline from the buffer using 2000 gradient iterations. The model is evaluated for target tracking using stochastic model-based control [73] over a set of uniformly randomly generating target locations uniform$(-2, 2) \in \mathbb{R}^3$.

We first illustrate that our method is more energetically efficient compared to other methods in Table 6.1. Here, energy is calculated using the resulting thrust of the quadcopter and we show the average commanded action u over the execution of the quadrotor in time. Our method is shown to be more energetically efficient (due to the direct exploitation of the equilibrium policy and the ergodic exploration process). Furthermore, our method is able to generate measurements in the

state-space that can learn a descriptive stochastic model of the dynamics for model-based tracking control (methods that could not learn a model for tracking control are indicated with a [*]). The resulting methods that could generate a model were comparable to our method (see Fig. 6-6), however; our method is able to directly target the regions of uncertainty (see Fig. 6-7) through dynamic exploration allowing the quadcopter to use less energy (and more directed exploratory actions).

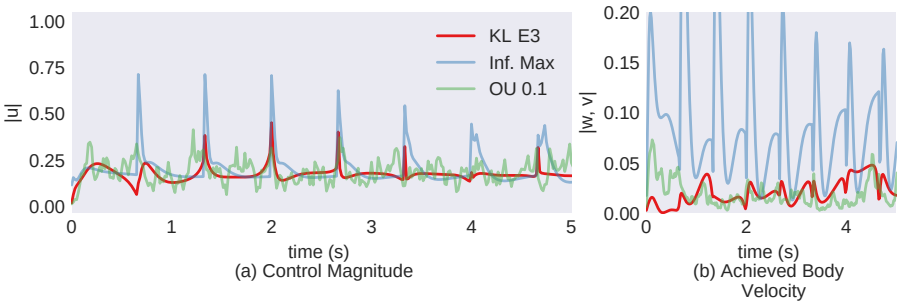


Figure 6-7: Control signal $\|u(t)\|$ and resulting body linear and angular velocities ω, v for the quadcopter system using KL-E^3, information maximization, and OU noise with 0.1 maximum noise level. KL-E^3 generates smoother control signals while exploring the necessary regions of the state-space without destabilizing the system. For videos of this example visit `https://sites.google.com/view/kle3` .

The following example illustrates how KL-E^3 can be used to aide exploration for off-policy robot skill learning by viewing the learned skill as an equilibrium policy.

6.3.3 Learning motor skills

In this example, I explore KL-E^3 for improving robot skill learning (here we consider off-policy reinforcement learning). For all examples, we assume that the learned policy is the equilibrium policy and is simultaneously learned and utilized for safe exploration. As a result, we cannot confirm Lyaponov attractiveness, but assume that the learned policy will eventually yield the Lyaponov

property. Thus, the goal is to show that we can consider a robot skill as being at equilibrium (using a feedback policy) where our method can explore within the vicinity of the robot skill in an intentional manner, improving the learning process.

In many examples of robot skill learning, a common mode of failure is that the resulting learned skill is highly dependent on the quality of the distribution of data generated that is used for learning. Typically, these methods use the current iteration of the learned skill (which is often referred to as a policy) with added noise (or have a stochastic policy) to explore the action space. Often the added noise is insufficient towards gaining informative experience which improves the quality of the policy. Here, we show that our method can improve robot skill learning by generating dynamic coverage and exploration around the learned skill, reducing the likelihood of suboptimal solutions, and improving the efficiency of these methods.

Deep deterministic policy gradient (DDPG) [78] is used as the choice of off-policy method. Given a set of data, DDPG calculates a Q-function defined as

$$Q(x,u) = \mathbb{E}\left[r(x,u) + \gamma Q(x', \pi(x'))\right] \tag{6.26}$$

where $r(x,u) : \mathbb{R}^{n\times m} \to \mathbb{R}$ is a reward function, x' is the next state subject to the control u, the expectation $\mathbb{E}$ is taken with respect to the states, $0 > \gamma > 1$ is known as a discounting factor [113], and the function $Q(x,u) : \mathbb{R}^{n\times m} \to \mathbb{R}$ maps the utility of a state and how the action at that state will perform in the next state given the policy $\pi(x)$. DDPG simultaneously learns $Q(s)$ and a policy $\pi(x)$ by sampling from a set of collected states, actions, rewards, and their resulting next state. We refer the reader to the pseudo-code of DDPG in [78].

Our method uses the learned $Q(x,u)$ and $\pi(x)$ as the target distribution and the equilibrium policy respectively. We modify the Q function such that it becomes a distribution using a Boltzmann

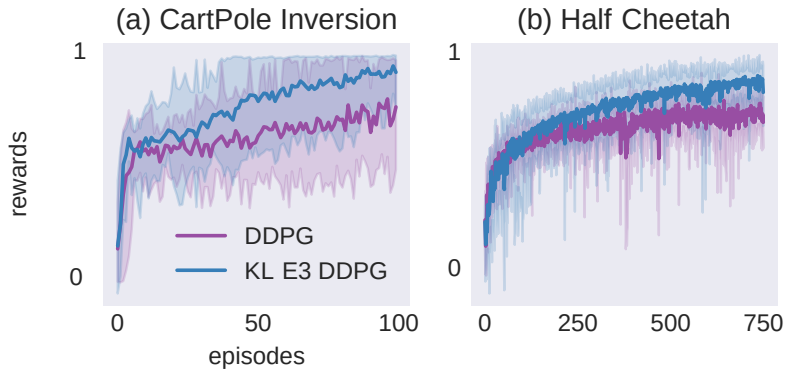


Figure 6-8: Comparison of KL-E^3 enhanced DDPG against DDPG using the cart pole inversion and the half cheetah running tasks. KL-E^3 provides a more informative distribution of data which assists the learning process, improves the overall performance, and achieves better performance faster for DDPG. For videos of this example visit `https://sites.google.com/view/kle3`.

softmax

$$p(s) = \frac{\exp(cQ(s))}{\int_{\mathcal{S}^v} \exp(cQ(\bar{s}))d\bar{s}} \tag{6.27}$$

where $s \in \mathbb{R}^{n+m}$ includes both states and actions. This form of Equation (6.27) has been used previously for inverse reinforcement learning [117, 118]. Here, Eq. (6.27) is used as a guide for the ergodic exploration where our exploration is centered around the learned policy and the utility of the learned skill (Q-function). Since most reinforcement learning deals with large state-spaces, we use the approximation to the time-averaged statistics in (6.17) to improve the computational efficiency of our algorithm. A parameterized dynamics model is built using the first 200 points of each simulation (see Appendix for more detail) and updated as each trial continues. OU noise is used for exploration in the DDPG comparison with the same parameters shown in [78]. We provide a pseudo-code of a KL-E^3 enhanced DDPG in the Appendix Alg. 11.

KL-E^3 is tested on the cart pole inversion and the half-cheetah running task (see Figure 6-8 for results). For both robotic systems, KL-E^3 is shown to improve the overall learning process, making

learning a complex robot skill more sample efficient. Specifically, inverting the cart pole starts to occur within 50 episodes and the half cheetah begins generating running gaits within 250 episodes of the half cheetah (each episode consists of 200 time steps of each environment). In contrast, DDPG alone generates suboptimal running (as shown in (`https://sites.google.com/view/kle3/home`)) and unstable cart inversion attempts. Because our method is able to explore within the vicinity of the learned skill in an intentional, active ergodic manner, it is able to quickly learn skills and improve the overall quality of the exploration.

6.3.4 Learning backflips through interactive imitation learning

In this last example, I present KL-E^3 for use in active imitation learning. As presented in Chapters 3 and 4, imitation learning, specifically behavior cloning, is an approach for robot systems to learn tasks from expert demonstrations by imitating a set of N demonstrations $\mathcal{D} = \{\{x_t^n, u_t^n\}_{t=0}^{T-1}\}_{n=1}^{N}$ of length T where x_i, u_i is the state-action pair associated with a task. The goal is then to learn a policy $\pi(x)$ that perfectly mimics the set of demonstrations $\mathcal{D}$ and can generalize to states outside the distribution of demonstrations. The difficultly in this task is that the expert policy $\pi^\star$ is not accessible and the task objective function is also unknown.

In Chapter 4, active imitation learning (or interactive imitation learning) is motivated as an approach that allows the robotic system to interactively learn a task by querying an expert for demonstrations $(x_i^\star, u_i^\star)$ at query states $x^\star$ which can be infeasible states. Through interactive querying, it is possible to acquire more informative demonstrations from an expert by inferring how an expert will behave in states which provide the most information about a task. By collecting these informative (but often infeasible) states, it becomes possible for a robot to actively learn a policy with fewer demonstrations as opposed to passively acquiring demonstrations from an expert.

Here, I investigate the case where the robotic system cannot query any arbitrary state to the expert but instead can on query states it passes through. In addition, I test the capabilities of KL-E^3 for learning highly dynamics tasks solely from queried demonstrations under strict safety requirements.

The test system is a planar lunar-lander system whose task is to learn how to backflip by actively querying an expert backflipper (or master-flipper) using only the state the system is currently visiting. In addition, a restriction of maintaining stable hovering is enforced. The stable hovering is defined by an LQR policy at an equilibrium hover height (see Appendix C for details on the dynamics and policy). The imitated policy is learned through behavior cloning which corresponds to the optimization following problem:

$$\theta^\star = \arg\min_\theta \sum_{\mathcal{D}} \|u_i^n - \pi(x_i^n, \theta)\|^2$$

where $\pi(x, \theta)$ is the θ parameterized deterministic behavior cloned policy. Demonstrations are obtained by visiting a state $x^\star$ and querying the expert for a demonstration of length T. Each demonstration is an example of how to backflip from the current query state $x^\star$ defined as $\{x_t, u_t\}_{t=0}^{T-1}$ (with a full backflip typically lasting $T = 100$ steps). The target distribution in this example is defined using the Fisher information matrix (as described in Chapter 4 and 5) as

$$p(s) = c\text{tr}\left(\mathcal{I}(\theta)\right)$$

where

$$\begin{aligned}\mathcal{I}(\theta) &= \frac{\partial \pi}{\partial \theta}^\top \mathcal{I}(\pi) \frac{\partial \pi}{\partial \theta} \\ &\approx \text{diag}\left((\sigma^{-1})^\top \left(\frac{\partial \pi}{\partial \theta} \right)^2 \right),\end{aligned} \tag{6.28}$$

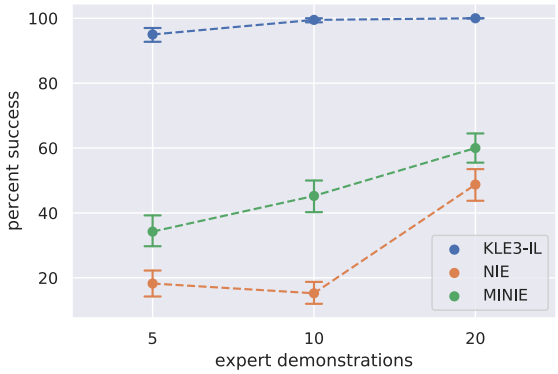


Figure 6-9: Comparison of KL-E^3 for active imitation learning to Noise Injected Exploration (NIE) and Max Information NIE or MINIE for learning lunar-lander backflip task from demonstrations. Error bars indicate 95 % confidence bound over behavior-cloned policy evaluations for 100 normally distributed initial conditions using 4 random seeds. KL-E^3 queries informative expert demonstrations that yield learned policies that successfully clones a backflip behavior.

σ is the known measurement uncertainty from the expert demonstration, and c is a normalization constant. The motivation for using the Fisher information comes from the Cramér-Rao bound and the fact that in imitation learning through behavior cloning, the best a policy can do is minimize the uncertainty of the learned parameters. This is a consequence of the nature of learning tasks from demonstrations and the lack of an external reward or cost.

In order to improve the impact of KL-E^3, rather than querying the expert at every state, the querying occurs when a state is reached which has an information measure defined by $\mathrm{tr}\left(\mathcal{I}(\theta)\right)$ that is greater than the previous queried state. At each planning iteration, the robot state is fed into KL-E^3 in receding horizon which returns a sequence of exploratory actions. In addition, we assume we have a model of the dynamics of the lunar-lander. The behavior cloned policy (and target distribution) is updated at each sampling instance of the robot and a final training loop is run until convergence once a finite set of demonstrations are acquired. Pseudo-code for KL-E^3 for active query-based imitation learning is provided in Appendix C.

KL-E^3 is compared to two methods for active learning (equivalent to the methods presented in Chapter 6.3.2). The first method, Noise Injected Exploration (NIE), uses normally distributed random noise injection with variance at 100% of the control saturation to the equilibrium hover

policy of the lunar-lander. [7] A expert demonstration is obtained at every state the robot visited during the exploration. The second method, Max Information Noise Injected Exploration (MINIE) is similar to the random query-based method with maximum information filtering presented in Chapter 4. The exploration method is the same random noise injection to the equilibrium policy at 100% control saturation with the added information filtering. Only queries that have a information measure greater than the previous queried state are forwarded to the expert demonstrator. Each example is run and tested for a finite set of queried demonstrations. The demonstration set is then used to train the behavior-cloned policy until convergence is reached. The policy is then evaluated on a series of 100 normally distributed random initial states for a set of 4 random seeds (totaling 400 policy evaluations) that tests the ability of the policy to successfully backflip.

Comparative results are presented in Figure 6-9 for active imitation learning after collecting a set of 5, 10, and 20 expert demonstrations using KL-E^3, NIE, and MINIE. Here, KL-E^3 is shown to query expert demonstrations which yield behavior cloned policies that have near 100% success rate over the backflip evaluation samples within 5 active queries. In contrast, both NIE and MINIE methods barely reach the 50 % threshold by 20 queried expert demonstrations. These results are a direct consequence of the noise injected exploration methods fighting the equilibrium policy. KL-E^3 uses the equilibrium policy as part of the planning and control of the robot for active query-based imitation learning. In addition, the use of indirect sampling of the target distribution using the Fisher information matrix avoids issues with local minima that can be problematic with direct active learning optimization methods. These results are further reinforced if we look at the maximum information acquired for each query in Fig. 6-10. We can see that simply choosing informative states to query the expert already improves the quality of the behavior cloning process and increases

[7] The aggressive equilibrium policy prevented any control-injected noise from changing the state of the lunar-lander at any value less than 100% control saturation.

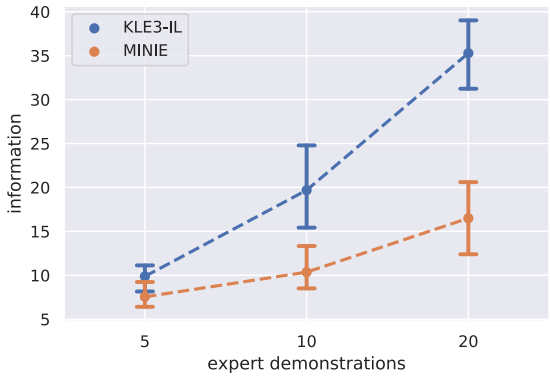


Figure 6-10: Maximum obtained information using KL-E^3 compared to MINIE for active query-based imitation learning for lunar-lander backflip task. KL-E^3 exploits the equilibrium policy and works with the policy to reach more informative query points to obtain an expert demonstration. The information gain is reflected in the percent success rates for MINIE and KL-E^3 in Fig. 6-9.

generalization of the task when comparing NIE to MINIE. However, adding stability constraints that impede sampling highly dynamic states will result in a loss of quality in the resulting behavior cloned policy. Both the ergodic sampling strategy and the equilibrium-based control synthesis developed in KL-E^3 improve these issues that allow robotic systems to actively learn from expert demonstrators in a safe and reliable manner.

6.4 Discussion and summary

I present KL-E^3, a method which is shown to enable safe active learning in robots through the KL–divergence ergodic measure. The proposed method synthesizes ergodic sampling using the KL-divergence measure that generates data through exploiting dynamic movement proportional to the utility of the data. I show that ideas and tools from hybrid control theory can be used to synthesize a schedule of exploratory actions that can incorporate learned policies and models in a systematic manner that improves safety and reliability of robot learning. Last, I present examples that illustrate the effectiveness of KL-E^3 for various robot active learning examples and provide theoretical analysis which bounds stability to equilibrium policies through Lyapunov attractiveness.

The following chapter examines some of the fundamental assumptions made with how we model the dynamics of robotic systems and the ability to measure the state of the robot and the world. In doing so, I present a novel way to predict and plan with learned dynamic models that both improves control authority and results in simpler mathematical structure. I then show how this structure can be used to improve upon active learning and stable learning using the methods presented in the previous chapters. I then present various examples which show the improved capabilities that one can find through just considering structure in the learning problem.

Chapter 7

Active learning in infinite linear embedding space

In most problems of robot learning and control, the robot, the task, and the environment are often represented as set of nonlinear systems of equations generally described through the Markov decision process. If we wanted to expand the scope of the robot dynamics and its interaction in the world, we would need to figure out what needs to be modeled and measured for the robotic system to operate successfully. Often the world can be infinitely complex, and arbitrarily expanding the scope for robot operations can be impossible to do directly. Even if it were possible, the problem of operating autonomously in the world under such a broad scope would generally yield more complex, nonlinear models which are exceedingly difficult for optimal control solvers to exploit.

How then can we increase the capabilities of robotic systems without inheriting all the nonlinearities and complexities that come with such a problem? Is it possible to construct representations that are mathematically simple, but provide robotic systems with the necessary information to successfully operate in the world? In this chapter, I investigate infinite linear embeddings as a

candidate class of models that allow robotic systems to exceed the constraints of non-linearity and model complexities. I illustrate the improved control authority from using such models for control, and I present a method for safe active learning that uses infinite linear embeddings to achieve instant learning of robot dynamics in a single execution of the robotic system.

7.1 Control in infinite linear embedded space

As discussed in Chapter 2, Koopman operators are a class of infinite linear embeddings that project nonlinear dynamic constraints into a linear dynamical system in a modified (often infinite) vector-space. Here, we examine the Koopman operator structure in the case of optimal linear quadratic (LQ) control of robotic systems. The usefulness of Koopman operators comes from the idea that the lifted linear features embed all the non-linear features into a linear representation that we can exploit for use with linear quadratic optimal control.

7.1.1 Linear optimal control > nonlinear optimal control

Let us consider control of the nonlinear forced Van der Pol oscillator, the dynamics of which are defined in Appendix D, as an example. We specify the control task as minimizing the following LQ objective

$$\mathcal{J} = \int_{t_i}^{t_i+T} x(t)^\top \mathbf{Q} x(t) + u(t)^\top \mathbf{R} u(t) dt + x(t_i+T)^\top \mathbf{Q}_f x(t_i+T)$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{R} \in \mathbb{R}^{m\times m}$, and $\mathbf{Q}_f \in \mathbb{R}^{n\times n}$. Choosing a set of function observable (Appendix D), we can compute a Koopman operator $\mathcal{K}$ by repeated simulation of the Van der Pol oscillator given a set of 5000 uniformly sampled state-control measurements.

Since the Van der Pol oscillator dynamics are nonlinear, an optimal control solution to the

LQ control problem is to linearize the dynamics about the equilibrium state $x_t = [0, 0]^\top$ and form a linear quadratic control regulator (LQR) feedback controller. Using the Kooman operator formulation of the Van der Pol dynamics, we can compute a controller in a similar manner using the following objective

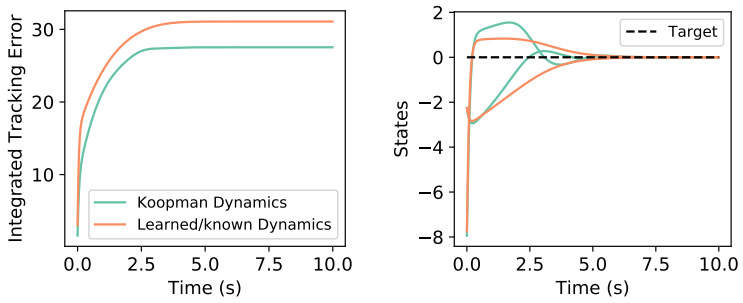
$$\mathcal{J} = \int_{t_i}^{t_i+T} z(t)^\top \tilde{\mathbf{Q}} z(t) + u(t)^\top \mathbf{R} u(t) dt + z(t_i + T)^\top \tilde{\mathbf{Q}}_f z(t_i + T)$$

where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{a \times a} \text{ and } \tilde{\mathbf{Q}}_f = \begin{bmatrix} \mathbf{Q}_f & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{a \times a}.$$

where a is the dimensionality of the subspace of state lifted features $z(x) \in \mathcal{F}^a$. Setting $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{Q}}_f$ to only include the observations of state allows us to compare the same control objective using the linearized dynamics against the Koopman operator dynamics where the first terms in the function observable $z(x(t))$ is the state of the Van der Pol system itself.

Figure 7-1 illustrates the improvement in control performance when using the the Koopman operator dynamics for LQ control instead of linearizing the dynamics around a local region. We compare the control authority using a learned dynamics model in the original state-space using Bayesian optimization with the same functions used for the Koopman operator. Figure 7-1 illustrates that the data used to compute the Koopman operator can learn a nonlinear model of the Van der Pol dynamics in the original state-space. The Koopman operator formulation of the Van der Pol approximates the dynamic constraints as a linear dynamical systems in a higher dimensional space that captures nonlinear dynamical behavior. As a result, the Koopman operator formulation coupled with LQ methods can be used to enhance the control the Van der Pol system as shown in Figure 7-1b. Computing the resulting trajectory error (Figure 7-1a) shows that the trajectory



(a) Int. Tracking Error **(b)** Trajectory

Figure 7-1: Control performance of a forced Van der Pol oscillator with an LQR control using the learned Koopman operator, the linearization of the known system dynamics, and the linearizion of a learned state-space model using the same data and basis functions as the Koopman operator. The control performance using the Koopman operator dynamics is shown to outperform the LQR control with known dynamics. The learned dynamics model performs equally to the known dynamics model and is overlayed on top of the known dynamics results.

taken from the Koopman operator controller results in less overall integrated error. This is due to formulating the LQ controller with additional information in the form of a dynamical system that evolves nonlinear functions of state in time.

While this example illustrates the possible benefits of utilizing the Koopman operator formulation, we ignored how the data was collected for constructing the model the Van der Pol dynamical system. The following sections introduce methods that enable robotic systems to fully utilize linear embeddings in real scenarios through active learning.

7.2 Active learning formulation with linear embeddings

As discussed in prior chapters, active learning controllers need to consider the safety of the robot itself based on the underlying task and manage the non-linearities often created through defining additional active learning objectives. As showed before, control with the Koopman operator models

gives us improved control performance and mathematical simplicity due to its linear structure. In this section, I formulate a controller for active learning that exploits the linear structure of the Koopman operator for learning and control of robotic systems. Using results from Chapters 3, the active learning problem is approached as a hybrid control problem where the goal is to actively learn the system dynamics while successfully solving an underlying stability task. Additional difficulty is introduce through the added constraint of simultaneously learning the system dynamics and using the models to construct stabilization policies online.[1] Rather than converting the active learning objectives using ergodic sampling, the linear structure of the Koopman operator is used to construct simpler learning objectives. In addition, the Koopman operator model is used to assist with control authority and construction of stabilization policies *online* to improve the learning and control of the system.

The active learning problem is formulated as a hybrid switching problem [75] where the goal is to switch between a task-based policy and an information maximizing controller that assists the dynamical system in collecting informative measurements for learning. Consider a general objective function of the form

$$\mathcal{J} = \int_{t_i}^{t_i+T} \ell(z(t), \pi(z(t)))dt + m(z(t_i + T)) \tag{7.1}$$

where $z(t) : \mathbb{R} \to \mathcal{F}^a$ is the value of the linear embedded features at time t subject to the Koopman dynamics

$$\dot{z}(t) = \mathcal{K}_z z(t) + \mathcal{K}_v v(x(t), u(t)) \tag{7.2}$$

with initial condition $z(x(t_i))$, $\ell(z, u) : \mathcal{F}^a \times \mathcal{U}^m \to \mathbb{R}$ is the running cost function, $m(z) : \mathcal{F}^a \to \mathbb{R}$ is the terminal cost, and $\pi(z) : \mathcal{F}^a \to \mathcal{U}^m$ is a continuous and differentiable policy defined in the

[1] During training, the policies derived from the Koopman operator dynamics will be inaccurate; however, over time and with informative measurements, both the model and policy will converge. This is a common approach in most model-based reinforcement learning techniques [119].

lifted feature space. Here, $\mathcal{K}_z \in \mathbb{R}^{a\times a}$, $\mathcal{K}_v \in \mathbb{R}^{a\times b}$ are the sub-components of $\mathcal{K}$ with $v(x,u) : \mathcal{X}^n \times \mathcal{U}^m \to \mathcal{F}^b$. In this work, the running cost is split into two parts:

$$\ell(z,u) = \ell_{\text{learn}}(z,u) + \ell_{\text{task}}(z,u)$$

where ℓ_{learn} is the information maximizing objective (learning task) and ℓ_{task} is the task objective for which the policy $\pi(z)$ is a solution when $\ell_{\text{learn}} = 0$.

Given equation (7.1), we want to synthesize a controller that is bounded to the policy $\pi(z)$, but also allows for improvement of an information measure for active learning. To do so, we follow the same procedure as in Lemma 1 for a switching mode system.

Proposition 3. *The sensitivity of switching from $\pi(z)$ to an arbitrary control $\hat{u}$ for all $\tau \in [t_i, t_i + T]$ for an infinitesimally small λ, is given by*

$$\frac{\partial \mathcal{J}}{\partial \lambda}\Big|_{\tau,\lambda=0} = \rho(\tau)^\top (f_2 - f_1) \tag{7.3}$$

where $f_2 = f(z(\tau), \hat{u}(\tau))$, $f_1 = f(z(\tau), \pi(z(\tau)))$, with f defined in (7.2) and

$$\dot{\rho} = -\left(\frac{\partial \ell}{\partial z} + \frac{\partial \pi}{\partial z}^\top \frac{\partial \ell}{\partial u}\right) - \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial z}\right)^\top \rho$$

subject to the terminal condition $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$.

Proof. See Appendix A. □

As in Chapter 3, we can write an unconstrained optimization problem for calculating the optimal $\hat{u}(\tau)$ over the interval $\tau \in [t_i, t_i + T]$ that will minimize[2] the mode insertion gradient. We can write

[2]It is a minimization since we want to reduce the cost function.

this optimization problem using an auxiliary objective function

$$\mathcal{J}_2 = \int_{t_i}^{t_i+T} \frac{\partial}{\partial \lambda} \mathcal{J}\Big|_{\tau=t,\lambda=0} + \frac{1}{2}\|\hat{u}(t) - \pi(z(t))\|^2_{\tilde{\mathbf{R}}} dt, \tag{7.4}$$

where $\tilde{\mathbf{R}} \in \mathbb{R}^{m\times m}$ bounds the change of $\hat{u}$ to $\pi(z)$, and $\frac{\partial}{\partial \lambda}\mathcal{J}\Big|_{\tau=t,\lambda=0}$ is evaluated at $\tau = t$. Solving equation (7.4) with respect to $\hat{u}(t)$ can be viewed as a functional optimization over $\hat{u}(t) \forall t \in [t_i, t_i + T]$. Since equation (7.4) is quadratic in $\hat{u}$, we obtain a closed form solution for any application time $\tau \in [t_i, t_i + T]$.

Proposition 4. *Assuming that $v(x, u)$ is differentiable, the control solution that minimizes (7.4) is*

$$u^\star(t) = -\tilde{\mathbf{R}}^{-1}\left(\mathcal{K}_v \frac{\partial v}{\partial u}\right)^\top \rho(t) + \pi(z(t)). \tag{7.5}$$

Proof. Since (7.4) is separable in time, we take the derivative of (7.4) with respect to $\hat{u}(t)$ at each point in t which gives the following expression:

$$\begin{aligned}\frac{\partial}{\partial \hat{u}}\mathcal{J}_2 &= \int_{t_i}^{t_i+T} \frac{\partial}{\partial \hat{u}}\left(\rho(t)^\top (f_2 - f_1)\right) + \tilde{\mathbf{R}}\left(\hat{u}(t) - \pi(z(t)\right) dt \\ &= \int_{t_i}^{t_i+T} \left(\mathcal{K}_v \frac{\partial v}{\partial u}\right)^\top \rho(t) + \tilde{\mathbf{R}}(\hat{u}(t) - \pi(z(t)))dt. \end{aligned} \tag{7.6}$$

Solving for $\hat{u}(t)$ in (7.6) gives the optimal closed-form solution

$$u^\star(t) = -\tilde{\mathbf{R}}^{-1}\left(\mathcal{K}_v \frac{\partial v}{\partial u}\right)^\top \rho(t) + \pi(z(t)).$$

□

We can use equation (7.5) with (7.3) to show that our approach improves the active learning

objective subject to bounds placed on arbitrarily defined tasks.

Corollary 3. *Assume that the Koopman operator dynamics for a system are defined by the following control affine structure:*

$$\dot{z} = \mathcal{K}_z z(x(t)) + \mathcal{K}_v v(x(t))u(t) \tag{7.7}$$

where $v(x) : \mathcal{X}^n \to \mathbb{R}^{b\times m}$ *where* $\mathcal{K}_v \in \mathcal{F}^{a\times b}$. [3] *Moreover, assume that* $\frac{\partial}{\partial \pi}\mathcal{H} \neq 0$ *where* $\mathcal{H}$ *is the control Hamiltonian for (7.1). Then*

$$\frac{\partial}{\partial \lambda}\mathcal{J} = -\| \left(\mathcal{K}_v v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} < 0 \tag{7.8}$$

for $u^\star(t) \in \mathcal{U}\ \forall t \in [t_i, t_i + T]$.

Proof. Inserting (7.5) into (7.3) gives

$$\frac{\partial}{\partial \lambda}\mathcal{J} = \rho(t)^\top \left(\mathcal{K}_v v(x(t))\right)\left(-\tilde{\mathbf{R}}^{-1}\left(\mathcal{K}_v v(x(t))\right)^\top \rho(t)\right)$$

which can be written as the norm

$$\frac{\partial}{\partial \lambda}\mathcal{J} = -\| \left(\mathcal{K}_u v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} < 0.$$

□

The following subsection provides a candidate learning objective inspired by the results in Chapter 3 using the Fisher information measure based on the Koopman operator model. We first describe the Fisher information matrix for the linear Koopman operator parameters. We then show that

[3] This formulation assumes that we can recover $x(t)$ from $z(t)$ for computing $v(x)$.

using (7.5) and Corollary 3, that we can approximately calculate to first order the gain in information.

7.2.1 Fisher information maximization of linear embeddings

Using the controller defined in (7.5), we investigate the Fisher information maximization to actively learn the Koopman operator dynamics for robotic systems. We can learn the approximate Koopman operator dynamics using the maximum likelihood objective

$$\mathcal{K}^\star = \arg\max_{\mathcal{K}} \sum_{i=1}^{N} \log p(z(x_{i+1}, u_{i+1}) \mid z(x_i, u_i), \mathcal{K})$$

where linear structure of the Koopman operator makes $p(z(x_{i+1}, u_{i+1}) \mid z(x_i, u_i), \mathcal{K}) = \mathcal{N}(\mathcal{K}z(x_t, u_t), \Sigma)$ where $\Sigma \in \mathbb{R}^{a \times a}$ is the variance. Thus, the Fisher information matrix over the parameters that compose of the Koopman operator $\mathcal{K}$ is computed as

$$\mathcal{I}(\mathcal{K}) = \frac{\partial f}{\partial \kappa}^\top \Sigma^{-1} \frac{\partial f}{\partial \kappa} \in \mathbb{R}^{|\kappa| \times |\kappa|} \tag{7.9}$$

where $\kappa = \{\mathcal{K}_{i,j} \mid \mathcal{K}_{i,j} \in \mathcal{K}\}$, $|\kappa|$ is the cardinality of the vector κ, and f is defined by the Koopman continuous dynamics (7.2). Here, we assume that the model is held fixed in order to avoid computing the parameter dynamics illustrated in Chapter 4. This allows us to compute reactive active learning strategies which can be executed on a robot at high frequencies without worrying about controller lag time.

Because the Fisher information defined here can be positive semi-definite, we use the T-optimality condition (different from the A-optimality) of the Fisher information matrix [120] as the learning

objective defined as

$$\ell_{\text{learn}}(z, u) = (\text{tr}\mathcal{I}(\mathcal{K}) + \epsilon)^{-1} \tag{7.10}$$

where $\epsilon \ll 1$ is a small number to prevent singular solutions due to the positive semi-definite Fisher information matrix [121–123]. Here, $\mathcal{I}(\mathcal{K})$ is computed using the evaluation of $\mathcal{K}$ at time t_i which is equivalent to greedily optimizing with respect to a single step of the parameter dynamics.

Assumption 1. *Assume that $\mathcal{I}(\tilde{\mathcal{K}}) > 0$ implies $\mathcal{I}(\mathcal{K}) > 0$ where $\tilde{\mathcal{K}}$ is an approximation to the Koopman operator $\mathcal{K}$ computed from the data set $\mathcal{D} = \{x(t_m), u(t_m)\}_{m=0}^{i}$ that contains data up until the current sampling time t_i.*

Theorem 5. *Given Assumption 1 and dynamics (7.7), then the change in information* [4] $\Delta\mathbf{I}$ *subject to (7.5) is given to first order*

$$\Delta\mathbf{I} \approx (\|(\mathcal{K}_v v(x))^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} + \ell_{task}(z, u^\star) - \ell_{task}(z, \pi))\mathfrak{I}_{u^\star}\mathfrak{I}_\pi + \mathcal{O}(\Delta t), \tag{7.11}$$

where $\mathfrak{I}_u = tr\mathcal{I}(\mathcal{K})_u + \epsilon$ is the T-optimality measure (7.10) from applying an arbitrary control u.

Proof. See Appendix A. □

Theorem 5 shows that our active learning controller with the Koopman operator dynamics increases the rate of information that a robot would have normally acquired if it had only used the control policy $\pi(z)$. Thus, through the Cramér-Rao bound, this implies that the bound

$$\text{var}\left[\mathcal{K}^\star\right] \geq \mathcal{I}\left(\mathcal{K}\right)^{-1}$$

is continuously improving through the information maximizing actions. Weighing the information

[4] With respect to the information acquired from applying only $\pi(z)$.

measure against the task objective allows us to ensure that the relative information gain is positive when using the active learning controller. That is, the difference between the information from using the policy $\pi(z)$ and the control $u^\star(t)$ will be positive. Other heuristics can be used such as a decaying weight on the information gain or setting the weight to 0 at a specific time so that the robot attempts the task. We provide a basic overview of the control procedure in Algorithm 7. Videos of the experiments and example code can be found at `https://sites.google.com/view/active-learning-koopman-op`.

Algorithm 7 Online Active Learning via Infinite Linear Embedded Information Maximization

1: **initialize:** objective $\ell(z, u)$, initial policy $\pi(z)$, normally distributed random $\mathcal{K} \sim \mathcal{N}(0, \mathbf{1})$.
2: sample state measurement $x(t_i)$
3: add $x(t_i)$ to dataset $\mathcal{D}$, update $\mathcal{K}$ and $\pi(z)$
4: simulate $z(t), \rho(t)$ for $t \in [t_i, t_i + T]$ with conditions $z(t_i) = z(x(t_i))$ and $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$ with $\pi(z)$
5: compute $u^\star(t) = -\tilde{\mathbf{R}}^{-1} \left(\mathcal{K}_v \frac{\partial v}{\partial u}\right)^\top \rho(t) + \pi(z(t))$
6: **return** $u^\star(t_i)$
7: update timer $t_i \to t_{i+1}$

The following sections use our derived controller to enable active-learning of Koopman operator dynamics.

7.3 Single rollout active learning of free-falling quadcopters

In this example, we illustrate the capabilities of combining the Koopman operator representation of a dynamical systems and active learning for single execution model learning of a free-falling quadcopter for stabilization. Additionally, we compare our approach to other common learning strategies such as active learning with Gaussian processes [25, 124, 125], online model adaptation through direct attempts at the tasks of stabilization (common online reinforcement learning and adaptive control approach [126–130]), and a two-stage noisy motor input (often referred to as

"motor babble" [131–133]).

7.3.1 Problem definition

The task is as follows: The quadcopter, with dynamics described in Appendix D and [134], must learn a model within the first second of free-falling and then use the model to generate a stabilizing controller, preventing itself from falling any further. We define success of the quadcopter in the task when $\|x - x_d\|^2 < 0.01$ where x_d is the desired target state defined by zero linear and angular velocity. The controllers are designed as linear quadratic regulators using the model that was learned the LQ formulation. The parameters used for this example are defined in Appendix D.

We compare the information gained (based on the T-optimality condition) and the stabilization error in time against various learning strategies. Each learning strategy is tested with the same 20 uniformly sampled initial velocities (and angular velocities) between -2 and 2 radians/meters per second. After each trial, the learned dynamics model is reset so that no information from the previous trials are used.

7.3.2 Comparison with other active learning strategies

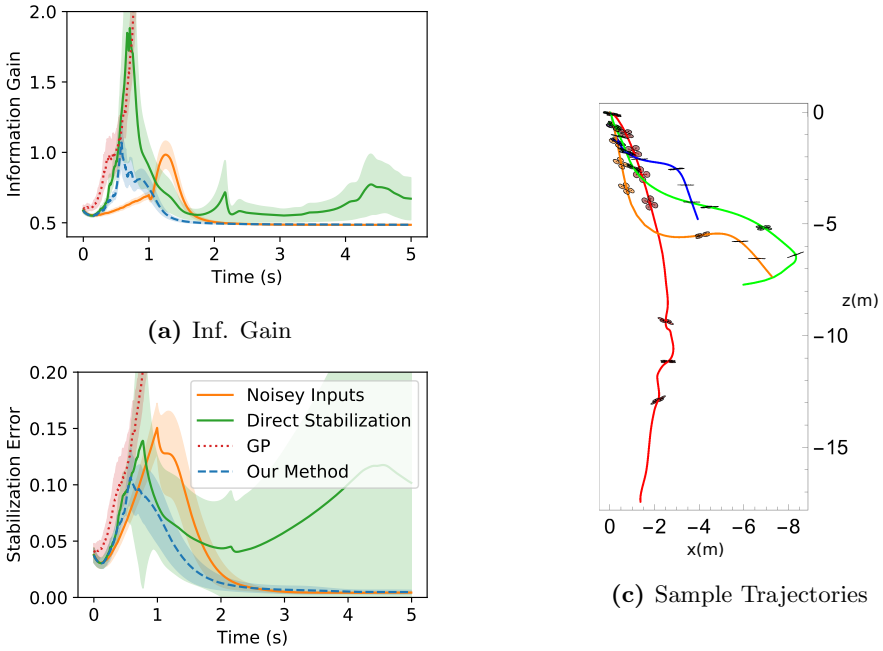
We compare our method for active learning against common dynamic model learning strategies. Specifically, we compare three model learning approaches against our method, a two-stage noisy control input approach [131], a direct stabilization with adaptive model using least squares [126, 127], and an active learning strategy using a Gaussian process [135, 136]. Each of these strategies are generating a Koopman operator using the functions of state defined in Appendix D to generate a dynamic model of the quadcopter. The Gaussian process formulation is the only model where the functions map to the original state-space resulting in a nonlinear dynamics model.

Least Squares Adaptive Stabilization The first strategy we compare to is to do the task of stabilization at the while updating the model of the dynamics recursively [126, 127]. This is often a strategy used in model-based reinforcement learning [132] and adaptive control [126].

Two-Stage Motor Babble The second strategy is a two stage approach using noisy motor input (motor babble) for the first second and then pure stabilization [131]. Rather than directly attempting to stabilize the dynamics, the priority is to simply try all possible motor inputs regardless of the model of the dynamics that is being constructed. The motor babble strategy allows us to bound the motor excitation which prevents the rotor from destabilizing once the learning stage is complete. As with the direct stabilization method, we use a recursive least squares to update the model of the Koopman operator.

Active Learning with Gaussian Process The last strategy is an active Gaussian process strategy [135, 136]. In this active learning strategy, we build a model of the dynamics of the quadcopter by generating a Gaussian process dynamics model [125, 135]. Using the variance estimate [136], we uniformly sample points around the current state bounded by some ϵ constant and find the state which maximizes the variance. The sampled state with the largest variance is then used to generate a local LQ controller to guide the quadcopter dynamics to that state to collect the data. After the first second, the Gaussian process model is used to generate a stabilizing controller by linearizing the model about the final desired stabilization state. The kernel function used is computed using the functions of state provided in Appendix D for a fair comparison. Note that for the two-stage, least squared adaptive, and our approach, we learn a Koopman operator dynamics model which we use to compute an LQ controller. The Gaussian process model is in in the original state-space as described in [125].

Figure 7-2 (a) illustrates the information (T-optimality of the Fisher information matrix) for each method. Our approach to active learning is shown to improve upon the information when com-



(a) Inf. Gain

(b) Stabilization Error

(c) Sample Trajectories

Figure 7-2: Monte-Carlo simulation comparing various learning strategies to stabilize a quadcopter falling for 20 trials with uniformly sampled initial linear and angular velocities. (a) Information gain (trace of the Fisher information matrix) is shown for the various learning strategies. (b) Stabilization error and standard deviation is shown over time for each learning strategy over 20 trajectories. (c) Representative time series snapshots are shown depicting the various learning strategies. With our approach, maximization of the information measure, coupled with the Koopman operator formulation of the dynamics, enables quick stabilization of the quadcopter. For videos of this example visit `https://sites.google.com/view/active-learning-koopman-op`

pared to motor babble (the most basic method for active learning). The other methods outperform our approach in terms of the overall information gain by overly exciting the dynamics. The direct adaptive stabilization method utilizes the incorrect dynamics model to self-adjust and eventually stabilize the quadcopter (as shown in the variance). The active Gaussian process approach uses the covariance estimate to actuate the quadcopter towards uncertain regions. Collecting data in uncertain regions allows the active Gaussian process approach to actively select where the quadcopter

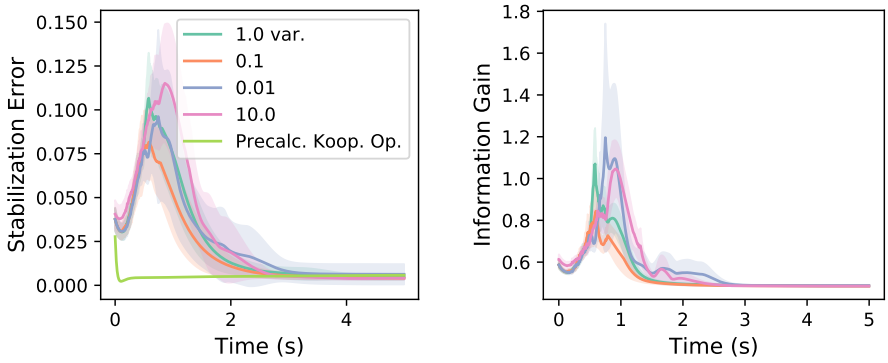
should collect data next.

It is worth noting that these approaches will often lead the quadcopter towards unstable regions, making it difficult to stabilize the dynamics in time. Our approach actively synthesizes when it is best to explore and stabilize which assists in quickly stabilizing the quadcopter dynamics (see Figure 7-2 (b)). The addition of the Koopman operator dynamics further enhances the control authority of the quadcopter as shown with the direct adaptive stabilization, motor babble, and our approach to active learning. While the active Gaussian process model does at times succeed, the method relies on both the quality of data acquired and the local linear approximation to the dynamics. This results in a deficit of nonlinear information that is needed to successfully achieve the learning task in a single execution.

7.3.3 Sensitivity to initialization and parameters

We further test our algorithm against sensitivities to initialization of the Koopman operator. Our algorithm requires an initial guess at the Koopman operator in order to boot-strap the active learning process. We accomplish this using the same experiment described in the previous section which used a zero mean, variance of 1 normally distributed initialization of the Koopman operator. We vary the variance that initializes the Koopman operator parameters using a normal distribution with zero mean and a variance experiment set of $\{0.01, 0.1, 1.0, 10.0\}$.

In Fig. 7-3, we find that so long as the initialization of the Koopman operator is within a reasonable initialization (non-zero and within an order of magnitude), the performance is comparable to active learning described in Fig. 7-2. However, this may not be true for all autonomous systems and results may vary depending on the sampling frequency and the behavior of the underlying system. A benchmark is provided for stabilizing the quadcopter when the Koopman operator is precom-



(a) Stabilization Error **(b)** Information Gain

Figure 7-3: Resulting sensitivities in stabilization error and information gain with respect to variance levels in Koopman operator initialization. Benchmark stabilization performance is provided for known/precalculated Koopman operator.

puted in Fig 7-3 illustrating the performance of the control authority when using the Koopman operator-based controller.

The choices in the parameters of our algorithm can also effect its performance. Specifically, setting the value of the regularization term $\tilde{\mathbf{R}}$ too large will prevent the robot from significantly exploring the states of the robot. In contrast, if the regularization term is set too low, the robot will widen its breath of exploration which can be harmful to the robot if the states are not bounded. A similar effect is achieved by adding a weight on the active learning objective.

Changes in the time horizon T will also effect the performance of the algorithm. Generally, smaller T will result in more reactive behaviors where larger T tends to have more intent driven control responses. Choosing these values appropriately will be problem specific; however, the limited number of tunable parameters (not including choosing a task objective) provides the advantage of ease of implementation.

7.4 Active learning with automatic discovery of linear embedded features

As a solution to automating the choice of function observables, the use of deep neural networks [137] have been used to automatically discover the function observables. In this section, we illustrate that we can use these neural networks coupled with our approach for active learning to automatically discover the Koopman operator and the associated functions of state.

Revisiting Chapter 2, we can parameterize $z(x)$ and $v(x,u)$ using a multi-layer neural network with parameters $\theta \in \mathbb{R}^d$. We denote the parameterization of z, v as $z_\theta(x)$ and $v_\theta(x,u)$ where the subscript θ denotes the function observables are parameterized by the same set of parameters θ. Given the same data set that was defined previously, $\mathcal{D} = \{x(t_m), u(t_m)\}_{m=0}^{M}$, the new optimization problem that is to be solved is

$$\min_{\mathcal{K},\theta} \frac{1}{2} \sum_{m=0}^{M-1} \|\tilde{z}_\theta(x(t_{m+1}), u(t_{m+1})) - \mathcal{K}\tilde{z}_\theta(x(t_m), u(t_m))\|^2, \tag{7.12}$$

where $\tilde{z}_\theta(x,u) = \left[z_\theta(x)^\top, v_\theta(x,u)^\top\right]^\top$. Equation (7.12) can be solved using any of the current techniques for gradient descent (Adam method [83] is used in this work). The continuous time Koopman operator is obtained similarly using the matrix log of $\mathcal{K}$, resulting in the differential equation

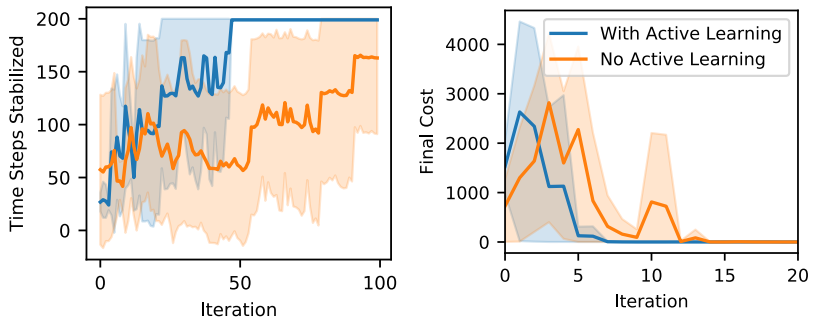
$$\dot{z}_\theta = \mathcal{K}_z z_\theta(x(t)) + \mathcal{K}_v v_\theta(x(t), u(t)). \tag{7.13}$$

Because we are now optimizing over θ, we lose the sample efficiency of single execution learning that was illustrated in the example in Section 7.3. Active learning can be used; however, adding the additional parameters θ to the information measure significantly increases the computational cost

of calculating the Fisher information measure (7.9). As a result, we only compute the information measure with respect to $\mathcal{K}$ in order to avoid the computational overhead of maximizing information with respect to θ.

7.4.1 Examples

We illustrate the use of deep networks for automating the function observables for the Koopman operator for stabilizing a cart pendulum and controlling a 2-link robot arm to a target. A neural network is first initialized (see Appendix D for details) for the Koopman operator functions z_θ, v_θ as well as an LQ controller for the task at hand. At each iteration, the robot attempts the task and learns the Koopman operator dynamics by minimizing (7.12). We compare against decaying additive control noise as well as our method for active learning where a weight on information measure is used which decays at each iteration according to γ^{i+1} where $0 < \gamma < 1$ and i is the iteration number. The data collected is then used to update the parameters θ and $\mathcal{K}$ using (7.12) and the LQ controller is updated with the new $\mathcal{K}_z, \mathcal{K}_v$ parameters.



(a) Cart Pendulum **(b)** 2-Link Robot

Figure 7-4: (a) Resulting stabilization time of a cart pendulum using Koopman operators with automatic embedding discovery. (b) Control response of a 2-link robot using Koopman operators with automatic function discovery. Active learning improves the rate of success of each task.

Figure 7-4 illustrates that we can automate the process of learning the function observables as well as the Koopman operator. With the addition of active learning, the process of learning the Koopman operator and the function observables is improved. In particular, stabilization of the cart pendulum is achieved in only 50 iterations in comparison to additive noise which takes over 100 iterations. Similarly, the 2-link robot can be controlled to the target configuration within 5 iterations with our active learning approach.

While this method is promising, there still exist significant issues that merit more investigation in future work. One of which is the trivial solution where $z_\theta, v_\theta = 0$. This issue often occurs with how the parameters θ were initialized. This trivial solution has been addressed in [138]; however, their approach requires significantly complicating how the regression (7.12) is formulated. We found that adding the state x as part of the neural network output of z_θ was enough to overcome the trivial solution.

7.5 Robot experiments

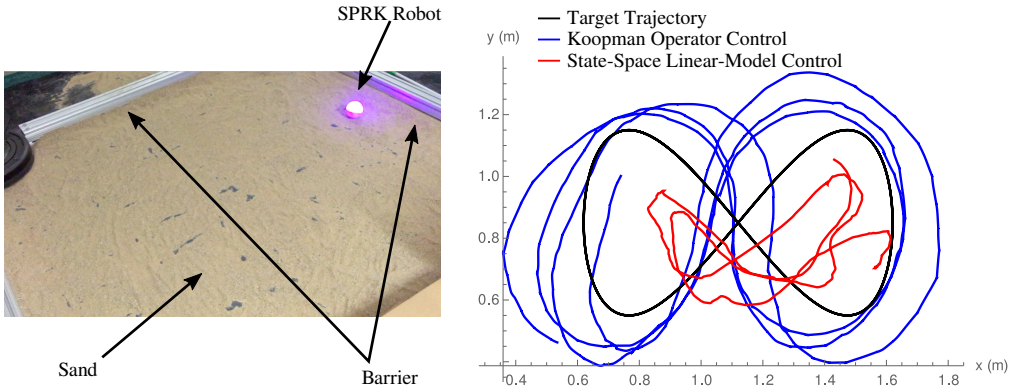
Our last set of examples test our active learning strategy with robot experiments. We use the robots depicted in Figure 7-5 to illustrate control and active learning with Koopman operators. The sphero SPRK robot (Figure 7-5a) is a differential drive robot inside of a clear outer ball. We test trajectory tracking of the SPRK robot in a sand terrain where the challenge is that the SPRK must be able to learn how to maneuver in sand. The Sawyer robot (Figure 7-5b) is a 7-link robot arm whose task is to track a trajectory defined at the end effector where the challenge is the high dimensionality of the robot.

(a) Sphero SPRK **(b)** Sawyer Robot

Figure 7-5: Depiction of robots used for experimentation. For videos of the robot examples visit `https://sites.google.com/view/active-learning-koopman-op`

7.5.1 Playing in a sand pit

Active learning is applied in an experimental setting using the Sphero SPRK robot (Fig. 7-5a) in sand. The interaction between sand and the SPRK robot makes physics-based models challenging. The parameters for the experiment are defined in Appendix D. The experiment starts with 20 seconds of active learning. After actively identifying the Koopman operator, the weight on information maximizing is set to zero at $t = 20$ and the objective is switched to track the trajectory shown in Fig. 7-6b. In Fig. 7-6c, we show the average root mean squared error (RMSE) of the $x-y$ trajectory tracking, the average $x - y$ Pearson's correlation using a two-sided hypothesis testing (values close to 1 indicate responsive controllers), and the phase lag of the experimental results. Note that in contrast to previous work in [52], the method of actively learning with Koopman operator improves the performance of the model-based controller. In particular, we find that the overall responsiveness and phase lag of the Koopman-based controller improved after active learning in sand.



(a) Experimental Setup **(b)** SPRK Trajectories

Method	RMSE	Correlation	Phase Lag (rad)
Koopman-based Control	0.3010	0.4028	1.1262
Controller in [52]	0.3535	0.1034	1.4667

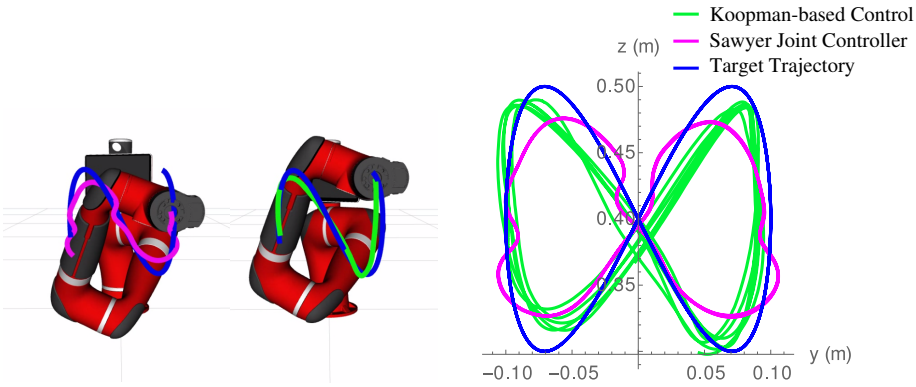
(c) Controller performance

Figure 7-6: Experiment using the Sphero SPRK robot in sand. (a) The experimental setup is depicted with the SPRK robot inside the sand pit. Position information is calculated with an overhanging Xbox Kinect using OpenCV [139] for tracking. (b) Performance of the SPRK robot using the Koopman operator-based controller after active learning. Performance is compared with results from [52]. (c) Performance measures showing active learning significantly outperforms non-active learning in robot experiment. The attached multimedia shows the experiment executed. For videos of this experiment visit `https://sites.google.com/view/active-learning-koopman-op`

7.5.2 Learning Sawyer dynamics from wiggling

In this experiment, we use active learning with the Koopman operator to model the dynamics of a 7 DoF Sawyer robot arm from Rethink Robotics. The 7-DoF system is of interest because it is both high dimensional and inertial effects tend to dominate the dynamics of the system. We define the parameters used for this experiment in Appendix D.

This example starts with the robot first actively learning its own dynamics (see videos in `https://sites.google.com/view/active-learning-koopman-op`). The resulting active learning trajectory is sequence of "wiggling" which excites the information rich dynamics of the robot arm directly



(a) Experimental Visualization **(b)** Sawyer Trajectories

Method	RMSE	Correlation	Phase Lag (rad)
Koopman-based Control	0.0228	0.9777	0.2826
Sawyer Joint Controller	0.0443	0.6026	0.7041

(c) Controller performance

Figure 7-7: Experiment using Sawyer. Experimental data visualized using RViz [140]. (a) End-effector trajectory paths using the embedded Rethink Joint controller and Koopman operator controller. Both controllers are running at 100 Hz. (b) Trajectory overlaid from both controller responses. (c) Controller performance shows that active learning for Koopman operator-based controllers performs comparably. We refer the reader to the attached multimedia to view clips of this experiment. For videos of this experiment visit `https://sites.google.com/view/active-learning-koopman-op`

from computed torques. The measurements are used to learn a Koopman model which we subsequently test on torque based tracking. Figure 7-7 illustrates a comparison of the embedded controller in the Sawyer robot and the model-based Koopman operator controller. Here, we show the average root mean squared error of the tracking position, the Pearson's correlation using a two-sided hypothesis testing (values close to 1 indicate responsive controllers), and the phase lag of the trajectory tracking. The resulting controller using the Koopman operator is shown to be comparable to the built-in controller with the inclusion of the evolution of the nonlinearities on the Sawyer robot which improve overall trajectory tracking performance. The trajectories of the two methods are overlaid which illustrates the improvement in control from the Koopman operator after

active learning has occurred. Since data is always being acquired online, the Koopman operator is continuously being updated as the robot is tracking the trajectory. The Koopman operator-based controller is able to capture dynamic effects of the individual joints from data. Note that one can build a model to solve for similar, if not better, inverse dynamics of the Sawyer robot that can be computed for control. In particular, the Sawyer robot provides an implementation of inverse dynamics in the robot's embedded controller. However, the proposed approach provides high accuracy without needing such a model ahead of time and without linearizing the nonlinear dynamics.

7.6 Discussion and summary

This chapter reexamines robot learning and control through the use of the linear embedding model structure. Linear models are well studied and can provide mathematical guarantees that many nonlinear models can not provide. This chapter shows that infinite linear embedding provide nonlinear information (at the cost of state-expansion) that makes them capable models for enhancing robot learning and control. Furthermore, I show that we can utilize results from prior chapters and apply them for actively learning infinite linear embedded models. The results in this chapter illustrate that robotic systems are capable of achieving fast and immediate learning performance that is safe and reliable through just a modification of how the learned models are structured.

The goal for this chapter, within the context of this thesis, is to illustrate that subtle modifications to the underlying learning problem can result in highly capable robotic systems. Simply changing the structure of what is learned can result in significant improvements for robot learning. It is possible that there exist a redefinition of the robot learning problem that makes the interconnectedness of action and sensing more clear and mathematically more elegant. However, without further breadth of study these modifications might ultimately never be discovered. Thus,

it is necessary for future work to seek out these hidden structures that can exploited to improve the capabilities of robotic systems as much as possible.

Chapter 8

Conclusions and outlook

Things are only impossible until they're not

Captain Picard

Welcome to the end! If you have made it this far, than congratulations! I applaud you for your tenacity. I can now tell you the secret meaning of life, and it is in fact, 42. By now, I hope that the you (the reader) have had the opportunity to learn and be inspired by the many examples and algorithms presented in this thesis that bring to light what is possible in robot learning and decision making. I started off this thesis with the question of what exactly does robot learning mean within the more general context of "machine" learning. The field of robot learning does not share the same requirements as the more general machine learning problem when it comes to how data is acquired and used. Robotic systems are inherently *active* systems that learn through interaction. They operate in a world that has physical constraints with real consequences for interacting and learning in the world. For most general machine learning problems, data is already acquired and

is used to learn in a *passive* manner. And for methods like Bayesian optimization that do consider active sampling, the physical constraints for sampling are never present or discussed as part of the problem formulation. This problem motivates the theme of this thesis: how do we make robotic systems capable of efficiently learning with intent where actions have physical consequences, where the world is complex to model, and where safety is of the upmost importance?

8.1 Discussion

In Chapter 3, I first try address this problem through improving the state of the art in robot learning. I introduced hybrid learning as an approach for viewing the current methods for robot learning as a hybrid mode scheduling problem. The take-away message being that it is desirable to first improve the current state of the art without necessarily relying on incremental improvements of the various approaches. I show that combining different modes of learning through hybrid control theory improves the overall learning process for robotic systems making them capable of learning motor skills efficiently. Unfortunately, proposed method does not address the problem of synthesizing intentional and informative actions for robotic systems, but instead tries to mask the problem of consequential interactions through fast learning. That being said, the formal tools used in this chapter are later adopted for more capable approaches for robot learning.

In the following Chapter 4, I first posed and motivated the problem of intentional learning in robotic systems as a direct active learning problem through information maximization. I presented the canonical maximum likelihood problem and derived an approach that uses the Cramér-Rao bound to relate the uncertainty of posterior model parameters to the Fisher information matrix. This relationship is used to construct optimizations problems that synthesize intentional robot actions for learning in various scenarios. In addition, I derived a more tractable approach to the

problem for complex neural-network style function approximators through an approximate Fisher information measures. I used these examples as an opportunity to motivate the subsequent chapters on why indirectly optimizing and sampling information measures for active learning is a better approach to active learning.

Chapters 5 and 6 posed the question of active learning for intentional robot interaction in the world through the use of ergodicity. I showed that optimizing the ergodic metric gives robotic systems the capability of learning models of spatially sparse environments through contact sensing. I showed that the ergodic control strategy is adaptable to the non-convex nature of information measures and allows robotic systems to avoid the downfall of local optima. In Chapter 6, I expanded the ergodic metric as a measure for exploring safely in high-dimensional dynamics spaces. The theoretical results from Chapter 3 are used as the foundation for KL-E^3 which enables active learning for more complex robot tasks while bounding exploration to equilibrium policies that does not impede on the acquisition of informative measurements.

In the last Chapter 7, I took the time to look back at the previous work and pose the question: would active learning and control be mathematically simpler and more efficient if we just used linear models to represent robotic systems? I posed this incredibly suggestive question not only to stir the curiosity of the reader, but to put the reader into the mind-set of questioning the small, ignored assumptions that we often impose on robotic systems that limit what our robots can achieve. It is at this point where I introduce the Koopman operator, a class of infinite linear embeddings that is capable of representing nonlinear systems as linear systems. Now, all of a sudden, it is possible to use linear models as a way of making planning, control, and active learning of robotic systems. I showed that this approach is not only simpler but provides more control authority in model-based control settings. These results are extended to active learning where the linear structure of the Koopman operator is used to optimize mathematically simpler information measures and provide

the control authority for a series of tasks that concludes this dissertation.

8.2 Future outlook

In this thesis, I presented a variety of methods for active learning and control for robotic systems which illustrate that under the right circumstances, it is possible for robotic systems to achieve learning within a single roll-out. However, these results merely open up a set of open questions that need to be expanded upon and investigated to further develop robotic system that can operate autonomously in the world.

8.2.1 Stochastic systems

In many of the examples presented, I assume deterministic dynamics with predictable outcomes. The main reasoning for choosing this for the dissertation is that the approaches presented were developed with the intent generate emergent exploration without the need for heuristic stochasticity as a boot-strap approach for learning. However, stochasticity and process noise is something that is unavoidable in robotic systems and must be accounted for during active learning. This is particularly true for the ergodic methods presented in this thesis. Having a way to reason about state uncertainty through predicted stochastic processes is beneficial and can provide insight on how we equip robotic systems with specific sensors.

Stochastic systems are incredibly interesting and insightful towards understanding active sensing behavior and developing new strategies that exploits new understandings. One of great benefits of Koopman operators is the ability to analyze the underlying systems' dynamic modes. These modes are a novel way to analyze nonlinear systems and expanding this kind of analysis to stochastic systems can only improve our understanding of nonlinear stochastic process.

8.2.2 Trajectory generation in active learning

Another avenue of research is trajectory generation in active learning. In Chapter 4, I presented various examples with direct active learning using Fisher information maximization. While I illustrated that these approaches are often intractable without approximations for arbitrarily complex models, constructing methods which make this approach tractable regardless of complexity or numerical approximation is a rich avenue for future work. Often one can find equivalences that can be exploited to indirectly solve the active learning problem (as done in Chapters 5 and 6). These equivalences can be used to further our understanding of the nature of active learning and the kinds of objective functions we an pose to solve these problems on larger settings. These new insights can often provide guidance for constructing and setting up learning problems for robotics systems in a variety of situations.

8.2.3 Continual learning and exploration

Inline with the idea of seeking out hidden assumption, what happens when we need to run our robots as time tends to infinity is as important as active learning and how robots learn in general. In many examples we see in research, our robotic systems generally are executed in episodes lasting a short amount of time. How do we construct and build learning autonomous systems that still function and operate even after a task is completed. How does one model "stand-by" mode and should the robot be doing something else during the downtime? These questions are ones related to continual learning and exploration in robotics. Creating robotic systems that can continually construct learning tasks is a interesting research avenue and the obvious next steps towards developing autonomous robotic systems. At some point, as human operators, we will not be able to teach the robotic system all the things in the world (either through hard-coding or with demonstrations). At this point, we need

to consider what it means for the robotic system to learn on its own; to adapt and comprehend through observation and interaction is the key to a possible renaissance in capabilities of robotic systems.

Bibliography

[1] R. A. Howard, "Dynamic programming and Markov processes." 1960.

[2] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

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

[4] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

[5] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.

[6] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox, "Model-based generalization under parameter uncertainty using path integral control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2864–2871, 2020.

[7] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7559–7566.

[8] A. Havens, Y. Ouyang, P. Nagarajan, and Y. Fujita, "Learning latent state spaces for planning through reward prediction," *arXiv preprint arXiv:1912.04201*, 2019.

[9] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," *arXiv preprint arXiv:1907.01657*, 2019.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[12] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[13] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 16–17.

[14] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3381–3388.

[15] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[16] D. Pomerleau, "An autonomous land vehicle in a neural network," *Advances in neural information processing systems(Morgan Kaufmann Publishers Inc., 1989)*, vol. 1, 1998.

[17] T.-C. Lin and Y.-C. Liu, "Direct learning coverage control based on expectation maximization in wireless sensor and robot network," in *Conference on Control Technology and Applications*, 2017, pp. 1784–1790.

[18] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *International Conference on Intelligent Robots and Systems*, vol. 1, 2002, pp. 540–545.

[19] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2016.

[20] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *International Conference on Intelligent Robots and Systems*, 2017, pp. 5204–5210.

[21] R. Arnold and A. Wellerding, "On the sobolev distance of convex bodies," *aequationes mathematicae*, vol. 44, no. 1, pp. 72–83, 1992.

[22] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Robotics research*, 2017, pp. 455–472.

[23] D. Ucinski, *Optimal measurement methods for distributed parameter system identification*. CRC Press, 2004.

[24] A. Emery and A. V. Nenarokomov, "Optimal experiment design," *Measurement Science and Technology*, vol. 9, no. 6, p. 864, 1998.

[25] D. Nguyen-Tuong and J. Peters, "Incremental online sparsification for model learning in real-time robot control," *Neurocomputing*, vol. 74, no. 11, pp. 1859–1867, 2011.

[26] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Dynamic task execution using active parameter identification with the baxter research robot," *Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 391–397, 2017.

[27] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4-5, pp. 432–442, 2011.

[28] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[29] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[30] 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.

[31] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[32] A. D. Ames and M. Powell, "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs," in *Control of Cyber-Physical Systems*, 2013, pp. 219–240.

[33] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multi-robot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[34] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[35] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[36] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *IEEE 56th Annual Conference on Decision and Control*, 2017, pp. 2242–2253.

[37] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 1424–1431.

[38] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," *arXiv preprint arXiv:2004.07584*, 2020.

[39] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, 2017, pp. 908–918.

[40] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in neural information processing systems*, 2016, pp. 2352–2360.

[41] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[42] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[43] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[44] I. Mezić, "Analysis of fluid flows via spectral properties of the Koopman operator," *Annual Review of Fluid Mechanics*, vol. 45, pp. 357–378, 2013.

[45] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[46] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS one*, vol. 11, no. 2, p. e0150171, 2016.

[47] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," *arXiv preprint arXiv:1707.01146*, 2017.

[48] G. Mamakoukas, M. L. Castano, X. Tan, and T. Murphey, "Local koopman operators for data-driven control of robotic systems." in *Robotics: science and systems*, 2019.

[49] I. Mezić, "On applications of the spectral theory of the Koopman operator in dynamical systems and control theory," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2015, pp. 7034–7041.

[50] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *arXiv preprint arXiv:1611.03537*, 2016.

[51] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2016, pp. 6492–6499.

[52] I. Abraham, G. de la Torre, and T. Murphey, "Model-based control using koopman operators," in *Proceedings of Robotics: Science and Systems*, 2017.

[53] A. Broad, T. Murphey, and B. Argall, "Learning models for shared control of human-machine systems with unknown dynamics," in *Proceedings of Robotics: Science and Systems*, 2017.

[54] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning data-driven stable koopman operators," *arXiv preprint arXiv:2005.04291*, 2020.

[55] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, 2015, pp. 2746–2754.

[56] I. Abraham, A. Broad, A. Pinosky, B. Argall, and T. D. Murphey, "Hybrid control for learning motor skills," in *Workshop on the Algorithmic Foundations of Robotics*, 2020.

[57] I. Abraham, A. Mavrommati, and T. D. Murphey, "Data-driven measurement models for active localization in sparse environments," in *Robotics: Science and Systems*, 2018.

[58] I. Abraham, A. Prabhakar, and T. D. Murphey, "Active area coverage from equilibrium," in *Workshop on Algorithmic Foundations of Robotics*, 2019.

[59] I. Abraham, A. Prabhakar, and T. D. Murphey, "An ergodic measure for active learning from equilibrium," *IEEE Transactions on Automation Science and Engineering*, 2020.

[60] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[61] I. Abraham and T. D. Murphey, "Decentralized ergodic control: distribution-driven sensing and exploration for multiagent systems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2987–2994, 2018.

[62] A. Prabhakar, I. Abraham, A. Taylor, M. Schlafly, K. Popovic, G. Diniz, B. Teich, B. Simidchieva, S. Clark, and T. Murphey, "Ergodic specifications for flexible swarm control: From user commands to persistent adaptation," *Robotics: Science and Systems*, 2020.

[63] A. Broad, I. Abraham, T. Murphey, and B. Argall, "Data-driven koopman operators for model-based shared control of human–machine systems," *The International Journal of Robotics Research*, 2020.

[64] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[65] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[66] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 377–382, 2002.

[67] H. Cramir, "Mathematical methods of statistics," *Princeton U. Press, Princeton*, vol. 500, 1946.

[68] C. R. Rao, "Information and the accuracy attainable in the estimation of statistical parameters," in *Breakthroughs in statistics*, 1992, pp. 235–247.

[69] B. Franchi, P. Hajłasz, and P. Koskela, "Definitions of sobolev classes on metric spaces," in *Annales de l'institut Fourier*, vol. 49, no. 6, 1999, pp. 1903–1924.

[70] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 62–80, 2018.

[71] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest, "Gradient descent approach to optimal mode scheduling in hybrid dynamical systems," *Journal of Optimization Theory and Applications*, vol. 136, no. 2, pp. 167–186, 2008.

[72] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and KL control," in *IEEE Conference on Decision and Control (CDC)*, 2012, pp. 1466–1473.

[73] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1433–1440.

[74] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[75] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.

[76] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[77] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 661–668.

[78] 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 preprint arXiv:1509.02971*, 2015.

[79] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Real-time trajectory synthesis for information maximization using sequential action control and least-squares estimation," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 4935–4940.

[80] F. Pukelsheim, *Optimal Design of Experiments*. SIAM, 2006.

[81] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory synthesis for Fisher information maximization," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1358–1370, 2014.

[82] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[84] E. Guić-Robles, C. Valdivieso, and G. Guajardo, "Rats can learn a roughness discrimination using only their vibrissal system," *Behavioural Brain Research*, vol. 31, no. 3, pp. 285–289, 1989.

[85] G. E. Carvell and D. Simons, "Biometric analyses of vibrissal tactile discrimination in the rat," *The Journal of Neuroscience*, vol. 10, no. 8, pp. 2638–2648, 1990.

[86] J. A. Hobbs, R. B. Towal, and M. J. Hartmann, "Spatiotemporal patterns of contact across the rat vibrissal array during exploratory behavior," *Frontiers in Behavioral Neuroscience*, vol. 9, p. 356, 2015.

[87] B. Mitchinson, C. J. Martin, R. A. Grant, and T. J. Prescott, "Feedback control in active sensing: rat exploratory whisking is modulated by environmental contact," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 274, no. 1613, pp. 1035–1041, 2007.

[88] L. M. Miller and T. D. Murphey, "Optimal planning for target localization and coverage using range sensing," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 501–508.

[89] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, 2017.

[90] R. E. Kopp, "Pontryagin maximum principle," in *Mathematics in Science and Engineering*, 1962, vol. 5, pp. 255–279.

[91] R. S. John and N. R. Draper, "D-optimality for regression designs: a review," *Technometrics*, vol. 17, no. 1, pp. 15–23, 1975.

[92] V. Fedorov, "Optimal experimental design," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 581–589, 2010.

[93] I. Abraham, A. Prabhakar, M. J. Hartmann, and T. D. Murphey, "Ergodic exploration using binary sensing for nonparametric shape estimation," *IEEE robotics and automation letters*, vol. 2, no. 2, pp. 827–834, 2017.

[94] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, "Pose estimation for planar contact manipulation with manifold particle filters," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 922–945, 2015.

[95] I. Abraham, A. Prabhakar, M. J. Z. Hartmann, and T. D. Murphey, "Ergodic exploration using binary sensing for nonparametric shape estimation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 827–834, 2017.

[96] C. Kreucher, K. Kastella, and A. O. Hero Iii, "Sensor management using an active sensing approach," *Signal Processing*, vol. 85, no. 3, pp. 607–624, 2005.

[97] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3, pp. 195–207, 1998.

[98] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 650–668, 1999.

[99] C. Kreucher, J. Wegrzyn, M. Beauvais, and R. Conti, "Multiplatform information-based sensor management: an inverted uav demonstration," in *Defense Transformation and Net-Centric Systems 2007*, vol. 6578, 2007, p. 65780Y.

[100] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *2013 American Control Conference*, 2013, pp. 4196–4201.

[101] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[102] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[103] E. D. Sontag, "Control-Lyapunov functions," in *Open problems in mathematical systems and control theory*, 1999, pp. 211–216.

[104] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[105] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 7, no. 11, pp. 1163–1173, 1983.

[106] J. A. Primbs, V. Nevistić, and J. C. Doyle, "Nonlinear optimal control: A control lyapunov function and receding horizon perspective," *Asian Journal of Control*, vol. 1, no. 1, pp. 14–24, 1999.

[107] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest, "Gradient descent approach to optimal mode scheduling in hybrid dynamical systems," *Journal of Optimization Theory and Applications*, vol. 136, no. 2, pp. 167–186, 2008.

[108] A. Polyakov and L. Fridman, "Stability notions and Lyapunov functions for sliding mode control systems," *Journal of the Franklin Institute*, vol. 351, no. 4, pp. 1831–1865, 2014.

[109] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[110] J. J. Moré and D. J. Thuente, "Line search algorithms with guaranteed sufficient decrease," *ACM Transactions on Mathematical Software (TOMS)*, vol. 20, no. 3, pp. 286–307, 1994.

[111] W. Zhong and H. Rock, "Energy and passivity based control of the double inverted pendulum on a cart," in *IEEE International Conference on Control Applications*, 2001, pp. 896–901.

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

[113] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[114] T. Fan and T. Murphey, "Online feedback control for input-saturated robotic systems on lie groups," *arXiv preprint arXiv:1709.00376*, 2017.

[115] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with Bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, 2016.

[116] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[117] E. Bıyık and D. Sadigh, "Batch active preference-based learning of reward functions," *arXiv preprint arXiv:1810.04303*, 2018.

[118] D. S. Brown, Y. Cui, and S. Niekum, "Risk-aware active inverse reinforcement learning," *arXiv preprint arXiv:1901.02161*, 2019.

[119] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *IEEE International Conference on Robotics and Automation*, vol. 4, 1997, pp. 3557–3564.

[120] N. Nahi and G. Napjus, "Design of optimal probing signals for vector parameter estimation," in *IEEE Conference on Decision and Control*, vol. 10, 1971, pp. 162–168.

[121] T. Morimura, E. . i. e. j. Uchibe, and K. Doya, "Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces," in *International Symposium on Information Geometry and Its Applications*, 2005, pp. 256–263.

[122] H. Wei, J. Zhang, F. Cousseau, T. Ozeki, and S.-i. Amari, "Dynamics of learning near singularities in layered networks," *Neural computation*, vol. 20, no. 3, pp. 813–843, 2008.

[123] M. Inoue, H. Park, and M. Okada, "On-line learning theory of soft committee machines with correlated hidden units–steepest gradient descent and natural gradient descent–," *Journal of the Physical Society of Japan*, vol. 72, no. 4, pp. 805–810, 2003.

[124] X. Yan, V. Indelman, and B. Boots, "Incremental sparse gp regression for continuous-time trajectory estimation and mapping," *Robotics and Autonomous Systems*, vol. 87, pp. 120–132, 2017.

[125] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.

[126] T. Lai and C.-Z. Wei, "Extended least squares and their applications to adaptive control and prediction in linear systems," *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 898–906, 1986.

[127] K. S. Sin and G. C. Goodwin, "Stochastic adaptive control using a modified least squares algorithm," *Automatica*, vol. 18, no. 3, pp. 315–321, 1982.

[128] F. Ding, X. Wang, Q. Chen, and Y. Xiao, "Recursive least squares parameter estimation for a class of output nonlinear systems based on the model decomposition," *Circuits, Systems, and Signal Processing*, vol. 35, no. 9, pp. 3323–3338, 2016.

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

[130] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *International Conference on Intelligent Robots and Systems*, 2010, pp. 3232–3237.

[131] R. Saegusa, G. Metta, G. Sandini, and S. Sakka, "Active motor babbling for sensorimotor learning," in *International Conference on Robotics and Biomimetics*, 2009, pp. 794–799.

[132] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," *arXiv preprint arXiv:1708.02596*, 2017.

[133] R. F. Reinhart, "Autonomous exploration of motor skills by skill babbling," *Autonomous Robots*, vol. 41, no. 7, pp. 1521–1537, 2017.

[134] T. Fan and T. Murphey, "Online feedback control for input-saturated robotic systems on lie groups," in *Proceedings of Robotics: Science and Systems*, 2016.

[135] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 491–496.

[136] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, "Safe exploration for active learning with gaussian processes," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 133–149.

[137] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," *arXiv preprint arXiv:1708.06850*, 2017.

[138] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.

[139] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[140] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[141] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[142] J. A. Boyan, "Least-squares temporal difference learning," in *ICML*, 1999, pp. 49–56.

[143] D. Precup, R. S. Sutton, and S. Dasgupta, "Off-policy temporal-difference learning with function approximation," in *ICML*, 2001, pp. 417–424.

[144] O. Klimov and J. Shulman, "Roboschool," https://github.com/openai/roboschool, 2017.

Appendix A

Proofs

Proof of Lemma 1

Proof. First define the trajectory

$$x(t) = x(0) + \int_0^{\tau} f(x(t), \mu(x(t)))dt + \int_{\tau}^{\tau+\lambda} f(x(t), \hat{u}(t))dt \; + \int_{\tau+\lambda}^{t_H} f(x(t), \mu(x(t)))dt \tag{A.1}$$

generated from $u(t) = \begin{cases} \hat{u}(t), & \text{if } t \in [\tau, \tau+\lambda] \\ u_{\text{def}}(t) & \text{otherwise} \end{cases}$ where $a_{\text{def}}(t) = \mu(s(t))$. Next, let us take the derivative of (A.1) with respect to the time duration λ so that we have the following expression:

$$\frac{\partial}{\partial \lambda}\mathcal{J} = \int_{\tau+\lambda}^{t_H} \frac{\partial r}{\partial x}^{\top} \frac{\partial x}{\partial \lambda} dt. \tag{A.2}$$

Using (A.1), we can define $\frac{\partial x}{\partial \lambda}$ as

$$\frac{\partial x(t)}{\partial \lambda} = f_2 - f_1 + \int_{\tau+\lambda}^{t} \left(\frac{\partial f}{\partial x} + \frac{\partial \mu}{\partial x}^{\top} \frac{\partial f}{\partial u} \right)^{\top} \frac{\partial x(\sigma)}{\partial \lambda} d\sigma \tag{A.3}$$

where σ is a place holder for time under the integrand, and $f_1 = f(x(t), \mu(x(t)))$ and $f_2 = f(x(t), \hat{u}(t))$ are remaining boundary terms from applying Leibniz's rule.

Noting that (A.3) is a linear convolution (due to the repeating $\frac{\partial x}{\partial \lambda}$ terms) with initial condition $\frac{\partial x}{\partial \lambda}(\tau) = f_2 - f_1$, we can rewrite (A.3) using a state-transition matrix

$$\Phi(t, \tau) = \exp\left(\left(\frac{\partial f}{\partial x} + \frac{\partial \mu}{\partial x}^{\top} \frac{\partial f}{\partial u} \right)^{\top} (t - \tau) \right)$$

with initial condition $f_2 - f_1$ as

$$\frac{\partial x(t)}{\partial \lambda} = \Phi(t, \tau)(f_2 - f_1) \tag{A.4}$$

Using (A.4) in (A.2) and pulling out the term $f_2 - f_1$ from under the integrand, we can rewrite (A.2) as the following:

$$\frac{\partial}{\partial \lambda} \mathcal{J}(\tau) = \lim_{\lambda \to 0} \int_{\tau+\lambda}^{t_H} \frac{\partial r}{\partial x}^{\top} \Phi(t, \tau) dt \, (f_2 - f_1) \, .$$

Taking the limit as $\lambda \to 0$ gives the instantaneous sensitivity from switching from $\mu \to \hat{u}$ at any time τ. Let us define this term as the adjoint variable

$$\rho(\tau)^{\top} = \int_{\tau}^{t_H} \frac{\partial r}{\partial s}^{\top} \Phi(t, \tau) dt \tag{A.5}$$

which give us the mode insertion gradient

$$\frac{\partial}{\partial \lambda}\mathcal{J}(\tau) = \rho(\tau)^\top (f_2 - f_1) \tag{A.6}$$

where the adjoint can be rewritten as the following differential equation

$$\dot{\rho}(t) = -\frac{\partial r}{\partial x} - \left(\frac{\partial f}{\partial x} + \frac{\partial \mu}{\partial x}^\top \frac{\partial f}{\partial u}\right)\rho(t) \tag{A.7}$$

with terminal condition $\rho(t_H) = \mathbf{0}$. □

Proof of Theorem 2

Proof. Expanding the objective in (3.7), we can show that

$$\begin{aligned} u^\star &= \arg\min_u \mathbb{E}_{\mathbb{Q}^\star}\left[\log\left(\frac{q^\star(v)}{q(v)}\right)\right] \\ &= \arg\min_u \int_\Omega q^\star(v)\log\left(\frac{q^\star(v)}{p(v)}\frac{p(v)}{q(v)}\right)dv \\ &= \arg\min_u \int_\Omega q^\star(v)\log\left(\frac{q^\star(v)}{p(v)}\right) - \int_\Omega q^\star(v)\log\left(\frac{q(v)}{p(v)}\right)dv \\ &= \arg\max_u \int_\Omega q^\star(v)\log\left(\frac{q(v)}{p(v)}\right)dv. \end{aligned} \tag{A.8}$$

Defining the policy $\pi(v_t \mid x_t) = \mathcal{N}(\mu(x_t), \Sigma(x_t))$ as normally distributed, we can show that

$$\begin{aligned}\frac{q(v)}{p(v)} &\propto \exp\left(\sum_t -\frac{1}{2}(v_t - u_t)^\top \Sigma^{-1}(v_t - u_t) + \frac{1}{2}(v_t - \mu(x_t))^\top \Sigma^{-1}(v_t - \mu(x_t))\right) \\ &= \exp\left(\sum_t -\frac{1}{2}(v_t - u_t)^\top \Sigma^{-1}(v_t - u_t) + \frac{1}{2}(v_t - \mu(x_t))^\top \Sigma^{-1}(v_t - \mu(x_t))\right) \\ &= \exp\left(\sum_t -\frac{1}{2}u_t^\top \Sigma^{-1} u_t + u_t^\top \Sigma^{-1} v_t + \mu(x_t)^\top \Sigma^{-1}(\mu(x_t) - 2v_t)\right)\end{aligned}$$

where $\Sigma = \Sigma(x)$ is used as short-hand notation. Plugging this expression into Eq. (A.8) gives

$$u^\star = \arg\max_u \sum_t -\frac{1}{2}u_t^\top \Sigma^{-1} u_t + u_t^\top \int_\Omega q^\star(v)\Sigma^{-1} v_t dv + \mu(x_t)^\top \int_\omega q^\star(v)(\mu(x_t) - 2v_t)dv.$$

which we can solve for u_t at each time by setting the derivative with respect to u_t to zero to give the optimal solution

$$u_t^\star = \int_\Omega q^\star(v) v_t dv. \tag{A.9}$$

Note that the expression $q^\star(v) \propto \exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right) p(v)$ which allows us to rewrite (A.9) in the following way:

$$\begin{aligned}u_t^\star &= \int_\Omega q^\star(v) v_t dv = \int_\Omega \frac{1}{\eta}\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right) p(v) v_t dv \\ &= \mathbb{E}_{\mathbb{P}}\left[\frac{1}{\eta}\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right) v_t\right].\end{aligned}$$

Using the change of variable $v_t = u_t + \delta u_t$, we get the recursive, sample-based solution

$$u_t^\star = u_t + \sum_k \omega(v_t^k)\delta u_t^k$$

where

$$\omega(v) = \frac{\exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v)}{\sum_n \exp\left(\frac{1}{\lambda}\mathcal{J}(v)\right)p(v)}.$$

□

Proof of Proposition 1

Proof. Let us define the trajectory $x(t)$ switching from $\pi(x(\tau)) \to \hat{u}(\tau)$ for a duration of λ as

$$x(t) = x(t_0) + \int_{t_0}^{\tau} f(x, \pi(x))dt + \int_{\tau}^{\tau+\lambda} f(x, \hat{u})dt + \int_{\tau+\lambda}^{t_f} f(x, \pi(x))dt \tag{A.10}$$

where we drop the dependence on time for clarity. Taking the derivative of (6.2), using (A.10), with respect to the duration time λ gives us the following expression:

$$\frac{\partial}{\partial\lambda} D_{\mathrm{KL}} = -\sum_i \frac{p(s_i)}{q(s_i)} \int_{\tau+\lambda}^{t_f} \frac{\partial q}{\partial x}^{\top} \frac{\partial x}{\partial \lambda} dt. \tag{A.11}$$

We obtain $\frac{\partial x}{\partial \lambda}$ by using Leibniz's rule to evaluate the derivative of (A.10) with respect to λ at the integration boundary conditions to obtain the expression

$$\frac{\partial x(t)}{\partial \lambda} = (f_2 - f_1) + \int_{\tau+\lambda}^{t} \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right)^{\top} \frac{\partial x(s)}{\partial \lambda} ds \tag{A.12}$$

where s is a place holder variable for time, $f_2 = f(x(\tau+\lambda), \hat{u}(\tau+\lambda))$ and $f_1 = f(x(\tau+\lambda), \pi(x(\tau+\lambda))$. Noting that $\frac{\partial x}{\partial \lambda}$ is a repeated term under the integral, (A.12) is a linear convolution with initial

condition $\frac{\partial x(\tau+\lambda)}{\partial \lambda} = f_2 - f_1$. As a result, we can rewrite (A.12) using a state-transition matrix [141]

$$\Phi(t, \tau + \lambda) = \exp\left(\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right)^\top (t - \tau)\right)$$

with initial condition $f_2 - f_1$ as

$$\frac{\partial x(t)}{\partial \lambda} = \Phi(t, \tau + \lambda)(f_2 - f_1). \tag{A.13}$$

Using (A.13) in (A.11) gives the following expression

$$\frac{\partial}{\partial \lambda} D_{\mathrm{KL}} = -\sum_i \frac{p(s_i)}{q(s_i)} \int_{\tau+\lambda}^{t_f} \frac{\partial q}{\partial x}^\top \Phi(t, \tau + \lambda) dt \, (f_2 - f_1) \, . \tag{A.14}$$

Taking the limit as $\lambda \to 0$ we then set

$$\rho(\tau)^\top = -\sum_i \frac{p(s_i)}{q(s_i)} \int_{\tau}^{t_f} \frac{\partial q}{\partial x}^\top \Phi(t, \tau) dt \tag{A.15}$$

in (A.14) which results in

$$\frac{\partial}{\partial \lambda} D_{\mathrm{KL}} = \rho(\tau)^\top (f_2 - f_1) \, . \tag{A.16}$$

Taking the derivative of (A.15) with respect to time τ yields the following:

$$\frac{\partial}{\partial \tau} \rho(\tau)^\top = \sum_i \frac{p(s_i)}{q(s_i)} \frac{\partial q}{\partial x}^\top \Phi(\tau, \tau) - \sum_i \frac{p(s_i)}{q(s_i)} \int_{\tau}^{t_f} \frac{\partial q}{\partial x}^\top \frac{\partial}{\partial \tau} \Phi(t, \tau) dt.$$

Since $\Phi(\tau, \tau) = 1$, and

$$\frac{\partial}{\partial \tau} \Phi(t, \tau) = -\Phi(t, \tau) \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right),$$

we can show that

$$\frac{\partial}{\partial \tau}\rho(\tau)^\top = \sum_i \frac{p(s_i)}{q(s_i)}\frac{\partial q}{\partial x}^\top - \underbrace{\left(-\sum_i \frac{p(s_i)}{q(s_i)}\int_\tau^{t_f}\frac{\partial q}{\partial x}^\top \Phi(t,\tau)dt\right)}_{=\rho(\tau)^\top}\left(\frac{\partial f}{\partial x}+\frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right).$$

Taking the transpose, we can show that $\rho(t)$ can be solved backwards in time with the differential equation

$$\dot{\rho}(t) = \sum_i \frac{p(s_i)}{q(s_i)}\frac{\partial q}{\partial x} - \left(\frac{\partial f}{\partial x}+\frac{\partial f}{\partial u}\frac{\partial \pi}{\partial x}\right)^\top \rho(t) \tag{A.17}$$

with final condition $\rho(t_f) = \mathbf{0}$. □

Proof of Theorem 3

Proof. Writing the integral form of the Lyapunov function switching between $\mu(x(t))$ and $u^\star(t)$ at time τ for a duration of time λ starting at $x(0)$ can be written as

$$\begin{aligned} V(x_\lambda^\tau(t)) = V(x(0)) &+ \int_0^\tau \nabla V \cdot f(x(s),\pi(x(s)))ds \\ &+ \int_\tau^{\tau+\lambda} \nabla V \cdot f(x(s),u^\star(s))ds \\ &+ \int_{\tau+\lambda}^{t} \nabla V \cdot f(x(s),\pi(x(s)))ds \end{aligned} \tag{A.18}$$

where s is a place holder for time. Expanding $f(x,u)$ to $g(x)+h(x)u$ and using (6.10) we can show the following identity:

$$\begin{aligned}\nabla V \cdot f(x,u^\star) &= \nabla V \cdot g(x) + \nabla V \cdot h(x)u^\star \\ &= \nabla V \cdot g(x) + \nabla V \cdot h(x)\pi(x) - \nabla V \cdot h(x)\mathbf{R}^{-1}h(x)^\top \rho \\ &= \nabla V \cdot f(x,\pi(x)) - \nabla V \cdot h(x)\mathbf{R}^{-1}h(x)^\top \rho \end{aligned} \tag{A.19}$$

Using (A.19) in (A.18), we can show that

$$\begin{aligned}V(x_\lambda^\tau(t)) &= V(x(0)) + \int_0^t \nabla V \cdot f(x(s),\pi(x(s)))ds - \int_\tau^{\tau+\lambda} \nabla V \cdot h(x(s))\mathbf{R}^{-1}h(x(s))^\top \rho(s)ds \\ &= V(x(t)) - \int_\tau^{\tau+\lambda} \nabla V \cdot h(x(s))\mathbf{R}^{-1}h(x(s))^\top \rho(s)ds \end{aligned} \tag{A.20}$$

where $x(t)$ is given by (6.6).

Letting the largest value of $\nabla V \cdot h(x(t))\mathbf{R}^{-1}h(x(t))^\top \rho(t)$ be given by

$$\beta = \sup_{s\in[\tau,\tau+\lambda]} -\nabla V \cdot h(x(s))\mathbf{R}^{-1}h(x(s))^\top \rho(s) > 0,$$

we can approximate (A.20) as

$$\begin{aligned}V(x_\lambda^\tau(t)) &= V(x(t)) - \int_\tau^{\tau+\lambda} \nabla V \cdot h(x(s))\mathbf{R}^{-1}h(x(s))^\top \rho(s)ds \\ &\leq V(x(t)) + \beta\lambda.\end{aligned}$$

Subtracting both side by $V(x(t))$ gives the upper bound

$$V(x_\lambda^\tau(t)) - V(x(t)) \leq \beta\lambda$$

which quantifies how much (6.10) deviates from the equilibrium conditions in (6.4). □

Proof of Proposition 3

Proof. Consider the objective (7.1) evaluated at a trajectory $z(t)\forall t \in [t_i, t_i + T]$ generated from a dynamical system. Furthermore, assume that $z(t_i+T)$ is generated by a policy $\pi(z(t))\forall t \notin [\tau, \tau + \lambda]$ and a controller $u^\star(t)\forall t \in [\tau, \tau + \lambda]$ where τ is the time of application of control $u^\star$ and λ is the duration of the control. Formally, $z(t_i + T)$ can be written as

$$\begin{aligned} z(t_i + T) = z(t_i) &+ \int_{t_i}^{\tau} f(z(t), \pi(z(t)))dt \\ &+ \int_{\tau}^{\tau+\lambda} f(z(t), u^\star(t))dt \\ &+ \int_{\tau+\lambda}^{t_i+T} f(z(t), \pi(z(t)))dt, \end{aligned} \tag{A.21}$$

where $f(z, u) : \mathcal{F}^a \times \mathbb{R}^m \to \mathcal{F}^a$ is a mapping which describes the time evolution of the state $z(t)$.

Using (A.21) and (7.1), we compute the derivative of (7.1) with respect to the duration λ of control $u^\star$ applied at any time $\tau \in [t_i, t_i + T]$:

$$\frac{\partial}{\partial\lambda}\mathcal{J}\bigg|_\tau = \int_{\tau+\lambda}^{t_i+T} \left(\frac{\partial\ell}{\partial z} + \frac{\partial\pi}{\partial z}^\top \frac{\partial\ell}{\partial u}\right)^\top \frac{\partial z}{\partial\lambda} dt. \tag{A.22}$$

where

$$\frac{\partial z(t)}{\partial \lambda} = f_2 - f_1 + \int_{\tau+\lambda}^{t} \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial z} \right)^{\top} \frac{\partial z(s)}{\partial \lambda} ds \tag{A.23}$$

such that $f_2 = f(z(\tau), u^{\star}(\tau))$, $f_1 = f(z(\tau), \pi(z(\tau)))$ are boundary terms from applying Leibniz's rule.

Because (A.23) is a linear convolution with initial condition, $\frac{\partial z(\tau)}{\partial \lambda} = f_2 - f_1$, we are able to rewrite the solution to $\frac{\partial z(t)}{\partial \lambda}$ using a state-transition matrix $\Phi(t, \tau)$ [141] with initial condition $f_2 - f_1$ as

$$\frac{\partial z(t)}{\partial \lambda} = \Phi(t, \tau)\left(f_2 - f_1\right). \tag{A.24}$$

Since the term $f_2 - f_1$ is evaluated at time τ, we can write (A.22) as

$$\frac{\partial}{\partial \lambda}\mathcal{J}\bigg|_{\tau} = \int_{\tau+\lambda}^{t_i+T} \left(\frac{\partial \ell}{\partial z} + \frac{\partial \pi}{\partial z}^{\top}\frac{\partial \ell}{\partial u} \right)^{\top} \Phi(t, \tau) dt \left(f_2 - f_1\right). \tag{A.25}$$

Taking the limit of (A.25) as $\lambda \to 0$ gives us the sensitivity of (7.1) with respect to switching at any time $\tau \in [t_i, t_i + T]$. We can further define the adjoint (or co-state) variable

$$\rho(\tau)^{\top} = \int_{\tau}^{t_i+T} \left(\frac{\partial \ell}{\partial x} + \frac{\partial \pi}{\partial x}^{\top}\frac{\partial \ell}{\partial u} \right)^{\top} \Phi(t, \tau) dt \in \mathbb{R}^{c_x}$$

which allows us to define the mode insertion gradient [71] as

$$\frac{\partial}{\partial \lambda} J\Big|_{t=\tau} = \rho(\tau)^{\top}\left(f_2 - f_1\right)$$

where

$$\dot{\rho} = -\left(\frac{\partial \ell}{\partial z} + \frac{\partial \pi}{\partial z}^{\top}\frac{\partial \ell}{\partial u} \right) - \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \pi}{\partial z} \right)^{\top} \rho$$

subject to the terminal condition $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$. □

Proof of Theorem 5

Proof. First define (7.1) for a controller as

$$\mathcal{J}(u(t)) = \int_{t_i}^{t_i+\Delta t} \frac{1}{\mathfrak{I}_u} + \ell_{\text{task}}(z(t), u(t))dt \tag{A.26}$$

where $\Delta t < T$ is a time duration, $z(t)$ is subject to the controller $u(t)$, and $\mathfrak{I}_u = \text{tr}\mathcal{I}(\mathcal{K})_u + \epsilon$ is the measure of information from applying the control u. If we consider the difference between $\mathcal{J}(u^\star)$ and $\mathcal{J}(\pi)$ where π is a controller that minimizes $\ell_{\text{task}}(z, u)$, then

$$\begin{aligned}
\mathcal{J}(u^\star) - \mathcal{J}(\pi) &= \int_{t_i}^{t_i+\Delta t} \frac{1}{\mathfrak{I}_{u^\star}} - \frac{1}{\mathfrak{I}_\pi} + \ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi)dt \\
&\approx \Delta t \left(\frac{1}{\mathfrak{I}_{u^\star}} - \frac{1}{\mathfrak{I}_\pi} + \ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi) \right) \\
&\quad + \mathcal{O}(\Delta t).
\end{aligned} \tag{A.27}$$

From Corollary 3 and that,

$$\frac{\partial}{\partial \lambda} \mathcal{J} \Delta t \approx \mathcal{J}(u^\star) - \mathcal{J}(\pi),$$

we can show that

$$\begin{aligned}
\frac{\partial}{\partial \lambda} J \Delta t &\approx \mathcal{J}(u^\star) - \mathcal{J}(\pi) \\
&\approx \Delta t \left(\frac{1}{\mathfrak{I}_{u^\star}} - \frac{1}{\mathfrak{I}_\pi} + \ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi) \right) \\
&\quad + \mathcal{O}(\Delta t).
\end{aligned} \tag{A.28}$$

which we rearrange (A.28) and insert (7.8) to get

$$\begin{aligned} -\| \left(\mathcal{K}_u v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} &\approx \left(\frac{1}{\mathfrak{I}_{u^\star}} - \frac{1}{\mathfrak{I}_\pi} + \ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi) \right) \\ &\quad + \mathcal{O}(\Delta t). \\ &\approx \frac{\mathfrak{I}_\pi - \mathfrak{I}_{u^\star} + (\ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi))\mathfrak{I}_{u^\star}\mathfrak{I}_\pi}{\mathfrak{I}_{u^\star}\mathfrak{I}_\pi} \\ &\quad + \mathcal{O}(\Delta t). \end{aligned} \tag{A.29}$$

Setting $\Delta\mathbf{I} = \mathfrak{I}_{u^\star} - \mathfrak{I}_\pi$ in (A.29) and simplifying gives the relative information gain

$$\Delta\mathbf{I} \approx (\| \left(\mathcal{K}_u v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} + \ell_{\text{task}}(z, u^\star) - \ell_{\text{task}}(z, \pi))\mathfrak{I}_{u^\star}\mathfrak{I}_\pi + \mathcal{O}(\Delta t).$$

□

Appendix B

Supplementary material for Chapter 3

Here, we present implementation details for each of the examples provided in the main paper as well as additional algorithmic details presented and mentioned throughout the paper. Any parameter not explicitly mentioned as deterministic or stochastic variations of hybrid learning are equivalent unless otherwise specified. All simulated examples have reward functions specified as the default rewards found in the Pybullet environments [74] unless otherwise specified. Table C.1 provides a lists of all hyperparameters used for each environment tested.

Model configuration

For each simulated example using the experience-based method, we use the same model representation of the dynamics as $x_{t+1} = x_t + f(x_t, u_t)$ where $f(x_t, u_t) = \mathbf{W}_2 \sin(\mathbf{W}_1[x_t, u_t] + \mathbf{b}_1) + \mathbf{b}_2$, and $\mathbf{W}_1 \in \mathbb{R}^{200\times(n+m)}$, $\mathbf{W}_2 \in \mathbb{R}^{n\times 200}$, $\mathbf{b}_1 \in \mathbb{R}^{200}$, $\mathbf{b}_2 \in \mathbb{R}^n$ are learned parameters. For locomotion tasks we use the rectifying linear unit (ReLU) nonlinearity. The reward function is modeled as

a two layer network with 200 hidden nodes and rectifying linear unit activation function. Both the reward function and dynamics model are optimized using Adam [83] with a learning rate of 0.003. The model is regularized using the negative log-loss of a normal distribution where the variance, $\Sigma_{\text{model}} \in \mathbb{R}^{n \times n}$, is a hyperparameter that is simultaneously learned based on model-free learning. The predicted reward utility is improved by the error between the predicted target and target reward equal to $\mathcal{L} = \|r_t + 0.95\ r(s_{t+1}, a_{t+1}) - r(x_t, u_t)\|^2$ (similar to the temporal-difference learning [142, 143]). This loss encourages learning the value of the state and action that was taken for environments that have rewards that do not strictly depend on the current state (i.e., the reward functions used in Pybullet locomotion examples). A batch size of 128 samples are taken from the data buffer $\mathcal{D}$ for training.

Policy configuration

For the policy, we parameterize a normal distribution with a mean function defined as a single layer network with $\sin(x)$ nonlinearity with 128 nodes (similar to the dynamics model used). The diagonal of the variance is specified using a single layer with 128 nodes and rectifying linear unit activation function. Soft actor critic (SAC) is used to optimize the policy for the pendulum, cartpole, hopper, and half-cheetah environments respectively. All examples use the same hyperparameters for SAC specified by the shared parameters in [11] including the structure of the value and soft Q functions, and excluding the batch size and policy (which we match the 128 samples used with model learning and to utilize the simpler policy representation).

The ant and panda robot with behavior cloning use the policy structure defined in Table C.1, which is structured in a similar stochastic parameterization as mentioned in the paragraph above. The negative log loss of the normal distribution is used for behavior cloning expert demonstrations

Environment	H	T	K	λ	policy dim	nonlinearity
Pendulum Swingup	5	200	20	0.1	128	$\sin(x)$
Cartpole Swingup	5	200	20	0.1	128	$\sin(x)$
Hopper	5	1000	20	0.1	128	$\text{relu}(x)$
Half-Cheetah	10	1000	20	0.2	128	$\text{relu}(x)$
Sawyer	10	100	10	0.01	128×128	$\text{relu}(x)$
Ant	20	400	40	1.0	128×64	$\text{relu}(x)$
Franka Panda	40	200	40	1.0	32×24	$\text{relu}(x)$

Table B.1: Parameters for all examples used in this chapter (only when applicable). Each example using the deterministic variation of hybrid learning (Alg. 2) uses added action exploration of the form $\epsilon = 0.999^t$ where t is the total number of environment interactions.

with a learning rate of 0.01 for each method.

Robot experiment configuration

In all robot experiments, a camera is used to identify the location of objects in the environment using landmark tags and color image processing. For the Sawyer robot example, the state is defined as the pose of the end-effector of the arm to the block as well as the pose of the block to the target. The action space is the target end-effector velocity. The reward is defined as

$$r(s, a) = -5\|p_{\text{b2t}}\| - 10\|p_{\text{ee2b}}\| - 0.01\|a\|^2$$

where $p_{\text{b2t}}, p_{\text{ee2b}}$ denote the poses of the block to the target and the end-effector to the target location respectively.

For the Franka robot, the state is defined as the end-effector position, the block position, and the gripper state (open or closed) as well as the measured wrench at the end-effector. The action

space is defined as the commanded end-effector velocity. The reward function is defined as

$$r(s,a) = r_{\text{stage}}(s) - 1.0e^{-6}\left(\|F_{\text{ee}}\| + \|a\|\right)$$

where

$$r_{\text{stage}}(s) = \begin{cases} -1.25\|p_{\text{ee}} - p_{\text{stack}}\|, & \text{if grasped block} \\ -\|p_{\text{ee}} - p_{\text{block}}\|, & \text{if not grasped block} \end{cases}$$

denotes the stage at which the Franka is in at the block stacking task. Here, $p_{\text{ee}}, p_{\text{block}}, p_{\text{stack}}$, and F_{ee} denote the end-effector pose, the block pose, the target stacking position, and the measured wrench at the end-effector respectively.

Algorithm 8 Hybrid Learning (stochastic) with Behavior Cloning

Randomly initialize continuous differentiable models f, r with parameters ψ and policy π with parameter θ. Initialize memory buffer $\mathcal{D}$ and expert data buffer $\mathcal{D}_{\text{exp}}$, prediction horizon parameter H.
while task not done **do**
    ▷ get expert demonstrations
    for $t = 0, \ldots, T-1$ **do**
        observe state x_t, expert action u_t
        observe x_{t+1}, r_t from environment
        $\mathcal{D}_{\text{exp}} \leftarrow \{x_t, u_t, r_t, x_{t+1}\}$
        $\mathcal{D} \leftarrow \{x_t, u_t, r_t, x_{t+1}\}$
    end for
    ▷ update models using data
    update ψ using $\mathcal{D}$ any regression method
    update θ using $\mathcal{D}_{\text{exp}}$ with behavior cloning
    ▷ test in environment
    for $t = 0, \ldots, T-1$ **do**
        observe state x_t
        get action u_t Alg. 2 or 3
        observe x_{t+1}, r_t from environment
        $\mathcal{D} \leftarrow \{x_t, u_t, r_t, x_{t+1}\}$
    end for
    if task not done, continue
end while

Appendix C

Supplementary material for Chapter 6

Algorithmic variations of KL-E^3 and example configurations

This appendix provides additional details for each learning goal presented in Chapter 6. This includes pseudo-code for each method and parameters to implement our examples (see Table. C.1). Videos of each example and demo code are provided in (`https://sites.google.com/view/kle3/home`).

Algorithm 9 KL-E^3 for Bayesian Optimization

1: **init:** see Alg. 6 and Alg. 1
2: **while** task not done **do**
3: set $x(t_i) = \hat{x}(t_i)$
4: $\triangleright$ simulation loop (see Alg. 6 lines 4-18)
5: get $\delta\mu(t)$ from simulation
6: $\triangleright$ apply to real robot (see Alg. 6 lines 20-28)
7: chose $\tau \in [t_i, t_i + t_H]$ and $\lambda \leq t_H$
8: $\triangleright$
9: **for** $t \in [t_i, t_{i+1}]$ **do**
10: **if** $t \in [\tau, \tau + \lambda]$ **then**
11: apply $u^\star(t) = \delta u^\star(t) + \pi(\hat{x}(t))$
12: **else**
13: apply $\mu(x(t))$
14: **end if**
15: **if** time to sample **then**
16: measure true state $\hat{x}(t)$ and $y(t) = \phi(\hat{x}(t))$
17: append to data set $\mathcal{D} \leftarrow \{\hat{x}(t), y(t)\}$
18: **end if**
19: **end for**
20: update posterior on ϕ given $\mathcal{D}$
21: update $p(s)$ from posterior
22: $i \leftarrow i + 1$
23: **end while**

Algorithm 10 KL-E^3 for Model Learning

init: see Alg. 6, generate initial parameter θ^0 for model (6.22)
while task not done **do**
    set $x(t_i) = \hat{x}(t_i)$
    ▷ simulation loop (see Alg. 6 lines 4-18)
    get $\delta\mu(t)$ from simulation
    ▷ apply to real robot
    chose $\tau \in [t_i, t_i + t_H]$ and $\lambda \leq t_H$
    for $t \in [t_i, t_{i+1}]$ **do**
        if $t \in [\tau, \tau + \lambda]$ **then**
            apply $u^\star(t) = \delta u^\star(t) + \pi(\hat{x}(t))$
        else
            apply $\pi(x(t))$
        end if
        if time to sample **then**
            measure state $\hat{x}(t)$, change in state $d\hat{x}(t)$, and applied control $u(t)$
            append to data set $\mathcal{D} \leftarrow \{\hat{x}(t), d\hat{x}(t), u(t)\}$
        end if
    end for
    sample K batch $\{\hat{x}_k, d\hat{x}_k, u_k\}_{k=1}^K$
    $\theta^{i+1} \leftarrow \theta^i + \sum_k \nabla_\theta \log \mathcal{L}$ given batch
    update $p(s)$ from $\sigma_{\theta^{i+1}}$
    $i \leftarrow i + 1$
end while

Algorithm 11 KL-E^3 enhanced DDPG

init: see Alg. 6, [78],
while task not done **do**
  set $x(t_i) = \hat{x}(t_i)$
  ▷ simulation loop (see Alg. 6 lines 4-18)
  get $\delta u^\star(t)$ from simulation and apply to real robot
  chose $\tau \in [t_i, t_i + t_H]$ and $\lambda \leq t_H$
  for $t \in [t_i, t_{i+1}]$ **do**
    if $t \in [\tau, \tau + \lambda]$ **then**
      apply $u^\star(t) = \delta u^\star(t) + \pi(\hat{x}(t))$
    else
      apply $\mu(x(t))$
    end if
    if time to sample **then**
      measure state $\hat{x}(t)$, next state $\hat{x}'(t)$, applied control $u(t)$, reward $r(t)$
      append $\mathcal{D} \leftarrow \{\hat{x}(t), \hat{x}'(t), u(t), r(t)\}$
    end if
  end for
  if time to update and buffer is large enough **then**
    update Q, μ from [78]
    update $p(s)$ using Q (6.27)
  end if
  $i \leftarrow i + 1$
end while

Algorithm 12 KL-E^3 for Active Query-Based Imitation Learning

init: see Alg. 6, append known dynamics model, expert policy $\pi^\star$, behavior cloned policy $\pi(x, \theta)$, policy parameters θ, demonstration data set $\mathcal{D}$, $\mathcal{I}_d = -\infty$
while task not done **do**
  set $x(t_i) = \hat{x}(t_i)$
  ▷ simulation loop (see Alg. 6 lines 4-18)
  get $\delta\mu(t)$ from simulation
  ▷ apply to real robot (see Alg. 6 lines 20-28)
  if $\mathrm{tr}\mathcal{I}(\theta) \mid_{x(t_i)} > \mathcal{I}_d$ **then**
    query expert demonstration $\mathcal{D} \leftarrow \{x_t, \pi^\star(x_t)\}_{t=0}^{T-1}$ with $x_0 = x(t_i)$
    $\mathcal{I}_d \leftarrow \mathrm{tr}\mathcal{I}(\theta) \mid_{x(t_i)}$
  end if
  gradient update θ given $\mathcal{D}$ and behavior clone objective
  update $p(s)$ using $\mathrm{tr}\mathcal{I}(\theta)$
  $i \leftarrow i + 1$
end while

Example	t_H	λ	$f(x,u)$	$\pi(x)$	$\mathbf{R}$	Σ	N samples
Cart Double Pendulum Bayes. Opt.	0.2 s	line search $\lambda < t_H$	local linear model at inverted pose	LQR stabilizing policy	0.1	0.1	20
Quadcopter Model Learning	0.6 s	$\lambda = t_H$	local linear model at hoverheight	LQR hovering policy	$0.5\mathbf{I}$	$0.1\mathbf{I}$	100
DDPG Cart pole swingup	0.1 s	$\lambda = 0.02$ s	neural-net model $\dot{x} = f(x,u;\theta)$ learned from data	Learned swingup skill	$0.01 * 0.99^t$	$0.1\mathbf{I}$	20
DDPG Half Cheetah running	0.03 s	$\lambda = 0.0165s$	neural-net model $\dot{x} = f(x,u;\theta)$ learned from data	Learned running skill	$0.01 * 0.99^t$	$0.1\mathbf{I}$	50

Table C.1: Parameters used for each method presented in Chapter 6 . $\mathbf{I}$ indicates an identity matrix of size $m \times m$, $n \times n$ for the quadcopter model learning, and $v \times v$ where $v = n + m$ used in both DDPG examples. Neural net model parameters θ are learned by minimizing the error $\|\dot{x} - f(x,u;\theta)\|^2$ over a subset of K data points where $\dot{x} \approx (x(t_1) - x(t_0))/dt$ and dt is the time step of the system. A time-decaying $\mathbf{R}$ is used for both DDPG examples so that the largest exploring occurs earlier on in the episode to better assist the skill learning.

Supplementary material for interactive backflip learning example

The dynamics of the lunar-lander system are given by

$$\dot{x} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ -\sum_i \tan(u_i)\sin(x_3) \\ \sum_i \tan(u_i)\cos(x_3) - 1 \\ 2(u_1 - u_2) \end{bmatrix}$$

where the state is defined as $x = [x_1, x_2, x_3, \dot{x}_1, \dot{x}_2, \dot{x}_3]^\top$ corresponds to the planar global positions, the angular rotation, and the respective linear and angular velocities. The control $u = [u_1, u_2]^\top$ are the left and right thrusters respectively. Here, gravity and mass are normalized for simplicity and the control input is saturated using the tangent function. The stabilization policy is defined by an infinite horizon LQR policy with weights $Q = \mathbf{I} \in \mathbb{R}^6$ and $R = 0.001\mathbf{I} \in \mathbb{R}^2$ where $\mathbf{I}$ is the identity matrix. Sampling is done at a frequency of 10 hertz. The planning time horizon for KL-E^3 is set to 4 seconds and $\Sigma = 0.01\mathbf{I} \in \mathbb{R}^6$. The control regularization in KL-E^3 is set the same as for the LQR policy. The policy model for behavior cloning is defined as a single layer network model with sine as the nonlinear function with 64 nodes. The behavior cloned policies are trained by sampling 64 unique state-action pairs from the demonstration data set using the Adam [83] method with a learning rate of 0.003. After the total number of demonstrations are collected the policy is trained until the parameters converge to a minima.

The expert backflip policy is trained using policy gradient using the same structure and dimen-

sion as the behavior cloned model. The backflip cost function is defined as

$$\ell(x, u) = 10(x_3 + \pi)^2 + 5x_2^2 + 10\text{relu}(-x_2) + \sum_i \dot{x}_i^2 + 0.1 \sum_i u_i^2.$$

The policy is trained on normally distribution initial conditions at $x = 0$ with variance 0.1 using Adam [83] with a learning rate of 0.003 until the parameters converge.

Appendix D

Supplementary material for Chapter 7

Control of forced Van der Pol oscillator

The nonlinear dynamics that govern the Van der Pol oscillator are given by the differential equations

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + \epsilon(1 - x_1^2)x_2 + u \end{bmatrix}$$

where $\epsilon = 1$ and u is the control input.

The Koopman operator functions used are defined as

$$z(x) = \left[x_1, x_2, x_1^2, x_2 x_1^2\right]^\top$$

and $v(u) = u$. The same functions are used to compute a regression problem where the final

equation is given by

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{A}z(x) + \mathbf{B}v(u)$$

where $\mathbf{A} \in \mathbb{R}^{n \times a}$ and $\mathbf{B} \in \mathbb{R}^{n \times b}$ are both generated using linear regression.

The weight parameters for LQ control are

$$\mathbf{Q} = \text{diag}\left([1, 1]\right) \text{ and } \mathbf{R} = 0.1$$

where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{a \times a} \tag{D.1}$$

Quadcopter Free-Falling

The quadcopter system dynamics are defined as

$$\begin{aligned} \dot{h} &= h\begin{bmatrix} \hat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix}, \\ J\dot{\omega} &= M + J\omega \times \omega, \\ \dot{v} &= \frac{1}{m}Fe_3 - \omega \times v - gR^T e_3, \end{aligned}$$

where $h = (R, p) \in \mathrm{SE}(3)$, the inputs to the system are $u = [u_1, u_2, u_3, u_4]$, and

$$F = k_t(u_1 + u_2 + u_3 + u_4),$$
$$M = \begin{bmatrix} k_t l(u_2 - u_4) \\ k_t l(u_3 - u_1) \\ k_m(u_1 - u_2 + u_3 - u_4) \end{bmatrix}$$

(see [134] for more details on the dynamics and parameters used). Note that in this formulation of the quadcopter, the control vector u has bidirectional thrust.

The measurements of the state of the quadcopter are given by

$$[a_g, \omega, v]^\top \in \mathbb{R}^9 \tag{D.2}$$

where $a_g \in \mathbb{R}^3$ denotes the body-centered gravity vector and ω, v are the body angular and linear velocities respectively. The sampling rate for this system is 200 Hz.

We define the basis functions for this system as

$$z(x) = [a_g, \omega, v, g(v, \omega)]^T \in \mathbb{R}^{18}$$

where $g(v, \omega) = [v_3\omega_3, v_2\omega_3, v_3\omega_1, v_1\omega_3, v_2\omega_1,\ v_1\omega_2, \omega_2\omega_3, \omega_1\omega_3, \omega_1\omega_2]$ are the chosen basis functions such that ω_i, v_i are elements of the body-centered angular and linear velocity ω, v respectively. The functions for control are

$$v(u) = u \in \mathbb{R}^4.$$

The LQ control parameters for the stabilization problem are given as

$$\mathbf{Q} = \text{diag}\left([1, 1, 1, 1, 1, 1, 5, 5, 5]\right) \text{ and } \mathbf{R} = \text{diag}\left([1, 1, 1, 1]\right)$$

where the weight on the additional functions $\tilde{\mathbf{Q}}$ are set to zero as in (D.1) . The time horizon used in $0.1s$.

The active learning controller uses a weight on the information measure of 0.1 and a regularization weight $\tilde{\mathbf{R}} = \text{diag}(1000, 1000, 1000, 1000])$. Motor noise used in the two-stage method is given by uniform noise at 33% of the control saturation.

Models for automatic discovery of embeddings

In this example, we use the Roboschool environments [144] for the robot simulations. For the cart pendulum example, we use a three layer network with a single hidden layer for z_θ and v_θ with $\{4, 20, 40\}$ and $\{2, 20, 10\}$ nodes respectively for each layer making $a = 40$ and $b = 10$. The exploration noise used on the control is given by additive zero mean noise with a variance of 40% motor saturation decreasing at a rate of 0.9^{i+1}. The decay weight on the information measure is given by 0.2^{i+1}. The LQ weights are given by $\tilde{\mathbf{Q}} = \text{diag}([50.0, 1.0, 10.0, 0.1] + \vec{\mathbf{0}})$ where the first non-zero weights correspond to the states of the cart pendulum. A time horizon of $0.1s$ is used with a sampling rate of 50 Hz. The regularization weight $\tilde{\mathbf{R}} = 1 \times 10^6$.

For the 2-link robot example, we use a similar three layer network with a single hidden layer for z_θ and v_θ with $\{4, 20, 40\}$ and $\{2, 20, 20\}$ nodes respectively for each layer making $a = 40$ and $b = 10$. The exploration noise used on the control is given by additive zero mean noise with a variance of 40% motor saturation decreasing at a rate of 0.9^{i+1}. The decay weight on the information measure is given by 0.2^{i+1}. The LQ weights are given by $\tilde{\mathbf{Q}} = \text{diag}([10.0, 1.0, 20.0, 1.0] + \vec{0})$ where the first

non-zero weights correspond to the states of the cart pendulum. A time horizon of $0.05s$ is used with a sampling rate of 100 Hz. The regularization weight $\tilde{\mathbf{R}} = \text{diag}([1 \times 10^6, 1 \times 10^6])$.

SPRK playing in sand

The SPRK robot is running a 30 Hz sampling rate for control and state estimation. Control vectors are filtered using a low-pass filter to avoid noisy responses in the robot. The controller weights are defined as

$$\tilde{\mathbf{Q}} = \text{diag}([60, 60, 5, 5, \vec{\mathbf{1}}])\text{and } \mathbf{R} = \text{diag}([0.1, 0.1]).$$

The control regularization is $\tilde{\mathbf{R}} = \mathbf{R}$. A weight of 80 is added to the information measure. A time horizon of $0.5s$ is used to compute the controller.

We run the active learning controller for 20 seconds and then set the weight of the information measure to zero and track the end effector trajectory given by

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 0.5\cos(t) + 1.12 \\ 0.3\sin(2t) + 0.85 \end{bmatrix}.$$

In this example, the set of functions are chosen as a polynomial expansion of the velocity states $\mathbf{x} = [\dot{x}, \dot{y}]$ to the 3^{rd} order. The function observables are defined as

$$z(x) = \left[x, y, \dot{x}, \dot{y}, 1, \dot{x}^2, \dot{y}^2, \dot{x}^2\dot{y}, \ldots, \dot{x}^3\dot{y}^3\right]^T \in \mathbb{R}^{18}$$

and

$$v(x, u) = u \in \mathbb{R}^2.$$

Learning Sawyer dynamics

The Sawyer robot was run on a sampling rate of 100 Hz. Control vectors are filtered using a low-pass filter to avoid noisy responses in the robot. The controller weights are defined as

$$\tilde{\mathbf{Q}} = \text{diag}([200 \times \vec{\mathbf{1}} \in \mathbb{R}^{14}, \vec{\mathbf{1}}]) \text{and } \mathbf{R} = \text{diag}([0.001 \times \vec{\mathbf{1}} \in \mathbb{R}^{7}]).$$

The control regularization is $\tilde{\mathbf{R}} = \mathbf{R}$. A weight of 2000 is added to the information measure. A time horizon of $0.5s$ is used to compute the controller.

We run the active learning controller for 20 seconds and then set the weight of the information measure to zero and track the end effector trajectory given by

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.1\cos(2t) \\ 0.1\sin(4t) + 0.4 \end{bmatrix}.$$

The functions of state using to compute the Koopman operator are defined as

$$z(x) = \left[\mathbf{x}^T, 1, \theta_1\theta_2, \theta_2\theta_3, \ldots, \theta_6^3\theta_7^3, \dot{\theta}_1\dot{\theta}_2, \ldots, \dot{\theta}_6^3\dot{\theta}_7^3\right]^T \in \mathbb{R}^{51}$$

with $v(u) = u \in \mathbb{R}^7$ as the torque input control of each individual joint and states $\mathbf{x}$ containing the joint angles and joint velocities.