LEARNING, INFERENCE, AND CONTROL FOR CONSTRUCTION ROBOTS AND SPATIOTEMPORAL PROCESSES

BY

HARSHAL MASKE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Agricultural and Biological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Assistant Professor Girish Chowdhary, Chair
Associate Professor Timothy Bretl
Associate Professor Tony E. Grift
Professor Prabhakar Pagilla, Texas A&M University

ABSTRACT

Expert operators of real world robots, especially constructions robots, develop expertise from years of training and experience. In the absence of such experts, these robots are operated by novice operators, and this adversely affects the productivity. On the other hand, safety concerns and the nature of the operating environment limits the possibility of automating these robots. This thesis proposes a solution by considering a problem setting in which the robot learns a policy from experts to train novice human operators. Formally, this is posed as the problem of learning instructional policy from demonstration, given as $\pi^I : s \rightarrow \boldsymbol{i}$, that maps the state ($s$) of the robot to an instruction ($\boldsymbol{i}$) for a human operator. Existing methods learn policy from demonstration, however such policies do not relate to the human operator's action space and hence cannot be used to generate instructions for novice operators. We introduce action primitives that address this challenge of mapping continuous state action trajectories to human parse-able and executable instructions.

Construction tasks are complex as they consist of several subtasks with stochastic transitions. For such tasks, existing approaches learn policy for component subtask and then rely either on predefined decompositions or heuristics to generate policy for the entire task. To overcome this limitation, and to generate instructions for an entire construction task, this thesis proposes learning of a structured probabilistic model for instructional policy. This model utilizes hierarchy of Markov chains that incrementally captures the number of subtasks as well as their transitions. Switching between the subtasks is inferred using a likelihood rate based inference approach proposed in this thesis. Instructional policy model is tested based on a controlled group study involving 113 participants, who learn to perform the truck loading task on a hydraulic actuated scaled excavator robot.

Further this thesis investigates shared control design for construction robots. Existing work has established that shared control can improve cycle times in nominal conditions. However, these methods can be too slow to relinquish control in off-nominal cases, when the operator needs to deviate from the nominally optimal trajectory due to unforeseen obstacles or other uncertainties. With an objective to incorporate such capability, this thesis proposes a new shared control technique that utilizes the operator's intent to quickly relinquish control in off-nominal conditions. Theoretical

results with performance guarantees and improved obstacle reaction time are presented. Proposed design has been experimentally validated on Zermelo's navigation problem. The last part of the thesis introduces kernel observer for learning and inference of large-scale stochastic phenomena with both spatial and temporal (spatiotemporal) evolution. This work considers the problem of estimating the latent state of a spatiotemporally evolving continuous function using very few sensor measurements. The model consists of a dynamical systems prior over temporal evolution of weights of a kernel model. Theoretical results provide sufficient conditions on the number and spatial location of sensors required to guarantee state recovery. A lower bound on the minimum number of sensors required to robustly infer the hidden states is also derived. Finally, theoretical results for randomly selecting sensing or sampling locations based on the predictive kernel observer model are presented. Our approach outperforms state-of-the-art kernel based machine learning methods in numerical experiments on real world datasets.

To my mother, for her love and support.

ACKNOWLEDGMENTS

My under-grad days at the Indian Institute of Technology Kharagpur had given me a tremendous desire to do research in the area of Dynamics and control. I was especially fascinated by the ideas of physics based modeling for virtual simulation of real world dynamical systems. Credits for my research inclination goes to the illustrious Professor duo, Prof. A. B. Chattopadhyay and Prof. Amalendu Mukherjee. This inclination eventually lead to the enterprising endeavor of Doctoral study that began in August 2014, lasting for four long years, and culminating successfully in the form of this powerful thesis in August 2018.

The main person to be acknowledged is of course my advisor, Prof. Girish Chowdhary, who admitted me as his PhD student at the Oklahoma State University. And thanks to the generous $5000 fellowship granted by Okstate that allowed me to move around in a nice black Toyota Camry since March 2015, simply wow! At Okstate we were given a grand and spacious office space at the Advanced Technology Research Center, I loved it as this is where 50% of my PhD work was accomplished and this is where I wrote my most cited paper on 'Intent Aware Shared Control'.

About Girish, there is a lot to say and there is a lot to thank. As far as I recollect I have always left his office feeling inspired and motivated to do more and better research. IMHO, Girish is a highly optimistic, and self motivated person who is very passionate about research. I enjoyed doing research with him, and when I recollect the hours of research discussion with him, I feel it was a fabulous experience. Not to mention that our research discussion after playing badminton were amazing, and led to the idea of intent aware shared control.

Senior lab mentor, well I am Girish's first PhD student, fortunately I had Hassan Kingravi a former postdoc with Girish, with whom I eventually collaborated on Kernel observer. I owe a great deal to his coding skills, and hours of discussion on Skype, thank you Hassan. I would like to thank another senior member of our lab at Okstate, Ben Reish, he extended a lot of help in Latex problems. Well, Ben is a CTAN contributor and has written bloX package, which is useful to draw block diagrams in tikz.

I am thankful to my doctoral committee members: Timothy Bretl, Tony Grift, and Prabhakar Pagilla they have provided stimulating conversations, insights, mentoring, and mathematical advice whenever it was needed, and this dissertation wouldn't have been the same without them.

The most enjoyable parts of graduate school were undoubtedly the experiences I had with the lab members from DASLAB, whom I met at Okstate and UIUC. Beginning with Rakshit who took me as his roommate and had also arranged airport pick up on my first arrival. We stayed at the main place apartment, it was a really awesome place, and my home till I left Stillwater Oklahoma in May 2016. Other wonderful people I met at Stillwater - Alex, Milecia, Ali, Allan, Sri, Emilio, Max, David, Girish, Denis, and Shyam. Special thanks to Max, and Hardeep for being wonderful roommates. Certainly, I am forgetting many wonderful souls, special thanks to them as well.

For the next two years, Urbana-Champaign was the new home, new place, new people, and lots of new activities. Most fun activity was the summer dance classes organized by Illini Dancesport, thank you guys especially the trainers, John and Allen.

Thanks to the new lab members who joined at Illinois - Anay Pattnaik, and Anwesa Choudhuri, for bringing new life and new insights to the group. Special thanks to Joshi, Anay, and Anwesa for our afternoon tea breaks and the enlivening discussions we had. Jokes, discussions, and sometimes card games, really kept the wheels rolling, in the otherwise pressure cooker type situations of meeting deadlines for papers and report submissions. And a special mention and thanks to Anay, for being an amazing work out partner, and a person to whom I could talk freely, you really made the last two years of my PhD happier and easy going, thanks mate!

My mother Nalini Maske, has been an unending source of love, support, and encouragement my entire life. She always highly valued education, and supported my academic interests from a young age. Mom, thank you so much for weathering life's storms with me and never losing faith in me. I love you very much and could never have done any of this without your support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Many real world operations, such as those involving farming and construction robots, are safety critical and are characterized by uncertain and dynamically changing environment. For example, an excavator loading a truck in Figure 1.1, has to operate at different elevations, dig and scoop dynamically changing profiles of soil while ensuring safety of construction workers. Such characteristics makes the presence of human operators necessary and therefore robots such as heavy construction and farming equipments; examples include excavators, tractors, backhoes, etc have not been automated, though there exists substantial research [1–3]. However, the human operators of these robots require years of training and operation to develop high levels of skills and expertise [4]. Moreover, in case of increasing attrition or scarce number of expert operators, the robots are operated by inexperienced or entirely novice operators, causing an adverse impact on productivity. Thus, overcoming the relative skill-gap between expert and novice operators, without consuming the time of expert operators in training others, presents an important challenge. This paper conceives a solution to this problem by investigating a policy that allows the robot to teach or guide novice operators. Further, we develop a policy model that learns such a policy for an entire complex task. This is crucial to instruct an operator in doing and learning a complex task comprised of several subtasks.

1.1 Learning Instructional Policy Model for Construction Tasks

We investigate development of construction robots that can assist human operators by guiding them in doing a task as a solution to speed up the learning process of novice operators. Specifically, we want the robot to learn from expert operators, and generate instructions to assist and train novice operators. To date, Learning from Demonstration (LfD) has been widely studied in the context of robot learning a policy to do a task from teacher demonstrations (surveyed in [5]). In the context of present problem, we need to learn a policy that maps a robot's current state to an instruction for a human operator, we define this as the process of *learning instructional policy from demonstration.* We propose that a robot teaching a human operator needs to do much more than learning to execute the demonstrated task, such as, (i) It has to simplify and decompose the task into

Figure 1.1: An excavator robot and its human operator performing a truck loading task in a safety critical and dynamically changing environment. Our goal is to enable construction robots to learn instructional policy from skilled human operators, and use that policy to guide and train novice human operators in completing complex tasks.

human understandable task-primitives and communicate back the essential instruction (sequence of actions); (ii) These instructions must be in the same action-space that the human operates in, for example, in the excavator control problem, they must be in the joystick motion space; (iii) Finally the instructions must be delivered in a manner that adapts to the human operator's current task-state. Traditional learning from demonstration methods [6–9], have not considered the problem in which the robot instructs another (novice) human operator and hence do not addresses the above mentioned requirements of this new setting.

Our first contribution is the introduction of action primitives that enables mapping of continuous state-action trajectories to instructions in human operator's action space. As opposed to the widely studied concept of motion primitives, definition of action primitives (in section 3.2.1) is directly applicable for automatic segmentation of demonstration trajectories into simpler reusable primitives. This segmentation procedure does not require sampling based inference and is hence very efficient and scalable to larger datasets comprising of tens of thousands of state-action pairs. Second, we propose and demonstrate a single policy model representation for a complex task comprising of several subtasks. Existing methods typically learn individual policies for each subtasks, and rely on heuristics or pre-defined sequencing of subtasks to replay the task. Proposed policy representation models sequencing of subtasks along with the transitions between low level primitives within a subtask. This is accomplished using a hierarchy of Markov chains that uniquely associates low level primitives with the objects being manipulated in a task. Third, we created and tested two visual interfaces to communicate instructional policy to the human operator. Finally, we present exhaustive evaluation of the proposed instructional policy model through experiments

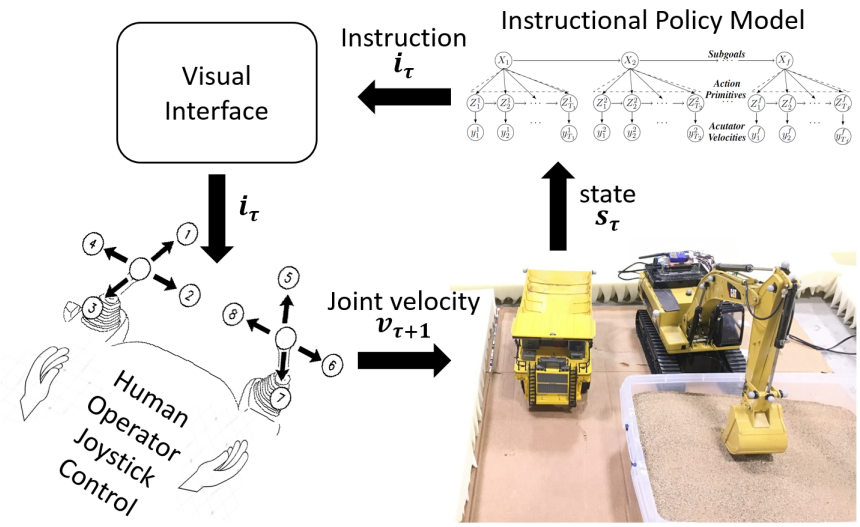


Figure 1.2: Excavator robot training a human operator to perform a construction task using Instructional Policy model (enlarged in Figure 3.3) learned from expert demonstrations. Instructional policy, $\pi^I : s \rightarrow \boldsymbol{i}$, maps robot's state ($s$) to an instruction ($\boldsymbol{i}$). The instruction $\boldsymbol{i}$ is communicated through the visual interface to the human operator who then commands an action here joint velocity.

involving 113 novice operators comparing performance between those assisted by the robot and those who learn to perform truck loading task based on observing other experts. Our approach is an attempt to investigate an architecture for a robot to train or assist their operators. Figure 1.2, is an example of such an architecture in which the excavator robot utilizes the feedback of its current state, and the instructional policy model, to generate instructions for the human operator.

1.1.1 Significance with respect to Relevant Literature

Existing LfD techniques can be broadly classified as those based on ideas from reinforcement learning ([10]) and direct policy approximation ([7]). Their objective is to infer a control policy that maps the robots state into actions, whereas in this thesis we develop a policy approximation approach that maps the robot's state to instructions for a human operator. Can the existing policy representations, such as linear, neural nets, dynamic motor primitives (DMPs), etc, generate instructions? These representations typically output actions in terms of desired trajectory and would require a series of transformations (see Figure 1.3) to generate instructions in the actuator space (human operator's action space). If the instructions are in trajectory space, the operator would be required to figure out suitable joystick actuations, rather than focusing on learning the task. In translating instructions to the actuator space, most critical part is to breakdown a skill in trajectory space to a sequence of discrete joystick commands. These positions should then be given as in-

structions at an operator friendly rate rather than continuous update of control inputs typically used by the robots. Hence, in our approach we pursue skill (or segment) identification in the actuator space, such skills can be communicated directly as operator friendly instructions.

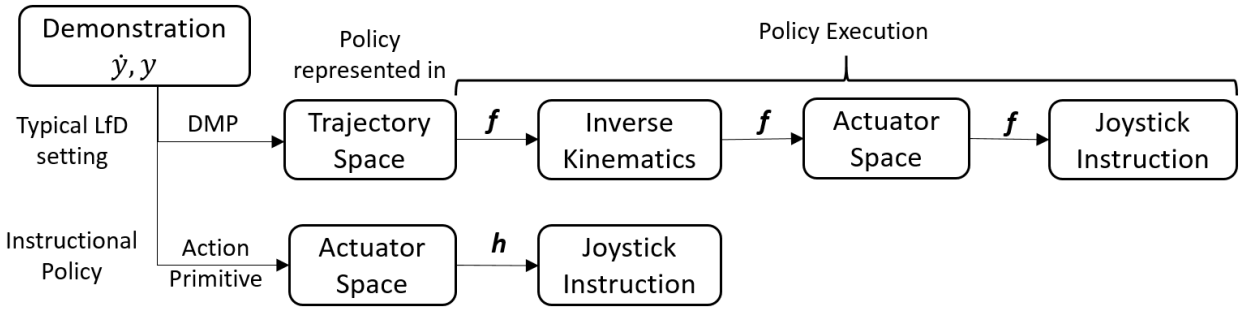


Figure 1.3: Transformation required for using existing LfD methods to generate instructional policy in human operator's action space i.e. the joystick space. Moreover, the policy is updated continuously at rate $\boldsymbol{f}$ in traditional setting, whereas an update rate $\boldsymbol{h}$ that updates instruction in an operator friendly manner is developed.

Further, our objective is to develop a single policy model representation that generates instructions for complex task comprising of several subtasks. Most of the existing research in policy learning ([9]) addresses the problem in which a robot learns a monolithic policy from a demonstration of a simple task that has a well-defined beginning and end. Recent approaches to learn complex tasks such as robot soccer goal scorer ([11]), playing table tennis ([12]), table leg assembly ([13]), and others, either utilize a predefined or hand-coded decomposition of a task into subtasks or resort to heuristics. [12] utilized a pre-defined sequence of skills, and used DMPs to represent the policy for each individual skill. [11] had utilized a hand-coded controller for transitions between subtasks for robot soccer goal scorer task while using infinite mixture of Gaussian process experts (IMoGPE) to model policies within each subtask and incrementally discovering each subtask. [13] had demonstrated success in learning table leg assembly from unstructured demonstration, utilizing a heuristic approach of finite state representation to sequence skills, where each skill (or segment) is modeled as a dynamic motor primitive. For these tasks, transition between skills occurs with respect to the task objects. For example, in table leg assembly, robot transitions from the reaching skill to grabbing of the leg after it positions its end-effector over the leg. Similarly, for robot soccer and table tennis, skill transitions occur with respect to the task objects being manipulated. We exploit this insight to learn subgoals in the trajectory space with respect to the task objects. Further, we let the sequence of skills (or segments) associated with such a subgoal to represent a subtask. And transition between these subgoals to represent the execution of a complex task.

Based on these insights and representations, we propose a structured policy representation to

learn a complex real world task as a probabilistic model, that captures transition among the low level primitives (skill or segments) associated with the higher level task context (subgoals). Such a policy parameterization model can generate policy for an entire task, enabling robots to perform or to instruct a complex real world task. We utilize a hierarchy of Markov chains to construct such a policy model. We utilize Dirichlet process means based clustering approach to discover subgoals. Transition among these subgoals is used as the higher level task context which is associated to the transition among the action primitives at the lower level. Though the construction of this model is inspired by the goal of generating instructional policy, with few modifications, the policy model is also demonstrated to perform autonomous task execution.

Another key aspect is the interface to communicate instructions to the human operator. This plays a key role in learning speed, and skill retention of novice operators. In our study, we utilize insights from operant conditioning (subjects conscious behavioral responses to an environment) and visual reinforcement. Videos are known to be effective at teaching tasks and problem solving strategies in both children and adults [14, 15]. With regards to interactive experiences, simple visual interfaces help reduce stimulant load during learning [16]. Additionally, the use of limited guidance or feedback, especially in the early training of novices, can also increase learning [17]. Based on this prior research, we propose and evaluate two different hypotheses for visual interfaces in training novices. Learning of the instructional policy model and the experimental results are presented in chapter 3.

1.2 Spatiotemporal Monitoring

Modeling of large-scale stochastic phenomena with both spatial and temporal (spatiotemporal) evolution is a fundamental problem in the applied sciences. Common examples include modeling ocean heat content and acidification in oceanography [18], future seismicity [19], land use change in urbanization [20], and extreme weather events [21], among others. While modeling spatiotemporal phenomena has traditionally been the province of the field of geostatistics, it has in recent years gained more attention in the machine learning community [22]. The data-driven models developed through machine learning techniques provide a way to capture complex spatiotemporal phenomena that are not easily modeled by first-principles alone, such as stochastic partial differential equations.

In the machine learning community, kernel methods represent a class of extremely well-studied and powerful methods for inference in spatial domains; in these techniques, correlations between the input variables are encoded through a covariance kernel, and the model is formed through a linear weighted combination of the kernels [23, 24]. In recent years, kernel methods have been ap-

plied to spatiotemporal modeling with varying degrees of success [22,23]. Many recent techniques in spatiotemporal modeling have focused on nonstationary covariance kernel design and associated hyperparameter learning algorithms [25–27]. These methods, which focus on the careful design of covariance kernels, have been proposed as an alternative to the naive approach of simply including time as an additional input variable in the kernel [28]. The careful design/optimization of covariance kernel avoids an explosion in the number of parameters (kernels utilized) of the model which would be inevitable in a model that simply adds time as an additional input variable, and has been shown to better account for spatiotemporal couplings. However, there are two key challenges with existing kernel based approaches: The first is ensuring the scalability of the model to large scale phenomena, which manifests due to the fact that the problem of optimizing the covariance kernel (known as hyperparameter optimization in the ML community) is not convex in general, leading to methods that are difficult to implement especially in online settings, susceptible to getting stuck at local minima, and highly computationally demanding for large datasets. The second key challenge is in using existing kernel-based machine learning models for analysis and synthesis of observers and controllers for the large scale spatiotemporal phenomena. While the first challenge can be addressed with increasing computational power for large datasets, addressing the latter (and vastly more fundamental) challenge is particularly important in the design of reliable engineering systems, such as distributed sensor/actuator networks intended for monitoring physical phenomena, autonomous soft-robots, or other physical systems with distributed sensing and actuation.

In this work, an alternative perspective of solving the spatiotemporal monitoring problem that brings together kernel-based modeling, systems theory, and Bayesian filtering is developed. We define the monitoring problem as follows: *Given an approximate predictive model of the spatiotemporal phenomena learned using historic data, estimate the current latent state of the phenomena in the presence of uncertainty using as few sensors as possible*. In particular, we argue that when it comes to predictive inference over spatiotemporal phenomena, a Kalman-filter type approach of predicting and correcting with feedback from a set of minimal sensors is a robust way of dealing with real-world uncertainties and inherent modeling errors. In the context of this specific problem, our *main contributions are two-fold*: first, we demonstrate that spatiotemporal functional evolution can be modeled using stationary kernels with a linear dynamical systems layer on their mixing weights. In particular, in contrast with existing work, this approach does not necessarily require the design of complex spatiotemporal kernels, and can accommodate positive-definite kernels on any domain on which it is possible to define them, which includes non-Euclidean domains such as Riemannian manifolds, strings, graphs and images [29]. Second, we show that such a model can be utilized to determine sensing locations that guarantee that the hidden states of functional evolution can be estimated using a Bayesian state-estimator (Kalman filter) with very few measurements. We provide sufficient conditions on the number and location of sensor measurements

required and prove non-conservative lower bounds on the minimum number of sampling locations by developing fundamental results on observability of kernel based models. The validity of the presented model and sensing techniques is corroborated using synthetic and large real datasets.

The fundamental idea of building observers and controllers introduced in this thesis is generalizable beyond the particular application of spatiotemporal monitoring. Indeed, the significance of the contributions of this work are in fusing machine learning theory with systems theory to provide a pathway to address major challenges in spatiotemporal monitoring and control. The problem of state estimation of a temporally evolving finite-dimensional state-space system has been extensively studied in the dynamical systems and feedback-control community [30]. Here, fundamental results in observability/controllability provide sufficient conditions on the structure of the state transition and measurement matrix such that the latent state can be estimated in the presence of measurement and process noise. Such filters can be naively extended to the functional domain (e.g. [31]), but have not typically been studied in context of the spatiotemporal monitoring problem studied here.

The marriage of systems theory with machine learning pursued in this paper is exciting because it can provide a formal way of answering fundamental questions about complex systems, such as: What is the least number of sensors required to observe a distributed system? Where to place sensors/actuators to guarantee observability/controllability of the system? And how does random sensor placement affect observability/controllability? We expect that follow on work will exploit the framework presented in this paper of utilizing linear models in feature spaces of machine learning models to enable practical and analyzable data-driven engineering systems. To facilitate the development of the theory, we have focused our work on the problem of monitoring spatiotemporal phenomena. However, the idea can be generalized to any distributed cyber-physical system that is changing with space and time.

We also addressed another very important challenge: given a predictive model of the spatiotemporal phenomena, can we randomly select sampling locations to estimate the current latent state of the phenomena? We present theoretical results that explain rigorously the interesting discovery in this work i.e. increasing the number of randomly placed sensors guarantee observability with probability that approaches one exponentially in section 6.2. Formulation of the kernel observer model, theoretical and experimental results are presented in chapter 6.

1.3 Mathematical Preliminaries

1.3.1 Gaussian Process Regression

A common approach in machine learning to model spatial dynamics is to use the nonparametric Bayesian framework known as Gaussian Processes [23]. Being a non-parametric model, the model structure need not be specified a priori but is instead determined from data. Typically, Gaussian Process Regression (GPR) is used to learn input-output mapping function f from the training data set $\mathcal{D}$ of n observations, $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$, where x denotes the input vector of dimension D, and y is the scalar output (or target); the column vector inputs for all n cases are aggregated in the $D \times n$, matrix X. Once the mapping function f is known for the set of inputs X, it can then be used to make predictions for all possible set of test values X_* through the derivation of the posterior function $f(X_*)$.

By definition, a Gaussian process describes distribution over functions and is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$ of a real process $f(x)$ as

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)], \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x) - m(x'))] \end{aligned}$$

which can be denoted as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

In our case the random variables represent the value of the function $f(x)$ at location x. Often, Gaussian processes are defined over time, i.e. the index set of the random variables is time. This is not the case in our use of GPs; here the index set $\mathcal{X}$ is the set of possible inputs, which could be more general, e.g. $\mathbb{R}^D$. In present work, we use squared exponential covariance function defined as,

$$cov(f(x), f(x')) = k(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma})$$

To derive $f(X_*)$ using GPR given the dataset $\mathcal{D}$, we begin by defining a zero mean prior over the functions as

$$f \sim \mathcal{N}(0, K(X, X))$$

where $K(X, X)$ is a covariance matrix, with entries $k(x_i, x_j)$ for $i, j = 1, \ldots, n$. Next, we incorporate measurement noise in the output as $y = f(x) + \epsilon$, assuming additive independent identically distributed Gaussian noise ϵ with variance σ_n^2 , hence the prior on the noisy observations now becomes $f \sim \mathcal{N}(0, K(X, X) + \sigma_n^2 I)$. The joint distribution of the measured target values and

the function values at the test locations according to the prior is

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

where $f_* \triangleq f(X_*)$. The posterior conditioned on the observations gives the key predictive equations for Gaussian process regression as

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, cov(\bar{f}_*)) \tag{1.1}$$

$$\bar{f}_* = K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} y \tag{1.2}$$

$$cov(\bar{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} K(X, X_*) \tag{1.3}$$

where $\bar{f}_*$ is the mean prediction at locations X_* and $cov(\bar{f}_*)$ is the predictive uncertainty.

For Spatiotemporal applications, we use GPR to model spatial dynamics. Equation (1.2), can be written as

$$\bar{f}_* = \sum_{i=1}^{M} K(X_*, x_i) w_i \tag{1.4}$$

where w_i is the i^{th} component of vector $w \in \mathbb{R}^M$ given as

$$w = [K(X,X) + \sigma_n^2 I]^{-1} y \tag{1.5}$$

We argue that a linear dynamical systems prior over the evolution of GP weights w, can be used to monitor temporal evolution of a process. Hence learning the GP weights at each instant of time and then its temporal evolution is a key approach in modeling processes evolving in space and time. Preliminary work using this key idea is elaborated in section 6.

1.3.2 Time Series Analysis

Hidden Markov models (HMMs) are generative Bayesian models that have long been used to make inferences about time series data. Time-series data, such as robot task demonstrations, present unique challenges for analysis, since observations are temporally correlated. Clearly, time-series data are not independent and identically distributed (nor are they exchangeable), but weaker assumptions can often be made about the data to make inference tractable. Define a sequence to be a first-order Markov chain if:

$$p(x_n|x_1, x_2, \ldots, x_{n-1}) = p(x_n|x_{n-1}) \tag{1.6}$$

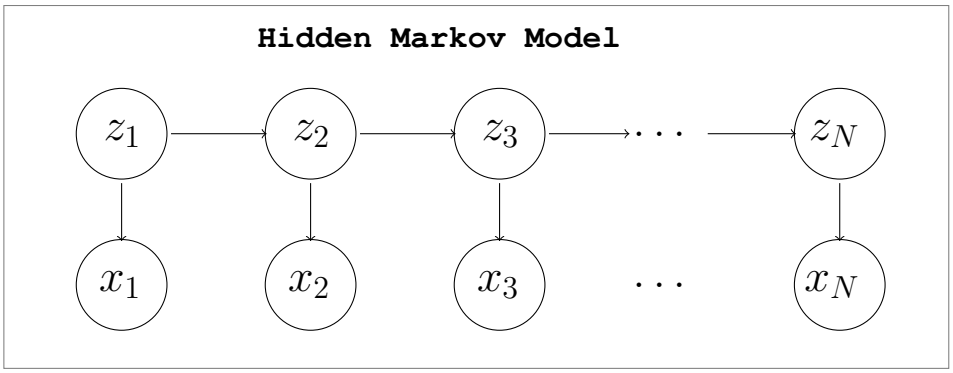


Figure 1.4: Hidden Markov Model, with latent variable z and conditionally independent observations x

In other words, given the previous observation x_{n-1}, an observation x_n is conditionally independent of all other previous observations. To capture longer-range interactions, this concept can be extended to higher orders, such that an observation is dependent on the previous M observations:

$$p(x_n|x_1, x_2, \ldots, x_{n-1}) = p(x_n|x_{n-1}, \ldots, x_{n-M}) \tag{1.7}$$

One way to tractably model time-series data is through the use of a state space model, in which each observation x_i has a corresponding latent variable or hidden state z_i associated with it. The latent variables $z_1, \ldots, z_n$ form a Markov chain and emit the observations $x_1, \ldots, x_n$ based on conditional distributions of the form $p(x|z)$. Figure 1.4 shows the graphical representation of a state space model. When the latent variables z in a state space model are discrete, we obtain the standard Hidden Markov Model (HMM). The standard HMM is defined by the number of states K that the latent variables can take on, a $K \times K$ transition probability matrix π (with rows π_k) that describes the probabilities $p(z_i|z_{i-1})$, and a parameterized distribution $F(\cdot)$ that describes the conditional probabilities $p(x_i|z_i)$. The generative model for an HMM can be written as:

$$z_i \sim \pi_{z_{i-1}}$$
$$x_i \sim F(\theta_{z_i})$$

where θ_{z_i} is a parameter vector associated with state z_i, and "$\sim$" can be read as "drawn from" or "distributed as". In other words, the HMM can describe time series data with a mixture model in which the latent mixture component indices have a temporal relationship as a first-order Markov chain. One drawback of the standard HMM is that the observation x_i is conditionally independent of any other observation x_j , given the generating hidden state z_i. This independence assumption

is clearly not well founded for much time-series data, such as robot demonstrations, in which the observations, as well as the hidden states, have temporal dependencies. The autoregressive HMM (AR-HMM) addresses this by adding links between successive observations, forming a Markov chain, as shown in Figure 1.5. This can also be extended to a Mth order AR-HMM, as shown in Figure 1.6.

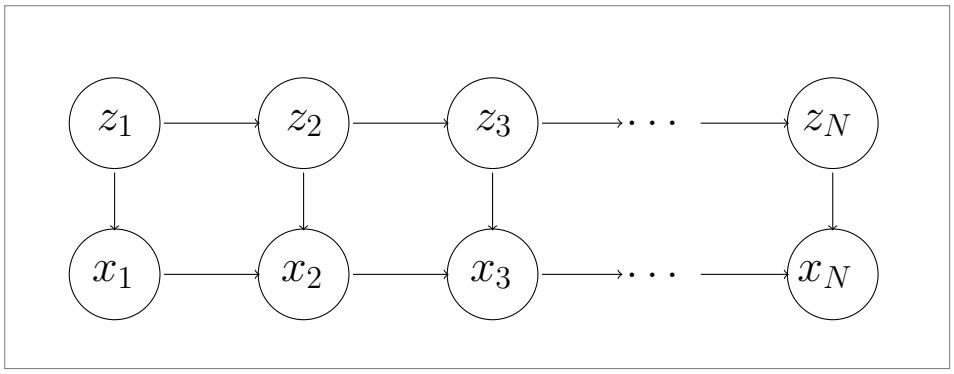


Figure 1.5: A first-order autoregressive HMM. Additional links can be added to make an M^{th}-order autoregressive HMM.

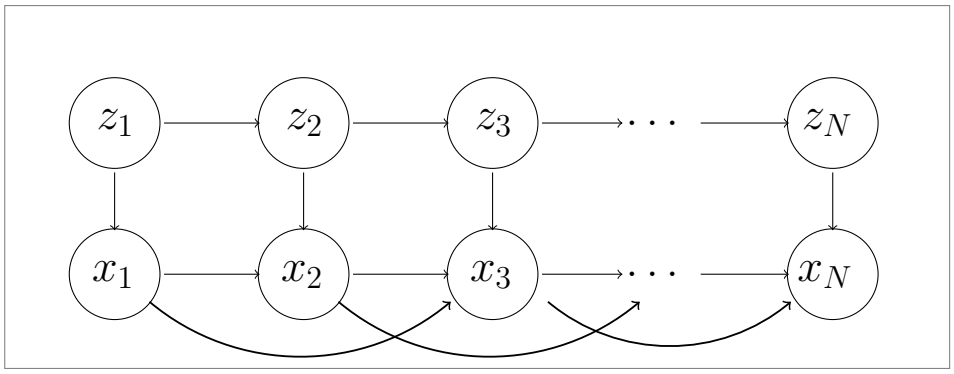


Figure 1.6: This example shows a second-order AR-HMM. Additional links can be added to make an M^{th}-order autoregressive HMM.

AR-HMMs face a problem in cases where the observations are not discrete - a simple transition matrix cannot be used to describe the probabilities $p(x_i|z_i, x_{i-1}, \ldots, x_{i-M})$. For example, demonstration data is often comprised of continuously valued state action pairs representing robot joint poses and actuations. In this case the conditional probability density function over x_i must be able to be written in terms of a continuous function of its predecessors. For example, a linear dynamical

system governing the conditional distribution can be used, such that:

$$p(x_i|z_i, x_{i-1}, \ldots, x_{i-M}) = \sum_{j=1}^{M} A_{j,z_i} x_{i-j} + e_i(z_i) \tag{1.8}$$

These fundamentals are used in our policy model to learn complex real world task from demonstrations.

Time series data available from demonstrations can be modeled using HMM, for e.g. [32] models the time series data from table assembly demonstration using HMM, in their case the latent variable of the HMM represents different skills (object manipulation, reaching a point in space etc.) with the corresponding joint positions and velocities representing the observations generated by these latent variables. These HMMs are limited by the fact that the number of latent modes need to be specified a priori. A principled way to overcome this limitation is to use a Bayesian non-parametric prior such as Beta Process (BP), or Hierarchical Dirichlet Process (HDP).

Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM) used by Niekum et.al. [32, 33], uses a Beta Process prior over the HMM such that each time series can exhibit a subset of the total number of discovered modes (latent skills) and switch between them in a unique manner. Moreover, Bayesian nonparametric prior allows the segmentation to be performed without the need for prior knowledge about the number or structure of skills involved in a task.

1.3.3 Preliminaries on Rational Canonical Structure and Jordan Decomposition

We take a geometric approach towards the choice of sampling locations for spatiotemporally evolving systems. This involves a major result in linear algebra related to rational canonical structure and Jordan canonical decomposition of a linear operator A [34]. Let $\mathcal{V}$ be a linear space over $\mathbb{R}$, with $\dim(\mathcal{V}) = M$, then $A : \mathcal{V} \to \mathcal{V}$ has a characteristic polynomial $\pi(\lambda)$ such that $\pi(A) = 0$ by the Cayley-Hamilton theorem. The minimal polynomial (MP) of A is the monic polynomial $\alpha(\lambda)$ of least degree (denoted by $\deg(\cdot)$) such that $\alpha(A) = 0$. The MP is unique and divides $\pi(\lambda)$, so that $\deg(\alpha) \leq M$. The MP of a vector $v \in \mathcal{V}$ (relative to A) is the unique monic polynomial ξ_v of least degree such that $\xi_v(A)v = 0$. If $\deg(\alpha) = M$, then A is said to be cyclic and there exists $c \in \mathcal{V}$, such that the vectors $\{c, Ac, \ldots, A^{n-1}c\}$ form a basis for $\mathcal{V}$; in the dual sense this is same as saying that the pair (c^T, A^T) is observable.

If $\deg(\alpha) < M$ and if MP of $c \in \mathcal{V}$ relative to A is $\alpha(\lambda)$, then the vectors $\{c, Ac, \ldots, A^{M-1}c\}$ span an M-dimensional A-cyclic subspace $\mathcal{V}_\mathcal{S}$ with cyclic generator c. The subspace $\mathcal{V}_\mathcal{S}$ decomposes $\mathcal{V}$ relative to A. The rational canonical structure theorem shows that A can be successively decomposed into subspaces $\mathcal{V}_i \subset \mathcal{V} (i \in \{1, \ldots, \ell\})$ with properties: $\mathcal{V} = \mathcal{V}_1 \oplus ... \oplus \mathcal{V}_l$, $A\mathcal{V}_i \subset \mathcal{V}_i$,

and $A|\mathcal{V}_i, (i \in \{1, \ldots, \ell\})$ is cyclic. The integer ℓ is unique and is called the cyclic index of A. One of our main results is to show that the cyclic index is a lower bound on the number of measurements required to reconstruct $w(t)$. A key relevant result in linear algebra is that any matrix $A \in \mathbb{R}^{M\times M}$ is similar to a unique block diagonal matrix Λ (i.e. $\exists P \in \mathbb{R}^{M\times M}$ invertible such that $A = P\Lambda P^{-1}$) whose diagonal blocks are matrices of the form

$$\Lambda_k(\lambda_i, \lambda_i^*) := \begin{bmatrix} M & I_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & I_2 \\ 0 & 0 & \cdots & M \end{bmatrix}. \tag{1.9}$$

where (λ_i, λ_i^*) is a complex conjugate eigenvalue of A, and $M = \begin{bmatrix} \mu_1 & \mu_2 \\ -\mu_2 & \mu_1 \end{bmatrix}$ and $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Real eigenvalues λ_i correspond to the case $M = \lambda_i$ and $I_2 = 1$. Thus the complete real Jordan form of A will be the appropriate diagonal array of these blocks. If all the eigenvalues λ_i are nonzero and real, we say the matrix has a *full-rank Jordan decomposition*.

1.3.4 Dirichlet Distribution for Transition Matrix (π)

For a K state Markov chain with uncertain transition matrix π, a well-known Bayesian approach [35], is to assume a prior Dirichlet distribution on each row π_m ($\pi_m = [\pi_{m,1}, \ldots, \pi_{m,K}]$) of the transition matrix π, and recursively update this distribution with observations. The Dirichlet density at any time τ defined over the simplex formed by the elements of the row π_m and hyper-parameters $\alpha(\tau) = [\alpha_1, \ldots, \alpha_K]$ (with each $\alpha_i > 1$) is given by

$$f_D(\pi_m|\alpha(\tau)) = \frac{1}{B(\alpha(\tau))} \prod_{i=1}^{K} \pi_{m,i}^{\alpha_i - 1}, \quad \sum_i \pi_{m,i} = 1 \tag{1.10}$$

where $B(\alpha(\tau)) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\alpha_0)}$ is a normalizing factor, where $\alpha_0 = \sum_i \alpha_i$. Dirichlet distribution being conjugate to multinomial facilitates posterior update of π_m based on the observed transitions. Moreover, the hyperparameters α_i can be interpreted as "counts", or times that a particular state transition was observed, this allows easy update to the distribution based on new observations. The mean and the variance of the Dirichlet distribution defined in (1.10) can be calculated as

$$\bar{\pi}_{m,i} = \alpha_i/\alpha_0$$
$$\Sigma_{ii} = \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)}$$

These are the mean and the variance of the i^{th} element of π_m, and is computed $\forall i$ and for all rows of π.

1.3.5 Mean-Variance Estimator

To update π_m one has to calculate the mean and variance at each time step. However, the Mean-Variance estimator developed by [36] allows recursive update for the mean and variance, in a form of predict and update step, written jointly (for brevity) as

$$\bar{\pi}_{m,i}(\tau+1) = \bar{\pi}_{m,i}(\tau) + \Sigma_{ii}(\tau)\frac{\delta_{i,i'} - \bar{\pi}_{m,i}(\tau)}{\bar{\pi}_{m,i}(\tau)(1-\bar{\pi}_{m,i}(\tau))} \tag{1.11}$$

$$\Sigma_{ii}^{-1}(\tau+1) = \gamma(\tau+1)\Sigma_{ii}^{-1}(\tau) + \frac{1}{\bar{\pi}_{m,i}(\tau)(1-\bar{\pi}_{m,i}(\tau))} \tag{1.12}$$

where $\gamma(\tau+1) = \frac{\bar{\pi}_{m,i}(\tau)(1-\bar{\pi}_{m,i}(\tau))}{\bar{\pi}_{m,i}(\tau+1)(1-\bar{\pi}_{m,i}(\tau+1))}$, and $\delta_{i,i'} = 1$, if $i = i'$ and is zero otherwise, refer [36] for proof and details. This estimator plays a key role in the evaluation of likelihood rate of a TPM.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Automation and Challenges in Construction Robotics

One of the earliest attempts at automating excavator was LUCIE by Seward et.al [37], a fully autonomous excavator. Their work used laser sensors to scan its surroundings and learn about its environment, but was not able to see objects completely above or below the sensors. Subsequently, automation of the excavation process has been studied with respect to trajectory planning, explicit modeling, and reactive strategies. However, LUCIE and other such attempts at autonomous excavator development [1–3,38,39] have suffered from limitations in perception, situational awareness, and safety. Other efforts [40,41] have focused on the optimization aspect for different tasks such as scooping and trenching. For example, Kim et.al. [40] developed technique for minimizing the requisite torque and time for excavator operation utilizing analytic gradient based motion optimization algorithms. Seo et.al. [41] developed a task planner to ensure that the planning is more reactive to real time changes. Different kinds of controllers and sensors have also been used to provide feedback for estimation of control parameters and tracking motions to get the optimal paths for excavation ([42–45]). Saeedi et.al. [43] developed Fuzzy logic based path tracking controller, while Jun et.al. [45] used fuzzy logic controllers to capture parameter uncertainties, external disturbance, and nonlinearity. However, most of these approaches encountered problems due to the limitations such as sensor range, field of view of the cameras, and the lack of human feedback.

On the other hand, construction industry has long been affected by high rate of workplace injuries and fatalities. More recently, safety has become an important concern for the construction industry, which recorded highest number of occupational fatalities according to the Census of Fatal Occupational Injuries (CFOI) report ([46]). Owing to the limitations discussed earlier, combined with, the safety critical nature of construction task, aforementioned research efforts into automating construction equipments have not seen the daylight. In addition to the safety concerns, shortage of skilled operators has been another significant concern in recent times due to the rapid increase in infrastructure projects. Job portals have numerous openings for construction equipment operators, a recent search on Jora Australia for “urgent excavator operator jobs”, alone had 900+ openings.

All such recent trends motivate our development of assistive autonomy in which the construction co-robots can assist in speeding up the training of novice operators. In the future, such an approach can also leverage the prior developments in automating construction equipments to share control with the operators to boost productivity when the operator follows the robot predicted trajectory [47], and relinquish control in off-nominal situations [48].

We discuss prior work in learning from demonstration and inverse reinforcement learning and point out the key ideas that we leverage in learning instructional policy model.

2.2 Robot Learning from Demonstration

Research in Learning from Demonstration (LfD) has focused on enabling robots to learn a policy for task execution from demonstrations provided by the human teacher. Considerable research efforts have been made in this field followed by some review and survey articles. First review of direct policy approximation methods from a computational point of view was by Schaal et.al. [7], it covered statistical and mathematical approaches that were used to tackle imitation learning problem. Billard et.al. [49] provides an overview of robot programming by demonstration with focus on incremental learning approaches that allow the robot to adapt the learned skill to different situations. A comprehensive survey of robot LfD, covering possible choices in terms of how teacher demonstrations are obtained, and various techniques for policy derivation, is given in Argall et.al. [5]. A more detailed survey into policy search techniques was given by Deisenroth et.al. [9]. While a detailed survey of reinforcement learning based approaches in robotics appeared in Kober et.al. [50]. In this thesis, we introduce and tackle a problem setting that requires the robot to learn from a human teacher a policy, that can be used to instruct or assist other operators. We leverage insights from the existing LfD literature in the development of our policy approximation approach that learns the instructional policy (π^I) to map the robot's state to an instruction for a human operator. The survey paper by Argall et.al. [8] provides a baseline for categorization of research work in LfD, with respect to the means of gathering demonstrations and policy derivation. There are different ways of gathering demonstrations based on various techniques for executing and recording demonstrations. We restrict to the case where teacher operates the robot platform (who is a learner) and the robot's sensors record the execution. There are cases where robot has to mimic the teacher motion and then record its sensor data.

Most of the existing research in LfD addresses the problem in which a robot learns a monolithic policy from a demonstration of a simple task that has a well-defined beginning and end, for example learning baseball [51], tennis swings [52], ball-in-the-cup [53], inverted helicopter hovering [54] etc. These approaches enable the learning of a single policy from data. While many approaches

enable the learning of a single policy from data, some approaches perform automatic segmentation of the demonstrations into simpler reusable primitives, followed by learning of policy for each primitive [11, 32, 55–57]. Such an approach is argued to allow recognition of repeated skills and its generalization to new settings. After segmentation into simpler task primitives, each segment is modeled using a policy representation such as, linear policies, radial basis function networks, neural network, Gaussian process or dynamic motor primitives ([52]).

Jenkins and Mataric introduced Spatio-Temporal Isomap in order to find the underlying low-dimensional manifolds within a set of demonstrated data [58]. This work extends the dimensionality reduction technique Isomap to include temporal information and allows the discovery of repeated motion primitives. However, segmentation is performed with a heuristic and the motion primitives cannot be improved through techniques like RL. Dixon and Khosla [59] demonstrate that generalizable motions can be parameterized as linear dynamical systems. This algorithm also uses heuristic segmentation and cannot recognize repeated instances of skills. Gienger et al. [60] segment skills based on co-movement between the demonstrator's hand and objects in the world and automatically find appropriate task-space abstractions for each skill. Their method can generalize skills by identifying task frames of reference, but cannot describe skills like gestures or actions in which the relevant object does not move with the hand.

More recent work has focused on using principled statistical techniques to perform automatic segmentation of demonstrations into simpler reusable primitives [11,33,55,56]. Such an approach is argued to allow recognition of repeated skills and its generalization to new settings. Incremental learning of subtasks by Grollman and Jenkins [11] uses a Chinese Restaurant Process (CRP) prior over subtask, allowing the possibility of assigning each new data point to an unseen subtask. Constructing skill trees approach by Konidaris et.al. [55] uses an online changepoint detection method to segment trajectories from multiple demonstrations and then merges the resulting chains of segments into a skill tree. But their approach cannot recognize repeated segments to assist with segmentation. Niekum et.al. [33] use the Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM) developed in Fox et.al. [61] to perform auto-segmentation of time series data from multiple demonstrations. Recent work by Figueroa and Billard [62] focuses on transform invariant learning of sub-tasks (referred to as action phases) from unstructured demonstrations. Their approach allows discovery of similar action phases that are performed in different geometrical transformations with respect to each other. In these methods, the segments or task primitives are defined in trajectory space, with no bounds on the number of possible segments. Moreover, these segments do not directly relate to the actuator space in which the human operates. Therefore in this paper, we first define segments as action primitives in actuator space, this definition is then utilized to perform segmentation of an unstructured demonstration. Moreover, our procedure does not rely on computationally inefficient Gibbs sampling for learning model parameters.

Aforementioned algorithms attempt to directly learn the policy; next we discuss reward based IRL techniques for LfD. We leverage key concepts from these two school of thoughts in our algorithm for learning instructional policy model.

2.2.1 Bayesian Nonparametric Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is an LfD technique concerned with finding hidden reward function of a human demonstrator from the demonstrated state and action samples [63], [64]. Recently, an approach to solve IRL by automatically decomposing the reward function into a series of subgoals, which were viewed as local reward functions, was proposed [65]. We utilize the notion of executing subgoals in a particular sequence to perform a task as a key component of our instruction policy model. This is based on research that deals with human expertise in complex environment [66], [67]. According to these findings, humans often form implicit decompositions of higher level tasks into several subgoals, so that the execution of each subgoal brings the task closer to completion.

In IRL [64] a Markov Decision Process (MDP) without the reward function $R(s)$ i.e. $MDP \backslash R$ is considered. A demonstration set O consists of state action pairs, $O = \{(s_1, a_1), \ldots, (s_N, a_N)\}$, where each pair $O_i = (s_i, a_i)$ indicates that the action a_i was performed from the state s_i. Multiple set of demonstrations are used as an input to the IRL algorithm to estimate the reward function $\hat{R}(s)$, such that the corresponding optimal policy π^* matches the observations. The IRL problem is ill-posed, since defining reward as, $\hat{R}(s) = c \; \forall s \in S$, would make any set of state-action pairs trivially optimal. Moreover, it is possible to encounter dissimilar actions from a particular state s_i. This ambiguity was resolved by restricting the reward function to be of certain form [68–70]. Later [71] developed a standard Bayesian inference procedure to learn reward function.

The IRL methods cited above attempt to explain the entire observations set O with a single, complex reward function, resulting in a large computational burden when the space of candidate reward functions is large. To overcome this limitation, Michini et. al. [65] developed Bayesian non-parametric IRL (BNIRL) that partitions the demonstration set O and explains each partition with a simple reward function or a "subgoal" $R_g(s)$, which consists of a positive reward at a single coordinate g in the state (or feature) space (zero elsewhere). Moreover, the candidate subgoal g is constrained to those states that were observed in demonstration. Thus the set of candidate reward functions scales with the size of demonstration set O (as opposed to an infinite set defined over the entire state space in other IRL methods). A Dirichlet process prior (Chinese Restaurant process (CRP) construction) is assumed over the unknown number of partitions that consists of observed state-action pairs $O_i \equiv (s_i, a_i)$. Partition assignment for each observation O_i is denoted by variable

z_i, and let $\boldsymbol{z}$ be the set containing all z_i's. Posterior over the partition assignment is given by

$$P(z_i|z_{-i}, \boldsymbol{O}) \propto \underbrace{P(z_i|z_{-i})}_{CRP} \underbrace{P(\boldsymbol{O}_i|R_{z_i})}_{likelihood} \tag{2.1}$$

where the first term denotes standard CRP prior with $z_{-i} = \boldsymbol{z} \backslash z_i$ being the set of all assignment variables except z_i. The second term evaluates the likelihood of the action a_i from state s_i given the subgoal reward function R_{z_i} corresponding to partition (or subgoal) identified by z_i. This likelihood term is evaluated using exponential rationality model [71]:

$$P(\boldsymbol{O}_i|R_{z_i}) = P(a_i|s_i, z_i) \propto e^{\alpha Q^*(s_i, a_i, R_{z_i})} \tag{2.2}$$

where the parameter α represents the degree of confidence in the demonstrator's ability to maximize reward. The evaluation of optimal action value function Q^* requires substantial computation and becomes infeasible for large state spaces. Hence, the author developed an approximation based on action comparison for the action likelihood (2.2), as follows

$$P(\boldsymbol{O}_i|R_{z_i}) = P(a_i|s_i, z_i) \propto e^{\alpha ||a_i - a_{CL}||_2} \tag{2.3}$$

where a_{CL} is the action given by some closed-loop controller attempting to go from state s_i to subgoal g_{z_i}. Note that z_i is a partition assignment variable for state s_i, if $z_i = k$, then g_k is the coordinate of k^{th} subgoal. This approximation to BNIRL [72], enables successful application of IRL to real-world learning scenario characterized by large state space such as quad-rotor domain. Another Bayesian non-parametric reward segmentation approach by [73], models skills as reward function instead of subgoal states, but relies on value iteration to determine likelihood. In this thesis, we propose and demonstrate a further simplified and computationally tractable BNIRL approach that performs clustering of states as opposed to state-action pairs, utilizing the Dirichlet process means approach that is discussed next.

2.2.2 Dirichlet Process Means for mixture model

Dirichlet process (DP) is a well known prior for the parameters of a Gaussian mixture model when the number of mixture components are not known a-priori. Recently Kulis and Jordan [74] have shown that the Gibbs sampling algorithm for the Dirichlet process mixture approaches a hard clustering algorithm when the covariances of the Gaussian variables tend to zero. This results in a

k-means-like clustering objective given by

$$\min \sum_{c=1}^{k} \sum_{x \in l_c} ||x - \mu_c||^2 + \lambda k$$
$$\text{where} \quad \mu_c = \frac{1}{|l_c|} \sum_{x \in l_c} x$$

where k is the number of clusters, μ_c is the mean for cluster c, and l_c is the set of data points x that belongs to the cluster c. This objective function is similar to that of k-means but includes a penalty λ for the number of clusters. DP-means algorithm behaves similarly to that of k-means with the exception that a new cluster is formed whenever a point is farther than λ away from every existing cluster centroid. To select λ, a simple farthest-first heuristic was found to be effective in all experiments, refer [74] for more details. We utilize this approach to cluster actuator velocities as well as the states observed in the demonstration set by assuming Gaussian distribution over their euclidean norm.

2.3 Modeling of Spatiotemporal Processes

Modeling of large-scale stochastic phenomena with both spatial and temporal evolution has been widely studied in Statistics and machine learning community. The data-driven models developed through machine learning techniques provide a way to model complex spatiotemporal phenomena that are not easily modeled by first-principles alone. Of the many techniques studied, kernel based techniques have seen increasing popularity [22, 75]. In these techniques, correlation between spatiotemporal variables is modeled through a covariance kernel, and the model is formed through a linear, weighted combination of the kernels. Also known as Kernel regression, the idea is to obtain estimates of the continuous dependent variable from a limited set of data points by convolving the data points' locations with the kernel function, approximately speaking, the kernel function specifies how to "blur" the influence of the data points so that their values can be used to predict the value for nearby locations. These techniques in spatiotemporal modeling focus on covariance kernel design and associated hyperparameter learning algorithms [25–27]. Next we elaborate the research in Geo-statistical as well as Machine learning community.

2.3.1 Non-stationary Kernel Methods

There is a large body of literature on spatiotemporal modeling with kernel-based methods [22,76]. A naive approach is to utilize both spatial and temporal variables as inputs to the kernel [77, 78]. However, this technique leads to an ever-growing kernel dictionary, which is computationally

taxing. Furthermore, constraining the dictionary size or utilizing a moving window will occlude the learning of long-term patterns. Periodic or nonstationary covariance functions and nonlinear transformations have been proposed to address this issue [23, 26]. Work focusing on nonseparable and nonstationary covariance kernels seeks to design kernels optimized for environment-specific dynamics, and to tune their hyperparameters in local regions of the input space. Seminal work in [79] proposes a process convolution approach for space-time modeling. This model captures nonstationary structure by allowing the convolution kernel to vary across the input space. A class of non-stationary covariance functions for Gaussian process (GP) regression (spatial modeling) was first introduced by [80]. They show that the covariance for Gaussian kernels takes the following form

$$cov\{Z(s_i), Z(s_j)\} = k(s_i, s_j) = \sigma^2 |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} |(\Sigma_i + \Sigma_j)/2|^{-\frac{1}{2}} exp(-Q_{ij}) \tag{2.4}$$

with the quadratic form

$$Q_{ij} = (s_i - s_j)^T ((\Sigma_i + \Sigma_j)/2)^{-1} (s_i - s_j), \tag{2.5}$$

where Σ_i, is the hyperparameter matrix of the Gaussian kernel at s_i. The evolution of the these matrices in space produces non-stationary covariance. However, model's hyperparameters are inferred using MCMC integration, its application has been limited to smaller datasets. To overcome this limitation, [27] proposes to use the mean estimates of a second isotropic GP (defined over latent length scales) to parameterize the nonstationary covariances. Finally, [25] considers non-isotropic variation across different dimension of input space for the second GP as opposed to isotropic variation by [27]. In this design they defined a generic non-stationary non-separable covariance function as

$$cov\{Z(s_i, t_i), Z(s_i, t_j)\} = |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} |(\Sigma_i + \Sigma_j)/2|^{-\frac{1}{2}} exp(-\sqrt{Q_{ij}^s Q_{ij}^t}) \tag{2.6}$$

where Q_{ij}^s can be defined in accordance to equation (2.5). The hyperparameters of these spatiotemporal covariance kernel and the weights are learned by solving an optimization problem, or through Bayesian inference techniques.The benefit of careful design of covariance kernels is that they can account for intricate spatiotemporal couplings. However, the key challenge with this approach is in ensuring the scalability of the model to large scale phenomena and manifests due to the fact that the hyperparameter optimization problem is not convex. Moreover the selection of an appropriate nonstationary covariance function for the task at hand is a nontrivial design decision (as noted in [81]).

2.3.2 Geostatistical Approach

On the other hand, Geostatistical approach to modeling spatio-temporal data rely on selection of appropriate space-time covariance functions. Selection of covariance function is based on the assumptions of stationarity, separability and full symmetry. Typically observations are modeled as Gaussian random function

$$Z(s,t), \qquad (s,t) \in \mathbb{R}^d \times \mathbb{R} \tag{2.7}$$

which is indexed in space by $s \in \mathbb{R}^d$ and in time by $t \in \mathbb{R}$. As per the definitions given in [82], the random field Z is said to have separable covariance if there exist purely spatial and purely temporal covariance functions cov_S and cov_T, respectively, such that

$$cov\{Z(s_1,t_1), Z(s_2,t_2)\} = cov_S(s_1,s_2) \cdot cov_T(t_1,t_2) \tag{2.8}$$

for all space-time coordinates (s_1,t_1) and (s_2,t_2) in $\mathbb{R}^d \times \mathbb{R}$. The condition of full symmetry implies separability and is fulfilled if

$$cov\{Z(s_1,t_1), Z(s_2,t_2)\} = cov\{Z(s_1,t_2), Z(s_2,t_1)\} \tag{2.9}$$

for all space-time coordinates (s_1,t_1) and (s_2,t_2) in $\mathbb{R}^d \times \mathbb{R}$. Stationarity can be spatial, if $cov\{Z(s_1,t_1), Z(s_2,t_2)\}$ depends on the observation sites s_1 and s_2 only through the spatial separation vector, $s_1 - s2$. Temporally stationary covariance implies $cov\{Z(s_1,t_1), Z(s_2,t_2)\}$ depends on the observation times t_1 and t_2 only through the temporal lag, $t_1 - t_2$. Covariance function design usually involves checking for the conditions of stationarity, separability and symmetry of the observations to select an appropriate positive definite covariance function. Gneiting et. al. [82], claim through their experiments, that a complex and physically realistic correlation and covariance models results in improved predictive performance. Their approach utilizes particular type of stationary or symmetric covariance function based on their analysis of the data set at hand (e.g. evaluating correlation among different measurement locations). However such an approach is limited by the availability of physical insights as well as the comprehensive data analysis which guides the design. Most importantly, they are not robust to previously unobserved evolution.

2.3.3 Sensor Placement

The problem of placing a small number of sensors with the goal of making accurate global predictions about a spatial field with temporal variation, has received attention in past few years. A well known approach for near-optimal sensor placement was developed by [83]. Their work assumes a Gaussian process model over sensor data obtained from a given spatial phenomena, specifically

they utilize complex non-stationary covariance function of [84]. Given a GP model, sensor placement is obtained by optimizing mutual information between the set of sensor locations $\mathcal{A}$ and the rest of the space $\mathcal{V}\backslash\mathcal{A}$. This problem is shown to be NP-complete and hence [83] proposed a greedy algorithm that selects sensors in sequence such that a chosen sensor y provides maximum increase in mutual information given by

$$\delta_y = MI(\mathcal{A} \cup y) - MI(\mathcal{A}) = \frac{\sigma_y^2 - \Sigma_{y\mathcal{A}}\Sigma_{\mathcal{A}\mathcal{A}}^{-1}\Sigma_{\mathcal{A}y}}{\sigma_y^2 - \Sigma_{y\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1}\Sigma_{\bar{\mathcal{A}}y}} \tag{2.10}$$

where $\bar{\mathcal{A}} = \mathcal{V}\backslash\mathcal{A}$, $\Sigma_{y\mathcal{A}}$ denotes $k(y, \mathcal{A})$ for abbreviation. Thus a sensor $y^* \leftarrow \arg\max_{y\in\mathcal{V}\backslash a}\delta_y$ is selected. Mutual information is shown to be a submodular function, this property is then used to show that the greedy algorithm results in near optimal (approximately 63% of optimal) sensor placement. However, this approach does not account for temporal evolution of the spatial phenomenon while selecting the sensors, and thus nothing can be concluded about the optimality of sensor placement in that regime. Moreover, their approach seems limited to cases where one can design and select appropriate non-stationary covariance function as the reported results show poor performance for stationary covariance function. To overcome this problem, we approach sensor placement through a holistic spatiotemporal model that derives the placement using the model dynamics.

Bayesian optimization approach proposed by [85], first defines an objective function that measures the efficacy of a particular sensor layout, and builds a corresponding Gaussian process for the observed function values. The GP is then used to make predictions for possible sets of sensor layouts and selects the set that minimizes the objective function value. The approach suffers from explore exploit dilemma, and also requires selection of appropriate covariance function for the GP. Their approach does not cater to the question of what number of sensors should be selected in order to achieve optimal accuracy in prediction.

Joshi and Boyd [86], describe a heuristic based on convex optimization, for approximately solving the problem of choosing a set of k sensor measurements, from a set of m possible or potential sensor measurements, that minimizes the error in estimating some parameters. Thus the literature surveyed assumes selection of k sensors from a set of m potential sensor locations. The question whether these k sensors ensure optimal prediction is not answered. As per our knowledge, this question has never been addressed in the literature. Our aim is to provide a non-conservative lower bound on the minimum number of sensors required to ensure optimal prediction using the available observation data. Further we propose to develop methods for sensor selection with theoretical guarantees for making predictions.

CHAPTER 3

CONSTRUCTION ROBOTS: LEARNING INSTRUCTIONAL POLICY MODEL

3.1 Problem Formulation

We consider a problem in which a robot learns to train human operators of construction equipment to perform a task. Typically, in a learning from demonstration setting, a robot utilizes policy search methods ([9]) to learn a policy $\pi : s \rightarrow a$, mapping its current state s, to an action a. In contrast, the present problem has three key aspects, first the robot should map its current state to an instruction for a human operator, second the robot should learn a generative model of policy to train humans in performing an entire task, and third is the interface between the robot and the human to communicate the instructions. We define the first aspect of this problem as learning instructional policy, $\pi^I : s \rightarrow \boldsymbol{i}$, that maps the state s, to an instruction $\boldsymbol{i}$, based on the state-action pairs observed in a demonstration trajectory $O = \{(s_i, a_i)\}_{i=1}^{N}$, and the locations of all the M number of manipulated objects, given by m_j, $j \in \{1, \ldots, M\}$ in the robot's base frame. As a solution, we introduce action primitives to enable the mapping of continuous state action trajectories to an instruction $\boldsymbol{i}$ in the human operator's action space. Further it is desired that the valid domain $\Omega_O \subset \Omega$, (Ω being the set of all possible states) for the function π^I is much larger than the set of states in O.

For the second aspect, we seek a holistic policy representation for a complex task composed of several subtasks. We propose to learn a generative policy model of an entire task that will sequentially generate policy π^I to train humans on a task in a cyclic manner. Section 3.2 elaborates these two aspects and the learning process of the instructional policy model which is demonstrated on an excavator robot in the experiment sections 3.4.1- 3.4.2. For the third aspect, we hypothesize two fundamental instruction interface for robots to train humans in section 3.3, these are: (i) visually depicting the instructional policy; (ii) generating positive reinforcement for desired actions. These hypotheses are evaluated through exhaustive experiments on a 1/14th scale hydraulic actuated excavator model in sections 3.4.3-3.4.4.

3.2 Learning Instructional Policy Model from Demonstration

Our objective is to learn a holistic policy representation of a complex task comprised of several subtasks, with each subtask being a sequence of primitive segments (or skills). While doing so our main goal is to learn instructional policy that maps the robot's state to instructions in human operator's action space. Thus, the goal of the proposed instructional policy model is to achieve a single policy representation for complex tasks, and to generate instructions to train human operators. This is accomplished by using action primitives to decompose a task into human executable actions or segments and then use a structured probabilistic approach to model their sequence. Overview of the learning process is given in Figure 3.1. In the first step, trajectories are segmented into constituent action primitive segments. Generated segments are input to the subgoal learning algorithm. Finally, subgoals and action primitive segments are utilized in the construction of instructional policy model. We discuss the three key elements of this model, the action primitives, the segmentation of demonstration trajectories and the subgoal learning in the following three subsections 3.2.1-3.2.3, followed by a subsection 3.2.4 on the construction of the instructional policy model.

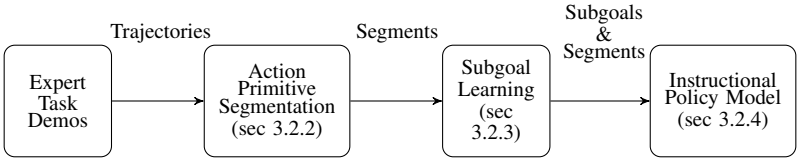
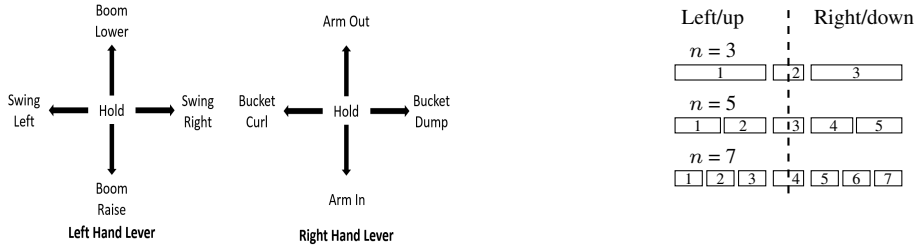


Figure 3.1: Robot learns instructional policy model from expert demonstrations.



(a) Joystick Controls

(b) Discretization

Figure 3.2: Action Space Decomposition: (a) Joystick lever movements for an excavator: Human operator's action space, (b) Levels of Discretization of joystick movements denoted by n.

3.2.1 Action Primitives

For a human to execute the instructional policy $\pi^I : s \rightarrow \boldsymbol{i}$, the instruction $\boldsymbol{i}$ has to be defined in the human operator's action space. An example of human operator's action space is shown in

Figure 3.2a, where an instruction would indicate the direction and the extent of joystick deflection. Human operator can act based on such an instruction $\boldsymbol{i}$ to generate desired movements for the robot. We define *action primitives* as a key to relate sampled joint velocities (i.e. the action a_i's) to an equivalent instruction in the operator's action space. Note that by sampled joint velocities we imply the velocities measured at fixed intervals of time during a demonstration.

Sampled joint velocities are directly related to the joystick position. Rather a finite interval of joint velocities is a result of a particular joystick position. We propose to segment continuously sampled joint velocities on the basis of significant change in joystick positions. The resulting segments are defined as action primitive segments. Most simple level of change in joystick control is the three categories: i) move to the left, ii) hold neutral position, and iii) move to the right, denoted by $n = 3$ in Figure 3.2b. Without loss of generality, these joystick positions produce equivalent states, such as: i) counterclockwise rotation, ii) no movement (or non-zero noisy perturbation), and iii) clockwise rotation. Thus, for a robot with one revolute joint, sampled joint velocities can be segmented into segments identified by three different action primitives denoted as $\mathbf{r} \in \{1, 2, 3\}$.

Further levels of change in joystick positions (i.e. $n > 3$) is obtained by dividing i) and iii) into more bands as depicted by $n = 5, 7$, in Figure 3.2b. These levels split rotation speeds in both directions into two or more bands, and is an indicator of higher level of precision used in an operation. Note, that these bands need not be uniform as shown, rather they might differ in their spans. For two or more joints, an action primitive $\mathbf{r}$ is a vector and we let r_j denote its j^{th} joint's component. For a given n, each component, $r_j \in \{1, \ldots, n\}$. For a robot with four revolute joints denoted as $m = 4$, choosing- $n = 5$ for all joints, $\mathbf{r} = [3, 1, 3, 3]$ is an example of action primitive that is a result of joystick position in the band given by $r_2 = 1$ for the second joint, and in the neutral band $r_1, r_3, r_4 = 3$ (see Figure 3.2b) for other joints. In the next section, we learn each joystick band as a Gaussian distribution over sampled velocities. Thus, each component of an action primitive represents a Gaussian distribution, i.e. $r_j = i$, has an associated Gaussian distribution $\mathcal{N}(\mu_{ji}, \sigma_{ji}^2)$, with mean μ_{ji}, and variance σ_{ji}^2, for each joint j, and each $i \in \{1, \ldots, n\}$. An instruction, $\boldsymbol{i}$, is a result of samples from each component which is then mapped to joystick position. Mathematically, an action primitive, $\mathbf{r}$, defines the distribution over joystick movement instruction for a human operator, given by the variable- r_j for each joint j.

3.2.2 Action Primitive based Segmentation

In this section, we utilize the definition of action primitives to segment demonstration trajectories into a sequence of action primitive segments. To identify the action primitives, we first cluster the sampled velocities for each joint into n clusters and obtain $n-$Gaussian distributions from cluster members. Note, the number of clusters n is not known a-priori. Let $\boldsymbol{v}_\tau \in \mathbb{R}^{m \times 1}$ denote

the sampled velocity at time τ, with $v_{j\tau}$ the velocity of joint j, and $v_{j.}$ be the set of all sampled velocities $\{v_{j1}, v_{j2}, \ldots, v_{jT}\}$ of the joint j, sampled at time instants $\tau = 1, 2, \ldots, T$. We utilize DP-means algorithm (see section 2.2.2) to cluster velocities in the set $v_{j.}$. To obtain the penalty parameter λ for the algorithm, we input an approximate number of desired clusters, $k = n'$. DP-means algorithm then generates n clusters for r_j. The value of n governs different levels of speed required to do a task. In our experiments, we set the initial guess n' (for n) based on the different levels of speed (or joystick control) used by an expert, determined visually from the speed plots.

An example of DP-means clustering for the four joints of the excavator robot is shown in Figure 3.7. From the cluster members generated by DP-means, we obtain the mean μ_{ji} and the variance σ^2_{ji}, for each cluster $i \in \{1, \ldots, n\}$, and for each joint $j \in \{1, \ldots, m\}$. An example of action primitive for $m = 4$, and $n = 5$ for each joint, is $\mathbf{r} = [2, 3, 3, 3]$. Such an action primitive $\mathbf{r}$, over a finite interval of time implies that all the observed joint velocities can be modeled under the normal distribution $\mathcal{N}(\mu_{12}, \sigma^2_{12})$ for joint 1, $\mathcal{N}(\mu_{23}, \sigma^2_{23})$ for joint 2 and so on.

Next, we describe how classification of sampled joint velocities into action primitives generates the segmentation of demonstration trajectories. Each segment will then be an action primitive that spans over finite instants of time. At any time instant τ, we define the probability for each element of the action primitive $r_{j\tau}$ as,

$$p(r_{j\tau} = i | v_{j\tau}) = \mathcal{N}(v_{j\tau} | \mu_{ji}, \sigma^2_{ji}) \tag{3.1}$$

These probabilities can be normalized to obtain, $\sum_{i=1}^{n} p(r_{j\tau} = i | v_{\tau j}) = 1$. We then assign $r_{j\tau}$ to the neutral band if that probability exceeds a threshold otherwise it is assigned to a band with highest probability, mathematically,

$$r_{j\tau} = \begin{cases} (n+1)/2 & p(r_{j\tau} = r_{db}) > \eta \\ \ell & \ell = \arg\max_i p(r_{j\tau} = i | v_{\tau j}) \end{cases} \tag{3.2}$$

where η is perturbation threshold to isolate noisy perturbation around neutral position denoted as r_{db}. This procedure is repeated for each joint j to obtain the action primitive $\mathbf{r}_\tau$ at each time instant τ. A key advantage of this procedure is in isolating inherent noise in demonstrations caused by numerous sources, such as unintended joystick motion, inertial movement of the actuator, noise in the measurement sensor, etc. For example, in Figure 3.7d the noisy perturbation band isolates movement of the bucket that takes place under load and not due to intended actuation by the operator.

Continuous repetition of an action primitive $\mathbf{r}_k$, starting at an instant τ_1 and repeating over an interval of finite time instants, say n_k, i.e. the set of instants $\{\tau_1, \tau_2, \ldots, \tau_{n_k}\}$, is defined as an action primitive segment $\mathcal{A}_k$. Thus, the process of classifying sampled velocities (i.e. the ac-

tions a) into action primitives generates action primitive segments, each comprising of a unique action primitive observed continuously over finite interval of time. An example segmentation (discussed later in section 3.4.1) is shown in Figure 3.8a, where each colored segment is an action primitive segment. Each action primitive segment $\mathcal{A}_k$ has an associated end-effector pose s_k at its beginning, and is hence defined by the pair $\mathcal{A}_k \equiv (s_k, \mathbf{r}_k)$. Thus, using segmentation procedure, we decompose the entire demonstration trajectory into a set of action primitive segments, $O_{\mathcal{R}} = \{(s_1, \mathbf{r}_1), (s_2, \mathbf{r}_2), \ldots, (s_N, \mathbf{r}_N)\}$, where N is the total number of segments observed in a demonstration. Let $n_1, n_2, \ldots, n_N$ be the corresponding number of sampling instants over which each action primitive is observed and $n' = \min\{n_1, n_2, \ldots, n_N\}$, then $\frac{\sum_{i=1}^{N} n_i}{N} \geq n'$. Note, $\sum_{i=1}^{N} n_i$ are the total number of state-action pairs, this implies that learning policy over state-action primitive pairs results in a reduction of data size by a factor of at least n'.

In general, our approach will ensure at least an order of magnitude reduction in data size for applications in which action primitives lasting less than t seconds are noisy perturbations and the sampling rate is at least $10/t$. For example, a sampling rate of at least $10Hz$ with a shortest valid action primitive lasting for a second, would imply $n' \geq 10$, i.e. an order of magnitude reduction. Learning policy using state-action primitive pairs can certainly improve the scalability of an approach by at least an order of magnitude. Thus, the definition of action primitive serves the dual purpose of discretizing continuous demonstration trajectory into finitely many action primitive segments and of a generative distribution for instructions in the human operator's action space. For latter, it is assumed that a map between actuator velocity and joystick position is known. Usually such a map is not linear, and might as well depend upon other variables. In such cases, nonlinear regression such as Gaussian processes can be utilized to learn the mapping.

Each action primitive segment $\mathcal{A}_k$ is defined by the robot's state at the start s_k and the action primitive $\mathbf{r}_k$. All such states $s_k \in O_{\mathcal{R}}$, are key states of robot's pose at which the human teacher takes significantly different joystick action, and are hence utilized to learn subgoals in the next subsection. Thus, the concept of action primitives enables efficient segmentation, which then serves as a means to learn subgoals and ultimately a holistic policy representation. The set $O_{\mathcal{R}}$ is used as an input to learn subgoals.

3.2.3 Learning Subgoals

Research in behavioral psychology [66, 67] have shown that the expert operators are good at decomposing a given ill-defined task into a series of actionable subgoals. Such decomposition is implicit within the human mind, and humans are not always able to clearly explain how they arrived at the decomposition. Nevertheless, an expert utilizes such decomposition to train others. The key idea that we leverage here is to decompose complex tasks into subgoals. We propose

Dirichlet process means based inverse reinforcement learning (DPMIRL) formulation: a reward based partitioning of task space into subgoals. This decomposes a complex tasks into subgoals such that the series of instructions could be used to navigate an operator through those subgoals. We define subgoals as the goal points in the robots trajectory space that are traversed in a sequence to accomplish a task.

BNIRL (Section 2.2.1) utilizes a Dirichlet process prior over state-action pairs observed in demonstration. We propose DPMIRL, a novel algorithm that utilizes Dirichlet process means method for clustering states observed in action-primitive segments, to partition the state-action pairs. For clustering, euclidean distance metric ($||\cdot||_2$) is defined for the states observed in action-primitive segments using appropriate coordinate frame of reference. Typically in an IRL problem, entire state space is modeled as an MDP, whereas in the present setup, set of states is reduced to the finite number of action-primitive segments obtained using action primitive based segmentation of demonstration trajectories. This step significantly reduces the computational burden, and facilitates the analysis of high dimensional continuous state action spaces feasible. In contrast, the BNIRL approach deals with the curse of dimensionality by using action comparison (equation (2.3)) for the likelihood term in equation (2.1). Next we show that how action comparison can generalize to pose-error comparison which further reduces computational requirements.

In action comparison the likelihood of an action a_i w.r.t to a subgoal g_{z_i} is computed as $P(a_i|s_i, z_i) \propto e^{\alpha||a_i - a_{CL}||_2}$. Due to this step the evaluation of computationally intensive optimal action value function Q^* is not required. As proposed in [72], a_{CL} is the action of a closed-loop controller that attempts to go from state s_i to subgoal g_{z_i}. A simple controller that would generate an action a_{CL} to reduce the pose error between the subgoal g_{z_i} and the present state s_i is $a_{CL} \propto (g_{z_i} - s_i)$. We argue that a demonstrator is also invariably reducing the pose error between the state s_i and the subgoals, thus the action a_i in the demonstration is also proportional to the pose error. Hence given a particular subgoal g_{z_i}, even $a_i \propto (g_{z_i} - s_i)$. Applying these arguments to equation (2.3) we obtain

$$P(a_i|s_i, z_i) \propto e^{\alpha||\kappa(g_{z_i} - s_i) - \lambda(g_{z_i} - s_i)||_2} \tag{3.3}$$

$$\propto e^{\alpha'||(g_{z_i} - s_i)||_2} \tag{3.4}$$

where we let κ and λ to be scalar proportionality constants, and thus the original action comparison reduces to pose error comparison between the state s_i and the subgoals. This result is very intuitive in the sense that any state-action pair (s_i, a_i) (or action primitive segement $(s_i, \mathbf{r}_i)$ for the case of DPMIRL) will be partitioned or assigned to the closest subgoal. Each subgoal is a mean of member states, thus any action primitive $\mathbf{r}_i$ from a state s_i, is likely to either attain the assigned subgoal or recede away towards the next. Though counter-intuitive, this property proves useful in generating instructions to navigate sequence of subgoals. Action primitive segments in the set

$O_{\mathcal{R}}$ are partitioned using euclidean distance metric on state locations i.e. $||s_i||_2$ for a state s_i. This is performed using DP-means algorithm discussed in section 2.2.2, that results in action primitive segments classified into clusters $\{c_1, \ldots, c_l\}$ where the number of clusters l is not known a-priori. Each cluster c_j consists of member segments $\{(s_i, \mathbf{r}_i) \in c_j : z_i = j\}$. We define the subgoal as a multivariate Gaussian $X_j \sim \mathcal{N}(\boldsymbol{\mu_j}, \Sigma_j)$ in n-dimensional space (formed by end-effector position and mechanism) whose parameters mean vector $\boldsymbol{\mu_j}$, and covariance matrix Σ_j are obtained from the member states of cluster c_j.

Algorithm 1 Subgoal Learning

Input: Set of action primitive segments $O_{\mathcal{R}}$, M objects of interest centered at m_j
- Assign each state $s_i \in O_{\mathcal{R}}$ to S_k s.t. $k = \arg\min_j ||s_i - m_j||_2$, here $k \in \{1, \ldots, M\}$
- Run DP-means for states in each S_k to obtain M set of clusters, where any set k is given as $C_k = \{c_{k_1}, \ldots, c_{k_l}\}$
- Set of subgoals $X = \emptyset$, $p = |X| = 0$
for each set of cluster C_k, $k \in \{1, \ldots, M\}$ **do**
  for each cluster i in C_k, $i \in \{1, \ldots, k_l\}$ **do**
    - Compute mean μ_{k_i} and variance Σ_{k_i} from the member states $s \in c_{k_i}$
    - Add $X_p \sim \mathcal{N}(\mu_{k_i}, \Sigma_{k_i})$ to set X, increment p,
  end for
end for
Output: Set of subgoals X.

To ensure that the policy model of a given task, is generalizable to any novel configuration we learn subgoals w.r.t each known object in the task. To do so, the euclidean norm is computed w.r.t the coordinate frames centered on each of the object in the task. Let M be the number of objects and m_j be the center of the j^{th} object's coordinate frame w.r.t the base frame of the robot. We group the states observed in the set $O_{\mathcal{R}}$ into M disjoint sets $\{S_1, \ldots, S_M\}$, where a state s_i is assigned to S_k if $k = \arg\min_j ||s_i - m_j||_2$, note $k \in \{1, \ldots, M\}$. We run DP-means separately for each set S_k, to obtain subgoals as described in Algorithm 1. For each identified subgoal X_i we assign member action primitive segments to the set $\mathcal{S}_i$. Defining subgoals as a multivariate Gaussian allows us to evaluate the likelihood of any state s in space w.r.t to each subgoal, to determine the most likely subgoal, followed by the selection of a sequence of action primitives that generates instructions to reach the next subgoal, this is discussed in next subsection.

3.2.4 Instructional Policy Model

With the goal of constructing instructional policy model for a complex task as subgoal transitions (or subtask execution) resulting from a series of instructions (or skills in LfD setting) emitted by the action primitives, we used a hierarchy of Markov chains also known as a dynamical Bayesian

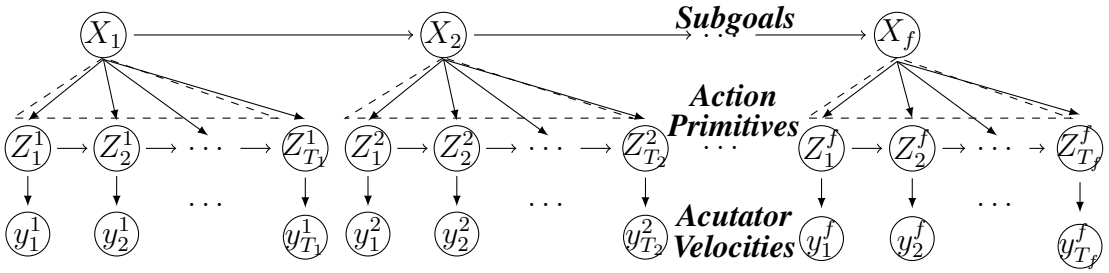


Figure 3.3: Policy model as a hierarchy of Markov chains to associate the high level task context defined by the subgoals, with the low level action primitives. Action primitives are sampled to obtain actuator velocities. Actuator velocities can either be mapped to joystick position to generate an instruction for a human operator or used as a policy for task replay.

network shown in Figure 3.3. The transition between subgoals is modeled by the topmost Markov chain. In this construction, we utilize a fact that each subgoal X_i has an associated set $\mathcal{S}_i$ of state-action primitive pairs, and there exists a Markov chain of latent modes $\{Z_1^i, \ldots, Z_{T_i}^i\}$, where each mode is an action primitive at time instants $\tau = 1, \ldots, T_i$, that generates instructions resulting in a translation to the next subgoal X_j. Hence the Markov chain under each subgoal X_i models the transition among the action primitives associated with that subgoal. These transition models for action primitives are obtained from the segmentation of demonstration data and counting the transitions between the action primitive segments. The final layer of the model is actuator velocity variable y_τ^i that is conditional on the action primitive Z_τ^i, and is modeled by Gaussian distributions over the sampled actuator velocities contained in an action primitive segment as discussed in the action primitives section 3.2.2.

At any given time instant τ, an instruction is inferred based on the robot's sate (s_τ) and the previous action primitive as follows. Given the computed most likely subgoal X_i for the current state s_τ and the previous action primitive $Z_{\tau-1}^i = \mathbf{r}_{k-1}$, instructional policy model is a generative model (Figure 3.3) to get the next action primitive $Z_\tau^i = \mathbf{r}_k$ and to sample actuator velocity to generate instruction (i.e. to generate π^I) as follows:

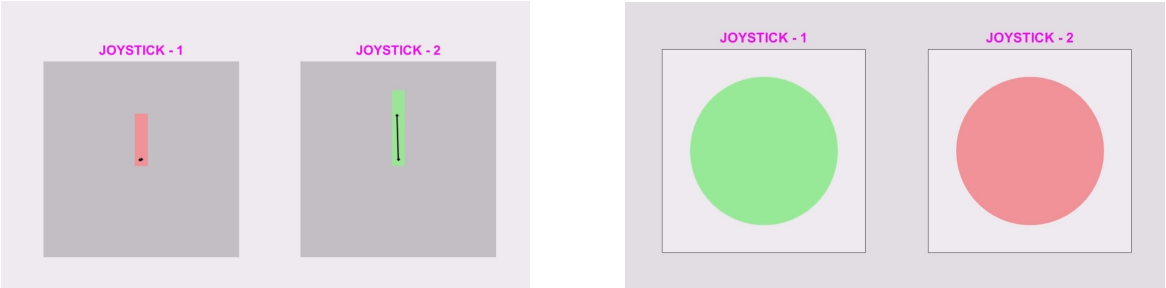
$$P(X_{i+1}|X_i) \sim \Pi \tag{3.5}$$

$$P(Z_\tau^i|Z_{\tau-1}^i, X_i) \sim \pi(X_i|X_{i+1}) \tag{3.6}$$

$$P(y_\tau^i|Z_\tau^i) \sim F(\theta_{Z_\tau^i}) \tag{3.7}$$

and the parameters for this model are the transition distribution Π for subgoals, the subgoal specific transition model $\pi(X_i|X_{i+1})$ for the action primitives associated with the subgoal X_i to achieve the next subgoal X_{i+1}, and the parameter vector $\theta_{Z_\tau^i}$ that models conditional distribution of actuator velocities given the action primitive. Actuator velocity sampled in equation (3.7) is mapped to

the joystick position to obtain an instruction $\boldsymbol{i}$, that is communicated to the human operator. In this article, we assume a hard-coded map between actuator velocities and joystick position to be known. Otherwise, a non-linear mapping function that accounts for complex relationship between actuator velocities and joystick positions especially during multiple actuations should be learned using a neural network, or a Gaussian process or other valid approaches. Note, at every transition to a new subgoal X_i, initial distribution $\pi_0(X_i)$ over action primitives is used to select the first action primitive associated with X_i. At any instant, likelihood of the robot's current state s is evaluated w.r.t each subgoal X_i using $P(s|\mathcal{N}(\mu_i, \Sigma_i))$. Thus, an instruction is generated unless this likelihood evaluates to zero (or $< \epsilon$, a very small number as it can never be exactly zero) for all the subgoals. Mathematically, the valid domain, Ω_O is given by the set of those states $s \in \Omega$, that satisfy, $\min_{X_i} P(s|X_i \sim \mathcal{N}(\mu_i, \Sigma_i)) > \epsilon$, with $\epsilon = 10^{-10}$. Clearly, the valid domain Ω_O for π^I is much larger than the set of states in O, as desired. Next section discusses two different interfaces used by the robot for communicating instructions generated by π^I to humans.



(a) GUI Bars: Complete Instruction

(b) GUI Circles: Positive Reinforcement

Figure 3.4: Visual interfaces for instructing humans, (a) Instruction depicted in terms of direction and extent of joystick motion, with thin black line indicating actual joystick position, (b) Color of the circle w.r.t a joystick changes from red to green if the predicted instruction is executed by the operator.

3.3 Instruction Interface

We evaluate two key and fundamentally different hypotheses to communicate instructions generated by the instructional policy model. Since our study is designed to look at the construction operators who normally work under noisy conditions, we chose to focus on visual interfaces. Moreover, research ([14,15]) have shown that videos are effective at teaching tasks and problem solving strategies in both children and adults. Interfaces were designed to look at operant conditioning and visual reinforcement in the robot-human interaction, but, more importantly, to determine which version of the visual feedback would result in retention of information and learned skills. Our first

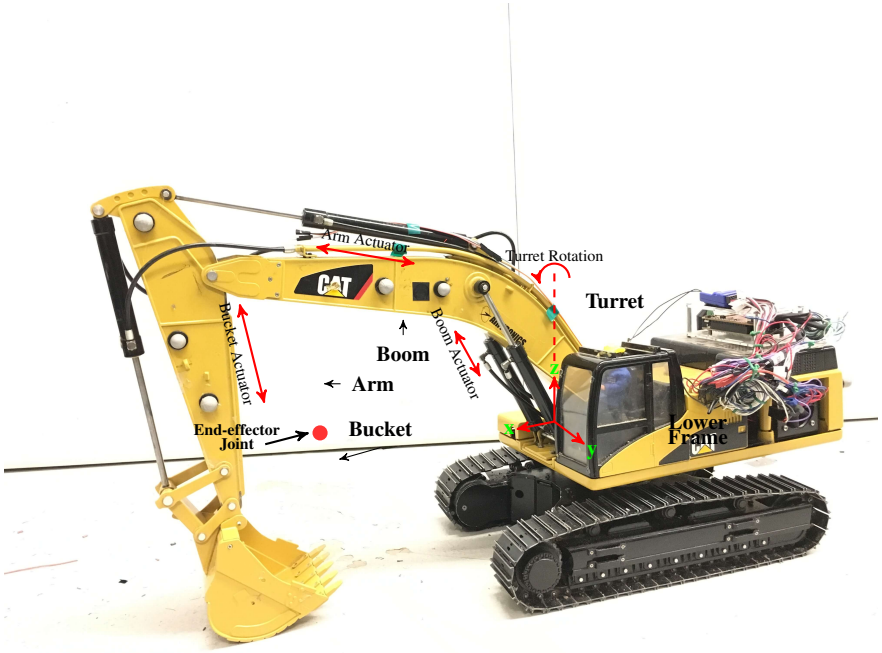


Figure 3.5: Experimental platform: 1/14th scaled excavator robot has four links Turret, Boom, Arm and Bucket. Turret is actuated by an electric motor (not seen in the figure). Other links are hydraulically actuated by piston-cylinder mechanism. Base reference frame of the robot is fixed to the lower frame and centered at the revolute joint between the lower frame and the turret, with one of its axis along the axis of turret rotation. End-effector pose $\boldsymbol{s}^e$ is given by the (x, y, z) position of the end-effector joint w.r.t the base reference frame together with the bucket angle.

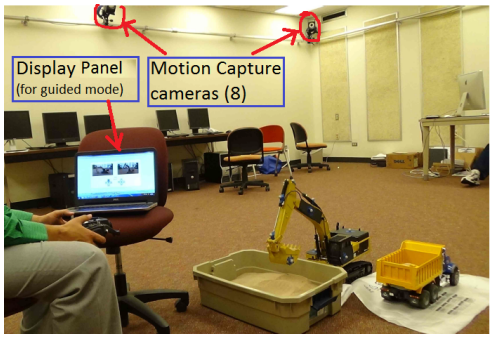


Figure 3.6: Experimental set-up consisting of motion capture system, scaled excavator model, and display panel for visual interface.

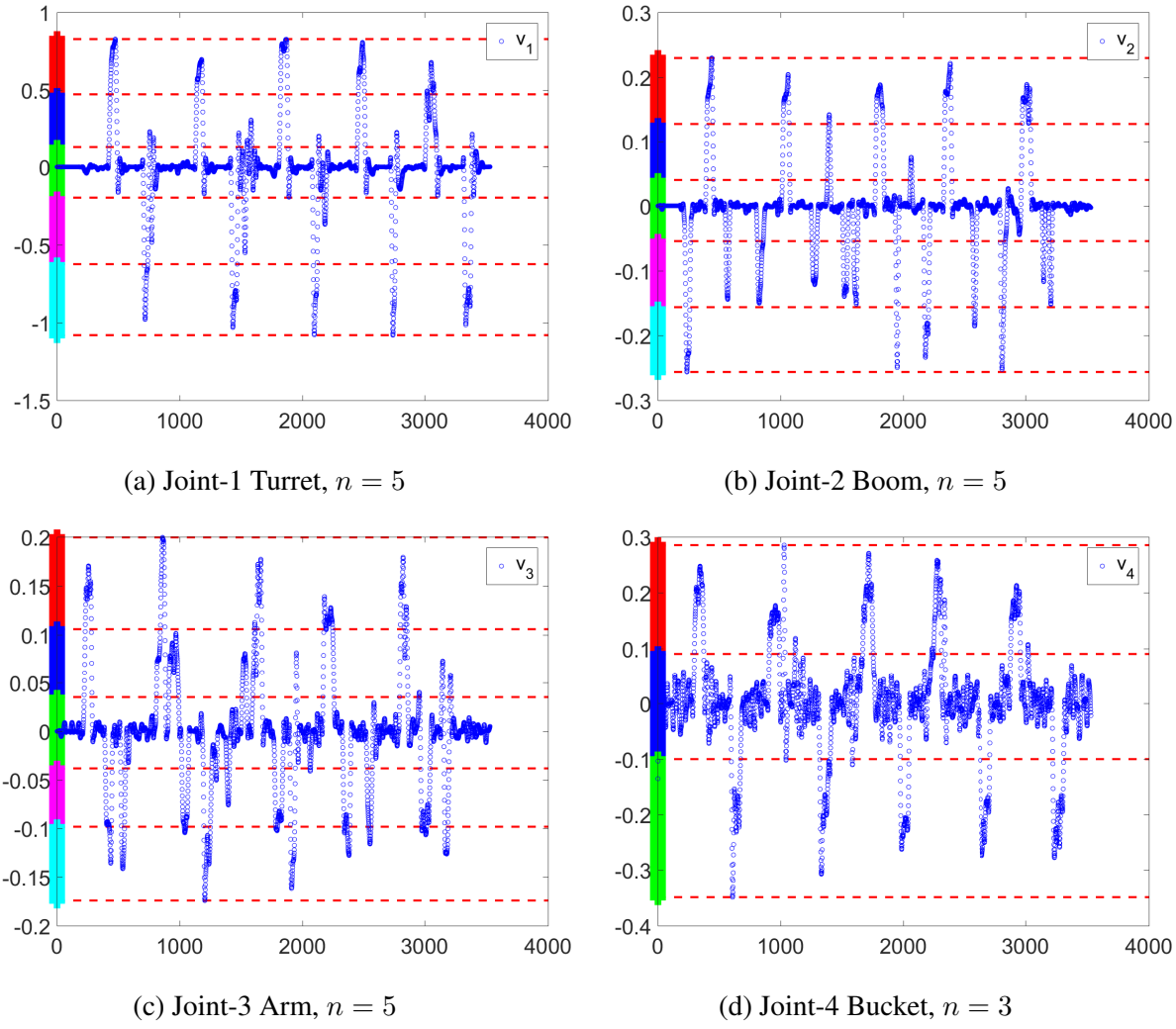
hypothesis is to visually reproduce instruction policy in human operators' action space as in Figure 3.4a. This interface depicts action primitive by showing desired joystick movement direction and magnitude using red colored bars with appropriate length. Thin black line corresponds to the actuation by the operator, which when matched, turns the red colored bar to green. Second hypothesis is based on several considerations. First, with regards to interactive experiences, [16] shows that simple visual interfaces help reduce stimulant load during learning, suggesting that a basic interface would promote better retention of acquired skills. Second, given the nature of uncertainty and human presence around construction equipments, the interface should demand minimal attention to ensure better situational awareness of the operator. Therefore, the second interface (Figure 3.4b) only generates positive reinforcement for desired actuation by change in color, from red to green, to reinforce desired skills while demanding minimal operator attention. We chose the easily-recognizable green and red colors to indicate correct (green) and incorrect (red). These colors come with subtle cultural associations of good and bad ([87, 88]) which helps to create the emotional associations needed to maintain them as reinforcement tools needed for this hypothesis.

3.4 Experiments

Experiments were performed on a $1/14^{th}$ scaled 345D Wedico excavator model, a 4 d.o.f hydraulic robotic arm manipulator, controlled by a radio transmitter (see Figure 3.5 for the robot's description). Figure 3.6 shows the human operator performing truck loading task with the excavator robot. The robot communicates instruction to a operator using the visual interfaces shown in Figure 3.4. The Wedico excavator lacked joint-angle encoders and internal proprioception, hence the experiments were performed within a motion capture facility to provide real-time measurements to the algorithm. End-effector position w.r.t the base frame and its mechanism that is the bucket angle gives 4-D end-effector pose vector $\boldsymbol{s}^{\boldsymbol{e}}$. To demonstrate our approach we selected the truck loading task which is a standard task performed using an excavator. A truck loading cycle begins with the bucket positioned over the sand pile and involves scooping of the sand, positioning of the bucket over the truck, dumping the sand into the truck, and coming back to the initial position, while avoiding spillage and damage to the truck. We obtained six set of demonstrations for the truck loading task from an expert operator. Each demonstration involved filling the truck with sand, for which joint positions $\boldsymbol{q}$, and state vector $\boldsymbol{s}^{\boldsymbol{e}}$, sampled at 25Hz were recorded and joint velocities $\boldsymbol{v}$ were computed through differentiation of joint positions $\boldsymbol{q}$.

3.4.1 Action Primitive based Segmentation

To learn the instructional policy model we first decompose continuous state-action trajectories from expert demonstrations into state-action primitive segments. To obtain action primitives we first cluster actuator velocities using DP-means algorithm according to the process described in section 3.2.2. Figure 3.7 plots actuator velocities for each joint, and the clusters of these velocities obtained using DP-means on the y-axis. This divides actuators velocities (and hence joystick movement) into n-bands, $n = [5, 5, 5, 3]$ for the four joints, turret, boom, arm, and bucket respectively, was obtained.



(a) Joint-1 Turret, $n = 5$

(b) Joint-2 Boom, $n = 5$

(c) Joint-3 Arm, $n = 5$

(d) Joint-4 Bucket, $n = 3$

Figure 3.7: Joint velocities sampled from a truck loading task demonstration using the excavator robot is plotted versus sampling instants on $x-$axis. Clustering output of the joint velocities using DP-means algorithm is shown on the $y-$axis. An initial guess of the discretization level, $n' = [6, 5, 6, 4]$, was used. Bucket velocity is very noisy due to the movement under load. It is important to isolate such noise in joint movement in the process of skill identification.

Table 3.1: List of action primitive classes. r_1 corresponds to the turret joint, r_2 to boom, r_3 to arm, and r_4 to bucket. This list is not exhaustive, but sufficient to infer actuator movement for any other action primitive class that is not listed.

Action Primitive Class $[r_1, r_2, r_3, r_4]$	Actuator Movement
$[3, 3, 3, 2]$	No motion or noisy perturbations
$[1, 3, 3, 2]$	Turret rotates anti-clockwise at high speed
$[2, 3, 3, 2]$	Turret rotates anti-clockwise at low speed
$[5, 3, 3, 2]$	Turret rotates clockwise at high speed
$[3, 1, 3, 2]$	Boom raise at high speed
$[3, 5, 3, 2]$	Boom lower at high speed
$[3, 3, 1, 2]$	Arm Inwards at high speed
$[3, 3, 5, 2]$	Arm outwards at high speed
$[3, 3, 3, 1]$	Bucket scoop
$[3, 3, 3, 3]$	Bucket dump

Next, we classify each sampled actuator velocity into an action primitive class according to the process described in section 3.2.2. For the excavator robot, each action primitive class is given as $\mathbf{r} = [r_1, r_2, r_3, r_4]$, where r_1 corresponds to the turret joint, r_2 to boom, r_3 to arm, and r_4 to bucket. Each component r_j takes a value $i \in \{1, \ldots, n_j\}$ and represent a Gaussian distribution $\mathcal{N}(\mu_{ji}, \sigma_{ji}^2)$, whose parameters were obtained from cluster members. Some action primitive classes for the excavator are defined in Table 3.1. Segmentation of three demonstration trajectories into action primitive segments is shown in Figure 3.8, and a similar segmentation in end-effector's pose w.r.t the robot's base frame is shown in Figure 3.10a. To isolate noisy perturbations, threshold η was set to 10^{-3}. In Figure 3.8a, action primitive labels at each time step are indicated by unique colors, with grid lines marking their start point except for the action primitive $\mathbf{r} = \mathbf{r}_{db}$, representing noisy perturbation or no movement which is shown in dark blue. Action primitives for first 122 samples represent no movement that is, $\mathbf{r}_1 = \cdots = \mathbf{r}_{122} = \mathbf{r}_{db}$, thus forming an action primitive segment $(s_1, \mathbf{r}_{db})$. Rest of the action primitive segments for this demonstration are given in Table 3.2. Action primitive segmentation has reduced around 700 state-action pairs to only 12 state-action primitive pairs (excluding $\mathbf{r}_{db}$). This is over 1.5 orders of magnitude reduction in data size for action primitive based policy learning. Such reduction in data size can certainly improve the scalability of policy search methods to high dimensional problems with continuous state action spaces. For learning instructional policy, proposed segmentation is certainly scalable to huge data-sets comprising of tens of thousands of state-action pairs in demonstration data.

Table 3.2: Detailed breakdown of action primitive segmentation for the first demonstration trajectory of truck loading cycle. Only the first occurrence of noisy perturbation action primitive $r_{db} = [3, 3, 3, 2]$ is shown, others are omitted. All missing indices belong to r_{db}. Index refers to the sampling instants.

Action Primitive Class $[r_1, r_2, r_3, r_4]$	Start Index	End Index	Segment Length	Actuator Movement
$[3, 3, 3, 2](= \mathbf{r}_{db})$	1	122	122	Neutral/Noisy perturbation
$[3, 5, 1, 2]$	123	148	26	Boom, Arm (Cycle Begins)
$[3, 3, 1, 2]$	149	169	22	Arm
$[3, 3, 3, 1]$	190	272	83	Bucket scoop
$[3, 1, 5, 2]$	302	327	26	Boom, Arm
$[1, 1, 5, 2]$	328	368	41	Turret, Boom, Arm
$[1, 3, 3, 2]$	374	393	20	Turret
$[3, 3, 5, 2]$	414	440	27	Arm
$[3, 5, 3, 2]$	443	463	21	Boom
$[3, 3, 3, 3]$	473	553	81	Bucket Dump
$[5, 3, 3, 2]$	583	636	54	Turret (cycle ends)
$[3, 5, 1, 2]$	679	721	43	Boom, Arm

Computation time required to perform segmentation using action primitives and BP-HMM [13] is in Table 3.3, computed using i7-6700K CPU @4GHz and 24 GB RAM machine. For latter, combined Metropolis-Hastings and Gibbs sampler were executed 10 times for 1000 iterations, selecting segmentation with the highest log likelihood w.r.t the feature settings. Computation time required by BP-HMM scaled geometrically with data size, in comparison, action primitive based segmentation scales almost linearly. Number of skills discovered by BP-HMM increases proportionally with data (see Table 3.3), but the number of action primitives discovered remained fairly constant. Figure 3.9 compares the segmentation obtained by these two methods, clearly both identify repetitive segments across the two cycles. It seems that the classification of noisy perturbation into different skills (depicted by solid red grid lines), caused BP-HMM to discover increasing number of skills. Moreover, several segments identified by the BP-HMM do not correspond to a monolithic action in human operator's action space, rather they span across multiple actions taken by the expert, for example the BP-HMM segment around the sampling instant $\tau = 400$, encompasses three action primitives. Thus to map a BP-HMM segment to an instruction for a human operator, it would be required to further break down those segments into component action primitives. Due to this limitation of relating segments with the human operator's action space, segmentation obtained from BP-HMM or any other statistical method cannot be applied to generate instructional policy.

Table 3.3: Comparison of the presented action primitive based segmentation with BP-HMM ([13]) in terms of the computation time (in seconds) and the number of action primitives versus number of BP-HMM skills, for 2, 4, and 6, demonstration trajectories taken together.

No. of Time Series	2	4	6
Data size N (4-D data)	6690	14033	23341
Action Primitive (sec)	21	41	68
BP-HMM (sec)	1639	6690	14752
# Action Primitives	14	17	18
# Segments (BP-HMM)	27	56	79

3.4.2 Learning Instructional Policy Model

To learn the instructional policy model we first learn the subgoals based on the segmentation output. For each demonstration j, we obtain the set of state-action primitive pairs, $O_{\mathcal{R}_j} = \{(s_1, \mathbf{r}_1), (s_2, \mathbf{r}_2), \ldots, (s_{N_j}, \mathbf{r}_{N_j})\}$, where N_j is the total number of action primitive segments. And the union of all the six sets $O_{\mathcal{R}} = \bigcup_{j=1}^{6} O_{\mathcal{R}_j}$, obtained from segmentation of six demonstration trajectories, is used as an input to Dirichlet process means inverse reinforcement learning (DPMIRL) algorithm to decompose the task space into finite subgoals. There are two ($M = 2$) objects in this task, the truck and the pile. Coordinate frames centered at these objects are used to first group and then cluster each state $s_i \in O_{\mathcal{R}}$ using algorithm 1, to obtain subgoals as multivarite Gaussian in $\hat{n} = 4-$D space. For presentation clarity we first consider a single expert demonstration and obtain the set $O_{\mathcal{R}_1}$. Total five subgoals were discovered using $O_{\mathcal{R}_1}$ as an input to the algorithm 1, three w.r.t the pile and two w.r.t the truck, whose mean location in three dimension is shown in Figure 3.11a. Figure also shows each state s_k associated with the action primitive segment $\mathcal{A}_k$ as unfilled circles. Subgoals obtained using original BNIRL approach by [65], are shown in Figure 3.11b. Clearly, the decomposition generated by the latter approach associates action primitive segments over the pile to the subgoal over the truck and vice-versa. Whereas the clustering approach using DP-means generates member action primitive segments that can lead to the execution of associated subgoal, hence it is better suited for generating instruction policy. Note that the clusters for subgoals 2 and 3 are co-located but are actually far apart in the fourth dimension of bucket angle, same holds for subgoals 4 and 5. Subgoals generated using $O_{\mathcal{R}}$ as an input to algorithm 1 is shown in Figure 3.11c, an additional subgoal is discovered w.r.t the dirt pile. This is attributed to the change is scooping position as the sand pile got exhausted in some demonstrations. Thus, the proposed policy model allows incorporation of additional demonstrations, though not incrementally but by relearning parameters over the entire data set.

Finally, we learn the parameters of instructional policy model given by equations (3.5)-(3.7).

We learn the parameters based on the action primitive segments, the subgoals and the data from the six demonstration trajectories. Each element Π_{ij} of the transition distribution matrix Π for subgoals, models the probability of transitioning to the subgoal X_j given the current subgoal X_i. To obtain Π_{ij}, we divide the total number of transitions from the subgoal X_i to X_j by the total number of transitions from the subgoal X_i. Similar counting process is followed to the learn the parameters of the transition model $\pi(X_i|X_{i+1})$ and the initial distribution $\pi_0(X_i)$, from the action primitives associated with the transition from subgoal X_i to X_{i+1}. Parameter vector $\theta_{Z_\tau^i}$ w.r.t an action primitive is known from the k-means clustering process used in action primitive based segmentation. The resulting model is generalizable to any novel configuration of objects in the task as the subgoals are identified w.r.t the task objects whose location is known to the robot during task execution. Computation time for the entire learning process, from processing six demonstration trajectories to learning instructional policy model was 85 secs. Thus, learning policy model from more and more demonstrations is feasible and computationally inexpensive.

Visualization of the learned instructional policy model: Refer figure 3.11a for subgoals and figure 3.5 for the robot's description. Truck loading cycle begins with the bucket in dump position (bucket cylinder retracted) and the entire excavator arm located above the sand pile. This position corresponds to subgoal no.4's initiating action primitive of lowering the boom given by $\mathbf{r} = [3, 5, 3, 2]$, followed by action primitives comprised of arm and bucket motion until the bucket scoops the sand, at this instant the robot enters the subgoal no.5. This subgoal involves raising of the boom and turning towards the truck, this gives way to subgoal no.1. Turning further to align with the truck results in subgoal no.2. In subgoal no.2, sequence of action primitives positions the bucket over the truck to execute dump action. Upon dumping subgoal no.3 becomes active and involves a single action primitive of turning towards the sand pile. Once over the pile subgoal no.4 becomes likely and the cycle continues. Subgoals related to scoop and dump subtasks are depicted as 3-d Gaussian ellipsoid in Figure 3.11d. Clearly, the subgoal w.r.t the sand is geometrically larger and allows for scooping at different locations, whereas the subgoal w.r.t the truck is smaller and corresponds to precise dump location. Thus, our definition of subgoals is useful in capturing the nature of manipulation utilized by the expert w.r.t each task object.

3.4.3 Testing Instructional Policy Model

Excavator robot utilizes the instructional policy model, learned from expert demonstrations, to train its novice operators. This process depicted in the Figure 1.2, was tested using the experimental set up shown in the Figure 3.6. Real time execution of a task with the robot generating instructions for the human operator follows the cycle shown in Figure 3.12. At any instant, the robot identifies the current action primitive based on a window of last ten sampled joint velocities. An action primitive

is detected, if each sample in the window belongs to a particular action primitive class, otherwise the stationary action primitive $\mathbf{r}_{db}$ is assumed. Robot queries instruction from the instructional policy model after consecutive detection of an action primitive $\mathbf{r}$ (other than $\mathbf{r}_{db}$) followed by $\mathbf{r}_{db}$. Based on the action primitive $\mathbf{r}$ and the current state $\boldsymbol{s}^e$, an instruction $\boldsymbol{i}$ is generated. This instruction is communicated to the operator. This process ensures that the operator executes a joystick control before a change in the visual instruction, thus maintaining a suitable update rate for instructional policy, recall Figure 1.3.

An instruction is communicated to the human operator through the visual interface, using the two different instruction hypotheses or GUIs (Figure 3.4). In case of GUI Bars, predicted instruction is depicted visually. For GUI Circles, the red color changes to green if the operator executes the predicted instruction. In the absence of a benchmark for the present problem, performance of a set of novice human operators without any visual instructions was recorded. These novices were allowed to learn by observing an expert operator perform truck loading task and becoming familiar with the joystick controls before they performed the task.

A total of 113 participants (novice operators) had volunteered under IRB guidelines and were comprised of active students, faculty, or visitors in the psychology and engineering departments. Upon arrival to the laboratory, each participant was randomly assigned to one of the three groups: Group with no GUI, group using the GUI circles (Figure 3.4b), and group using the GUI with speed bars (figure 3.4a). Participants were asked to complete a minimum of three truck loading cycles with the instruction interface available to those belonging to the last two groups. We will refer to the truck loading cycles performed in the these trials as cycle-1 (C1), cycle-2 (C2) and so on. To evaluate the instructional policy model, performance of the three groups is compared in terms of cycle completion time and erroneous action primitives per cycle. Detection of an action primitive that do not conform with the current joystick movement instruction is counted as an erroneous action primitive. Note that, since an action primitive is detected based on a window of sampled joint velocity, any noisy perturbation or inertial movement of joints is not counted as an error. The results are summarized in Figures 3.13-3.14. Figures 3.13a, and 3.13b, plots the cycle completion time and the cycle errors respectively, for the first three cycles. The bars group that used the instructional policy, shows a consistent improvement in doing the task as the median, and the spread, of their cycle completion time as well as errors, decrease over the three cycles. For the participants without instructions, non-significant improvement in the performance was observed. The circles group better their cycle completion time, though their errors do not reduce significantly. To reaffirm this observation, paried t-tests for cycle time and cycle errors between the first and the third cycle were performed within each group, using $\alpha = 0.05$. For cycle time, GUI bars ($\hat{\mu} = 8.94, \hat{\sigma} = 12$), and GUI circles ($\hat{\mu} = 10.51, \hat{\sigma} = 11.2$) group showed significant differences ($p < 0.001$) in comparison to no GUI group ($\hat{\mu} = 3.81, \hat{\sigma} = 19.19$). Corresponding

Table 3.4: Comparing novice operator performance statistics for the three groups, No GUI, GUI circles, and GUI bars, using paired sample $t-$tests between cycles for cycle time (first three rows) and cycle errors (last three rows). C1-C3 implies between cycle 1, and cycle 3, R1-R3 implies between retest cycle 1 and retest cycle 3.

Paired $t-$test	GUI Circles $(t(n-1), p)$	GUI Bars $(t(n-1), p)$	No GUI $(t(n-1), p)$
Cycle Time			
C1-C3	0.82(27), 0.420	3.06(25), 0.010	1.13(26), 0.270
C3-R1	0.68(27), 0.500	0.59(25), 0.560	1.13(26), 0.270
R1-R3	4.53(28), 0.001	1.76(25), 0.090	2.60(28), 0.020
Cycle Errors			
C1-C3	1.46(31), 0.154	1.31(30), 0.070	0.84(37), 0.410
C3-R1	1.27(19), 0.220	-0.58(26), 0.570	0.98(34), 0.330
R1-R3	2.29(19), 0.033	2.71(26), 0.012	1.98(34), 0.055

$t-$statistics and $p-$values are reported in Table 3.4 for the cycle time as well as for cycle errors. These results indicate that using instructional policy, statistically significant improvement in the performance of the novice operators was observed.

Another observation from the box-plots in Figure 3.13b, is a number of outliers for the groups using instruction interface. These are some of the apparent cases in which the instructional policy failed to generate instructions as the state of the robot deviates from the domain of π^I, i.e $s \notin \Omega_O$. Counting even a single failure to generate an instruction during a cycle as failure, overall π^I failed in 23.3% of truck loading cycles. One way to negotiate such failures would be to instruct the operator to drive the robot back to the previous subgoal position. Finally, we report the performance averaged over cycles as most of the participants performed around five truck loading cycles as box-plots in Figure 3.14. These plots show that the novice operators assisted by the instructional policy perform far better than the operators learning on the basis of observation or positive feedback.

3.4.4 Testing Instruction Hypotheses for Retention

Retention is an important aspect of training novice operators. To test retention among the three groups, participants were asked to return one to three days later (based on availability) to perform additional cycles (called retests) without the help of any instruction interface. Note, not all participants reported back for the test. Cycles performed in retests are referred to as retest-1 (R1), retest-2 (R2), etc. Paired sample $t-$tests were conducted within each group to compare the performance between the cycles - C3 and R1, and, R1 and R3. The goal of this test was to determine if there is a significant change in cycle time and cycle errors between each of these cycles, to infer retention among the novice operators.

Paired $t-$test for cycle times (Table 3.4) suggested that for all groups there were no significant differences between cycle times for C3-R1. However, between R1 and R3, GUI circles ($\hat{\mu} = 18.62, \hat{\sigma}$ = 22.16) group and no GUI ($\hat{\mu} = 10.76, \hat{\sigma} = 22.29$) group showed significant differences ($p < .001$ and $p = .015$ respectively). This suggests that both the groups, GUI circles and no GUI, demonstrated significant changes in cycle time, and hence efficient performance during the retest. An ANOVA was used to compare within groups the cycle time results of R3 and one-sample $t-$test to compare it with the expert time (24.9s). The ANOVA showed that the difference in R3 times between groups was significant $F(2, 81) = 6.41$ ($p < .01$) suggesting that the GUI circles group demonstrated significant change from R1 to R3 in comparison to other groups. Moreover, they came closest to the expert's cycle time as seen in Figure 3.15 that shows the average cycle times for each group. This Figure summarizes and compares cycle time performance results over the cycles C1, C3, R1 and R3. Overall, the groups assisted by the robot learn to perform better and come closer to the performance of an expert in comparison to the group learning from observation and experience whose performance improves but only at a slow rate.

Paired $t-$test results for the cycle errors shown in Table 3.4 suggests that, although significant improvement in errors did not occur during testing cycles C1 and C3 (except for GUI bars), the data suggests that latent learning may have occurred as the errors decrease significantly during retest. This behavior was observed for GUI circles and GUI bars groups, which suggests that the instructional policy model improved the performance of participants over unguided novices. Overall, the results of this experiment demonstrated that the simple user guidance system that utilized green and red circles was more effective towards retention.

3.5 Policy Parameterization for Robot Learning

In previous sections, policy model representation was utilized to parametrize instructional policy for a complex construction task. This parameterization approach is a probabilistic framework that also holds the potential for learning and executing complex tasks in pure LfD setting. In this section, we present the application of the proposed policy representation for autonomous execution of truck loading task. In the presented framework, sequencing of low level skills (or action primitives) is associated with the high level task context, with transitions at each level modeled using Markov chains. We believe that the utilization such hierarchy of Markov chains, based on the robot and the task at hand, provides additional modeling power and flexibility for policy parameterization which is lacking in the existing literature.

Such parameterization coupled with existing policy search approach, where the later comprises of different means for policy exploration, policy evaluation and policy update ([9]), holds the key

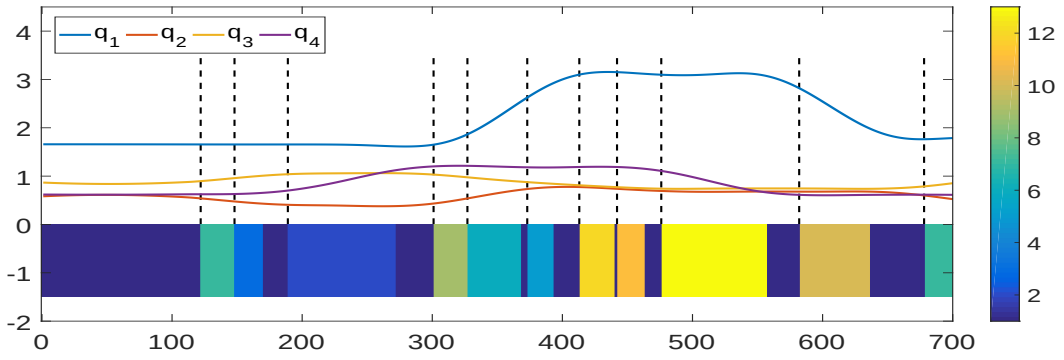
for robot learning a single policy for complex real world tasks. The presented approach can be modified for a robot to learn a usual policy, $\pi : s \rightarrow a$ based on the type of task. For a task that involves different acceleration profiles for a single motion such as playing tennis, baseball or beating drums, one straightforward modification is to model the policy for each action primitive segment using a dynamic motor primitive, while retaining the benefits of the hierarchical structure of the policy model. Presented policy parameterization can be directly used for executing goal based tasks such as the truck loading task presented in this paper, pick and place, assembly of parts, etc. To utilize the policy model in these types of task, it is not sufficient to identify the current subgoal rather the location of the initiating and the terminating states for each action primitive segment is required. Trained motor primitives can then be utilized to generate the policy between the current state and the terminating state as in [89]. For truck loading task, each segment being an action primitive, it suffices to command actuator velocities sampled from equation (3.7) to generate a policy that mimics an action primitive segment between the current and the terminating state.

The initiating and terminating states for each subgoal X_i can be obtained from the associated set $\mathcal{S}_i$ of state-action primitive pairs. We collect the states of all initiating action primitives in the set $\mathcal{S}_i$, and learn a multivariate Gaussian (as we did for the subgoals) to represent the initiating set $\mathcal{I}_i$ for the subgoal X_i. Similar initiating set $\mathcal{I}_{i+1}$ for the next subgoal X_{i+1}, acts as the terminating set for X_i, i.e., $\mathcal{T}_i = \mathcal{I}_{i+1}$. A chain of action primitives is used to generate policy starting from a state in $\mathcal{I}_i$ to a state in $\mathcal{T}_i$. Switch between the intermediate action primitives was decided on the basis of error between the mean states of the initiating and terminating sets. This is plausible for robots with one-to-one correspondence between the degree-of-freedom and the actuators. Otherwise, the time duration for intermediate action primitives can be modeled using semi-Markov models.

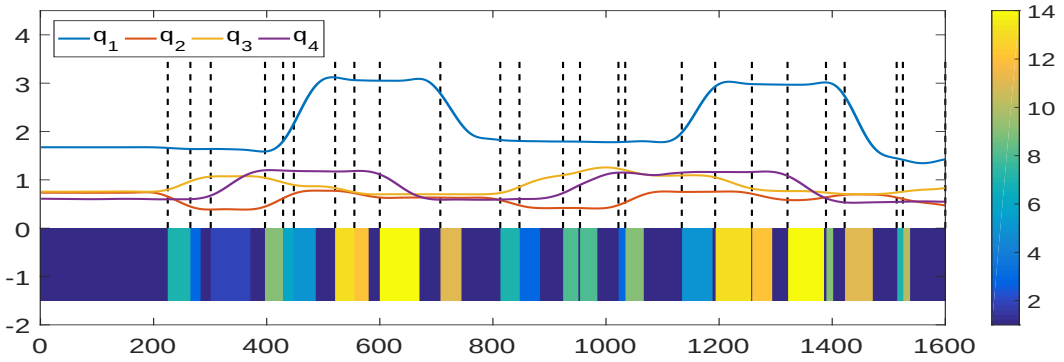
3.5.1 Autonomous Task Replay in Novel Configurations

We test the policy model for autonomous execution of truck loading task using the modifications outlined in the previous section. These experiments were performed to demonstrate policy model's performance in pure LfD setting, otherwise the truck loading task requires human presence for the perception of dynamically changing sand surface. Two different configurations for the demonstration of truck loading task were used as shown in the Figure 3.16. Policy model parameters were obtained following the procedures detailed in sections 3.2.2-3.2, in addition the initiating and terminating sets w.r.t each subgoal were also identified. Subgoals identified are shown in Figure 3.17. Algorithm identifies distinct subgoals w.r.t the truck and the sand pile to learn different set of action primitives for different configurations, for example the set of action primitives required to reach a truck subgoal to the right of pile will be different from those required to reach a subgoal on the left. During execution, $\arg\max$ operator is used to make inference using equations (3.5)-(3.6),

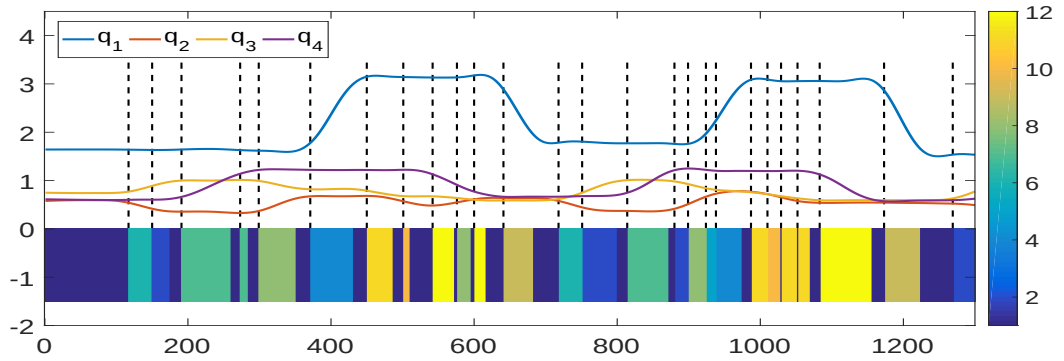
to choose the next subgoal and a corresponding action primitive. Policy model was tested for four different test configurations shown in Figure 3.16. The robot was able to generalize for the first three configurations and successfully performed the task. For the last case, it could not manage to raise its arm to clear the truck height, as the truck was placed very close to the sand in comparison to the demonstration cases. Using policy update strategies ([9]), the robot can learn to generalize in such situation as well as improve its performance. To compare the timing performance for a truck loading cycle, we also tested the policy model for a configuration quite similar to that of the expert. Two comparison cases shown in Figure 3.18, suggests that the policy model closely mimics the expert's performance, finishing the task four seconds earlier in the second case, due to quick and precise dumping over truck, and absence of humanly pauses. Thus, the proposed policy model successfully replayed the truck loading task in different configurations achieving better performance in some cases.



(a) Demonstration 1



(b) Demonstration 2



(c) Demonstration 3

Figure 3.8: Segmentation of three demonstration trajectories of truck loading task, along with the joint positions. Though each demonstration had five cycles, we show either one or two cycles for clarity. Position of each joint q_j is sampled 25 times per second. $x-$axis denotes sampling instants.

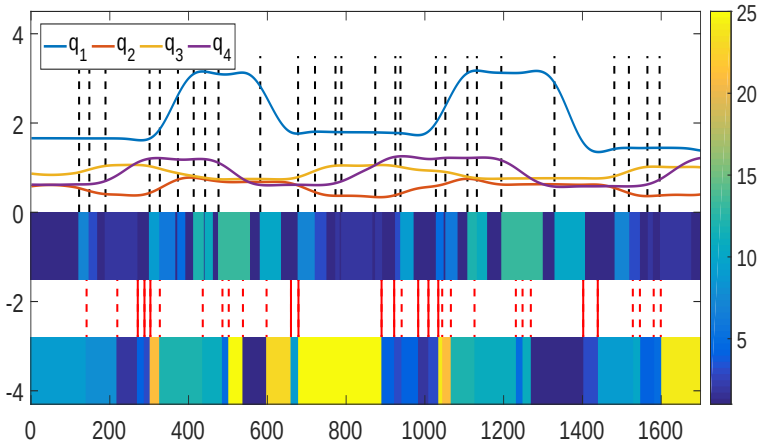
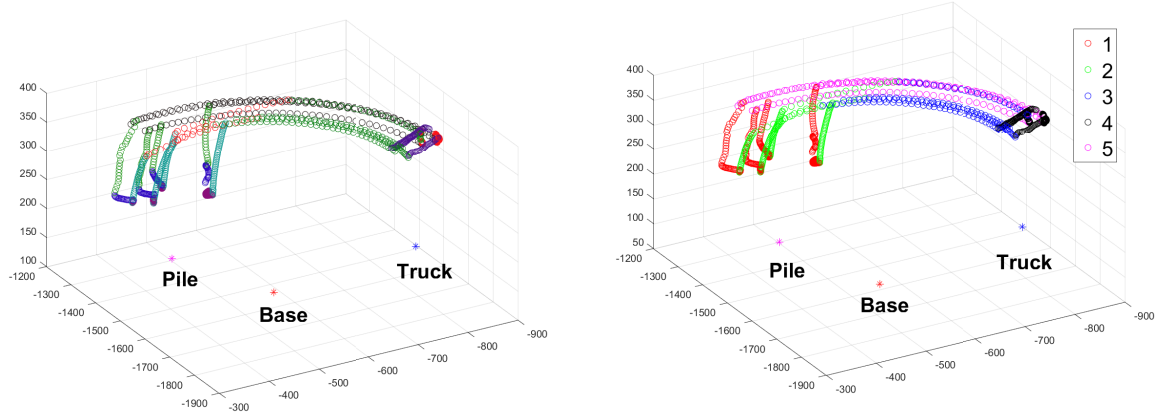


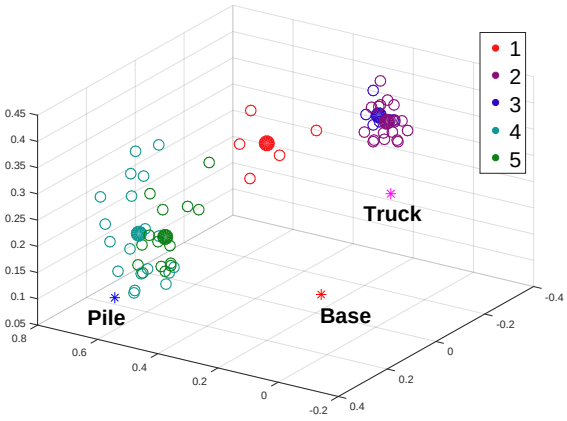
Figure 3.9: Segmentation using action primitives (top) and BP-HMM (bottom) for two cycles of truck loading task, along with the joint positions. Black grid lines depict the start of action primitives (except for the noisy perturbation $\mathbf{r}_{db}$), similarly red grid lines indicate the start of BP-HMM segments. Solid red grids indicate BP-HMM segments corresponding to noisy perturbations.



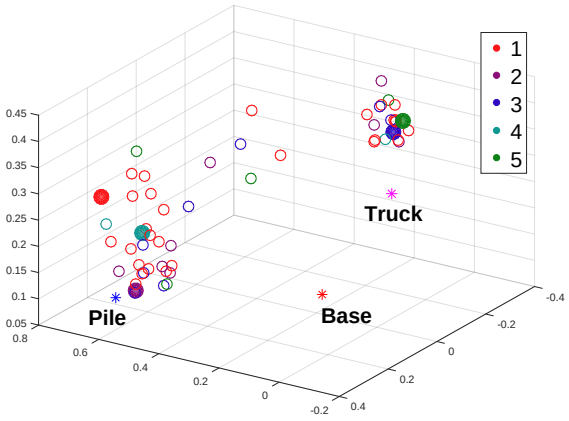
(a) Action primitive segmentation. Each color represents a unique action primitive segment.

(b) Subgoal based segmentation. Each subgoal comprises of one or more action primitive segments.

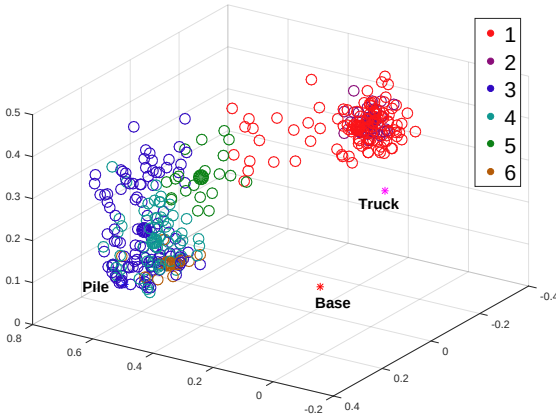
Figure 3.10: Segmentation in end-effector's pose (x, y, z) w.r.t the base frame of the robot for a single expert demonstration consisting of five truck loading cycle.



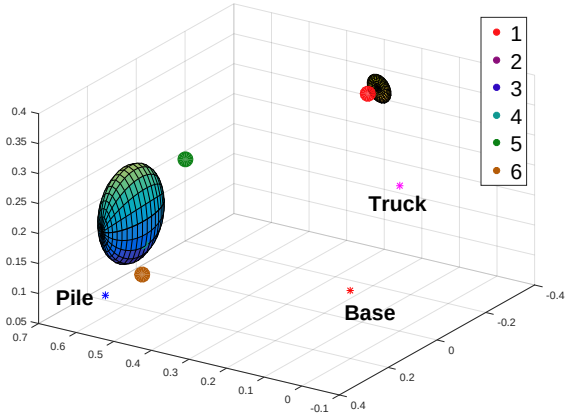
(a) DPMIRL



(b) BNIRL



(c) DPMIRL (For six demonstration sets)



(d) Scoop subgoal over pile and dump subgoal over truck

Figure 3.11: Learning subgoals: Task space depicts robot's base, pile and truck location. (a)-(b) For a single demonstration set both algorithms discovered five subgoals shown as filled circles. State s_k of each action primitive segment $\mathcal{A}_k$ is depicted as unfilled circle. (c) Subgoals obtained using modified BNIRL for all six demonstration trajectories. (d) 3-d Gaussian ellipsoids with one standard deviation from the mean for scoop and dump subgoal.

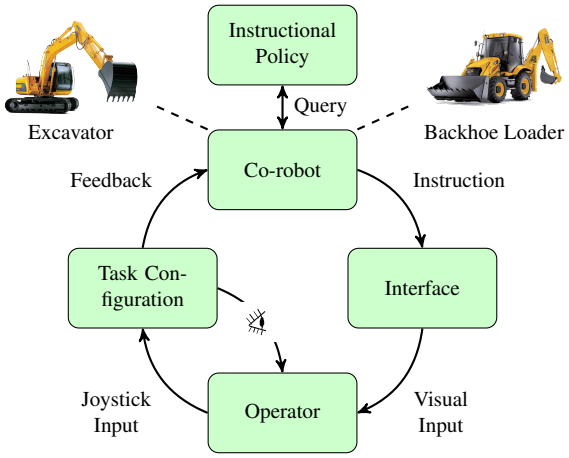
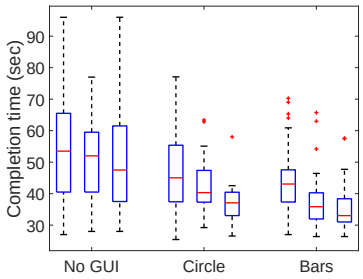
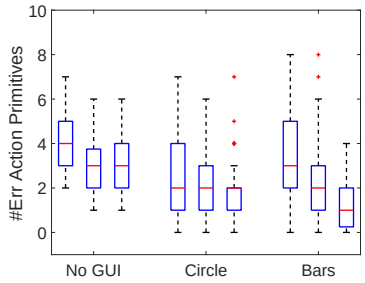


Figure 3.12: Learning instructional policy from expert demonstration can enable construction robots to generate instruction for human operators based on the current task space configuration in real time. Goal is to reduce skill-gap between experts and novice operators and to ensure efficient task execution.

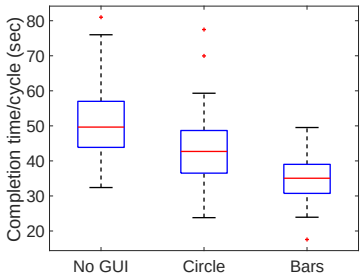


(a) Cycle time performance over first three cycles.

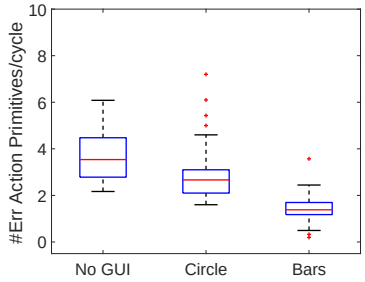


(b) Cycle errors over first three cycles.

Figure 3.13: Participant performance in terms of cycle time and cycle errors for the three different groups.



(a) Average cycle time performance.



(b) Average cycle error performance.

Figure 3.14: Average cycle time and cycle errors for the three different groups.

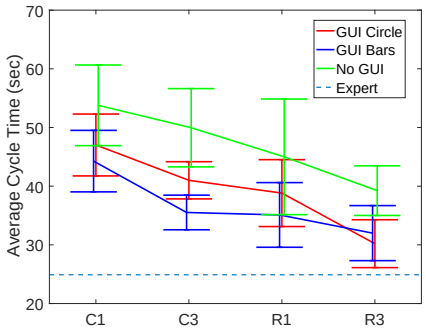


Figure 3.15: Average cycle time with 95% confidence interval error bars for C1, C2, R1, and R3. Means of the bars plots are connected to visualize and compare the rate of improvement between the cycles. Average cycle time of the expert operator is marked by the dashed line.

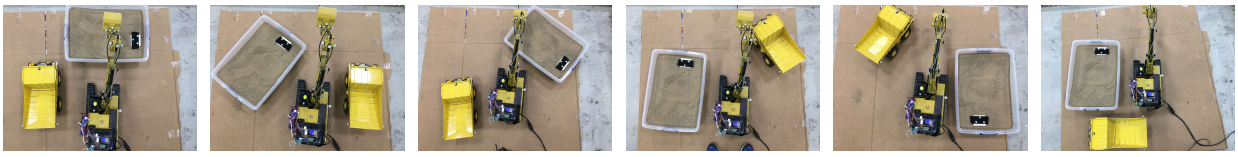


Figure 3.16: First two images are task demonstration configurations, followed by four novel test configurations.

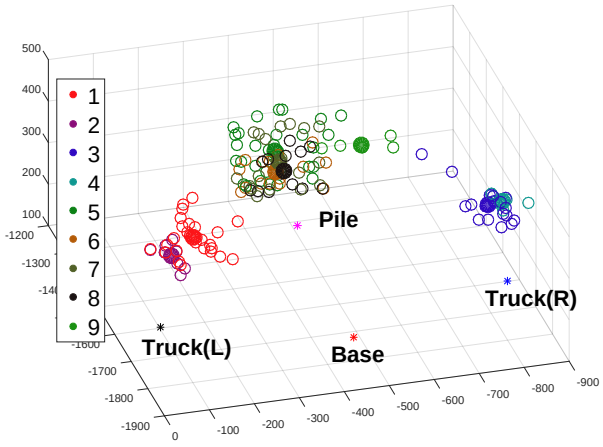


Figure 3.17: Distinct subgoals for different relative configurations of task objects. Note there are different subgoals w.r.t pile for approach from left/right truck position.

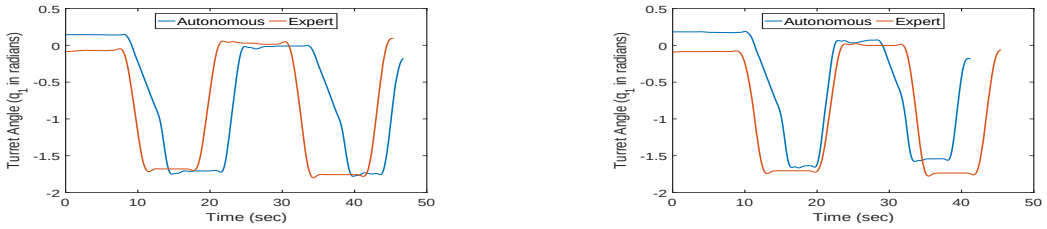


Figure 3.18: Two comparison cases between autonomous operation and expert demonstration for two cycles of truck loading operation. For clarity only turret angular position is shown.

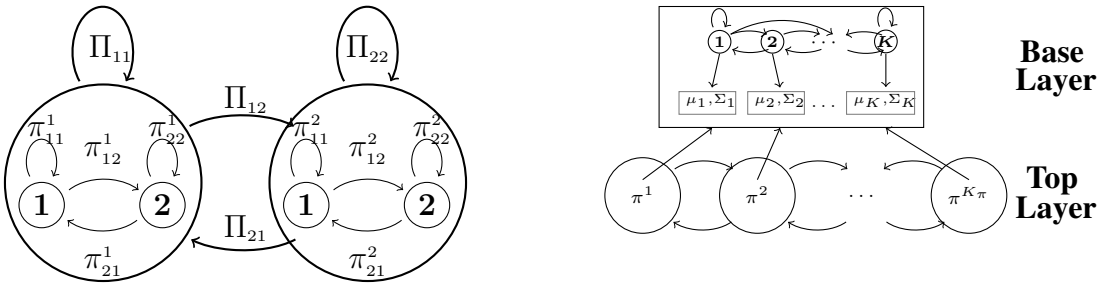
CHAPTER 4

CONSTRUCTION ROBOTS: INFER SWITCHING BETWEEN TASKS

For real world implementation of instructional policy model it is required to infer when an operator switches between different tasks. Construction activities are composed of several different tasks, for example a truck loading activity also involves intermediate digging, leveling and piling tasks along with the cycles of truck loading task. Our goal is to learn policy models for each task, and to infer switching between different construction tasks. To solve this problem we propose to utilize non-stationary Markov chain as a model of the overall task while inferring non-stationarity or switches between subtasks using a first order system to model the likelihood of each subtask. Most of the existing attempts to learn such complex tasks are either heuristic based or fail to capture the subtasks. We propose to investigate a Layered Non-stationary Markov Model (LNMM) that would learn a complete task model from demonstration. It will comprise of hidden Markov model at lower level to represent transition among the action primitives for any given subtask, while the Markov chain at higher level would capture the transitions between the subtasks. The hierarchical Markov model would basically capture the non-stationarity in the Markov chain at the lower level. In the following section, this is formulated as switching between the finite set of transition probability matrices, where each matrix is a represents a task.

4.1 Problem Formulation

For a Markovian observation sequence $\{z_1, \ldots, z_T\}$, where each z_t is discrete, such that $z_t \in \{1, \ldots, K\}$, a typical modeling approach is to learn the conditional distribution $p(z_t|z_{t-1})$ modeled using a $K \times K$ matrix known as the Transition probability model (matrix) π, where $\pi_{i,j} = p(z_t = j|z_{t-1} = i)$ is the probability of going from state i to state j. We consider the case where non-stationarity in Markov models is induced by time variant transition function $p(z_t|z_{t-1})$, that switches between a finite set of transition probability matrices given by $\Pi^s = \{\pi^1, \ldots, \pi^{K_\pi}\}$. Thus for any non-stationary Markov sequence there exists a latent sequence of transition models as shown in (4.1), such that each $\pi^k \in \Pi^s$, defined as Recurrent Transition Probability Models.



(a) Layered Non-stationary Markov Model (b) Layered Non-stationary Hidden Markov Model

Figure 4.1: (a) Number of modes in base layer (thin lines) is $K = 2$, with two elements in $\Pi^s = \{\pi^1, \pi^2\}$, and the elements of top layer transition matrix Π (thick lines), (b) Base layer is a Hidden Markov model with K modes and respective Gaussian emissions, and a top layer that models Markovian dynamics among the set (Π^s) of K_π transition probability matrices (TPMs).

$$\underbrace{z_{11}, z_{12}, \ldots, z_{1\tau_1}}_{\pi^1}, \quad \ldots \quad , \underbrace{z_{k1}, \ldots, z_{k\tau_k}}_{\pi^k}, \quad \ldots \quad , \underbrace{z_{n1}, \ldots, z_T}_{\pi^n} \tag{4.1}$$

Our goal is to estimate the transition probability matrices in the set Π^s where K_π is not known a-priori. This translates to unknown number of tasks in a given construction activity. We assume that K i.e., the number of states observed is known. This corresponds to the total number of action primitives observed in a demonstration, which is known from segmentation (see section 3.2.2). Developments in this chapter are also applicable to class of Markov models, such as HMMs, SLDS, MJS and MDPs. We learn the set Π^s of TPMs over the latent mode sequence (i.e. the hidden markov chain) for HMMs, and over the Markov chains that governs SLDS or MJS. For these cases the state of the Markov chain is only partially observed as oppose to perfect state observations in the case of Markov chains and MDPs. For presentational clarity, an observation sequence is represented by $\{x_1, \ldots, x_T\}$, and the corresponding latent Markov sequence by $\{z_1, \ldots, z_T\}$. And for the observation model $p(x_\tau|z_\tau) \sim F(\theta_{z_\tau})$, the parameter vector θ_{z_τ} can be representative of Gaussian or multinomial emissions, a linear dynamical system or a non-linear transitional function.

4.2 Layered Non-stationary Markov Model (LNMM)

We propose that the switching between the transition matrices in the set Π^s can be modeled by layering a Markov chain that models the transition function $p(\pi^\tau|\pi^{\tau-1})$, where each TPM $\pi^\tau \in \{\pi^1, \ldots, \pi^{K_\pi}\}$, with the corresponding $K_\pi \times K_\pi$ transition matrix Π. We call this model as Layered Non-stationary Markov Model. A simple example of a Markov chain with two modes is shown

in the Figure 4.1a, in this example the transition matrix π switches among $\{\pi^1, \pi^2\}$. In brief, the hierarchical Markovian dynamics captures the discrete time non-stationarity in the lower level Markov Model. Similar example for the case of Hidden Markov Model is shown in Figure 4.1b, with K_π possible transition matrices in the top layer. We assume the number of modes K in the base layer to be known whereas K_π is considered unknown. To learn the parameters of the proposed LNMM model for a non-stationary Markov sequence we need to estimate the number of transition probability matrices and their switching dynamics. To obtain such an estimate we propose the concept of likelihood rate elaborated in subsection 4.2.2, before that we look at LNMM as the generative model of a non-stationary Markov sequence.

4.2.1 Generative Model

Our aim is to learn the parameters of the following generative model of layered Hidden Markov sequence (Figure 4.1b) with non-stationarity induced by recurrent transition models in Π^s,

$$\pi^\tau \sim \Pi_{\pi^{\tau-1}} \tag{4.2}$$

$$z_{\tau+1} \sim \pi^\tau_{z_\tau} \tag{4.3}$$

$$x_{\tau+1} \sim F(\theta_{z_{\tau+1}}) \tag{4.4}$$

At any time τ, the next latent mode $z_{\tau+1}$ is drawn from the transition distribution π^τ conditioned upon the current latent mode z_τ, where the transition distribution π^τ is itself drawn from the hierarchical transition distribution matrix Π given $\pi^{\tau-1}$. We define $\Pi_{i,j} = p(\pi^\tau = j | \pi^{\tau-1} = i)$, where $\pi^\tau \in \Pi^s$ at any t, and $\{\pi^1, \ldots, \pi^{K_\pi}\}$ are the elements of set Π^s. Finally the observation $x_{\tau+1}$ is generated by the distribution $F(\theta_{z_{\tau+1}})$ where $\theta_{z_{\tau+1}}$ is a parameter vector associated with state $z_{\tau+1}$. For Markov chains (Figure 4.1a) or MDPs we do not have the equation (4.4).

4.2.2 Likelihood Rate (ΔL) of a TPM

To estimate multiple TPMs from a non-stationary sequence given in (4.1), an important question that needs to be answered is that: Is it possible to identify the observations $z_{1\tau_1}, \ldots, z_{k\tau_k}$, or the time points $\tau_1, \ldots, \tau_k$, in a sequence, at which there is a switch in the TPM? This is a hard question to answer, since we need to identify a change in TPM given only the transitions between the observations. To answer this question we first assume that the set of possible TPMs that generate data, i.e. the set Π^s is known (assumption made for exposition sake only). Given the set Π^s, our aim is to evaluate the likelihood of a transition $z_\tau \rightarrow z_{\tau+1}$ with respect to each transition matrix $\pi^k \in \Pi^s$.

To evaluate the likelihood of a transition $z_\tau \rightarrow z_{\tau+1}$ with respect to each transition matrix $\pi^k \in \Pi^s$, we utilize the mean variance estimator developed by [36], that updates a transition matrix $(\pi)_{\text{mv}}$ after every observed transition as elaborated in section 1.3.5. We propose that the likelihood of a transition $(z_\tau \rightarrow z_{\tau+1})$ with respect to each transition matrix in the set Π^s, can then be evaluated in terms of the likelihood of the updated $(\pi)_{\text{mv}}$ with respect to each π^k. We define the likelihood of a recursively updated transition matrix $(\pi)_{\text{mv}}$ with respect to a known transition probability matrix π^k as

$$L((\pi)_{\text{mv}}|\pi^k) = \prod_{m=1}^{K}\left[\frac{1}{B(\alpha^m)}\prod_{i=1}^{K}(\pi_{m,i})_{\text{mv}}^{\pi^k_{m,i}-1}\right] \tag{4.5}$$

where $(\pi_{m,i})_{\text{mv}}$ denotes the i^{th} element of m^{th} row of $(\pi)_{\text{mv}}$, same holds true for $\pi^k_{m,i}$, α^m is a vector consisting of elements of the row π^k_m, such that $B(\alpha^m) = \frac{\prod_{i=1}^{K}\Gamma(\pi^k_{m,i})}{\Gamma(\alpha^m_0)}$, and $\alpha^m_0 = \sum_i \pi^k_{m,i} = 1$, henceforth we abbreviate $L((\pi)_{\text{mv}}|\pi^k)$ as L_k. We demonstrate the utility of this likelihood estimation using a simple example. Suppose we have three sequences O_1, O_2 and O_3, with the maximum likelihood estimates of transition matrices being π^1, π^2 and π^3. These matrices are the elements of the set Π^s. We consider a sequence $O = \{O_3, O_2, O_1, \ldots, O_3, O_2, O_1\}$, for which we recursively estimate transition matrix $(\pi)_{\text{mv}}$ using mean variance estimator of section 1.3.5. And at each step we evaluate the likelihood of this matrix w.r.t the matrices in the set Π^s using equation (4.5). This results in an interesting plot of likelihood values shown in the Figure 4.2. It is observed that there is sharp change in the likelihood behavior of the recursively updated transition matrix $(\pi)_{\text{mv}}$ whenever there is a switch in TPM, for example the likelihood of π^3 keeps increasing in comparison to that of π^1 and π^2 for the sequence O_3, but at the start of sequence O_2 the likelihood of π^2 starts increasing and that of π^1 and π^3 are on a decrease, and the behavior continues. However, note that for the sequence O_2 and O_3, the absolute values of the likelihood L_2 and L_3 is lower than the likelihood L_1, hence the distinguishing feature for π^k is the slope or the rate of likelihood $\Delta L((\pi)_{\text{mv}}|\pi^k)$ (or in short ΔL_k). We estimate likelihood rate as $\Delta L^{\tau+1}_k = L^{\tau+1}_k - L^{\tau}_k$. Another notable aspect from Figure 4.2 is that the likelihood estimate does not increase monotonically, this is attributed to the fact that a single transition may be likely with respect to more than one transition matrix, however it is observed that over finite number of transitions the likelihood of the generating transition model keeps increasing. *Thus a possible answer to the question that we had posed earlier is that by observing the likelihood rate ΔL_k of the recursively updated transition matrix $(\pi)_{mv}$ w.r.t. each matrix k in the set Π^s, one can identify the observations or the points in a sequence at which there is a switch in the TPM.* Another interesting observation in Figure 4.2 is that the likelihood of all the TPMs is converging over time, this shows that the recursively updated transition matrix $(\pi)_{\text{mv}}$ converges to an estimate that is equally likely w.r.t the existing TPMs. This

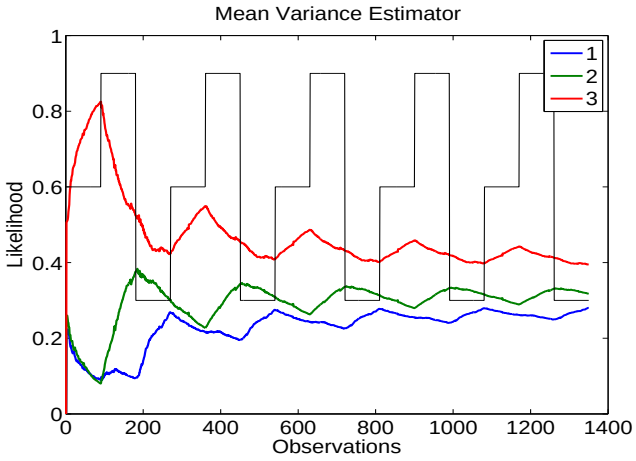


Figure 4.2: Likelihood of transition matrix $(\pi)_{\text{mv}}$ w.r.t π^k's $\in \Pi^s$ for $k \in \{1, 2, 3\}$. Black solid lines represent switch between sequences in the order $\{O_3, O_2, O_1\}$.

is a major limitation of the existing estimation techniques for transition matrix, in the presence of non-stationarity, they converge to a single most likely estimate or to the current matrix estimate rather than learning each of them individually.

Next we investigate theoretically whether likelihood rate can be used to distinguish between transition probability models. We show that for a transition from state $z_{\tau-1}(= m)$ to $z_\tau(= i)$, the likelihood rate $\Delta L_k > 0$ for all those transition matrices π^k, whose corresponding transition probability is higher as compared to transitions to other state $z_t \neq i$.

Proposition 1. *Given $(\pi)_{mv}$ updated recursively using equations (1.11)-(1.12), and a transition from state $z_{\tau-1}(= m)$ to $z_\tau(= i)$ of a Markov sequence, the likelihood rate $\Delta L_k^\tau > 0$ for all those transition matrices π^k that satisfy $\pi^k_{m,i} > \pi^k_{m,j}\ \forall j \in \{1, \ldots, K\}\backslash i$.*

Proof. Given that the transition is from $z_{\tau-1} = m$, we are interested in the m^{th} row of the matrix $(\pi)_{\text{mv}}$, let it be given as $\begin{bmatrix} x^o_{m1} & x^o_{m2} & \ldots & x^o_{mK} \end{bmatrix}$. After transition to $z_\tau(= i)$, elements of this row are updated to say $\begin{bmatrix} x^n_{m1} & x^n_{m2} & \ldots & x^n_{mK} \end{bmatrix}$ using equations (1.11)-(1.12). Clearly, the value of i^{th} element of this row increments i.e. $x^n_{mi} > x^o_{mi}$ as $\delta_{i,i'} = 1$, whereas all others decrease i.e. $x^n_{mj} < x^o_{mj}\ \forall j \neq i$. Let $L^n((\pi)_{\text{mv}}|\pi^k)$ denote the likelihood of the recursively updated transition matrix $(\pi)_{\text{mv}}$ w.r.t a arbitrary but known transition matrix π^k at instant t, while at instant $t-1$ let it be $L^o((\pi)_{\text{mv}}|\pi^k)$. Then the likelihood rate is given as $\Delta L_k^\tau = L^n((\pi)_{\text{mv}}|\pi^k) - L^o((\pi)_{\text{mv}}|\pi^k)$. Let the elements of the m^{th} row of transition matrix π^k be $\begin{bmatrix} p_1 & p_2 & \ldots & p_K \end{bmatrix}$, i.e. $\pi^k_{m,i} = p_i$. Given that $p_i > p_j\ \forall j \in \{1, \ldots, K\}\backslash i$, we need to show that $\Delta L_k^\tau > 0$. Note that since $p_i > p_j\ \forall j$ and

$\sum_i p_i = 1$, it is straightforward to deduce that $p_i > \frac{1}{K}$. Next using equation (4.5) we get,

$$\Delta L_k^\tau = \frac{C}{B(\alpha^m)} \Big[\prod_{j=1}^{K} (x_{mj}^n)^{p_j - 1} - \prod_{j=1}^{K} (x_{mj}^o)^{p_j - 1} \Big] \tag{4.6}$$

where constant C contains terms from rest of the rows that remained unchanged. Expanding the terms we get

$$\begin{aligned} \Delta L_k^\tau = C' \big[& (x_{m1}^n)^{p_1 - 1} * \cdots * (x_{mK}^n)^{p_K - 1} \\ & - (x_{m1}^o)^{p_1 - 1} * \cdots * (x_{mK}^o)^{p_K - 1} \big] \end{aligned} \tag{4.7}$$

We know that $x_{mi}^n > x_{mi}^o$, and we consider the worst possible case where rest of the elements are equal i.e., $x_{m1}^n = x_{m2}^n = \cdots = x_{mi-1}^n = x_{mi+1}^n = \cdots = x_{mK}^n = x^n$, and equal to x^o for terms at instant $\tau - 1$ thus we get,

$$\begin{aligned} \Delta L_k^\tau = C' \big[& (x_{mi}^n)^{p_i - 1} * (x^n)^{\sum_{l \neq i} (p_l - 1)} \\ & - (x_{mi}^o)^{p_i - 1} * (x^o)^{\sum_{l \neq i} (p_l - 1)} \big] \end{aligned} \tag{4.8}$$

Note $\sum_{l \neq i} (p_l - 1) = 1 - p_i - (K - 1) = -p_i - (K - 2) < 0$ as $K \geq 2$. Now for $\Delta L_k^\tau > 0$ implies

$$\big[(x_{mi}^n)^{p_i - 1} * (x^n)^{-p_i - K'} - (x_{mi}^o)^{p_i - 1} * (x^o)^{-p_i - K'} \big] > 0$$

where $K' = K + 2$, further rearranging terms and then taking log we get,

$$(1 - p_i) \log \left(\frac{x_{mi}^o}{x_{mi}^n} \right) < (p_i + K') \log \left(\frac{x^n}{x^o} \right) \tag{4.9}$$

as $\frac{x_{mi}^o}{x_{mi}^n} < 1$, the product $(1 - p_i) \log \left(\frac{x_{mi}^o}{x_{mi}^n} \right) < 0$, whereas $(p_i + K') \log \left(\frac{x^n}{x^o} \right) > 0$ as $\frac{x^n}{x^o} > 1$. Thus the equation (4.9) holds true and hence $\Delta L_k^\tau > 0$. □

Clearly from proposition 1, likelihood rate estimate from a single transition cannot decisively indicate the generating transition matrix π^k. Thus, we need to observe finite number of transitions (N_j) from each mode $j \in \{1, \ldots, K\}$, that would eventually narrow down $\Delta L_k > 0$ to a single π^k. But how many different π^k's are possible if the number of modes is K? Secondly what should be the value for N_j? Though infinitely many π^k's are possible, as each $\pi_{m,i}^k \in \mathbb{R}$ while satisfying $\sum_i \pi_{m,i}^k = 1$, we are rescued by the proposition 1 condition $\pi_{m,i}^k > \pi_{m,j}^k \; \forall j \in \{1, \ldots, K\} \backslash i$, which if applied to all the rows $m \in \{1, \ldots, K\}$ implies that effectively we need to distinguish between K^K transition matrices only, because for each row m there are K possible locations of i such that $\pi_{m,i}^k > \pi_{m,j}^k$. And thus we need to estimate likelihood rate for at least one transition ($N_j = 1 \; \forall j$)

from each of the K modes.

4.2.3 Estimation using Likelihood Rate

This section delineates the process to estimate the set Π^s from observed data sequence. For the case of hidden Markov models determination of the switches between the TPMs in the set Π^s becomes the part of getting expected sufficient statistics i.e. the **E** step of the Baum-Welch algorithm, and the **M** step remains unchanged. Hence the estimation proceeds similar to the **EM** algorithm, however the E step now requires determination of π^τ for every time step τ. We first discuss the process of learning the generating distribution π^τ $(\forall\tau)$ given the latent mode sequence z_τ, and then lay down the modified Baum-Welch algorithm for non-stationary hidden Markov model.

Estimating the set Π^s

We describe an estimation process that incrementally adds transition probability models to learn the set Π^s. We define a uniform transition probability matrix π^u having uniform Dirichlet distribution over the elements of each row parameterized by the K (recall $z_\tau \in \{1, \ldots, K\}$) dimensional vector α such that $\alpha(i) = 1\ \forall i$. For any transition $z_{\tau-1} \rightarrow z_\tau$, all the existing $\pi^k \in \Pi^s$ and a uniform transition probability matrix π^u are considered as priors for generating that transition. These priors are evaluated based on their likelihood rate ΔL and the posterior update is performed for the prior π^k or π^u with highest ΔL. Before we get into the details some definitions are in line. We define an index set $\mathcal{I}$ that stores the index k for every time step τ, such that $\pi^\tau = \pi^k$, hence the set $\mathcal{I}$ represents the generating distribution π^τ $(\forall\tau)$. Our aim is to learn the index set $\mathcal{I}$ together with the elements of the set Π^s. We had noted earlier that we need to observe ΔL for finite number of transitions (N_j) from each mode j before we evaluate our priors $\pi^k \in \Pi^s$, henceforth this is referred to as the update condition.

We begin with a single prior π^u in Π^s, which is updated after the update condition is satisfied, and a second prior π^u is added to the set Π^s so that the cardinality of Π^s denoted by K_π becomes 2. Thereafter, the posterior update is made for the prior π^k that has the maximum ΔL_k (we denote that index k by I). This update utilizes the transition counts between subsequent updates stored in the form of a matrix $\mathcal{M} \in \mathbb{N}^{K\times K}$. But if $k = K_\pi$ i.e. the current transitions are more likely w.r.t the uniform prior π^u then along with the posterior update, we also increment K_π by one and another π^u is appended as the last element of Π^s. In either case, the time steps in between the updates are assigned the corresponding value of I and are stored in $\mathcal{I}$. This learning process is summarized in algorithm 2. N_j's can take values in the range 1 to K to best fit the data. We do not suggest $N_j > K$ as this would limit the capability of the learning process to capture fast

switching between TPMs. The value set for N_j ($\forall j$) is denoted by n_π and is a tunable parameter in the learning process. Mathematically the update condition is represented by $N_j = n_\pi \ \forall j$ with $n_\pi \geq 1$.

Learning the Hierarchical transition matrix Π: The index set $\mathcal{I}$ obtained from Algorithm 2 represents hierarchical latent mode sequence for the transition distributions. Hence Π can then be obtained from I by using classical maximum likelihood estimate or by using the Bayesian approach discussed in section 1.3.5.

Algorithm 2 Estimating TPMs in LNMM

Input: Latent mode sequence $(z_1, \ldots, z_T)$, $\Pi^s = \{\pi^1 \ (= \pi^u)\}, K_\pi = 1$.
Initialize $(\pi)_{\text{mv}} = \pi^u, \mathcal{I} = \emptyset, \tau_p = 1, N_j = 0 \ \forall j$.
for $\tau = 2$ **to** T **do**
  Update $(\pi)_{\text{mv}}$ based on $z_{\tau-1} \rightarrow z_\tau$ using equations (1.11)-(1.12)
  Evaluate L_k^τ and its rate ΔL_k^τ for each $\pi^k \in \Pi^s$.
  if $N_j = n_\pi \ \forall j$ **then**
    if $|\Pi^s| = 1$ **then**
      Update π^1 using $\mathcal{M}$, augment $\Pi^s = \{\pi^1, \pi^2 \ (= \pi^u)\}, K_\pi = 2$.
    else
      $I = \arg\max_k \sum_{j=\tau_p}^{\tau} \Delta L_k^j$,
      if $I = K_\pi$ **then**
        Update π^I using $\mathcal{M}$, set $(\pi)_{\text{mv}} = \pi^I$.
        $K_\pi \leftarrow K_\pi + 1$, augment $\Pi^s = \{\pi^1, \ldots, \pi^{K_\pi} \ (= \pi^u)\}$
      else
        Update π^I using $\mathcal{M}$, set $(\pi)_{\text{mv}} = \pi^I$.
      end if
    end if
    $\mathcal{I}(\tau_p : \tau) = I, \tau_p \leftarrow \tau, \mathcal{M} \leftarrow \emptyset, \Delta L_k \leftarrow 0 \ \forall k, N_j \leftarrow 0 \ \forall j$.
  end if
end for
Output: $\Pi^s, \mathcal{I}$.

Modified Baum-Welch for Layered HMM

To elaborate the modified Baum-Welch algorithm for layered hidden Markov model we develop upon the details of classical EM algorithm given in section 17.5.2 of [90]. We begin with an initial guess on the parameters θ^{old} that comprises of distribution parameters corresponding to each latent mode, initial distribution on latent states $\pi^0(j) = p(z_1 = j)$ and $\Pi^s = \{\pi^u\}$ i.e. $\pi^\tau = \pi^u \ \forall \tau$. Every iteration of **E** step is then composed of two steps; one regular step followed by algorithm 2 as a second step that determines the set Π^s. In the first step, we compute smoothed marginals, $\gamma_\tau(j) = p(z_\tau = j | x_{1:T})$ by running the forwards-backwards algorithm with equations modified to accommodate different transition matrix π^τ at a time step τ as follows (shown in matrix-vector

Algorithm 3 Inference in LNMM

Input: Latent mode sequence $\{z_1, \ldots, z_T\}$, Π^s.
Initialize $(\pi)_{\text{mv}} = \pi^u, \mathcal{I} = \emptyset, \tau_p = 1, N_j = 0 \; \forall j$.
for $\tau = 2$ **to** T **do**
  Update $(\pi)_{\text{mv}}$ based on $z_{\tau-1} \rightarrow z_\tau$ using equations (1.11)-(1.12)
  Evaluate L_k^τ and its rate ΔL_k^τ for each $\pi^k \in \Pi^s$.
  if $N_j = n_\pi \; \forall j$ **then**
    $I = \arg\max_k \sum_{j=\tau_p}^{\tau} \Delta L_k^j$,
    $\mathcal{I}(\tau_p : \tau) = I, \tau_p \leftarrow \tau, N_j \leftarrow 0 \; \forall j, \Delta L_k \leftarrow 0 \; \forall k$
  end if
end for
Output: $\mathcal{I}$

form)

$$\boldsymbol{\alpha_\tau} \propto \boldsymbol{\psi_\tau} \circ ((\boldsymbol{\pi^\tau})^T \boldsymbol{\alpha_{\tau-1}}) \tag{4.10}$$

$$\boldsymbol{\beta_{\tau-1}} = (\boldsymbol{\pi^\tau})^T (\boldsymbol{\psi_\tau} \circ \boldsymbol{\beta_\tau}) \tag{4.11}$$

$$\boldsymbol{\gamma_\tau} \propto \boldsymbol{\alpha_\tau} \circ \boldsymbol{\beta_\tau} \tag{4.12}$$

where $\psi_\tau(j) = p(x_\tau | z_\tau = j)$ is the local evidence at time $\tau, \alpha_\tau(j) = p(z_\tau = j | x_{1:t})$ is filtered marginal initialized as $\alpha_1 = \text{normalize}(\psi_1 \circ \pi^0)$, and $\beta_\tau(j) = p(x_{t+1:T} | z_\tau = j)$, with base case $\beta_T(j) = 1 \; \forall j$. Replacing π^τ by a single matrix π results in original equations of [90]. In the second step, we run algorithm 2, for which the latent mode sequence is determined by assigning each observation (x_τ) to its most likely mode, i.e. $z_\tau(j) = \arg\max_i \gamma_\tau(i)$. The M step remains same and updates the distribution parameters θ_{z_τ} for each latent mode $z_\tau \in \{1, \ldots, K\}$. For second iteration onward, θ^{old} comprises of the set Π^s and the index set $\mathcal{I}$ (from Algorithm 2) along with the updated distribution parameters. Iterations are performed until parameter convergence or up to the maximum limit set for number of iterations.

4.2.4 Inference using LNMM

Given a observation sequence and the set of TPMs in Π^s, goal is to infer the generative transition probability matrix $\pi^\tau \in \Pi^s \quad \forall \tau$. Thus inference involves determination of the set $\mathcal{I}$. For hidden Markov sequence i.e. in case of HMM/SLDS/MJS, if the data is not fully observed, an initial guess of the latent mode sequence is generated by assigning each observation to its most likely mode. Set $\mathcal{I}$ is inferred based on this initial guess. Ultimate inference of the latent mode sequence is obtained by using modified forward-backwards equations (4.10)-(4.12) and the set $\mathcal{I}$. The inference

procedure to obtain $\mathcal{I}$ is given in Algorithm 3.

4.2.5 Existence of non-stationarity in Markov models

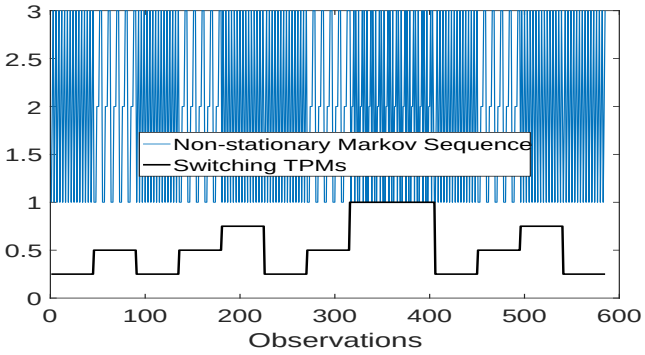
So far we have discussed the estimation process for multiple transition probability matrices in order to learn an expressive model for non-stationary Markov chains. But the question remains, how do we check whether the observed Markov sequence is really non-stationary? Answer to this question is hidden within the estimation process described in the previous section. Algorithm 2 identifies the existence of more than one transition model by evaluating and comparing the likelihood rate of the uniform transition matrix π^u with the transition model existing in the set Π^s. Based on this observation, we describe two step process to conclude existence of non-stationarity in a observed Markov sequence. First use a method of choice to learn the parameters of transition probability matrix (say π^1) and associated distribution parameter θ_{z_τ} over observations in case of HMM/SLDS/MJS. Second, infer likelihood rate for π^1 and π^u using the inference procedure described in section 4.2.4. If the likelihood rate for π^u increases while that of π^1 decreases over a number of observations (number given by the update condition in Algorithm 2) then it should be concluded that the Markov sequence under study is non-stationary. We demonstrate this check in the case of bee dance dataset in the results section.

4.3 Experiments on Synthetic Data-set

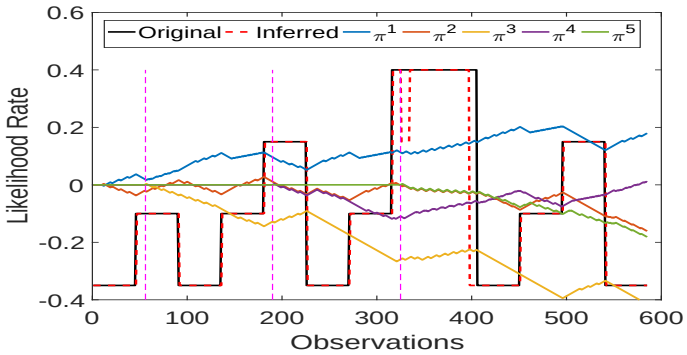
We first demonstrate and validate the estimation approach for a non-stationary Markov sequence utilizing Algorithm 2. We consider four randomly generated transition probability matrices π^k ($k \in \{1, 2, 3, 4\}$) for a random variable X that switches between three modes i.e. $K = 3$. A non-stationary Markov sequence of observations generated using these TPMs is shown in Figure 4.3a, along with the switching between the TPMs. We input this sequence to Algorithm 1, that infers switch between the TPMs as shown in Figure 4.3b. Our method successfully identified four generating transition models and almost perfectly captures the transitions happening among these models. Hamming distance error calculated by greedily mapping the indices of the estimated state sequence to those maximizing overlap with the true sequence (as defined by [91]) was 0.046. Figure 4.3b also plots the likelihood rate of recursively updated $(\pi)_{\text{mv}}$ w.r.t. each TPM π^k. Clearly the absolute increase in the likelihood rate ($\sum_{j=\tau_p}^{\tau} \Delta L_k^j$ summed between updates) of the transition model that generates a particular sequence is higher than that of the other models. Most importantly, the Algorithm initiates a new uniform transition model π^u as soon as the change in transition model is detected (dashed magenta line). The slight delay in detection is on account of meeting

the update condition. Likelihood rate for the last model π^5 never increases and is never updated, thus the Algorithm accurately detects four switching TPMs.

Parameter n_π was tuned to minimize the hamming distance error and was set equal to 2. Thus we need to validate whether this approach would generate similar performance for different realizations of randomly generated transition matrices without tuning n_π. We performed 100 experiments with matrices and sequences generated randomly. Average hamming distance in this case was 0.116 with a standard deviation of 0.057. Thus our approach performs very well in detecting and learning multiple TPMs existing in non-stationary Markov sequence.



(a)



(b)

Figure 4.3: (a) Nonstationary Markov chain generated by switching TPMs, (b) Estimated and the actual switching between the TPMs, along with the likelihood of recursively updated $(\pi)_{\text{mv}}$ w.r.t. each TPM π^k. Notice the rise in the likelihood of a transition matrix corresponding to its sequence.

4.3.1 Synthetic Dataset

We evaluate the performance of the proposed layered non-stationary hidden Markov model (LNHMM) using synthetic data generated from a 5 state HMM with 2d Gaussian emissions, Figure 4.4a is an

Table 4.1: Computation time for estimation using Baum-Welch (BW), MCMC, sticky HDP-HMM (sHDP) and LNHMM, computed on i7-6500K CPU @2.5GHz and 16 GB RAM machine.

METHOD	BW	MCMC	SHDP	LNHMM
TIME (SEC)	43.1	196	394.6	63.2

example showing the latent mode sequence (in black) laid over the true state sequence. The purpose is to quantify the benefits of using likelihood rate to estimate all the existing transition matrices that are generating the data as compared to the existing methods that learn a single transition matrix. As such this data was generated using three characteristically different transition probability matrices. One of the TPM had high probability of self transition, another had a high probability of leaving the current state and the last one was generated randomly. We use the estimation method described in section 4.2.3, to learn the model parameters as well as the set of transition matrices Π^s existing in the time series data of Figure 4.4a; the Algorithm identified three TPMs and the switching between them, this is overlaid on the latent mode sequence in Figure 4.4b. To compare we also learn the model using existing methods; standard Baum Welch Algorithm, Monte carlo Marko chain (MCMC [92]) and Bayesian non-parametric sticky-HMM approach [91]. We tested the models obtained using these methods by performing inference over a set of 100 different time-series data generated by the same 5 state HMM but with random switch between the three underlying TPMs. For MCMC we took 5000 draws after a burn-in of 1000 draws and chose the best from last 500. For sticky-HDPHMM we ran 10 chains of 1000 Gibbs iterations using blocked sampler and picked the best solution. Inference for LNHMM was performed as detailed in section 4.2.4. Performance was evaluated in terms of predictive likelihood of data and percentage error in Viterbi assignments. Results plotted in Figures 4.4c-4.4d, show that LNHMM has far higher predictive likelihood as compared to other methods and does very well in Viterbi assignments. And most significantly, the computational requirement for LNMM is closer to Baum-Welch (see Table 4.1). These experiments clearly suggest that the likelihood rate based estimation of multiple transition matrices achieves accurate inference for non-stationary Markov sequences. Further, the layered Markov model is an expressive model for non-stationarity induced by recurrent transition models and is computationally efficient.

4.4 Experiments on Honey Bee Dance data-set

We test the LNMM on a set of six honey bee dance sequences, this dataset has previously been used to demonstrate different switching linear dynamical systems (SLDS) approach developed by [93]

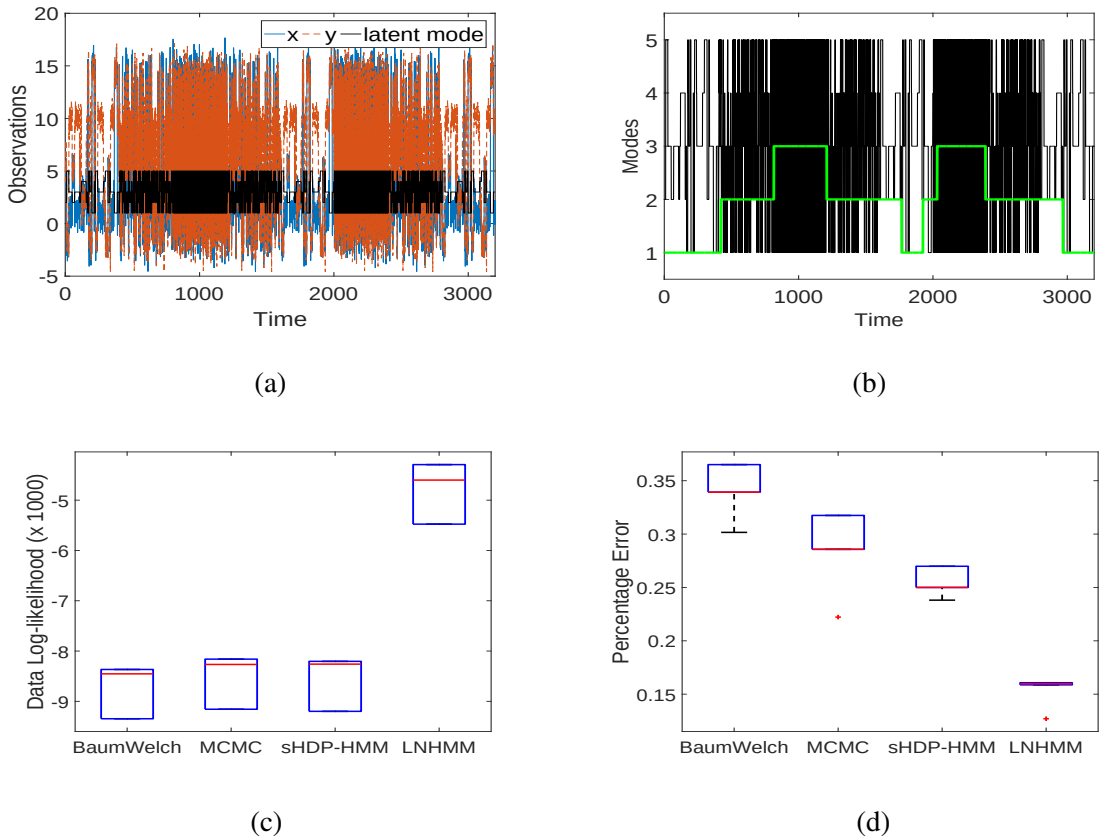
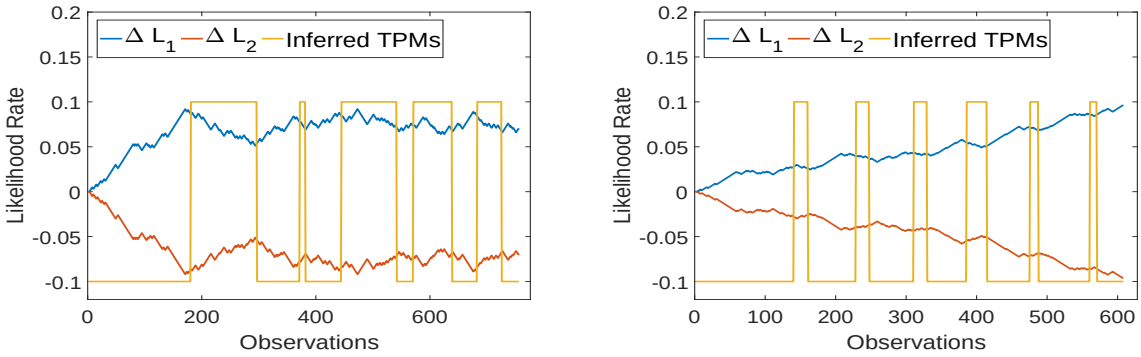


Figure 4.4: (a) Latent mode sequence (black) and true state sequence for a five state HMM (2d Gaussian emissions), (b) Enlarged view of latent mode sequence overlaid with switches between transition matrices obtained using Algorithm 2, (c) Predictive likelihood over 100 timeseries data generated by random switching between the TPMs present in (b), (d) Viterbi assignment errors.

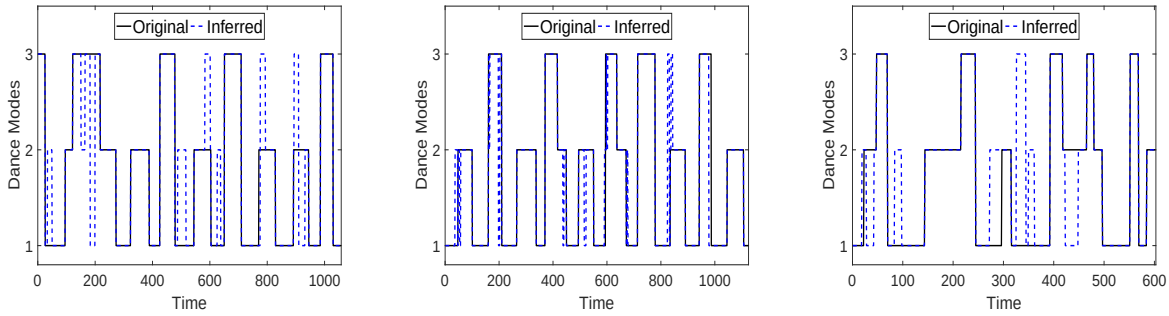
and [94]. Primary aim in this experiment is to segment the dance sequence into three distinct dance modes and compare performance with the known ground truth labels. The data consists of measurements $\mathbf{y}_t = [\cos(\theta_t)\ \sin(\theta_t)\ x_t\ y_t]$, where (x_t, y_t) is the 2-D coordinates of the bee's body and θ_t its head angle. At the outset we tested whether the bee dance sequence exhibit non-stationarity using the procedure described in section 4.2.5. We check whether it suffices to model the bee dance dataset using the dynamic parameters and transition model inferred by the approach given in [93]. Test results shown in Figure 4.5 indicates that the likelihood rate for uniform model π^u increases in comparison to π^1 for the sequences 4 and 6, hence Algorithm 3 infers π^u as the transition model. This exercise indicates that the bee dance sequences can indeed be better modeled using non-stationary Markov model.

In testing our approach for each of the bee dance sequence, we used *matrix-normal inverse-Wishart* prior to learn the dynamic parameters (θ_{z_τ}) of linear dynamical system for each dance mode $(z_\tau \in \{1, 2, 3\})$ similar to [93]. Our results in terms of median label accuracy for the six



(a) Sequence 4 (b) Sequence 6

Figure 4.5: Likelihood rate for models π^1 (ΔL_1) and π^u (ΔL_2), obtained using Algorithm 3, along with the inferred switch in TPMs.



(a) (b) (c)

Figure 4.6: (a)-(c) Estimated mode sequences using LNMM overlaid on ground truth for sequences 1, 2, and 3.

dance sequences along with the number of identified transition models is shown in Table 4.2. The table also reports the performance of Hierarchical Dirichlet process HMM (HDP-HMM) [93] using unsupervised and partially supervised Gibbs sampling, and that of the supervised "parameterized segmental SLDS" (PS-SLDS) and SLDS procedures [94]. Inference using LNMM for the sequences 1 to 3 is almost two times better than the unsupervised approach; see Figure 4.6 for inferred mode sequences. LNMM though unsupervised performed almost equivalent to the supervised approach of SLDS and PS-SLDS which employs ground truth labels for all but one sequence to perform inference. Thus likelihood rate and the resulting LNMM model is a key approach for estimation and learning for non-stationary Markov sequences observed in real world datasets.

Table 4.2: Median label accuracy for the six bee dance sequences of LNMM compared with HDP-HMM unsupervised (UN), HDP-HMM partially supervised (PS), and Data Driven-MCMC based SLDS and PS-SLDS.

SEQUENCE	1	2	3	4	5	6
LNMM (TPMS)	**83.6** (4)	**94.1** (4)	**84.6** (3)	91.5 (2)	**93.5** (3)	90.1 (2)
HDP-HMM UN	46.5	44.1	45.6	83.2	93.2	88.7
HDP-HMM PS	65.9	88.5	79.2	86.9	92.3	89.1
SLDS	74.0	86.1	81.3	**93.4**	90.2	90.4
PS-SLDS	75.9	92.4	83.1	**93.4**	90.4	**91.0**

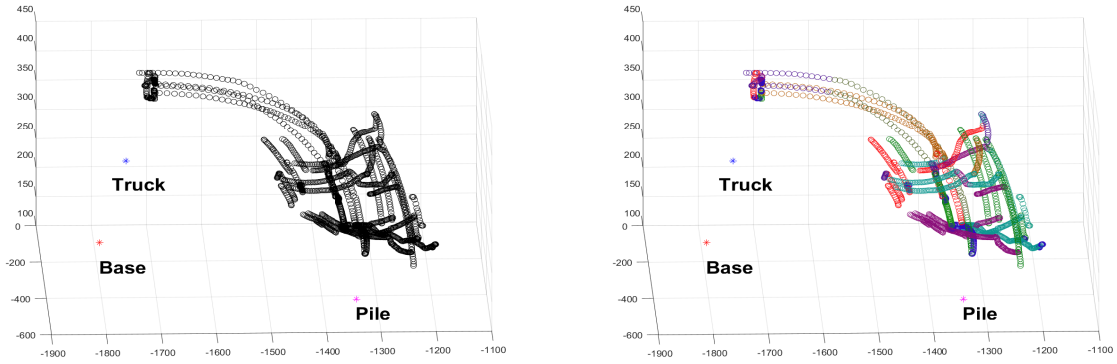
4.5 Experiments on Constructions Tasks

Our goal is to infer switch between different construction tasks during an entire construction activity. For this purpose we first learn transition model over action primitives for each task. This gives us the set Π^s that comprises of transition model for each construction task. We then demonstrate how the likelihood rate based inference can be used to infer switching between the tasks that occur in a single continuous demonstration of a construction activity. Finally, we also attempt to learn the number of different tasks that were present in a construction activity and simultaneously learn their transition models.

Experiments were performed on a $1/14^{th}$ scaled 345D Wedico excavator model, a 4 d.o.f hydraulic robotic arm manipulator, controlled by a radio transmitter (see Figure 3.5 for the robot's description). Refer section 3.4 for more details on the experimental platform and the set up. We collected a demonstration consisting of three different tasks constituting a truck loading activity. These tasks included leveling, truck loading and piling operations. A plot of unsegmented demonstration trajectories in the end-effector pose w.r.t the robot's base frame is shown in Figure 4.7a. From the trajectories, it is apparent that there were two cycles of truck loading operation and significant manipulations w.r.t the sand pile. Manipulation w.r.t sand consisted of piling operation, which was executed before truck loading, and leveling operation executed post truck loading cycles.

4.5.1 Likelihood rate based Inference of Construction Tasks

We first decompose continuous state-action trajectories from expert demonstrations into state-action primitive segments. This is done using the segmentation procedure detailed in section 3.4.1, resulting action primitive segments are depicted by unique colors in Figure 4.7b. Total to 13 action

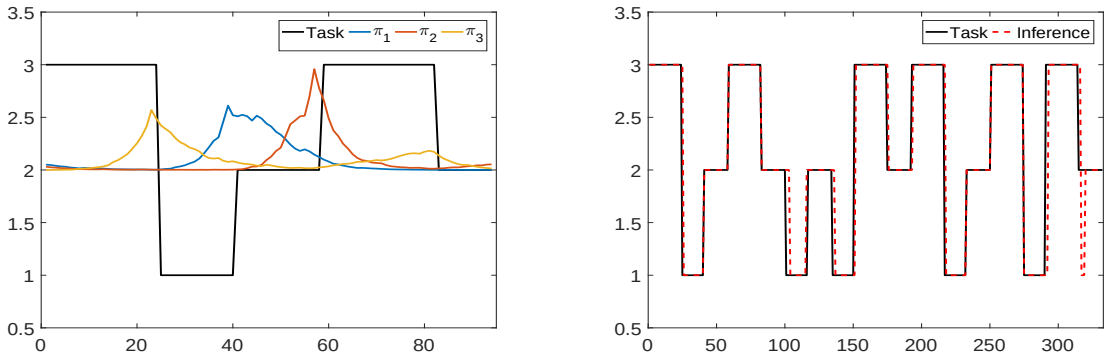


(a) Unsegmented demonstration trajectories

(b) Action primitive based segmentation

Figure 4.7: Segmentation in end-effector's pose (x, y, z) w.r.t the base frame of the robot for a truck loading activity comprising of piling and leveling of sand operations.

primitives were discovered. The sequence of action primitives is used to learn a transition model using the procedure described in section 1.3.4. We learn separate transition models for each of the three tasks, piling, truck loading, and leveling, denote by the set $\Pi^s = \{\pi_1, \pi_2, \pi_3\}$, respectively. Each transition model has the dimension, $\pi_i \in \mathbb{R}^{13\times 13}, \forall i$. The dimension of transition models in this problem is quite high in comparison to the models used in the experiments of previous two sections, and is thus a challenging problem. The known transition models are utilized to evaluate the likelihood as in the equation (4.5) w.r.t a recursively updated transition matrix $(\pi)_{\text{mv}}$ (that is updated as described in section 1.3.5).



(a) Likelihood of transition models

(b) Inferred switching between construction tasks

Figure 4.8: Likelihood rate based inference of switching between construction tasks. Plot of task represents switching between three different tasks indexed as $3-$leveling, $2-$truck loading, $1-$piling. (a) Plots the likelihood of transition models w.r.t the mean variance estimate $(\pi)_{\text{mv}}$. Clearly likelihood rate decisively determines the switch between the transition models. (b) Inferred transition model is overlaid with original switching.

Resulting likelihood values are plotted in Figure 4.8a. Figure also shows the construction task

that is switching between the three tasks. Clearly, the rate of change of likelihood, i.e. the likelihood rate decisively indicates the change in transition models. Figure 4.8b shows all possible switches between the three construction tasks and their inference. Overall, there were 16 switch between tasks, and an error of 1.13 action primitives was noted per switch between the tasks. Additionally, a minimum of three action primitives need to be observed in order to estimate the likelihood rate. This implies a minimum of 4 action primitives are required before the robot can figure out a switch in the task.

CHAPTER 5

SHARED CONTROL

5.1 Introduction

In many engineering applications where the operations are repetitive and within well-controlled surroundings, robots equipped with various sensing devices and control algorithms have replaced the need for human operators. However, there are many real-world applications where human presence is still crucial for guaranteeing safety and robustness. Examples include agricultural or construction tasks requiring co-robots such as excavators, tractors, backhoes, etc.; piloting of vehicle systems; and monitoring of safety-critical industrial processes. Although much work has been done to automate construction co-robots and provide a level of autonomy based on sensing their surroundings [2, 37], these have not come to fruition due to the level of uncertainty and dynamically changing environment due to the presence of off-nominal situations. Thus it seems that the presence of human operators will remain critical to these and many other real-world applications in the foreseeable future because of two key reasons: Firstly, the sensing and perception abilities of humans have been difficult to replicate in machines; and Secondly the higher level contextual knowledge of the world that humans possess has not yet been successfully imbibed in machines. As a result, even when autopilots have been around for over a generation in aviation, the presence of human pilot in the cockpit is still crucial in dealing with off-nominal situations.

In applications where humans and machines have to control a dynamical system together, the key question of interest is what should be the level of autonomy that is appropriate to ensure high task performance without compromising safety. This problem has been widely studied in the ergonomics and human-machine interaction literature [95]. The general consensus seems to indicate that while autonomy may be better suited to improve performance at the implementation level of task, ignoring human inputs can lead to costly mistakes when unforeseen things happen. Based on this insight, the most naive implementation of shared machine and human control system would involve a physical switch that determines whether the system is in human or autonomous control. However, this technique suffers from the drawback that the system does not benefit from the humans' perceptive abilities when in autonomous mode, and is otherwise hindered by humans'

potential lack of skill.

An alternative to the switched approach is to design shared control systems so that the total control input provided to the system is some combination of human and machine input [96]. Such a shared control design has been successfully applied to semi-autonomous wheelchair navigation problem [97–99]. In particular [97] proposes a mechanism that weights human and autonomous agent based on their local efficiency in terms of directiveness, safety and smoothness. In these applications, shared control plays a role of assistive technology. The design of shared control for real world co-robots remains a scarcely explored area of research. In a recent work, Enes and Wayne [100] pose an interesting and elegant navigation problem that captures the essential attributes of shard control problems typically suitable for mixed human-machine autonomy approach. Their setup is based around the Zermelo's navigation problem, that is the task for a human-machine team to navigate a ship to a desired location in the presence of varying currents. The blended shared controller proposed by Enes and Waynes adapts a shared control coefficient α so that if the operator brings the ship close to the optimal trajectory, then autonomous control takes over. Existing work by Enes and Wayne and others has repeatedly shown that the blended shared control approach guarantees improved performance in nominal conditions. However, the performance of these methods in off-nominal conditions has not been reported.

The key question that we seek to address in this paper is that whether a state-dependent shared control approach, similar to that proposed by Enes et al., will be successful in off-nominal task conditions. To emulate off-nominal conditions, we consider the case of suddenly appearing obstacle in the path of the ship in context of the Zermelo's navigation problem. In this case, our results show that the blended shared control approach will lead to suboptimal performance if the operational environment deviates from nominal. In particular, we show that when the suddenly appearing obstacle pops up, the blended control approach suffers, because it is too slow in yielding control back to the operator. To address this issue, we propose a new shared control technique that actively utilizes inferred operator intent. In this technique, intent of the operator is defined mathematically and is utilized to determine the shared control coefficient. In essence, our approach blends autonomous shared control with the operator's inputs if the operator's calculated intent indicates their intention to move towards the optimal trajectory, and quickly concedes control to the human operator when the operator makes a deliberate effort to move away from the optimal trajectory. Our theoretical results demonstrate that the Intent Aware Shared Control (IASC) approach proposed here leads to at least as good performance as blended shared control while providing larger obstacle reaction times. Experiments with human operators corroborate these findings, demonstrating a much lower number of obstacle collision incidences with IASC.

An alternate shared control approach [101], deals with control transfer to autonomous agent under off-nominal situations which are not perceived by human operators, an entirely opposite

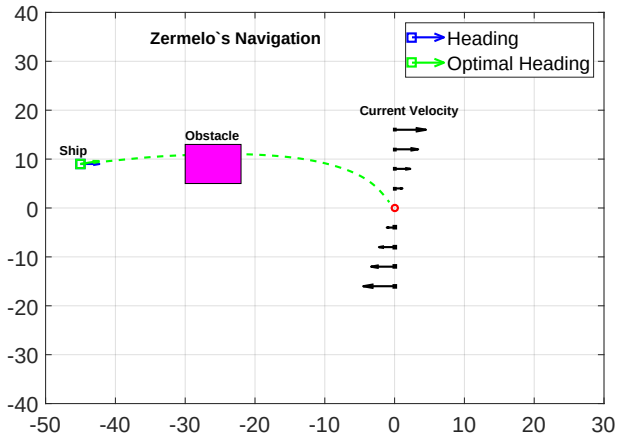


Figure 5.1: Zermelo's Navigation with Obstacle, with a typical Off-Nominal situation due to the presence of an obstacle not known a-priori to the autonomous agent.

viewpoint of the shared control problem than that discussed in this work. The rest of the paper is organized as follows, Section II presents background information and formulates the blended shared control problem. Section III provides the intent aware shared control formulation, while Section IV presents theoretical and simulation based results, before describing results with human subjects on Zermelo's navigation problem with a pop-up obstacle. The paper is concluded in Section V.

5.2 Blended Shared Control for Zermelo's Navigation

5.2.1 Zermelo's Navigation Problem

Zermelo's navigation problem is a classic optimal control problem, where the objective is to navigate a ship to the origin in the presence of linearly varying current as shown in the figure 5.1. The original objective of the problem as described in [102] is to minimize the transit time to the origin, for which the closed form solution is shown to exist. In its original form, this problem was used as a case study by [100] to investigate into shared control of a ship, where the control is combined from an human operator and an autonomous agent. In this paper, a slight modification described later in this section is used to further investigate shared control that captures the intention of the operator.

In Zermelo's problem, a ship (modeled as a particle) travels at a constant speed V relative to the water. The only control available to an operator is the ship's heading θ. The equations of motion

are given by

$$\begin{aligned} \dot{x} &= V cos\theta + u(y) \\ \dot{y} &= V sin\theta \end{aligned} \tag{5.1}$$

where θ is the ship's heading measured from the x-axis, (x, y) are its coordinates, and $u = Vy/h$ is the velocity of the current, with constant h that determines its intensity along the y-axis. In this paper we modify the Zermelo's Navigation problem to include an obstacle that appears dynamically during a simulation. Their existence is not known to the autonomous controller, and hence the operator control plays an important role in navigating the ship safely past the obstacle and finally to the origin.

5.2.2 Blended Shared Control

Enes and Wayne [100] had formulated Blended shared control law, that combines operator's command (u_0) with the optimal command calculated by the robot agent ($\tilde{u}$). The final control input x is given by

$$x = u_0 + \delta \tag{5.2}$$

where δ is a command perturbation term, which is calculated by shared controller depending on the machine model and/or optimization of cost function internal to the robot. In their formulation, with an objective of decreasing the time required to complete a given task, they considered the perturbation term to be a linear function of the error between the operator input u_0 and the robot agent's input ($\tilde{u}$) i.e. $\delta = -\alpha\Delta_u$, where $\Delta_u = u_0 - \tilde{u}$, replacing δ in (5.2) we have,

$$x = u_0 - \alpha\Delta_u \tag{5.3}$$

where $\alpha \in [0, 1]$ is the blended shared control parameter, which is the major subject of design in this paper. Clearly, for $\alpha = 0$ the system is under manual control (i.e. $x = u_0$) and when $\alpha = 1$ the system is fully autonomous (i.e. $x = \tilde{u}$).

5.2.3 Zermelo's Navigation using Blended Shared Control

In the application of blended shared control to Zermelo's navigation, ship's heading θ results from the combination of the operator's input u_0 and the autonomous agent's input $\tilde{u}$. The operator's input u_0 at the joystick materializes into the operator commanded heading $\theta_0 = \theta_i + \psi \int u_0 dt$, where θ_i is the present ship heading and ψ is a constant parameter for ship response. Blended

shared control law proposed in [100] is a function of the error Δ in commanded heading given by,

$$\Delta = \theta_0 - \tilde{\theta} \tag{5.4}$$

where $\tilde{\theta}$ is the optimal heading for the current ship location (x, y). And the parameter α was defined as,

$$\alpha = \max(0, 1 - d/d_0).\max(0, 1 - (\Delta/\Delta_0)^2) \tag{5.5}$$

this function allows manual operation if the ship is greater than distance d_0 away from the goal or if the operator commanded heading deviates from the optimal by value greater than Δ_0. This blended shared control relinquishes control authority to the operator only in the presence of large "errors" between the operator input and the autonomous agent's input. Note that at any given point the heading of the ship is given by $\theta = \psi \int x dt$, with x obtained using (5.3).

5.3 Intent Aware Shared Control

5.3.1 Introduction

Human operators have unique capability of quickly perceiving the state of their surrounding to make safety critical decisions in dynamically changing environment. The objective of the proposed Intent Aware Shared Control (IASC), is to utilize this perceptive capability of human operators while retaining the benefits of improved performance offered by shared control. Blended shared control law reviewed in section 5.2.2 does well in utilizing the effectiveness of the autonomous agent and hence in minimizing the time required to complete the Zermelo's navigation task. However the purpose of the shared control can truly be realized, if the control law design combines the utility of a human operator with the effectiveness of the autonomous agent. This is critical in ensuring safe execution of task at hand.

We present our formulation for a single input system. Extension to multi-input system is non-trivial and will be pursued as future work. Evaluation of single input system is expected to provide significant insight into the design of such blended shared control for different multi-input systems. We assume that a human operator provides rational control input (u_0) to the system, with an objective of performing stated task. So there are two important aspects of the human control input that are available to design a shared control law. First is the error between the optimal input ($\tilde{\theta}$) (determined by the autonomous or robotic agent) and the heading desired by the operator (θ_0), second is the rate ($\dot{\gamma}$) at which the human operator's input differ from that of the optimal. Mathematically,

former is given by the equation (5.4), and we define latter as

$$\dot{\gamma} = \frac{d}{dt}(|\theta_0 - \tilde{\theta}|) = \frac{d}{dt}(|\Delta|) \tag{5.6}$$

The key idea is that the intent of the operator can be captured by the rate $\dot{\gamma}$, and hence we chose to call it as the human intent. Investigation of equation (5.6) shows that there are following three facets to human intention,

- $\dot{\gamma} > 0$ implies that the human intends to differ from the optimal input $\tilde{\theta}$,
- $\dot{\gamma} < 0$, implies that the human is in agreement with the optimal input $\tilde{\theta}$ and intends to follow the same,
- $\dot{\gamma} = 0$, implies that the human is in passive agreement with the optimal input $\tilde{\theta}$.

First two facets are self explanatory, the third one implies that the human considers that the control inputs being generated by the autonomous agent are optimal for the intended task and hence does not bother to provide his/her input or in other words remains passive. Further clarity would result as we discuss the mathematical model of shared control based on the human intent $\dot{\gamma}$, in section 5.3.2.

Another shortcoming of the blended shared control is that the operator completely loses control authority to autonomous agent in the case $\alpha = 1$ (see equation 5.3). This situation can certainly occur as is apparent from equation (5.5) for the Zermelo's navigation problem. However, it is easy to avoid this situation by employing a simple fix of limiting the maximum attainable value for α to $\alpha_0 < 1$. In this paper, we perform experiments using a fixed α_0, and leave the investigation of the effects of variation of α_0 to future work.

5.3.2 Mathematical Model

We consider a system,

$$\begin{aligned} \dot{x} &= f(x, x) \\ x &= g(u_0, \tilde{u}, \alpha) \end{aligned} \tag{5.7}$$

where the control input u is the function of operator input u_0 and the autonomous controller input $\tilde{u}$ (also referred to as optimal input) and the blended shared control parameter α. In the previous blended shared control formulation [103], α is a function ξ of the error between the commanded

state x that results from independent application of u_0 and $\tilde{u}$,

$$\alpha = \xi(\Delta) \tag{5.8}$$

An example of Δ and the function ξ are equations (5.4) and (5.5) respectively. In contrast, in the presented formulation we define α as

$$\alpha = \xi(\dot{\gamma}) \tag{5.9}$$

where $\dot{\gamma}$ is a measure of human intent given by equation (5.6). Mathematically, intent is the rate of change in the absolute deviation between the states commanded by the operator input and the autonomous agent's input. Next we propose an appropriate function for $\xi(\dot{\gamma})$, based on the question that, how should the shared control react to the human intent $\dot{\gamma}$? Logic dictates that shared control should certainly decrease the impact of autonomous control (dictated by $\tilde{u}$) when the human intends to differ from the optimal trajectory i.e. $\dot{\gamma} > 0$ and vice-versa. Building on this reasoning, we propose the following design for the shared control parameter,

$$\xi(\dot{\gamma}) = \begin{cases} \alpha_0(1 - e^{-t/\tau}) & \dot{\gamma} < 0 \\ \alpha_0 e^{-t/\tau} & \dot{\gamma} > 0 \\ \alpha_{prev} & \dot{\gamma} = 0 \end{cases} \tag{5.10}$$

In our design, when the human remains passive $\dot{\gamma} = 0$, we continue with the previous value of α denoted as α_{prev}. In the other two cases, the effectiveness of the autonomous agent is decreased or increased based on the human intent. Time t is counted starting from zero whenever the intent switches between these cases, and the constant τ determines the response characteristics. The final form of control input for IASC is

$$x = u_0 - \xi(\dot{\gamma})(u_0 - \tilde{u}) \tag{5.11}$$

In the theoretical results section 5.4, we discuss this controller's performance with that of Blended shared control law given as

$$x = u_0 - \xi(\Delta)(u_0 - \tilde{u}) \tag{5.12}$$

5.4 Theoretical Results

First we analyze the performance of BSC and IASC controllers, in terms of their capability to relinquish control authority to human operator in the presence of an adhoc obstacle in Zermelo's navigation. Mathematically, control is relinquished by an autonomous agent if the function $\xi(\cdot) \rightarrow$

0. We assume that the operator takes a maximal evasive action in the event of unforeseen obstacle by driving u_0 to its maximum value of $\bar{u}_0$.

Proposition 2. *Given the system defined by (5.1), with the function g for the BSC and IASC controllers given by the equations (5.12) and (5.11) respectively. If t_1 and t_2 is the time taken by BSC and IASC controller respectively to achieve $\xi(\cdot) = \epsilon\alpha_0$ ($\epsilon \to 0$) under the maximal operator control input $u_0 = \bar{u}_0$ ($< \Delta_0$) and $\tau < \frac{1}{ln(1/\epsilon)}$, then $t_1 > t_2$.*

Proof. For both the controllers specified in the proposition, we consider that they remain active irrespective of the ship's distance d from the origin and that the maximum attainable value for α is α_0, thus from the equation (5.5) for BSC we have

$$\xi(\Delta) = \alpha_0 \text{max}(0, 1 - (\Delta/\Delta_0)^2) \tag{5.13}$$

From equation (5.12), for $\xi(\Delta) = \epsilon\alpha_0$, Δ should satisfy $1 - (\Delta/\Delta_0)^2 = \epsilon$, which implies $(\Delta/\Delta_0)^2 = 1 - \epsilon \approx 1$. Hence using equation (5.4), we obtain $\Delta = \Delta_0 = \theta_0 - \tilde{\theta}$, this implies $\theta_0 = \tilde{\theta} + \Delta_0$. Given that t_1 is the time taken by the ship to rotate to θ_0 given the rate $\bar{u}_0$, we have $\bar{u}_0 t_1 = \theta_0 = \tilde{\theta} + \Delta_0$,

$$t_1 = \frac{(\tilde{\theta} + \Delta_0)}{\bar{u}_0} > 1 + \frac{\tilde{\theta}}{\bar{u}_0} > 1$$

Now consider the case of IASC, since the operator is performing an evasive maneuver, his intent $\dot{\gamma} > 0$, hence for $\xi(\dot{\gamma}) = \epsilon\alpha_0$, we have $\alpha_0 e^{-t_2/\tau} = \epsilon\alpha_0$, which implies that

$$t_2 = \tau ln(1/\epsilon) < 1, \quad \text{given } \tau < \frac{1}{ln(1/\epsilon)}$$

Hence $t_2 < t_1$. □

Our next proposition shows that the IASC will never be strictly worse than the manual control in terms of minimum time required to complete a task and would perform better as compared to the BSC. We assume that the human operator always provides a rational control input of the form $u_0 = \tilde{u} + \eta$ such that the magnitude of η decreases monotonically with time, until the goal is reached. Here, the rationality of an operator is characterized by the fact that the operator drives his input u_0 towards the optimal $\tilde{u}$.

Proposition 3. *Given a starting location $\boldsymbol{x_0}$, and governing equation (5.1), and the existence of time-optimal path for $x = \tilde{u}$ starting at x_0. Let T_{MC}, T_{BSC}, T_{IA} be the time required for manual control, BSC and IASC respectively, to reach the goal. If the operator input is governed by $u_0 = \tilde{u} + \eta$ s.t. $\dot{\eta} < 0$ $\forall t$, then $T_{IA} < T_{MC}$ and $T_{IA} < T_{BSC}$.*

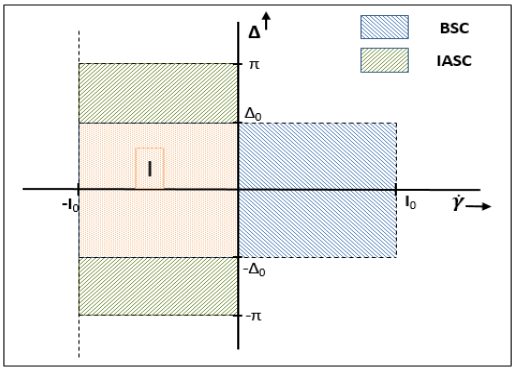


Figure 5.2: Phase plot for control effectiveness (i.e. $\alpha > 0$) in the error-intent ($\Delta - \dot{\gamma}$) plane, plotted for any location (x, y) in Zermelo's Navigation problem. $|\dot{\gamma}| < I_0$. Region I is common to both the controllers. A controller reactive to off-nominal situation should cover larger area in the left half plane. IASC occupies region larger than that of BSC in the left half plane (see proof of proposition 5).

Proof. For BSC or IASC, given $u_0 = \tilde{u} + \eta$, and using equation (5.12) we have that,

$$x = \tilde{u} + (1 - \xi(\cdot))\eta$$

since $\xi(\cdot) < 1$, the shared control law always drive the control input x closer to the optimal $\tilde{u}$, in comparison to the manual control where $u_0 = \tilde{u} + \eta$, hence $T_{BSC} < T_{MC}$. Also, $\eta = u_0 - \tilde{u}$, which implies that the intent $\dot{\gamma} = \dot{\eta} < 0$, and hence $\xi(\dot{\gamma}) > 0$, and hence $T_{IA} < T_{MC}$.

There exists two exhaustive cases based on the error between the ship's heading θ and the optimal heading $\tilde{\theta}$, given by $\Delta > \Delta_0$ and $\Delta \leq \Delta_0$. For the first case $\xi(\Delta) = 0$, whereas $\xi(\dot{\gamma}) > 0$ for both cases. If $\theta > \tilde{\theta} + \Delta$, BSC begins with $\xi(\Delta) = 0$, hence IASC would drive x faster to $\tilde{u}$ as compared to BSC, resulting in $T_{IA} < T_{BSC}$.

However, for $\theta \leq \tilde{\theta} + \Delta$, consider the phase plot between Δ and the $\dot{\gamma}$ shown in the figure 5.2. The region for which BSC is effective is shown in blue, and the green depicts the region where IASC is effective. Region I is common to both. Since $\dot{\gamma} < 0$, we are interested only in the left half of the phase plot. Clearly in the left half, the area covered by the IASC is greater than that of BSC if $|\Delta_0| \leq \pi$, which is always true. Hence qualitatively $T_{IA} \leq T_{BSC}$. □

5.5 Zermelo's Navigation with Obstacles

Simulations were performed on Zermelo's navigation problem with obstacle. An example scenario is shown in the figure 5.1, where the objective is to navigate the ship to the origin (shown in red) by

controlling the heading angle θ. Figure 5.1 also shows the optimal heading angle $\tilde{\theta}$, as computed by the autonomous agent without the knowledge of the Obstacle. We performed a base simulation to understand and compare the performance of BSC and IASC. For this simulation we considered an obstacle as shown in the figure 5.3. Using only manual control of the ship, we first obtained a start position that results in obstacle avoidance; corresponding position and the trajectory is shown. Next moving backwards from this position, we simulated BSC and IASC assuming maximal evasive maneuver from the operator i.e. $u_0 = \bar{u}_0$. Figure 5.3 shows the resulting collision and successful obstacle evasive trajectories for both the controllers.

To gain further understanding, we investigated the performance of the two controllers w.r.t the successful trajectories in figure 5.3. This is depicted in figure 5.4, which shows the optimal heading $\tilde{\theta}$, operator's heading θ_0 (in absence of shared control) and the ship's actual heading θ with time; with the variation in the shared control parameter α or $\xi(\cdot)$ shown in a separate subplot. BSC control responds poorly to the evasive operator input, this is because the function $\xi(\dot{\gamma})$ decays very slowly causing the ship to follow the optimal heading rather than the operator intended heading (figure 5.4a). In contrast for IASC, ship responds quickly to operator's intent. First the control authority is ceded to the operator when he moves away from the optimal heading, later the ship follows the optimal heading as soon as the operator intends to do so (figure 5.4b).

Another way to assess the safety of the shared control techniques is through computation of their respective reachable sets [104]. Given the nonlinear nature of both the system dynamics and the control evolution, we utilized simulations to determine the reachable sets of both the controllers from a given start position from which the application of operator's maximal evasive maneuver resulted in obstacle collision. Results from these simulations is summarized in the form of *zone of no-return* in figure 5.5. The zone of no-return characterizes the region in which the system definitely fails, if the control authority is not transferred from the autonomous controller to the human operator in off-nominal situations. Figure 5.5 shows that the zone of no return is considerably smaller for IASC as compared to that of BSC. These results suggest that operators utilizing IASC should be more successful in avoiding obstacles, something that is later corroborated through human experiments. The key reason for smaller zone of no-return for IASC is because $\xi(\dot{\gamma})$ decays to zero much faster than $\xi(\Delta)$, as seen in Proposition 4.

5.5.1 Experimental Results

We designed experiments to evaluate the proposed IASC in comparison to the BSC. In a typical experiment an operator starts navigating the ship towards the origin from one of the three locations: $(0, -60), (40, 40)$, and $(0, -60)$, though similar locations, these had different location for the obstacle (see Table 5.1). The operator is given assistance in the form of optimal heading direction

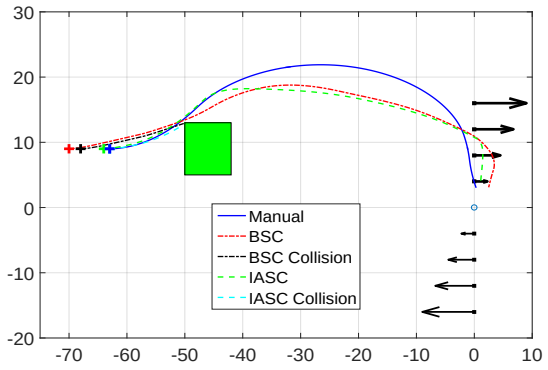
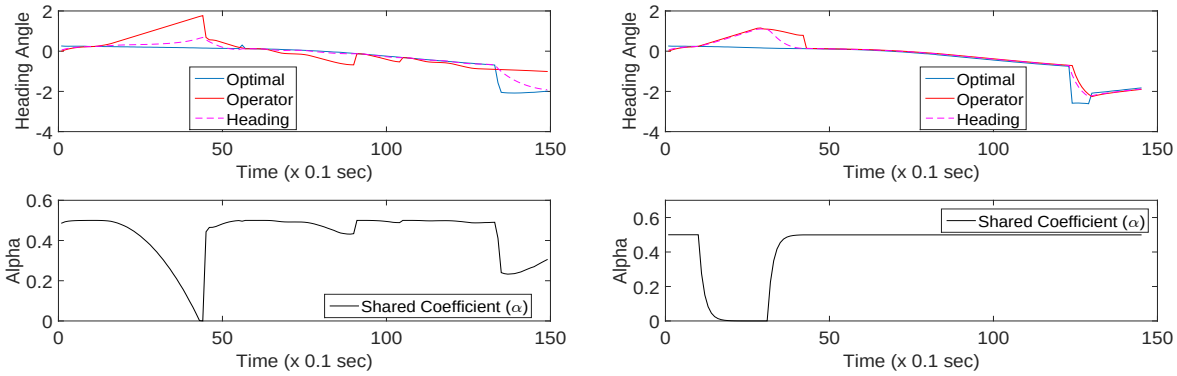


Figure 5.3: Trajectory for different control types



(a) Blended Shared Control

(b) Intent Aware Shared Control

Figure 5.4: Shared Control Performance while negotiating an obstacle in Zermelo's Navigation.

given by the green arrow shown in the figure 5.1, this assistance is however removed in the case of manual operation. While performing this task, a square obstacle of side 10 pops up at a distance of $d_{ob} = 25$ from the current location. This distance was chosen based on the zone of no return (figure 5.5), in order to provide sufficient time for an operator to take evasive action and avoid collision with the obstacle. The participants were instructed beforehand to ignore the optimal heading suggestion in the presence of an obstacle.

Prior to the experiment, a participant is allowed four practice runs starting from various locations in the field. During the practice runs, the optimal heading suggestion is active, giving the operator a sense of how an expert would navigate the currents. Currents were dictated by the constants $h = 4$ and $V = 2$. Obstacles were not introduced during the practice runs. After the completion of practice runs, a participant starts performing the actual experiment. An experiment consists of nine runs of navigation, which is a combination of three types of control for each of the three start location (or scenarios). Each run terminates in either of the following three cases,

- The participant navigates within d = 3 of the origin.

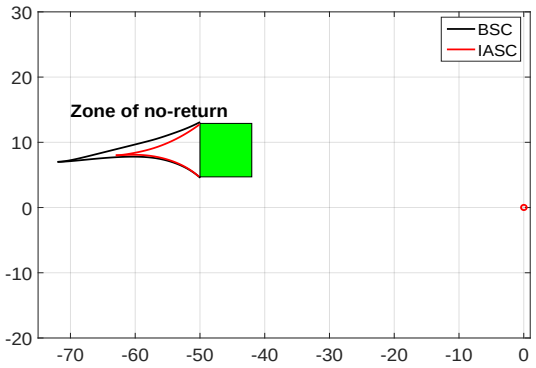


Figure 5.5: Zone of no-return: Region enclosed between the obstacle and the outermost collision trajectories of controllers.

- The ship collides with the obstacle,
- The ship exceeds a distance of d = 150 from the origin.

During an experiment, the sequence in which a scenario is presented or the controller type that is active, is random and hence not known to the operator.

Total of nine participants with Engineering background volunteered for the experiment, summary of the results in the form of box-plot that compares the task completion time for different control types is shown in the figure 5.6. Box-plots clearly show that the overall performance of IASC is far better than the blended shared control. For the first and the third location performance of IASC is comparable in mean (red lines) with others, but outperforms them for the second. This can probably be attributed to the time delay in operator response induced by directional ambiguity, since the second obstacle is approached from right to left whereas the other two obstacles are approached in opposite direction. However the IASC control remains robust to this time delay in operator's response and performs well. These results also verifies and supports the claim made in the proposition 5. The number of obstacle collisions observed during the trials given in Table 5.1 show that the proposed design of IASC is very responsive to off-nominal situations, resulting in fewer obstacle collision in comparison to the BSC architecture.

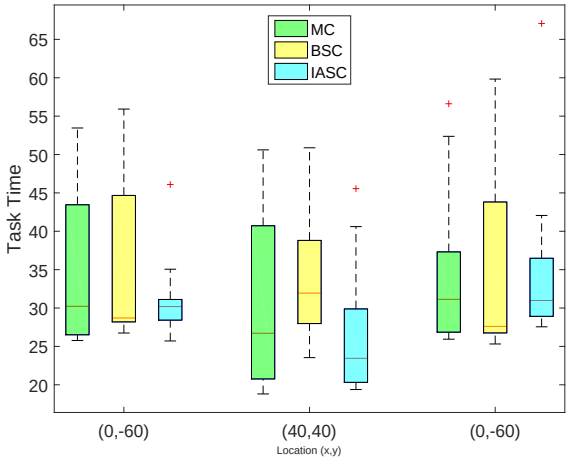


Figure 5.6: Boxplot comparison of completion times from three different starting locations to the origin, for Manual Control (MC), Blended shared control (BSC) and Intent Aware Blended shared control.

Table 5.1: Obstacle Collision: Design of IASC controller is responsive to off-nomial situations

Location (x, y)	Obstacle (x, y)	Manual Control	BSC	IASC
$(0, -60)$	$(-45, 8)$	1	10	2
$(40, 40)$	$(25, -8)$	0	4	3
$(0, -60)$	$(-75, 3)$	0	7	1

CHAPTER 6

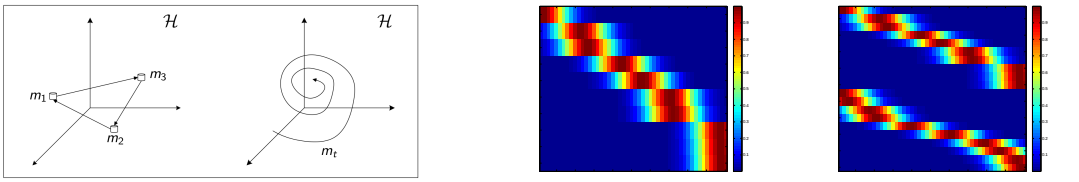
SPATIOTEMPORAL PROCESSES: KERNEL OBSERVER

6.1 Kernel Observers

This section formulates the problem, introduces kernel observers, and develops the main theoretical and algorithmic results. Section 6.2 presents a result on the expected number of randomly placed sensors required to monitor a spatiotemporal process in the context of our model. Section 6.3 demonstrates the efficacy of the algorithms on several challenging and large real-world datasets.

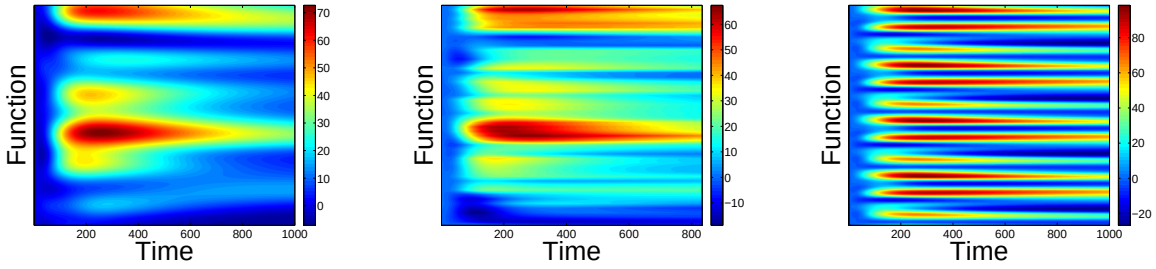
6.1.1 Problem Formulation

We focus on predictive inference of a time-varying stochastic process, whose mean f evolves temporally as $f_{\tau+1} \sim \mathbb{F}(f_\tau, \eta_\tau)$, where $\mathbb{F}$ is a distribution varying with time τ and exogenous inputs η. Our approach builds on the fact that in several cases, temporal evolution can be hierarchically separated from spatial functional evolution. A classical and quite general example of this is the *abstract evolution equation* (AEO), which can be defined as the evolution of a function u embedded in a Banach space $\mathcal{B}$: $\dot{u}(t) = \mathcal{L}u(t)$, subject to $u(0) = u_0$, and $\mathcal{L} : \mathcal{B} \to \mathcal{B}$ determines spatiotemporal transitions of $u \in \mathcal{B}$ [105]. This model of spatiotemporal evolution is very general (AEOs, for example, model many PDEs), but working in Banach spaces can be computationally taxing. A simple way to make the approach computationally realizable is to place restrictions on $\mathcal{B}$: in particular, we restrict the sequence f_τ to lie in a reproducing kernel Hilbert space (RKHS), the theory of which provides powerful tools for generating flexible classes of functions with relative ease [23]. In a kernel-based model, $k : \Omega \times \Omega \to \mathbb{R}$ is a positive-definite Mercer kernel on a domain Ω that models the covariance between any two points in the input space, and implies the existence of a smooth map $\psi : \Omega \to \mathcal{H}$, where $\mathcal{H}$ is an RKHS with the property $k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}}$. The key insight behind the proposed model is that spatiotemporal evolution in the input domain corresponds to temporal evolution of the mixing weights of a kernel model alone in the functional domain. Therefore, f_τ can be modeled by tracing the evolution of its mean embedded in a RKHS using switched ordinary differential equations (ODE) when the evolution is continuous, or switched difference equations when it is discrete (Figure 6.1a). The advantage of this approach is



(a) Hilbert Space Evolution (b) 1-shaded (Def. 1) (c) 2-shaded (Eq. (6.7))

Figure 6.1: (a) Two types of Hilbert space evolutions. Left: discrete switches in RKHS $\mathcal{H}$; Right: smooth evolution in $\mathcal{H}$, (b)-(c) Shaded observation matrices for dictionary of atoms.



(a) Gaussian (b) Laplacian (c) Periodic

Figure 6.2: One-dimensional function evolution over a fixed transition matrix A, initial condition w_0 and centers $\mathcal{C}$, but with different kernels $k(x, y)$. Each y-vector at a given value of x represents the output of the function, which evolves from left to right. As seen, changing the kernel creates quite different dynamic behaviors.

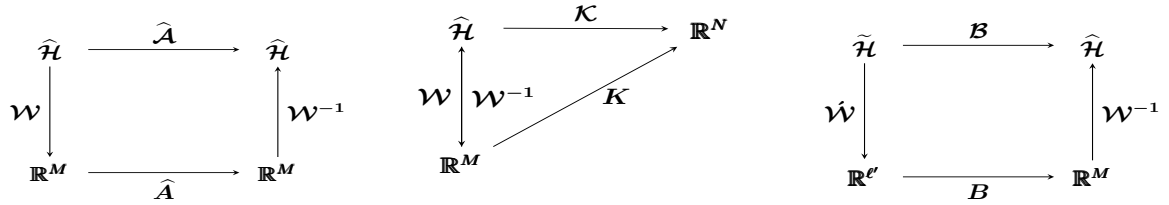
that it allows us to utilize powerful ideas from systems theory for deriving necessary and sufficient conditions for spatiotemporal monitoring.

In this paper, we restrict our attention to the class of functional evolutions $\mathbb{F}$ defined by linear Markovian transitions in an RKHS. While extension to the nonlinear case is possible (and non-trivial), it is not pursued in this paper to help ease the exposition of the key ideas. The class of linear transitions in RKHS is rich enough to approximately model many real-world datasets, as suggested by our experiments.

Let $y \in \mathbb{R}^N$ be the measurements of the function available from N sensors, $\mathcal{A} : \mathcal{H} \to \mathcal{H}$ be a linear transition operator in the RKHS $\mathcal{H}$, and $\mathcal{K} : \mathcal{H} \to \mathbb{R}^N$ be a linear measurement operator. The model for the functional evolution and measurement studied in this paper is:

$$f_{\tau+1} = \mathcal{A} f_\tau + \eta_\tau, \quad y_\tau = \mathcal{K}_\tau f_\tau + \zeta_\tau, \tag{6.1}$$

where η_τ is a zero-mean stochastic process in $\mathcal{H}$, and ζ_τ is a Wiener process in $\mathbb{R}^N$. Classical treatments of kernel methods emphasize that for most kernels, the feature map ψ is unknown, and possibly infinite-dimensional; this forces practioners to work in the dual space of $\mathcal{H}$, whose



(a) Relationship between $\mathcal{A}$ and A (b) Relationship between $\mathcal{K}$ and K (c) Relationship between $\mathcal{B}$ and B

Figure 6.3: Commutative diagrams between primal and dual spaces

dimensionality is the number of samples in the dataset being modeled. This conventional wisdom precludes the use of kernel methods for most tasks involving modern datasets, which may have millions and sometimes billions of samples [106]. An alternative is to work with a feature map $\widehat{\psi}(x) := \left[\, \widehat{\psi}_1(x) \;\cdots\; \widehat{\psi}_M(x) \,\right]^T$ to an approximate feature space $\widehat{\mathcal{H}}$, with the property that for every element $f \in \mathcal{H}$, $\exists \widehat{f} \in \widehat{\mathcal{H}}$ and an $\epsilon > 0$ s.t. $\|f - \widehat{f}\| < \epsilon$ for an appropriate function norm. A few such approximations are listed below.

Dictionary of atoms Let Ω be compact. Given points $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, we have a dictionary of atoms $\mathcal{F}^{\mathcal{C}} = \{\psi(c_1), \cdots, \psi(c_M)\}$, $\psi(c_i) \in \mathcal{H}$, the span of which is a strict subspace $\widehat{\mathcal{H}}$ of the RKHS $\mathcal{H}$ generated by the kernel. Here,

$$\widehat{\psi}_i(x) := \langle \psi(x), \psi(c_i) \rangle_{\mathcal{H}} = k(x, c_i). \tag{6.2}$$

Low-rank approximations Let Ω be compact, let $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, and let $K \in \mathbb{R}^{M \times M}$, $K_{ij} := k(c_i, c_j)$ be the Gram matrix computed from $\mathcal{C}$. This matrix can be diagonalized to compute approximations $(\widehat{\lambda}_i, \widehat{\phi}_i(x))$ of the eigenvalues and eigenfunctions $(\lambda_i, \phi_i(x))$ of the kernel [107]. These spectral quantities can then be used to compute $\widehat{\psi}_i(x) := \sqrt{\widehat{\lambda}_i}\widehat{\phi}_i(x)$.

Random Fourier features Let $\Omega \subset \mathbb{R}^n$ be compact, and let $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$ be the Gaussian RBF kernel. Then random Fourier features approximate the kernel feature map as $\widehat{\psi}_\omega : \Omega \to \widehat{\mathcal{H}}$, where ω is a sample from the Fourier transform of $k(x, y)$, with the property that $k(x, y) = \mathbb{E}_\omega[\langle \widehat{\psi}_\omega(x), \widehat{\psi}_\omega(y) \rangle_{\widehat{\mathcal{H}}}]$ [106]. In this case, if $V \in \mathbb{R}^{M/2 \times n}$ is a random matrix representing the sample ω, then $\widehat{\psi}_i(x) := \left[\, \frac{1}{\sqrt{M}} \sin([Vx]_i), \frac{1}{\sqrt{M}} \cos([Vx]_i) \,\right]$. Similar approximations exist for other radially symmetric kernels, as well as dot-product kernels.

In the approximate space case, we replace the transition operator $\mathcal{A} : \mathcal{H} \to \mathcal{H}$ in (6.1) by $\widehat{\mathcal{A}} : \widehat{\mathcal{H}} \to \widehat{\mathcal{H}}$. This approximate regime, which combines the flexibility of a truly nonparametric approach with computational realizability, still allows for the representation of rich phenomena, as will be seen in the sequel, and in Figure 6.2. The finite-dimensional evolution equations approxi-

mating (6.1) in dual form are

$$w_{\tau+1} = \widehat{A} w_\tau + \eta_\tau, \quad y_\tau = K w_\tau + \zeta_\tau, \tag{6.3}$$

where we have matrices $\widehat{A} \in \mathbb{R}^{M\times M}$, $K \in \mathbb{R}^{N\times M}$, the vectors $w_\tau \in \mathbb{R}^M$, and where we have slightly abused notation to let y_τ, η_τ and ζ_τ denote their $\widehat{\mathcal{H}}$ counterparts. Here K is the matrix whose rows are of the form $K_{(i)} = \widehat{\Psi}(x_i) = \left[\, \widehat{\psi}_1(x_i) \;\; \widehat{\psi}_2(x_i) \;\cdots\; \widehat{\psi}_M(x_i) \,\right]$. In systems-theoretic language, each row of K corresponds to a *measurement* at a particular location, and the matrix itself acts as a measurement operator.

The equations (6.1) suggest an immediate extension to functional control problems. Pick another basis for $\mathcal{H}$ as $\widetilde{\psi}(x) := \begin{bmatrix}\widetilde{\psi}_1(x) & \cdots & \widetilde{\psi}_{\ell'}(x)\end{bmatrix}^T$, where the functions $\widetilde{\psi}_j(x)$ are used to approximate the RKHS $\mathcal{H}$ generated by the kernel. We denote the span of these functions as $\widetilde{\mathcal{H}}$. In the dictionary of atoms case, an example would be another set of atoms $\mathcal{F}^D = \begin{bmatrix}\psi(d_1) & \cdots & \psi(d_{\ell'})\end{bmatrix}, \psi(d_j) \in \mathcal{H}$, $d_j \in \Omega$, with $\widetilde{\mathcal{H}}$ being a strict subspace of the RKHS $\mathcal{H}$ generated by the kernel. The functional evolution equation is then as follows:

$$f_{\tau+1} = \mathcal{A} f_\tau + \mathcal{B}\delta_\tau + \eta_\tau, \quad y_\tau = \mathcal{K}_\tau f_\tau + \zeta_\tau, \tag{6.4}$$

where the control functions δ_τ evolve in $\widetilde{\mathcal{H}}$, and $\mathcal{B} : \widetilde{\mathcal{H}} \to \widehat{\mathcal{H}}$. To derive the finite-dimensional equivalent of $\mathcal{B}$, we have to work out the structure of the matrix B: since $\widehat{\mathcal{H}}$ is not, in general, isomorphic to $\widetilde{\mathcal{H}}$, this imposes strict restrictions on B. We can derive B using least squares using the inner product of $\mathcal{H}$. An instructive example is where both $\widehat{\mathcal{H}}$ and $\widetilde{\mathcal{H}}$ are generated by dictionaries of atoms; recall that in this case, $\mathcal{F}^{\mathcal{C}} = \begin{bmatrix}\psi(c_1) & \cdots & \psi(c_M)\end{bmatrix}$ is the basis for $\widehat{\mathcal{H}}$, and let $\delta = \sum_{j=1}^{\ell'} \acute{w}_j \psi(d_j)$. Then the projection of δ onto $\widehat{\mathcal{H}}$ can be derived as

$$\begin{bmatrix} \langle \delta, \psi(c_1)\rangle_{\mathcal{H}} \\ \vdots \\ \langle \delta, \psi(c_M)\rangle_{\mathcal{H}} \end{bmatrix} = \underbrace{\begin{bmatrix} \langle \psi(d_1), \psi(c_1)\rangle_{\mathcal{H}} & \cdots & \langle \psi(d_{\ell'}), \psi(c_1)\rangle_{\mathcal{H}} \\ \vdots & \ddots & \vdots \\ \langle \psi(d_1), \psi(c_M)\rangle_{\mathcal{H}} & \cdots & \langle \psi(d_{\ell'}), \psi(c_M)\rangle_{\mathcal{H}} \end{bmatrix}}_{K_{CD}} \begin{bmatrix} \acute{w}_1 \\ \vdots \\ \acute{w}_{\ell'} \end{bmatrix}.$$

Note that in the dictionary of atoms case, the entries of K_{CD} can be computed in closed form as $K_{CDij} := k(d_i, c_j)$, using the reproducing property. This derivation shows that the operator B is simply $K_{CD} \in \mathbb{R}^{M\times \ell'}$, the kernel matrix between the data C generating the atoms $\mathcal{F}^{\mathcal{C}}$ of $\widehat{\mathcal{H}}$ and the data D generating the atoms $\mathcal{F}^D$ of $\widetilde{\mathcal{H}}$. Relation between the operators $\mathcal{B}$ and B, is outlined in the commutative diagram in Figure 6.3c.

Thus, the finite-dimensional evolution equations equivalent to (6.4) are

$$w_\tau = \widehat{A} w_\tau + K_{CD} \acute{w}_\tau, \quad y_\tau = K_\tau w_\tau. \tag{6.5}$$

We define the *generalized observability matrix* [108] as $\mathcal{O}_\Upsilon = \begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L} \end{bmatrix}$ where $\Upsilon = \{\tau_1, \ldots, \tau_L\}$ are the set of instances τ_i when we apply the operator K. A linear system is said to be *observable* if $\mathcal{O}_\Upsilon$ has full column rank (i.e. $\text{Rank}\mathcal{O}_\Upsilon = M$) for $\Upsilon = \{0, 1, \ldots, M-1\}$ [108]. Observability guarantees two critical facts: firstly, it guarantees that the state w_0 can be recovered exactly from a finite series of measurements $\{y_{\tau_1}, y_{\tau_2}, \ldots, y_{\tau_L}\}$; in particular, defining $y_\Upsilon = \left[y_{\tau_1}^T, y_{\tau_2}^T, \cdots, y_{\tau_L^T}\right]^T$, we have that $y_\Upsilon = \mathcal{O}_\Upsilon w_0$. Secondly, it guarantees that a feedback based *observer* can be designed such that the estimate of w_τ, denoted by $\widehat{w}_\tau$, converges exponentially fast to w_τ in the limit of samples. Note that all our theoretical results assume $\widehat{A}$ is available: while we perform system identification in the experiments (Section 6.3.3), it is not the focus of the paper.

We are now in a position to formally state the spatiotemporal modeling, control, and inference problems being considered: given a spatiotemporally evolving system modeled using (6.3), choose a set of N sensing locations such that even with $N \ll M$, the functional evolution of the spatiotemporal model can be estimated (which corresponds to *monitoring*), can be predicted robustly (which corresponds to *Bayesian filtering*), and which can be controlled (which corresponds to *functional control*). Our approach to solve the monitoring and prediction problem relies on the design of the measurement operator K so that the pair $(K, \widehat{A})$ is observable: any Bayesian state estimator (e.g. a Kalman filter) utilizing this pair is denoted as a **kernel observer** [1]. In the controls case, given a spatiotemporally evolving system modeled using (6.5), we need to choose a set of N sensing locations and ℓ' control locations, such that even with $N \ll M,\ \ell' \ll M$, the functional evolution of the spatiotemporal model can be controlled; in this case, we must design both a measurement operator K and a control operator K_{CD} such that the pair $(K_{CD}, \widehat{A})$ is controllable: a controls system utilizing this pair and the measurement operator K is denoted as a **kernel controller**.

6.1.2 Main Results

In this section, we prove results concerning the observability of spatiotemporally varying functions modeled by the functional evolution and measurement equations (6.3) formulated in Section 6.1.1. In particular, observability of the system states implies that we can recover the current state of the spatiotemporally varying function using a small number of sampling locations N, which allows us to 1) track the function, and 2) predict its evolution forward in time. We work with the approxi-

[1] In the case where no measurements are taken, for the sake of consistency, we denote the state estimator as an **autonomous kernel observer**, despite this being somewhat of an oxymoron.

mation $\widehat{\mathcal{H}} \approx \mathcal{H}$: given M basis functions, this implies that the dual space of $\widehat{\mathcal{H}}$ is $\mathbb{R}^M$. Proposition 4 shows that if $\widehat{A}$ has a full-rank Jordan decomposition, the observation matrix K meeting a condition called *shadedness* (Definition 1) is sufficient for the system to be observable. Proposition 5 provides a lower bound on the number of sampling locations required for observability which holds for any $\widehat{A}$. Proposition 6 constructively shows the existence of an abstract measurement map $\widetilde{K}$ achieving this lower bound. Since the measurement map does not have the structure of a kernel matrix, a slightly weaker sufficient condition for the observability of any $\widehat{A}$ is in Theorem 1. Finally, since both K and K_{CD} are kernel matrices generated from a shared kernel, these observability results translate directly into controllability results.

Definition 1. ***(Shaded Observation Matrix)*** *Given $k : \Omega \times \Omega \to \mathbb{R}$ positive-definite on a domain Ω, let $\{\widehat{\psi}_1(x), \ldots, \widehat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\widehat{\psi} : \Omega \to \widehat{\mathcal{H}}$, and let $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \Omega$. Let $K \in \mathbb{R}^{N \times M}$ be the observation matrix, where $K_{ij} := \widehat{\psi}_j(x_i)$. For each row $K_{(i)} := \begin{bmatrix} \widehat{\psi}_1(x_i) & \cdots & \widehat{\psi}_M(x_i) \end{bmatrix}$, define the set $\mathcal{I}_{(i)} := \{\iota_1^{(i)}, \iota_2^{(i)}, \ldots, \iota_{M_i}^{(i)}\}$ to be the indices in the observation matrix row i which are nonzero. Then if $\bigcup_{i \in \{1,\ldots,N\}} \mathcal{I}^{(i)} = \{1, 2, \ldots, M\}$, we denote K as a* shaded observation matrix *(see Figure 6.1).*

This definition seems quite abstract, so the following remark considers a more concrete example.

Remark 1. *let $\widehat{\psi}$ be generated by the dictionary given by $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$. Note that since $\widehat{\psi}_j(x_i) = \langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}} = k(x_i, c_j)$, K is the kernel matrix between $\mathcal{X}$ and $\mathcal{C}$. For the kernel matrix to be shaded thus implies that there does not exist an atom $\psi(c_j)$ such that the projections $\langle \psi(x_i), \psi(c_j) \rangle_{\mathcal{H}}$ vanish for all x_i, $1 \leq i \leq N$. Intuitively, the shadedness property requires that the sensor locations x_i are privy to information propagating from every c_j. As an example, note that, in principle, for the Gaussian kernel, a single row generates a shaded kernel matrix*[2].

Proposition 4. *Given $k : \Omega \times \Omega \to \mathbb{R}$ positive-definite on a domain Ω, let $\{\widehat{\psi}_1(x), \ldots, \widehat{\psi}_M(x)\}$ be the set of bases generating an approximate feature map $\widehat{\psi} : \Omega \to \widehat{\mathcal{H}}$, and let $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \Omega$. Consider the discrete linear system on $\widehat{\mathcal{H}}$ given by the evolution and measurement equations (6.3). Suppose that a full-rank Jordan decomposition of $\widehat{A} \in \mathbb{R}^{M \times M}$ of the form $\widehat{A} = P \Lambda P^{-1}$ exists, where $\Lambda = \begin{bmatrix} \Lambda_1 & \cdots & \Lambda_O \end{bmatrix}$, and there are no repeated eigenvalues. Then, given a set of time instances $\Upsilon = \{\tau_1, \tau_2, \ldots, \tau_L\}$, and a set of sampling locations $\mathcal{X} = \{x_1, \ldots, x_N\}$, the system (6.3) is observable if the observation matrix K_{ij} is shaded according to Definition 1, Υ has distinct values, and $|\Upsilon| \geq M$.*

Proof. To begin, consider a system where $\widehat{A} = \Lambda$, with Jordan blocks $\{\Lambda_1, \Lambda_2, \ldots, \Lambda_O\}$ along the

[2]However, in this case, the matrix can have many entries that are extremely close to zero, and will probably be very ill-conditioned.

diagonal. Then $\widehat{A}^{\tau_i} = \mathrm{diag}(\begin{bmatrix}\Lambda_1^{\tau_i} & \Lambda_2^{\tau_i} & \cdots & \Lambda_O^{\tau_i}\end{bmatrix})$. We have that

$$\mathcal{O}_\Upsilon = \underbrace{\begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L}. \end{bmatrix}}_{\mathcal{O}_\Upsilon \in \mathbb{R}^{NL \times M}}$$

We need to prove that the column rank of $\mathcal{O}_\Upsilon$ is M, which is not immediately obvious since typically $N \ll M$. To prove the statement, we will show that computing the rank of $\mathcal{O}_\Upsilon$ is equivalent to the rank computation of the product of two simple matrices. In what follows, we use the notation $\mathbf{0}_{\mathbb{R}^{I\times J}}$ to denote an $I \times J$ matrix of all zeros.

In the first step, we write the above matrix as the product of two matrices. Then it can be shown that $\mathcal{O}_\Upsilon$ is the product of two block matrices

$$\mathcal{O}_\Upsilon = \underbrace{\begin{bmatrix} K & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & K \end{bmatrix}}_{\widehat{\mathbf{K}} \in \mathbb{R}^{NL \times ML}} \underbrace{\left[\begin{array}{ccc} \Lambda_1^{\tau_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Lambda_O^{\tau_1} \\ \hline \vdots & \ddots & \vdots \\ \hline \Lambda_1^{\tau_L} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Lambda_O^{\tau_L} \end{array}\right]}_{\widehat{\mathbf{A}} \in \mathbb{R}^{ML \times M}}.$$

We need to simplify $\widehat{\mathbf{K}}$ even further. Recall that a matrix's rank is preserved under a product with an invertible matrix. Design a matrix of elementary row operations $U \in \mathbb{R}^{N\times N}$ such that $\breve{K} := UK$ is a matrix with at least one row vector of nonzeros; this can be achieved by having an elementary matrix that adds rows together. By the shadedness assumption, such a matrix exists. We can write this operation as

$$UK = \begin{bmatrix} \breve{K}_{11} & \breve{K}_{12} & \cdots & \breve{K}_{1M} \\ \breve{K}_{21} & \breve{K}_{22} & \cdots & \breve{K}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \breve{K}_{N1} & \breve{K}_{N2} & \cdots & \breve{K}_{NM}. \end{bmatrix}$$

Without loss of generality, and abusing notation slightly, let this multiplication lead to one nonzero

row, with the rest of the elements of the matrix being zero, as

$$UK = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Since elementary matrices are full-rank, we then have that $\text{rank}(UK) = \text{rank}(K)$.

To analyze the rank of $\mathcal{O}_\Upsilon$, we apply these elementary matrices to every $K \in \widehat{\mathbf{K}}$. To do so, consider the block-diagonal matrix $\mathcal{U} \in \mathbb{R}^{NL \times NL}$ with $U \in \mathbb{R}^{N \times N}$ along the diagonal, and zeros everywhere else, i.e.

$$\mathcal{U} := \begin{bmatrix} U & \mathbf{0}_{\mathbb{R}^{N\times N}} & \cdots & \mathbf{0}_{\mathbb{R}^{N\times N}} \\ \mathbf{0}_{\mathbb{R}^{N\times N}} & U & \cdots & \mathbf{0}_{\mathbb{R}^{N\times N}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times N}} & \mathbf{0}_{\mathbb{R}^{N\times N}} & \cdots & U \end{bmatrix}. \tag{6.6}$$

It can be shown that $\mathcal{U}$ is full-rank, i.e. has rank NL. Going back to the observability matrix, we have that

$$\begin{aligned} \mathcal{U}\mathcal{O}_\Upsilon &= \mathcal{U}\widehat{\mathbf{K}}\widehat{\mathbf{A}} \\ &= \underbrace{\begin{bmatrix} UK & \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & UK & \cdots & \mathbf{0}_{\mathbb{R}^{N\times M}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{\mathbb{R}^{N\times M}} & \mathbf{0}_{\mathbb{R}^{N\times M}} & \cdots & UK \end{bmatrix}}_{\mathcal{U}\widehat{\mathbf{K}} \in \mathbb{R}^{NL \times ML}} \underbrace{\widehat{\mathbf{A}}}_{\in \mathbb{R}^{ML \times M}}, \end{aligned}$$

since $\mathbf{0}_{\mathbb{R}^{N\times N}}\mathbf{0}_{\mathbb{R}^{N\times M}} = \mathbf{0}_{\mathbb{R}^{N\times M}}$. Due to the fact that $\text{rank}(\mathcal{U}\mathcal{O}_\Upsilon) = \text{rank}(\mathcal{O}_\Upsilon)$, we can therefore

perform our rank analysis on the simpler matrix $\mathrm{rank}(\mathcal{U}\mathcal{O}_\Upsilon)$. Note that

$$UK\widehat{A}^{\tau_j} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \widehat{A}^{\tau_j}$$
$$= \begin{bmatrix} k_{11}\lambda_1^{\tau_j} & \binom{\tau_j}{1}\lambda_1^{\tau_j-1} + k_{12}\lambda_1^{\tau_j} & \cdots & k_{1M}\lambda_O^{\tau_j} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Therefore, following some more elementary row operations encoded by $V \in \mathbb{R}^{ML\times ML}$, we have that

$$V\mathcal{U}\mathcal{O}_\Upsilon = \begin{bmatrix} k_{11}\lambda_1^{\tau_1} & \cdots & k_{1M}\lambda_O^{\tau_1} \\ k_{11}\lambda_1^{\tau_2} & \cdots & k_{1M}\lambda_O^{\tau_2} \\ \vdots & \ddots & 0 \\ k_{11}\lambda_1^{\tau_L} & \cdots & k_{1M}\lambda_O^{\tau_L} \\ \mathbf{0}_{\mathbb{R}^{M(L-1)\times 1}} & \cdots & \mathbf{0}_{\mathbb{R}^{M(L-1)\times 1}} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{\Phi} \\ \mathbf{0}_{\mathbb{R}^{M(L-1)\times M}} \end{bmatrix}.$$

If the individual entries k_{1i} are nonzero, and the Jordan block diagonals have nonzero eigenvalues, the columns of $\mathbf{\Phi}$ become linearly independent. Therefore, if $L \geq M$, the column rank of $\mathcal{O}_\Upsilon$ is M, which results in an observable system.

To extend this proof to matrices $\widehat{A} = P\Lambda P^{-1}$, note that

$$\mathcal{O}_\Upsilon = \begin{bmatrix} K\widehat{A}^{\tau_1} \\ \cdots \\ K\widehat{A}^{\tau_L} \end{bmatrix}$$
$$= \begin{bmatrix} KP\Lambda^{\tau_1}P^{-1} \\ \cdots \\ KP\Lambda^{\tau_L}P^{-1}. \end{bmatrix}$$
$$= \widehat{\mathbf{K}}\boldsymbol{P}\boldsymbol{\Lambda}^t\boldsymbol{P}^{-1},$$

where $\boldsymbol{P} \in \mathbb{R}^{ML\times ML}$, $\boldsymbol{\Lambda}^t \in \mathbb{R}^{ML\times ML}$, and $\boldsymbol{P^{-1}} \in \mathbb{R}^{ML\times ML}$ are the block diagonal matrices

associated with the system. Since $\boldsymbol{P}$ is an invertible matrix, the conclusions about the column rank drawn before still hold, and the system is observable. □

When the eigenvalues of the system matrix are repeated, it is not enough for K to be shaded. In the next proposition, we take a geometric approach and utilize the rational canonical form of $\widehat{A}$ to obtain a lower bound on the number of sampling locations required. Let r be the number of unique eigenvalues of $\widehat{A}$, and let γ_{λ_i} denote the geometric multiplicity of eigenvalue λ_i. Then the *cyclic index* of $\widehat{A}$ is defined as $\ell = \max_{1 \leq i \leq r} \gamma_{\lambda_i}$ [34] (see section 1.3.3 for details).

Proposition 5. *Suppose that the conditions in Proposition 4 hold, with the relaxation that the Jordan blocks* $\begin{bmatrix} \Lambda_1 & \cdots & \Lambda_O \end{bmatrix}$ *may have repeated eigenvalues (i.e.* $\exists \Lambda_i$ *and* Λ_j *s.t.* $\lambda_i = \lambda_j$*). Then there exist kernels* $k(x, y)$ *such that the lower bound* ℓ *on the number of sampling locations* N *is given by the cyclic index of* $\widehat{A}$*. In other words, the system in (6.3) is observable if* $N \geq \ell$*.*

Proof. **By Contrapositive.** We will show that if the number of sampling locations are $N = \ell - 1$ (i.e. $N < \ell$), then the system is not observable. Pick the Gaussian kernel in the dictionary of atoms framework, with sampling locations $x_i \in \mathcal{X}$ and centers $c_j \in \mathcal{C}$, with the additional property that $x_i \neq x_j \forall i, j \in \{1, \ldots, N\}, i \neq j$. In this case, $\mathbf{K}$ has $\ell - 1$ nonzero, linearly independent rows, and can be written as

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1M} \\ \vdots & \vdots & \cdots & \vdots \\ k_{(\ell-1)1} & k_{(\ell-1)2} & \cdots & k_{(\ell-1)M} \end{bmatrix}.$$

Since the cyclic index is ℓ, this implies that at least one eigenvalue, say λ, has ℓ Jordan blocks. Define indices $j_1, j_2, \ldots, j_\ell \in \{1, 2, \ldots, M\}$ as the columns corresponding to the leading entries of the ℓ Jordan blocks corresponding to λ. WLOG, let $j_1 = 1$. Using ideas similar to the last proof, we can write the observability matrix as

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\lambda^{\tau_1} & \cdots & k_{1j_\ell}\lambda^{\tau_1} & \cdots \\ \vdots & \ddots & \vdots & \ddots \\ k_{11}\lambda^{\tau_L} & k_{1j_\ell}\lambda^{\tau_L} & \cdots & \\ \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\lambda^{\tau_1} \cdots & k_{(\ell-1)j_\ell}\lambda^{\tau_1} & \cdots & \\ \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\lambda^{\tau_L} & \cdots & k_{(\ell-1)j_\ell}\lambda^{\tau_L} & \cdots \end{bmatrix}.$$

Define $\boldsymbol{\lambda} := \begin{bmatrix} \lambda^{\tau_1} & \lambda^{\tau_2} & \cdots \lambda^{\tau_L} \end{bmatrix}^T$. Then the above matrix becomes

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\boldsymbol{\lambda} & \cdots & k_{1j_2}\boldsymbol{\lambda} & \cdots & k_{1j_\ell}\boldsymbol{\lambda} & \cdots \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ k_{(\ell-1)1}\boldsymbol{\lambda} & \cdots & k_{(\ell-1)j_2}\boldsymbol{\lambda} & \cdots & k_{(\ell-1)j_\ell}\boldsymbol{\lambda} & \cdots \end{bmatrix}.$$

We need to show that one of the columns above can be written in terms of the others. This is equivalent to solving the linear system

$$\begin{bmatrix} k_{1j_1} \\ k_{2j_1} \\ \vdots \\ k_{(\ell-1)j_1} \end{bmatrix} = \begin{bmatrix} k_{1j_2} & \cdots & k_{1j_\ell} \\ k_{2j_2} & \cdots & k_{2j_\ell} \\ \vdots & \ddots & \vdots \\ k_{(\ell-1)j_2} & \cdots & k_{(\ell-1)j_\ell} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{(\ell-1)} \end{bmatrix}.$$

Since the kernel matrix on the RHS is generated from the Gaussian kernel, from [109], it's known that every principal minor of a Gaussian kernel matrix is invertible, which implies that $\mathcal{O}_\Upsilon$ cannot be observable. □

Section 6.1.3 gives a concrete example to build intuition regarding this lower bound. We now show how to construct a matrix $\widetilde{K}$ corresponding to the lower bound ℓ.

Proposition 6. *Given the conditions stated in Proposition 5, it is possible to construct a measurement map $\widetilde{K} \in \mathbb{R}^{\ell \times M}$ for the system given by (6.3), such that the pair $(\widetilde{K}, \widehat{A})$ is observable.*

Proof. The construction of the measurement map $\widetilde{K}$ is based on the rational canonical structure of $\widehat{A}^T$, which decomposes $\mathcal{V}$ into $\widehat{A}^T$-cyclic direct summands such that $\mathcal{V} = \mathcal{V}_1 \oplus \cdots \oplus \mathcal{V}_\ell$, where ℓ is the cyclic index of $\widehat{A}$ as defined in Proposition 5. Let ξ_v be the minimal polynomial (m.p.) of v (relative to $\widehat{A}^T$): it is then the unique monic polynomial of least degree such that $\xi_v(\widehat{A}^T)v = 0$. Let $\alpha_1(\lambda)$ be the m.p. of $\widehat{A}^T_{|\mathcal{V}_1}$: then $\deg(\alpha_1(\lambda)) < M$. By the rational canonical structure theorem [34], there exists a vector $\widehat{v}_1$, such that $\xi_{v_1}(\lambda) = \alpha_1(\lambda)$. Similarly there exists a vector $\widehat{v}_2$, such that $\xi_{v_2}(\lambda) = \alpha_2(\lambda)$, where $\alpha_2(\lambda)$, is the minimal polynomial of $\widehat{A}^T_{|\mathcal{V}_2}$ and so on. Thus we can obtain ℓ such vectors that form the measurement map $\widetilde{K} = [\widehat{v}_1, \widehat{v}_2, \cdots, \widehat{v}_\ell]^T$. Construction of these vectors $\widehat{v}_i$, can be simplified by first performing the Jordan decomposition as $\widehat{A}^T = P\Lambda P^{-1}$. Then the vectors $\widetilde{v}_i,\ i \in \ell$ for Λ, can be constructed such that the entries corresponding to the leading entries of Jordan blocks of $\Lambda_{|\mathcal{V}_i}$ are nonzero. Such a construction ensures that the m.p. of vector $\widetilde{v}_i$ w.r.t $\Lambda_{|\mathcal{V}_i}$, is also the corresponding m.p. of $\Lambda_{|\mathcal{V}_i}$. Hence the required map is given by $\widetilde{K} = [\widetilde{v}_1, \widetilde{v}_2, \ldots, \widetilde{v}_\ell]^T P^{-1}$. □

Algorithm 4 Measurement Map $\widetilde{K}$

Input: $\widehat{A} \in \mathbb{R}^{M \times M}$
Compute Rational canonical form, such that $C = Q^{-1}\widehat{A}^T Q$. Set $C_0 := C$, and $M_0 := M$.
for $i = 1$ **to** ℓ **do**
  Obtain MP $\alpha_i(\lambda)$ of C_{i-1}. This returns associated indices $\mathcal{J}^{(i)} \subset \{1, 2, \ldots, M_{i-1}\}$.
  Construct vector $v_i \in \mathbb{R}^M$ such that $\xi_{v_i}(\lambda) = \alpha_i(\lambda)$.
  Use indices $\{1, 2, \ldots, M_{i-1}\} \setminus \mathcal{J}^{(i)}$ to select matrix C_i. Set $M_i := |\{1, 2, \ldots, M_{i-1}\} \setminus \mathcal{J}^{(i)}|$
end for
Compute $\mathring{K} = [v_1^T, v_2^T, ..., v_\ell^T]^T$
Output: $\widetilde{K} = \mathring{K} Q^{-1}$

The construction provided in the proof of Proposition 6 is utilized in Algorithm 4, which uses the rational canonical structure of $\widehat{A}$ to generate a series of vectors $v_i \in \mathbb{R}^M$, whose iterations $\{v_1, \ldots, \widehat{A}^{m_1-1}v_1, \ldots, v_\ell, \ldots, \widehat{A}^{m_\ell-1}v_\ell\}$ generate a basis for $\mathbb{R}^M$. Unfortunately, the measurement map $\widetilde{K}$, being an abstract construction unrelated to the kernel, does not directly select $\mathcal{X}$. We will show how to use the measurement map to guide a search for $\mathcal{X}$ in Remark 2. For now, we state a sufficient condition for observability of a general system.

Theorem 1. *Suppose that the conditions in Proposition 4 hold, with the relaxation that the Jordan blocks* $\begin{bmatrix}\Lambda_1 & \cdots & \Lambda_O\end{bmatrix}$ *may have repeated eigenvalues. Let* ℓ *be the cyclic index of* $\widehat{A}$*. Define*

$$\mathbf{K} = \begin{bmatrix} K^{(1)^T} & \ldots & K^{(\ell)^T} \end{bmatrix}^T \tag{6.7}$$

as the ℓ-shaded matrix *which consists of* ℓ *shaded matrices with the property that any subset of* ℓ *columns in the matrix are linearly independent from each other. Then system (6.3) is observable if* Υ *has distinct values, and* $|\Upsilon| \geq M$*.*

Proof. A cyclic index of ℓ for this system implies that there exists an eigenvalue λ that's repeated ℓ times. We prove the theorem for repeated eigenvalues of dimension 1: the same statement can be proven for repeated eigenvalues for Jordan blocks using the ideas in the proof of Proposition 4. WLOG, let $\mathbf{K}$ have ℓ fully shaded, linearly independent rows, and, assume that the column indices corresponding to this eigenvalue are $\{1, 2, \ldots, \ell\}$. Define $\boldsymbol{\lambda}_i := \begin{bmatrix}\lambda_i^{\tau_1} & \lambda_i^{\tau_2} & \cdots \lambda_i^{\tau_L}\end{bmatrix}^T$. Then

$$\mathcal{O}_\Upsilon := \begin{bmatrix} k_{11}\boldsymbol{\lambda}_1 & k_{12}\boldsymbol{\lambda}_2 & \cdots & k_{1M}\boldsymbol{\lambda}_M \\ \vdots & \vdots & \ddots & \vdots \\ k_{\ell 1}\boldsymbol{\lambda}_1 & k_{\ell 2}\boldsymbol{\lambda}_2 & \cdots & k_{\ell M}\boldsymbol{\lambda}_M \end{bmatrix}.$$

Let $\boldsymbol{\lambda}_1 = \boldsymbol{\lambda}_2 = \cdots \boldsymbol{\lambda}_\ell := \boldsymbol{\lambda}$. Focusing on these first ℓ columns of this matrix, this implies that we

Algorithm 5 Sampling locations set $\mathcal{X}$

Input: $\widehat{A} = C$, lower bound ℓ
Decompose C to generate invariant subspaces $\mathcal{H}_j^{\mathcal{C}}$, $j \in \{1, 2, \ldots, \ell\}$ (see section 1.3.3)
for $j = 1$ **to** ℓ **do**
  Obtain centers $\mathcal{C}^{(j)}$ w.r.t subspace $\mathcal{H}_j^{\mathcal{C}}$,
  Generate samples $x_i^{(j)}$ to create a kernel matrix $K^{(j)}$ that is shaded only with respect to centers $\mathcal{C}^{(j)}$
end for
Output: Sampling locations set $\mathcal{X} = \{x^{(1)}, x^{(2)} \cdots, x^{(l)}\}$.

need to find constants $c_1, c_2, \ldots, c_{\ell-1}$ such that

$$\begin{bmatrix} k_{11} \\ \vdots \\ k_{\ell 1} \end{bmatrix} = c_1 \begin{bmatrix} k_{12} \\ \vdots \\ k_{\ell 2} \end{bmatrix} + \cdots + c_{\ell-1} \begin{bmatrix} k_{1\ell} \\ \vdots \\ k_{\ell\ell} \end{bmatrix}.$$

However, these columns are linearly independent by assumption, and thus no such constants exist, implying that $\mathcal{O}_\Upsilon$ is observable. □

While Theorem 1 is a quite general result, the condition that any ℓ columns of $\mathbf{K}$ be linearly independent is a very stringent condition. One scenario where this condition can be met with minimal measurements is in the case when the feature map $\widehat{\psi}(x)$ is generated by a dictionary of atoms with the Gaussian RBF kernel evaluated at sampling locations $\{x_1, \ldots, x_N\}$ according to (6.2), where $x_i \in \Omega \subset \mathbb{R}^d$, and x_i are sampled from a non-degenerate probability distribution on Ω such as the uniform distribution. For a semi-deterministic approach, when the dynamics matrix $\widehat{A}$ is block-diagonal, we can utilize a simple heuristic:

Remark 2. *Let Ω be compact, $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, and let the approximate feature map be defined by (6.2). Consider the system (6.3) with $\widehat{A} = \Lambda$, and let $\Upsilon = \{0, 1, \ldots, M-1\}$. Then the measurement map $\widetilde{K}$'s values lie in $\{0, 1\}$; in particular, each row $\widetilde{K}^{(j)}$, $j \in \{1, \ldots, \ell\}$, corresponds to a subspace $\mathcal{H}_j^{\mathcal{C}}$, generated by a subset of centers $\mathcal{C}^{(j)} \subset \mathcal{C}$. Generate samples $x_i^{(j)}$ to create a kernel matrix $K^{(j)}$ that is shaded only with respect to centers $\mathcal{C}^{(j)}$. Once this is done, move on to the next subspace $\mathcal{H}_{j+1}^{\mathcal{C}}$. When all ℓ rows of $\widetilde{K}$ are accounted for, construct the matrix $\mathbf{K}$ as in (6.7). Then the resulting system $(\mathbf{K}, \widehat{A})$ is observable.*

This heuristic is formalized in Algorithm 5. Note that in practice, the matrix $\widehat{A}$ needs to be inferred from measurements of the process f_τ. If no assumptions are placed on $\widehat{A}$, it's clear that at least M sensors are required for the system identification phase. Future work will study the precise conditions under which system identification is possible with less than M sensors.

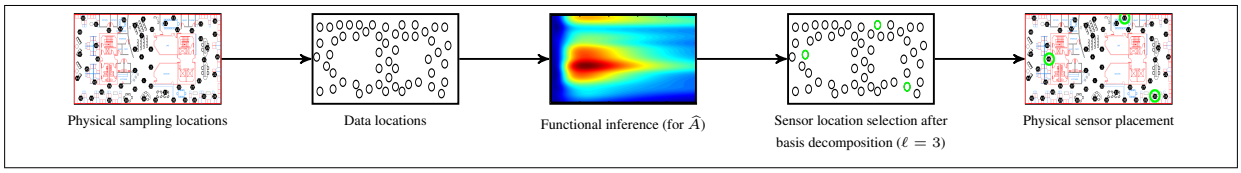


Figure 6.4: Overall description of how the kernel observer fits in the sensing framework. Physical locations are mapped to data locations, over which historical data is collected as a time series. Functional inference is performed over $\widehat{\mathcal{H}}$ to solve for $\widehat{A}$. The measurement operator K is then computed (see Figure 6.5), leading to sensor placement.

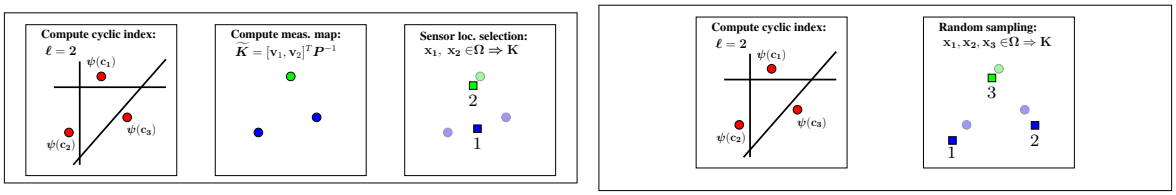


Figure 6.5: Diagram demonstrating sensor placement using the measurement map or random sampling approaches. The circles represent data locations associated to bases (e.g. $c_j \Leftrightarrow \psi(c_j)$) and the squares represent sensor locations (e.g. $x_i \Leftrightarrow \psi(x_i)$) . The cyclic index ($\ell = 2$) indicates how many possible couplings of bases exist, which can be represented as a choice of $\binom{M}{\ell}$ hyperplanes in Ω. If the measurement map is computed (left), the correct couplings are chosen (green vs. blue), and a smaller number of sensors (2) can be placed. Alternatively, random sampling (right) is more computationally efficient, but generally requires more sensors (3).

6.1.3 Discussion of Theoretical Results

The systems-theoretic approach taken in this paper reveals something rather surprising: functions with complex dynamics (with a small cyclic index) can be recovered with less sensor placements than functions with simpler dynamics. Although seemingly counterintuitive, it becomes clear that this is because complex dynamics, which are characterized by a lower geometric multiplicity of the eigenvalues, ensure that the orbit $\mathbf{\Theta} := \{\widehat{A}w_\tau\}_{\tau\in\Upsilon}$ traverses a greater portion of $\mathbb{R}^M \equiv \widehat{\mathcal{H}}$ and thus that fewer sensors can recover more geometric information. On the other hand, in 'simpler' functional evolution, $\mathbf{\Theta}$ evolves along strict subspaces of $\mathbb{R}^M$, and so more independent sensors are required to infer the same amount of information.

In the case described in Remark 2, we have a set of centers $\mathcal{C} = \{c_1, \ldots, c_M\}$, which generate the bases $\mathcal{F}^{\mathcal{C}} = \{\psi(c_1), \cdots, \psi(c_M)\}$. Let the cyclic index be ℓ: this implies that there exist ℓ subsets $\Psi^{(i)}$ of $\mathcal{F}^{\mathcal{C}}$ with at least one element $\psi(c_j)$ each, leading to $\binom{M}{\ell}$ possible choices: Figure 6.5 represents these choices as hyperplanes separating the subsets. The measurement map described in Alg. 4 induces this *decomposition of bases* $\mathcal{F}^{\mathcal{C}} = \{\Psi^{(1)}, \ldots, \Psi^{(\ell)}\}$ in polynomial time. Further, each subset $\Psi^{(i)}$ is directly associated to a subset of centers $\mathcal{C}^{(i)} \subset \mathcal{C}$, which allows us to pick targeted sensor locations $x_i \in \Omega$. In particular, for radially symmetric kernels such as the Gaussian,

the centroid of the convex hull of $\mathcal{C}^{(i)}$ is sufficient for generating a sensor placement. The measurement map is a significant theoretical insight into sensor placement for dynamically changing environments, because it directly takes into account the dynamics of the process. Of course, in practice, this may be too expensive for approximate feature spaces with M very large, so one can use random sampling to generate the sensor locations instead, at the cost of N being larger than ℓ. The advantage here though is that since random sampling is computationally inexpensive, different choices of sensor placements can be generated and evaluated relatively quickly.

Another point to note is that since the collection of bases $\{\widehat{\psi}_i(x)\}_{i=1}^{M}$ determines the richness of the function space $\widehat{\mathcal{H}} \approx \mathcal{H}$ we operate in, it determines the fidelity of the model approximation to the true time-varying function. As a consequence, observability of the system in $\widehat{\mathcal{H}}$ refers to the best possible approximation in $\widehat{\mathcal{H}}$. The greater the number of bases, the higher the dimensionality, which results in greater model fidelity, but which may require a much greater number of measurements for state recovery. This is where the lower bounds presented in the paper are particularly useful, because they show that for functional evolutions corresponding to certain $\widehat{A}$, *the number of sensor placements are essentially independent of the dimensionality* M, but depend rather on the cyclic index of $\widehat{A}$.

6.2 Random Sensor Placement

We now elaborate on how the challenging problem of sensor placement can be tackled through random selection. This process of random selection is a product of the kernel observer model described in the section 6.1. We present the theoretical background required to prove Theorem 2, which states the expected number of randomly placed sensors required to monitor a given spatiotemporal process, and Theorem 3, which determines the probability with which optimal sensor placement is ensured given that, N number of sensors have been placed.

As discussed earlier, we work with an approximate feature space $\widehat{\mathcal{H}}$, with the corresponding transition operator $\widehat{A} : \widehat{\mathcal{H}} \to \widehat{\mathcal{H}}$, representing finite-dimensional functional evolution. To achieve observability for the pair $(\widehat{A}, K)$, row vectors of the corresponding observability matrix, $\mathcal{O}$, should form the basis for the $\mathbb{R}^M$-dimensional space $\widehat{\mathcal{H}}$. According to the rational canonical structure Theorem [34], $\widehat{A}$ can successively decompose the dual space $\mathbb{R}^M$ into subspaces, $\mathcal{V}_i \subset \mathcal{V}$, $i \in \{1, \ldots, \ell\}$, with properties, i) $\mathcal{V} = \mathcal{V}_1 \oplus ... \oplus \mathcal{V}_\ell$, ii) $\widehat{A}\mathcal{V}_i \subset \mathcal{V}_i$, and iii) $\widehat{A}|\mathcal{V}_i, i \in \{1, \ldots, \ell\}$, are cyclic. The integer ℓ is unique and is called the *cyclic index of* $\widehat{A}$. Each of these properties contribute towards the theorem on the number of random samples required to achieve observability. The first property shows that the space $\mathbb{R}^M$ can be decomposed into ℓ independent subspaces. The second property shows that the vector $v_i \in \mathcal{V}_i$ stays in $\mathcal{V}_i$ even when operated upon by $\widehat{A}$. Thus, to

generate bases for $\mathbb{R}^M$, one needs at least ℓ vectors, say, $v_1, \ldots, v_\ell$, with respect to each subspace $\mathcal{V}_1, \ldots, \mathcal{V}_\ell$. This holds due to the third property, but requires that the vectors $v_1, \ldots, v_\ell$, are the cyclic generators of their corresponding subspaces. Our analysis is based on whether a randomly selected sensor can generate a cyclic generator. To examine this, recall that a row vector $K_{(i)}$ generated by a randomly selected sensor location x_i takes the form,

$$K_{(i)} = \Big[k(x_i, c_1), \ldots, k(x_i, c_M)\Big]. \tag{6.8}$$

Here, for radial kernels for example, the entries corresponding to the centers closer to x_i tend to be non-zero, whereas the others tend to be zero. The rows $K_{(i)}$ from random sensor placement must be able to generate a basis for a subspace $\mathcal{V}_i$, and thus must be cyclic generators. We will derive the expected number of random sensor placements sufficient for observability for the case where $\widehat{A} = \Lambda$ and then attempt to generalize the result for any $\widehat{A}$. Note Λ is a block diagonal Jordan form. In this case, the cyclic generator for each subspace $\mathcal{V}_i$, is a vector v_i with non-zero entries corresponding to the leading entry of the Jordan blocks of $\mathcal{V}_i$. An example of a subspace, and its cyclic generator is,

$$\mathcal{V}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad v_1 = \begin{bmatrix} 0 \\ 0 \\ s \\ s' \end{bmatrix},$$

where s, s' are non-zero.

Overall, our construction is as follows: for each subspace $\mathcal{V}_i$, let $\mathcal{C}_{\mathcal{V}_i} \subset \mathcal{C}$ be the centers corresponding to those leading entry of Jordan blocks: then the minimum number of random samples required to generate the bases for $\mathcal{V}_i$ is equal to the number of Jordan blocks comprising $\mathcal{V}_i$. Altogether, the minimum number of random samples required to generate a basis for $\mathbb{R}^M$ is equal to the total number of Jordan blocks in $\widehat{A}$. Let ς be the total number of Jordan blocks in $\widehat{A}$, then

$$\varsigma = \sum_{\lambda \in \sigma(\widehat{A})} \gamma_\lambda \tag{6.9}$$

where $\sigma(\widehat{A})$ represents the spectrum of $\widehat{A}$, whose elements are the eigenvalues of $\widehat{A}$, and γ_λ is the geometric multiplicity corresponding to the eigenvalue λ, which is also equal to the total number of Jordan blocks corresponding to the eigenvalue λ. Define a set of centers $\mathcal{C}_\varsigma$ with elements $\{c_1, c_2, \ldots, c_\varsigma\}$, to be the centers corresponding to the leading entries of the Jordan blocks. For sensor location $x \in \Omega$, and $\epsilon > 0$, let $k(x, c_j) > \epsilon$, denote the region $\Omega_j \subset \Omega$, such that the kernel evaluation with respect to center c_j is greater than ϵ, that is $\Omega_j \equiv \{x \in \Omega : k(x, c_j) > \epsilon\}$. We

define p_ϵ as

$$p_\epsilon = \min_{c_j \in \mathcal{C}_\varsigma} \frac{\nu(k(x, c_j) > \epsilon)}{\nu(\Omega)}, \tag{6.10}$$

where ν is a measure in the real analysis sense. Hence, p_ϵ corresponds to a lower bound on the probability that a random sample lies within the $\epsilon-$shaded region of a particular center c_j. With all of this in place, we can prove the following theorem.

Theorem 2. *Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D, \tau \in \mathbb{Z}^+$ its kernel observer model (6.3), and a tolerance parameter $\epsilon > 0$, the expected number of randomly placed sensor locations required to achieve observability for the pair $(K, \widehat{A})$ is ς/p_ϵ where ς is the summation over geometric multiplicities of each $\lambda \in \sigma(\widehat{A})$ given by Equation (6.9).*

Proof. For each random sample, the probability that it lies within the $\epsilon-$shaded region of a particular center $c_j \in \mathcal{C}_\varsigma$ is at least p_ϵ. The series of random samples can be considered as Bernoulli trials in which p_ϵ is the probability of a successful outcome. Note this is assuming worst case scenario that the intersection between any two ϵ-shaded region of centers belonging to the set $\mathcal{C}_\varsigma$ is empty. Observability for the pair $(K, \widehat{A})$ is achieved after ς successful outcomes are obtained because each success ensures a row vector with non-zero entry corresponding to the leading entry of the Jordan block.

Let $X_1, X_2, \ldots, X_N$ be i.i.d. random variables whose common distribution is the Bernoulli distribution with parameter p_ϵ. The random variable $X = X_1 + X_2 + \cdots + X_N$ denotes the number of success after N random samples. Since each X_i has the Bernoulli distribution, X will have binomial distribution,

$$P(X = h) = \binom{N}{h} p_\epsilon^h (1 - p_\epsilon)^{N-h},$$

in which h is the number of success. The expectation of the binomial distribution, that is the expected number of success is Np_ϵ, and thus the expected number of trials required will be $N = \varsigma/p_\epsilon$. □

Theorem 3. *Given the spatiotemporal function $f(x, \tau)$ with $x \in \Omega \subseteq \mathbb{R}^D, \tau \in \mathbb{Z}^+$, its kernel observer model (6.3), a tolerance parameter $\epsilon > 0$, summation over geometric multiplicities of each $\lambda \in \sigma(\widehat{A})$ denoted by ς as in Equation (6.9), and a constant $\delta \in (0, 1]$, the probability that pair $(K, \widehat{A})$ is unobservable after the selection of N random sensors is at most $e^{\frac{-1}{2}(Np_\epsilon - 2\varsigma)}$, where p_ϵ is given by Equation (6.10) and $N \geq \varsigma/p_\epsilon$.*

Proof. The random variable X from the proof of Theorem 1 has a binomial distribution, which enables the application of a Chernoff-type bound on its tail probabilities. A well known result on multiplicative Chernoff bound [110] is directly applied to establish this Theorem. If X is binomially distributed, $\delta \in (0, 1]$, and $\mu = \mathbb{E}[X]$, then $P[X \leq (1 - \delta)\mu] \leq \exp(-\mu\delta^2/2)$, in

which we let $\delta = 1 - \frac{\varsigma}{Np_\epsilon}$. The expression in the exponent can be simplified to $-\frac{1}{2}Np_\epsilon + \varsigma - \frac{\varsigma^2}{2Np_\epsilon}$, using $\mu = Np_\epsilon$. Note that $e^{\frac{-\varsigma^2}{2Np_\epsilon}} \leq 1$. This implies that,

$$\exp(-\mu\delta^2/2) = e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}.e^{\frac{-\varsigma^2}{2Np_\epsilon}} \leq e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}$$

Note, $(1-\delta)\mu = \varsigma$, hence we obtain that $P[X \leq \varsigma] \leq e^{-\frac{1}{2}(Np_\epsilon - 2\varsigma)}$. □

For the case when $\widehat{A} \neq \Lambda$, a change of basis can be used to obtain $\Lambda = P^{-1}\widehat{A}P$, where P is the projection map. There are two challenges in performing the above analysis for Λ so obtained: first, the leading entries of Jordan blocks do not directly correspond to the centers $\{c_1, \ldots, c_M\}$ which was the case for $\widehat{A} = \Lambda$. Second, although we can obtain the transformation of the row vector (Equation (6.8)) using the projection map P, we can no longer arrive at the definition of the probability p_ϵ as in Equation (6.10). The existence of the similarity transform hints that the results in Theorems 2-3 should hold for any $\widehat{A}$, but the mathematical tools utilized in the paper seem to be insufficient to prove them. However, we present some empirical evidence for these claims for when $\widehat{A} \neq \Lambda$ in Section 6.3.1.

6.3 Experimental Results

6.3.1 Sampling Locations for Synthetic Data Sets

The goal of this experiment is to investigate the dependency of the observability of system (6.3) on the shaded observation matrix and the lower bound presented in Proposition 5. The domain is fixed on the interval $\Omega = [0, 2\pi]$. First, we pick sets of points $\mathcal{C}^{(\iota)} = \{c_1, \ldots, c_{M_\iota}\}$, $c_j \in \Omega$, $M = 50$, and construct a dynamics matrix $A = \Lambda \in \mathbb{R}^{M \times M}$, with cyclic index 5. We pick the RBF kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, $\sigma = 0.02$. Generating samples $\mathcal{X} = \{x_1, \ldots, x_N\}$, $x_i \in \Omega$ randomly, we compute the ℓ-shaded property and observability for this system. Figure 6.6a shows how shadedness is a necessary condition for observability, validating Proposition 4: the slight gap between shadedness and observability here can be explained due to numerical issues in computing the rank of $\mathcal{O}_\Upsilon$.

Next, we consider a system with a cyclic index $\ell = 18$ to verify random sensor placement results. We constructed the measurement operator K using the heuristic in Remark 2 (Algorithm 5), and random sensor selection to generate the sampling locations $\mathcal{X}$. These results are presented in Figure 6.6b. The plot for random sampling which has been averaged over 100 runs, resembles a c.d.f function of an exponential distribution $F(X = x) = 1 - \exp(-\lambda x)$. This verifies the claim made in Theorem 3, as the probability of becoming unobservable decays exponentially with

Algorithm 6 Kernel Observer (Transition Learning)

Input: Kernel k, basis centers $\mathcal{C}$, final time step T.
while $\tau \leq T$ **do**
1) Sample data $\{y_\tau^i\}_{i=1}^M$ from f_τ.
2) Estimate $\widehat{w}_\tau$ via standard kernel inference procedure.
3) Store weights $\widehat{w}_\tau$ in matrix $\mathcal{W} \in \mathbb{R}^{M \times T}$.
end while
To infer $\widehat{A}$, define matrix $\Phi = \mathcal{W}^T\mathcal{W}$. Then:
for $i = 1$ **to** M **do**
At step i, solve system

$$\widehat{A}^{(i)} = \left(\left(\Phi + \lambda I\right)^{-1} \left(\mathcal{W}^T \mathcal{W}^{(i)}\right) \right)^T, \tag{6.11}$$

where $\widehat{A}^{(i)}$, and $\mathcal{W}^{(i)}$ are the ith columns of $\widehat{A}$ and $\mathcal{W}^{(i)}$ respectively.
end for
Compute the covariance matrix $\widehat{B}_\mathcal{C}$ of the observed weights $\mathcal{W}$.
Output: estimated transition matrix $\widehat{A}$, predictive covariance matrix $\widehat{B}_\mathcal{C}$.

Algorithm 7 Kernel Observer (Monitoring and Prediction)

Input: Kernel k, basis centers $\mathcal{C}$, estimated system matrix $\widehat{A}$, estimated covariance matrix $\widehat{B}_\mathcal{C}$.
Compute Observation Matrix: Compute the cyclic index ℓ of $\widehat{A}$, and compute K.
Initialize Observer: Use $\widehat{A}$, $\widehat{B}_\mathcal{C}$, and K to initialize a state-observer (e.g. Kalman filter (KF)) on $\widehat{\mathcal{H}}$.
while measurements available **do**
1) Sample data $\{y_\tau^i\}_{i=1}^N$ from f_τ.
2) Propagate KF estimate $\widehat{w}_\tau$ forward to time $\tau + 1$, correct using measurement feedback with $\{y_{\tau+1}^i\}_{i=1}^N$.
3) Output predicted function $\widehat{f}_{\tau+1}$ of KF.
end while

the number of sensor placed. Also, fitting an exponential distribution we found that the mean λ^{-1} comes close to the ratio ς/p, which is the expected number of randomly placed sensors required for observability as per Theorem 2. Note that observability is not achieved if the number of samples $N < \ell$ verifying the result in Proposition 5.

6.3.2 Comparison With Nonstationary Kernel Methods on Real-World Data

We use three real-world datasets to evaluate and compare the kernel observer with the two different lines of approach for non-stationary kernels discussed in Section 2.2. For the Process Convolution with Local Smoothing Kernel (PCLSK) and Latent Extension of Input Space (LEIS) approaches, we compare with NOSTILL-GP [25] and [111] respectively, on the Intel Berkeley, Irish Wind and Ozone data-sets.

Model inference for the kernel observer involved three steps: 1) picking the Gaussian RBF

kernel $k(x, y) = e^{-\|x-y\|^2/2\sigma^2}$, a search for the ideal σ is performed for a sparse Gaussian Process model (with a fixed basis vector set $\mathcal{C}$ selected using the method in [112]. For the data set discussed in this section, the number of basis vectors were equal to the number of sensing locations in the training set, with the domain for input set defined over $\mathbb{R}^2$; 2) having obtained σ, Gaussian process inference is used to generate weight vectors for each time-step in the training set, resulting in the sequence $w_\tau, \tau \in \{1, \ldots, T\}$; 3) matrix least-squares is applied to this sequence to infer $\widehat{A}$. This process is described in Algorithm 6. For prediction in the autonomous setup, $\widehat{A}$ is used to propagate the state w_τ forward to make predictions with no feedback, and in the observer setup, a Kalman filter (see Algorithm 7) with N determined using Proposition 5, and locations picked randomly, is used to propagate w_τ forward to make predictions. We also compare with a baseline GP (denoted by 'original GP'), which is the sparse GP model trained using all of the available data.

Our first dataset, the Intel Berkeley research lab temperature data, consists of 50 wireless temperature sensors in indoor laboratory region spanning 40.5 meters in length and 31 meters in width[3]. Training data consists of temperature data on March 6th 2004 at intervals of 20 minutes (beginning 00:20 hrs) which totals to 72 timesteps. Testing is performed over another 72 timesteps beginning 12:20 hrs of the same day. Out of 50 locations, we uniformly selected 25 locations each for training and testing purposes. Results of the prediction error are shown in box-plot form in Figure 6.7a and as a time-series in Figure 6.7b, note that 'Auto' refers to autonomous set up. Here, the cyclic index of $\widehat{A}$ was determined to be 2, so N was set to 2 for the kernel observer with feedback. Note that here, even the autonomous kernel observer outperforms PCLSK and LEIS overall, and the kernel observer with feedback with $N = 2$ significantly outperforms all other methods, which is why we did not include results with $N > 2$.

The second dataset is the Irish wind dataset, consisting of daily average wind speed data collected from year 1961 to 1978 at 12 meteorological stations in the Republic of Ireland[4]. The prediction error is in box-plot form in Figure 6.8a and as a time-series in Figure 6.8b. Again, the cyclic index of $\widehat{A}$ was determined to be 2. In this case, the autonomous kernel observer's performance is comparable to PCLSK and LEIS, while the kernel observer with feedback with $N = 2$ again outperforms all other methods.

Finally, the Ozone dataset measures ozone concentration (in parts per billion) measured at 60 stations by the United States Environmental Protection Agency [113] across USA. Due to missing measurements, we only selected data from year 1997 to 2013 for training and evaluation. For each station, we averaged ozone concentration over a period of three months, resulting in four quarters per year. Out of 60 sensor locations, we uniformly selected 30 for training and the remaining locations for testing purposes. The prediction error results are presented in box-plot form in Figure

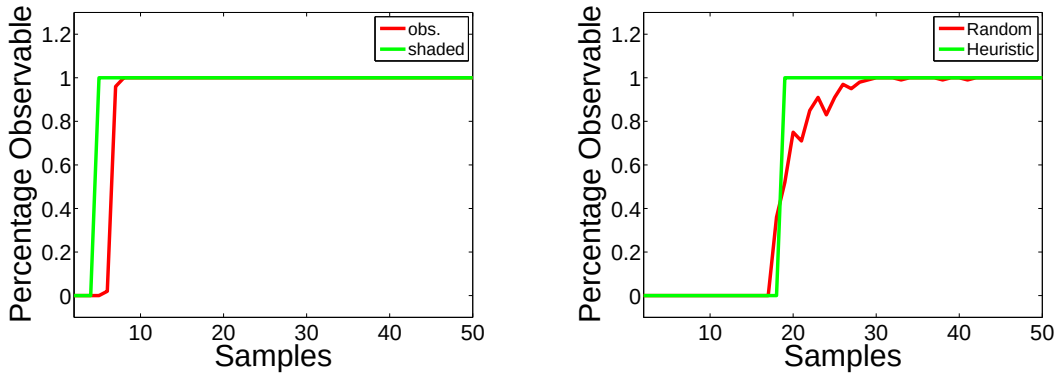
[3] `http://db.csail.mit.edu/labdata/labdata.html`

[4] `http://lib.stat.cmu.edu/datasets/wind.desc`

6.9a and as a time-series in Figure 6.9b. Here, the cyclic index of $\widehat{A}$ was determined to be 1. In this case, the performance of autonomous kernel observer is comparable to PCLSK and LEIS, with kernel observer with feedback with $N = 1$ performing the best. Table 6.1 reports the total training and prediction times associated with PCLSK, LEIS, and the kernel observer. We observed that, 1) the kernel observer is an order of magnitude faster than the competing methods, and 2) even for small sets, competing methods did not scale well.

Table 6.1: Total training and prediction times for Figs. 6.7 and 6.9

	Intel Berkeley	**Irish** Wind	**Ozone**
Data Size (*bases-timesteps*)	25-72	12-36	30-68
Kernel Observer	2.1 sec	0.1 sec	1.61 sec
PCLSK	121.4 sec	7.0 sec	91.90 sec
LEIS	43.8 sec	2.8 sec	37.41 sec



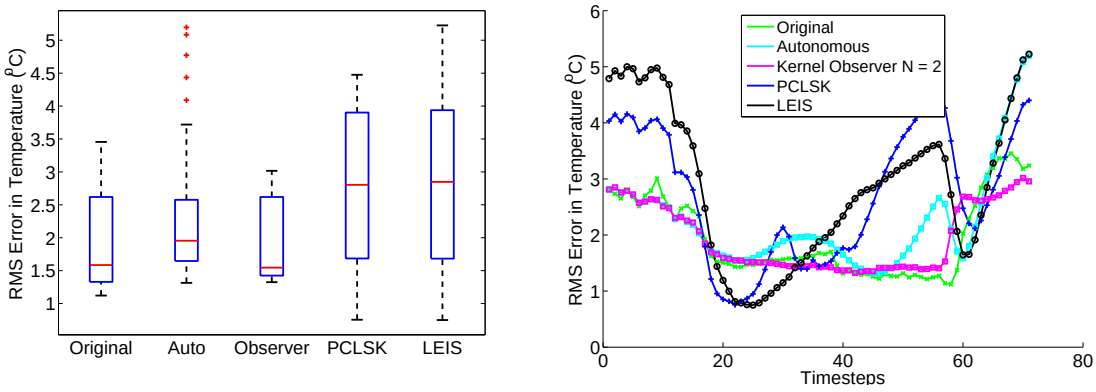
(a) Shaded vs. observability

(b) Heuristic vs. Random

Figure 6.6: Kernel observability results

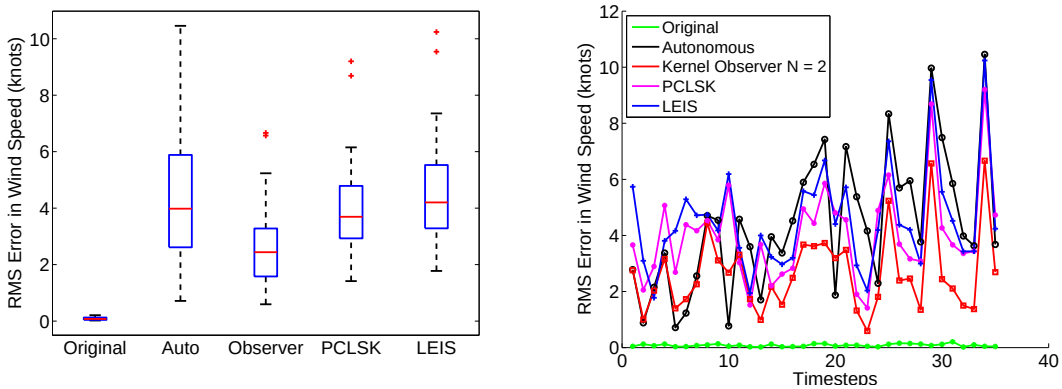
6.3.3 Prediction of Global Ocean Surface Temperature

We analyzed the feasibility of our approach on a very large dataset from the National Oceanographic Data Center: the 4 km AVHRR Pathfinder project, which is a satellite monitoring global ocean surface temperature. Figures 6.10a and 6.10b show an example of the raw data from the satellite and GP estimate (using 400,000 training points) respectively, for a specific date (01/02/2012)



(a) Error (boxplot) (b) Error (time-series)

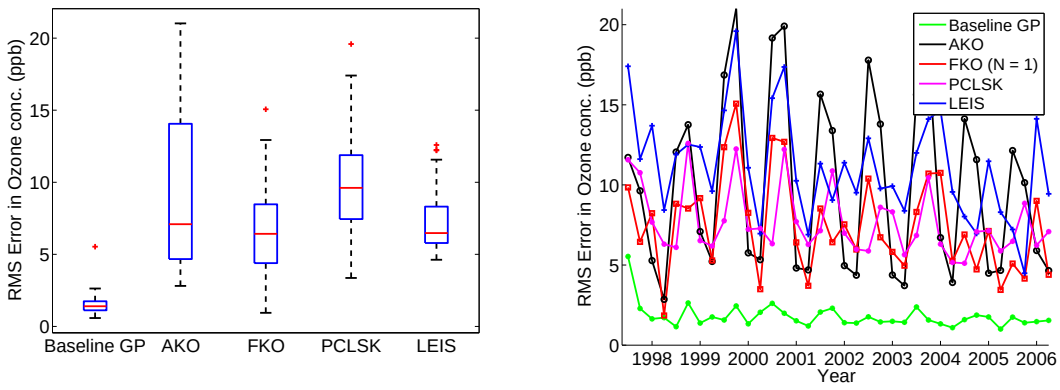
Figure 6.7: Comparison of kernel observer to PCLSK and LEIS methods on Intel Berkeley dataset.



(a) Error (boxplot) (b) Error (time-series)

Figure 6.8: Comparison of kernel observer to PCLSK and LEIS methods on Intel Berkeley dataset.

for day temperatures. This dataset is challenging, with measurements at over 37 million possible coordinates, but with only around 3-4 million measurements available per day, leading to a lot of missing data. The goal was to learn the day and night temperature models on data from the year 2011, and then to monitor thereafter for 2012. Success in monitoring would demonstrate two things: 1) the modeling process can capture spatiotemporal trends that generalize across years, and 2) the observer framework allows us to infer the state using a number of measurements that are an order of magnitude fewer than available. Note that due to the size of the dataset and the high computational requirements of the nonstationary kernel methods, a comparison with them was not pursued. To build the autonomous kernel observer and general kernel observer models, we followed the same procedure outlined in Section 6.3.2, but with $\mathcal{C} = \{c_1, \ldots, c_M\}$, $c_j \in \mathbb{R}^2$, $|\mathcal{C}| = 300$. The Kalman filter for the general kernel observer model used $N \in \{250, 500, 1000\}$ at random locations to track the system state given a random initial condition w_0. As a fair baseline, the observers are compared to training a sparse GP model (labeled 'original') on approximately $400,000$



(a) Error (boxplot)

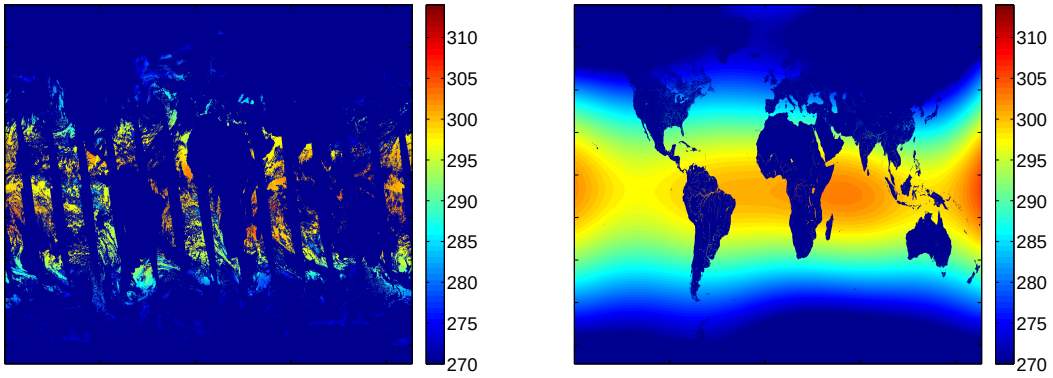
(b) Error (time-series)

Figure 6.9: Comparison of kernel observer to PCLSK and LEIS methods on Intel Berkeley dataset.

measurements per day. Figures 6.11a and 6.11b compare the autonomous and feedback approach with $1,000$ samples to the baseline GP; here, it can be seen that the autonomous does well in the beginning, but then incurs an unacceptable amount of error when the time series goes into 2012, i.e. where the model has not seen any training data, whereas FKO does well throughout. Figures 6.11c and 6.11d show a comparison of the RMS error of estimated values from the real data. This figure shows the trend of the observer getting better and better state estimates as a function of the number of sensing locations N [5]. Time required for kernel observer is much lesser than retraining the model every time step, see figure 6.11e.

Weather Anomaly in 2012: We further investigated the poor performance of autonomous kernel observer in the year 2012 as observed in Figures 6.11a and 6.11b. Clearly, the error in prediction blows up at the start of May in the year 2012. Indicating that the autonomous model trained using the data of the year 2011 does well in capturing the annual weather dynamics up to the month of May 2012. We turned our attention towards the weather in May 2012, as changes in ocean temperatures are directly related to the weather. Surprisingly, severe weather onset was reported on the east coast of United States in May 2012, and this anomaly continued over the period of May to June 2012. Thus, the apparent poor performance of autonomous kernel observer can actually be a useful indicator in detecting the anomalous behavior, as was the case with the ocean temperature in May 2012 which had deviated from the nominal weather dynamics observed in the year 2011 in which no severe weather anomalies were reported. We took a step ahead and identified the locations at which the prediction error was two standard deviations above the mean error. These error locations are plotted in Figure 6.12. Error locations during 6-7th May correlate with the severe weather onset and that of 28-29th May were along the track of storm Beryl.

[5]Note that we checked the performance of training a GP with only $1,000$ samples as a control, but the average error was about 10 Kelvins, i.e. much worse than FKO.



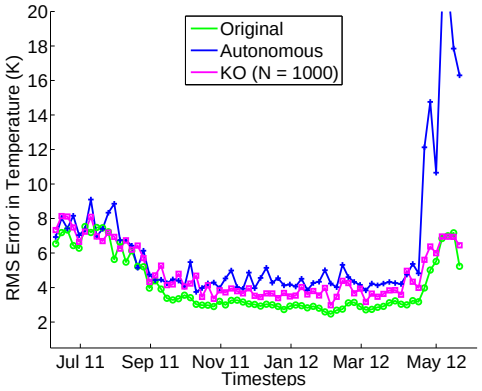
(a) Day Temperature Measurements

(b) Day Temperature Estimated

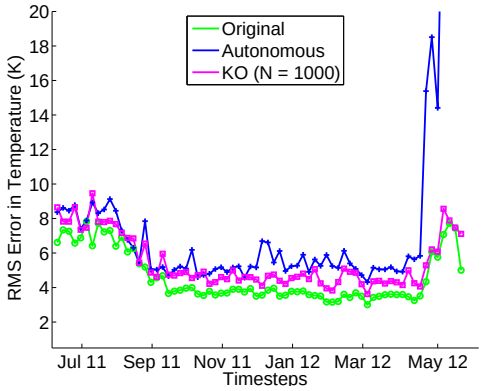
Figure 6.10: Pathfinder satellite temperature values, 01/02/2012. Blue values indicate missing data.

6.3.4 Control of a linear PDE

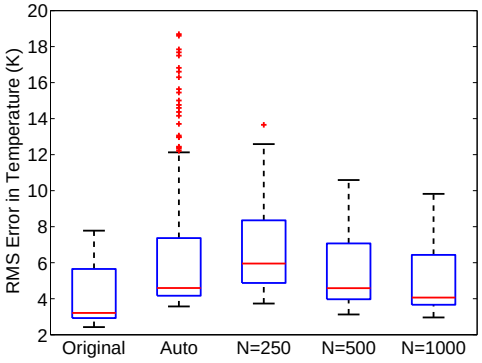
We then employed kernel controllers for controlling an approximation to the scalar diffusion equation $u_t = bu_{xx}$ on the domain $\Omega = [0, 1]$, with $b = 0.25$. The solution to this equation is infinite-dimensional, so we chose a kernel $k(x, y) = e^{-(\|x-y\|^2/2\sigma^2)}$, and a set of atoms $\mathcal{F}^{\mathcal{C}} = \{c_1, \ldots, c_M\}$, $c_i \in \Omega$, with $M = 25$ generating $\mathcal{H}^{\mathcal{C}}$, the space approximating $\mathcal{H}$, and another set of atoms $\mathcal{F}^{D} = \{\psi(d_1), \ldots, \psi(d_{\ell'})\}$, $d_j \in \Omega$, $\ell' = 13$, generating the control space $\widetilde{\mathcal{H}}$. The number of, and the location of the observations was chosen to be the same as that of the actuation locations d_j. First, tests (not reported here) were conducted to ensure that the solution to the diffusion equation is well approximated in $\mathcal{H}^{\mathcal{C}}$. Matrix least-squares was used to infer $\widehat{A}$. Figure 6.13a shows an example of an initial function f_{init} evolving according to the PDE. A reference function $f_{\text{ref}} \in \mathcal{H}^{\mathcal{C}}$ was chosen to drive f_{init} to f_{ref} under the action of the PDE. Finally, Algorithm 8 was used to control the PDE, driving f_{init} to f_{ref}; Figure 6.13b shows the absolute value of the error between f_k and f_{ref} as a function of time.



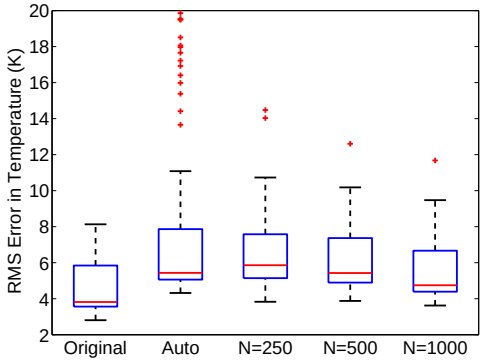
(a) Error/day (time-series)



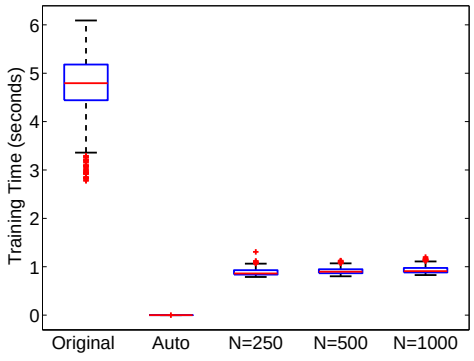
(b) Error/night (time-series)



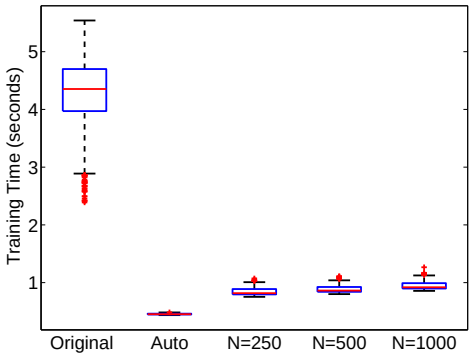
(c) Error (day)



(d) Error (night)

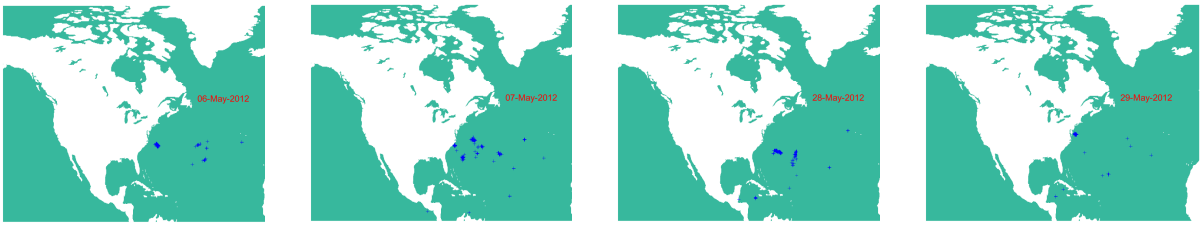


(e) Estimation time (day)



(f) Estimation time (night)

Figure 6.11: Performance of the kernel observer over AVVHR satellite 2012 data with different numbers of observation locations.

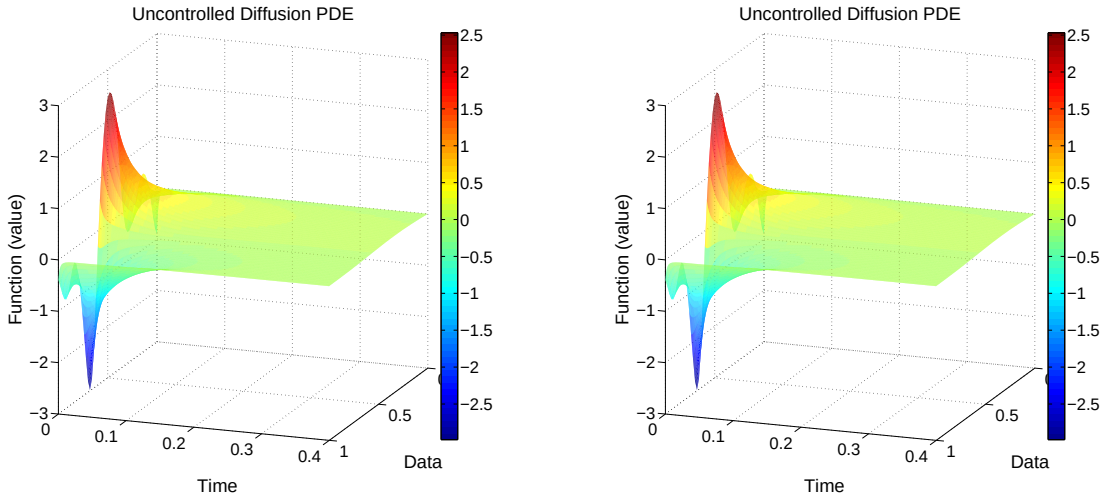


(a) 6th May (b) 7th May (c) 28th May (d) 29th May

Figure 6.12: Locations of weather anomaly obtained based on the error between the acutual temperature and the prediction of autonomous kernel observer. Landmass is shown in white and the ocean is in green. Locations marked have error greater than two standard deviations above the mean error.

Algorithm 8 Kernel Controller

Input: Kernel k, basis points $\mathcal{C}$, estimated system matrix $\widehat{A}$, estimated covariance matrix $\widehat{B}_{\mathcal{C}}$, and function f_{ref} to drive initial function to.
Initialize Observer: (see Algorithm 7).
Initialize Controller: Use Jordan decomposition of $\widehat{A}$ to obtain no. of control locations $\mathcal{D}$, compute kernel matrix $K_{CD} \in \mathbb{R}^{\ell' \times M}$ between $\mathcal{D}$ and $\mathcal{C}$, and initialize controller (e.g. LQR) utilizing $(\widehat{A}, \widehat{B}_{\mathcal{C}})$.
while measurements available **do**
1) Sample data $\{y_k^i\}_{i=1}^N$ from $f(x, \tau)$.
2) Utilize observer to estimate $\widehat{w}_{\tau+1}$.
3) Use $\widehat{w}_{\tau+1}$ and f_{ref} as input to controller to get feedback.
end while



(a) Evolution of initial function f_{init} according to diffusion equation.

(b) Error in absolute value between controlled pde and f_{ref}.

Figure 6.13: Demonstration of the control of a linear diffusion equation.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This thesis has developed enabling algorithms for learning, inference, and control for two challenging problem settings: i) Construction robots, and ii) Complex spatiotemporal processes. As a solution, this thesis has presented several contributions, these are elaborated as follows.

In Chapter 3, learning of instructional policy model from expert task demonstration has been presented. This policy model enables robots to generate instructions for assisting, and training novice operators in the execution of complex construction tasks. We have introduced action primitives that addresses the problem of mapping observed continuous state action trajectories to human parse-able instructions. Contributions in this chapter included:

- Efficient and meaningful segmentation of demonstration trajectories using action primitives. Existing state of the art methods rely on sampling based inference and are hence inefficient. The generated segments are meaningful as they directly translate to human operator's action space, which is not the case for any of the existing methods.

- Learning policy from state-action primitive pairs $(s, \mathbf{r})$, instead of state-action pairs (s, a) improves scalability. We have shown that the number of $(s, \mathbf{r})$ pairs are at least an order of magnitude less than the number of (s, a) pairs.

- Structured probabilistic policy model approach for learning policy for complex construction tasks that are comprised of several subtasks. Whereas the existing methods rely on heuristics or hand-coded approach to model transition between subtasks.

- Exhaustive experiments to test instructional policy involving 113 human participants as novice operators were conducted. Our experiments demonstrate statistically significant improvement in the learning rate and retention of skills by the novice operators.

In Chapter 4, we proposed a likelihood rate based estimation technique to capture non-stationarity in Markov sequences to allow accurate state inference. This approach is further utilized to infer switch between different construction tasks. Contributions of this method are:

- Proposed Layered Non-stationary Markov Model (LNMM), as a generative model for non-stationary Markov chains.

- Likelihood rate approach estimates multiple transition models that are generating the observations. This allows quick inference of the current transition model in comparison to the existing methods that slowly converge to the current model.

- Experiments on synthetic and honey bee dance data-set show that the inference using LNMM is two times more accurate than the existing unsupervised learning methods while being computationally efficient.

- Our experiments demonstrate that the likelihood rate based estimation successfully infers switch between different construction tasks.

In Chapter 5, a continuous shared control design has been proposed that retains the benefits of shared control while ensuring safety in off-nomial situations. Contributions of this chapter are:

- Proposed intent aware shared control that takes into account the operator's intent and quickly relinquishes control to the operator in off-nominal conditions.

- Human experiments on the Zermelo's navigation problem with pop-up obstacles demonstrated significantly better performance in terms of safety and efficiency over existing blended shared control technique.

In Chapter 6, Kernel Observer, a systems theoretic approach for modeling and inference of spatiotemporally varying processes has been presented. Contributions of this chapter included:

- Modeled spatiotemporal functional evolution using stationary kernels with a linear dynamical systems layer on their mixing weights.

- Harnessed kernel observer model to determine sensing locations with the guarantee that the hidden states of functional evolution can be estimated using a Bayesian state-estimator (Kalman filter) with very few measurements.

- Sufficient conditions on the number and location of required sensor measurements. Non-conservative lower bounds on the minimum number of sensing locations were derived.

- Theoretical results on random sensor placement that establish i) the expected number of randomly placed sensors required to monitor a given spatiotemporal process, and ii) the probability with which optimal sensor placement is ensured given the number of randomly placed sensors.

7.1 Future Work

In continuation of the contributions made in this thesis, there are several extensions which could serve as future work. These include, i) Utilization of the structure of the proposed policy model to learn policy for real world construction tasks, ii) Learning policy from multiple experts, iii) Policy improvement through reward feedback, and iv) Shared control for an excavator.

7.1.1 Policy Model for Real World Construction Tasks

Work done in this thesis provides a baseline for further development in the policy model to realize its utilization for a real world construction task. The proposed approach of designing policy model as a dynamical Bayesian network provides the flexibility to restructure the policy model to capture the dependence between additional variables and actuator velocities.

For a real world task it is crucial to account for robot's interaction with different types of material. Since the dynamics of excavation might differ based on the material being removed. For example, excavation can encounter hard rocks, loose gravels, sand, soft sedimentary deposits or wet soil, with different set of task dynamics in each case. Interactions w.r.t different types of materials can be captured through the forces acting on the end-effector as well as by recording pressures in the hydraulic actuators. Designing a suitable Bayesian network is required to capture the dependence of these additional variables onto the actuator velocities, ultimately to learn a policy for such real world construction scenarios.

Further, the subgoal learning aspect of the policy model provides an additional flexibility to capture different types of manipulations performed using an excavator. In this thesis, all subgoals were modeled using a multivariate Gaussian distribution which captured the nature of manipulation w.r.t the sand and w.r.t the truck. However, to capture the digging operation it would be required to utilize a distribution that outputs higher depths for digging as the operation proceeds, skewed Beta distribution could be a solution. Similarly, different distribution can be utilized to capture different manipulations w.r.t the manipulated task objects. The proposed probabilistic structure of the policy model has enormous potential to incorporate additional variables and learn a suitable policy model required for real world construction tasks.

7.1.2 Learning from Multiple Experts

While much of the thesis has focused on learning policy model from a single demonstrator, a framework can be developed to identify multiple demonstrators from a set of observed trajectories, and then selectively learn policy from an optimal demonstrator. In this scenario, it is likely that in-

dividual demonstrators will perform the same overall tasks but do so in potentially different ways. The policy learning framework would then need the ability to not only learn the overall tasks, but differentiate between demonstrators within those tasks. This could be accomplished by evaluating divergence between the distributions learned from different demonstration trajectories. For each demonstration trajectory we obtain distributions for action primitives as well as the subgoals. Thus, KL divergence or any other information-theoretic divergence measure can then be utilized to evaluate the number of different demonstrators. The learned measure would also provide insight into the commonalities and differences between different demonstrators solving the same task.

7.1.3 Policy Update

Beyond the work of learning policy models for each task, policy search can be applied to further improve a robot's performance through practice. This process involves updating policy model parameters based on the reward feedback obtained from task execution. For example, in case of truck loading reward can be formulated as a function of cycle time, and quantity of sand moved. Based on this reward function, policy parameters can be updated using policy gradient or expectation maximization techniques. These techniques work on the principle of policy search through parameter perturbation, followed by reward evaluation and policy update. With regards to parameter perturbation, the policy model developed has the advantage of guided search around the action primitive and subgoal parameters defined by the corresponding Gaussian distribution parameters.

7.1.4 Shared Control for Excavator Robot

Another key benefit of the policy model is to utilize its prediction to share control with the operator. Benefits of shared control are well established, and our work on intent aware shared control incorporates safety in shared control design. Another follow up work is to implement intent aware shared control on an actual excavator.

As discussed earlier, continuous shared control explores a form of arbitration between user input (u_o) and automatic controller prediction ($\tilde{u}$), to obtain control input u as

$$u = (1 - \alpha)u_o + \alpha\tilde{u}$$

where $\alpha = 0$ results in full manual control and $\alpha = 1$ in fully autonomous control. Beginning from preset values for α, the literature has recently focused on using linear policy blending i.e. the value of α varying linearly between 0 and 1. Till date, literature has mostly been concerned with providing assistance or taking over control i.e. $\alpha \to 1$. These includes cases: i) where robot's

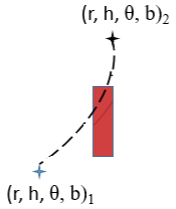


Figure 7.1: An example optimal trajectory in cylindrical coordinate system, with an unforeseen obstacle along the trajectory.

prediction and user's input are in agreement, ii) Task being difficult for humans to execute, iii) When user inputs are violating constraints or can potentially harm the system. However, the case when human operator should take over the control from the automatic controller has not received much attention. Such an action is required if the autonomous control input becomes undesirable due to the presence of unforeseen situations such as an obstacle, we define this earlier as off-nominal situations.

The intent aware shared control (IASC) proposed in section 5.3 ensures that the control is relinquished to the operator ($\alpha \to 0$) in off-nominal situations. This design quantifies human intent to derive the value of α, and uses exponential function to switch between the two extremes. In this thesis, IASC was implemented on a single control input Zermelo's navigation problem. Results in section 5.5.1 demonstrated that this design performed as good as the existing methods while achieving the objective of relinquishing control in off-nominal situations.

The end goal of IASC design is to ensure safety while implementing shared control for real world robots such as an excavator. Benefits of shared control for an excavator robot has been demonstrated through simulations by Enes and Book [103] and recently through experiments by Mitch et al. [114]. Reduction in cycle time is the main advantage, but we claim reduction in operator fatigue as another advantage for long hours of operation. However, these shared control designs do not guarantee safety in off-nominal situations as was evident in the case of Zermelo's navigation. Thus, we propose to develop IASC design for excavator robot a constrained multi-input system, to gain the benefits of shared control while ensuring safety. Next, we outline a formulation that will be tested on the scaled excavator robot.

As a first step for IASC, we will quantify operator's intent for motion along a trajectory. Consider a trajectory in cylindrical coordinate system augmented with a dimension that defines bucket motion as shown in Figure 7.1. In this representation, to a large extent, there is one to one relation between each actuator and a d.o.f: arm controls radius (r), boom controls the height (h), turret rotation control the angle (θ), and bucket actuator controls the bucket motion (b). Let $\tilde{q}$ be the desired optimal end-effector postion along the trajectory, and q^o be the one commanded by the operator,

then the intent ($\dot{\gamma}$) can be evaluated as

$$\dot{\gamma} = \max_i \frac{d}{dt}(|q_i^o - \tilde{q}_i|) \tag{7.1}$$

with the arbitration α evaluated as

$$\alpha = \begin{cases} \alpha_0(1 - e^{-t/\tau}) & \dot{\gamma} < -\eta \\ \alpha_0 e^{-t/\tau} & \dot{\gamma} > \eta \\ \alpha_{prev} & |\dot{\gamma}| < \eta \end{cases} \tag{7.2}$$

where α_0 is the max value, α_{prev} stands for the current value of α. Thus, if the operator deviates from the optimal trajectory along any dimension such that $\dot{\gamma} > \eta$, then the control will be relinquished to the operator. Finally, the control input incorporating the constraint of maximum hydraulic flow Q will be

$$u = u_o - \alpha(u_o - \tilde{u}) \tag{7.3}$$

$$\text{subject to} \quad \sum_{i=1}^{n} A_i u_i \leq Q \tag{7.4}$$

where $u, u_o, \tilde{u} \in \mathbb{R}^n$, with n being the number of actuators, and A_i is the area of piston head or rod end of the hydraulic piston cylinder actuator.

REFERENCES

[1] David A Bradley and Derek W Seward. The development, control and operation of an autonomous robotic excavator. *Journal of Intelligent and Robotic Systems*, 21(1):73–97, 1998.

[2] Anthony Stentz, John Bares, Sanjiv Singh, and Patrick Rowe. A robotic excavator for autonomous truck loading. *Autonomous Robots*, 7(2):175–186, 1999.

[3] Guilherme J Maeda, David C Rye, and Surya PN Singh. Iterative autonomous excavation. In *Field and Service Robotics*, pages 369–382. Springer, 2014.

[4] Ian Cornford and James Athanasou. Developing expertise through training. *Industrial and Commercial Training*, 27(2):10–18, 1995.

[5] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[6] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20, 1997.

[7] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547, 2003.

[8] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[9] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[11] Daniel H Grollman and Odest Chadwicke Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 261–266. IEEE, 2010.

[12] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[13] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[14] Zhe Chen and Robert S Siegler. Young childrens analogical problem solving: Gaining insights from video displays. *Journal of experimental child psychology*, 116(4):904–913, 2013.

[15] Emma Flynn and Andrew Whiten. Dissecting childrens observational learning of complex actions through selective video displays. *Journal of experimental child psychology*, 116(2):247–263, 2013.

[16] Fred Paas, Juhani E Tuovinen, Huib Tabbers, and Pascal WM Van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational psychologist*, 38(1):63–71, 2003.

[17] Jeroen JG Van Merriënboer, Liesbeth Kester, and Fred Paas. Teaching complex rather than simple tasks: Balancing intrinsic and germane load to enhance transfer of learning. *Applied cognitive psychology*, 20(3):343–352, 2006.

[18] Tim P Barnett, David W Pierce, and Reiner Schnur. Detection of anthropogenic climate change in the world's oceans. *Science*, 292(5515):270–274, 2001.

[19] Hilmar Bungum, Conrad Lindholm, and JI Faleide. Postglacial seismicity offshore mid-norway with emphasis on spatio-temporal–magnitudal variations. *Marine and Petroleum Geology*, 22(1):137–148, 2005.

[20] Jin S Deng, Ke Wang, Yang Hong, and Jia G Qi. Spatio-temporal dynamics and evolution of land use change and landscape pattern in response to rapid urbanization. *Landscape and Urban Planning*, 92(3):187–198, 2009.

[21] Matthew J Heaton, Matthias Katzfuss, Shahla Ramachandar, Kathryn Pedings, Eric Gilleland, Elizabeth Mannshardt-Shamseldin, and Richard L Smith. Spatio-temporal models for large-scale indicators of extreme weather. *Environmetrics*, 22(3):294–303, 2011.

[22] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2011.

[23] Carl Edward Rasmussen. Gaussian processes for machine learning. Technical report, Max Planck Institute for Biological Cybernetics, 2006.

[24] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[25] Sahil Garg, Amarjeet Singh, and Fabio Ramos. Learning non-stationary space-time models for environmental monitoring. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[26] Chunsheng Ma. Nonstationary covariance functions that model space–time interactions. *Statistics & Probability Letters*, 61(4):411–419, 2003.

[27] Christian Plagemann, Kristian Kersting, and Wolfram Burgard. Nonstationary gaussian process regression using point estimates of local smoothness. In *Machine learning and knowledge discovery in databases*, pages 204–219. Springer, 2008.

[28] Girish Chowdhary, Hassan A Kingravi, Jonathan P How, and Patricio A Vela. Bayesian nonparametric adaptive control using gaussian processes. *IEEE transactions on neural networks and learning systems*, 26(3):537–550, 2015.

[29] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[30] A. Gelb. *Applied Optimal Estimation*. MIT press (QA402.A5), Cambridge, MA, 1974.

[31] Kanti V Mardia, Colin Goodall, Edwin J Redfern, and Francisco J Alonso. The kriged kalman filter. *Test*, 7(2):217–282, 1998.

[32] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, 2013.

[33] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5239–5246. IEEE, 2012.

[34] W Murray Wonham. *Linear multivariable control*. Springer, 1974.

[35] Robin Jaulmes, Joelle Pineau, and Doina Precup. Active learning in partially observable markov decision processes. In *European Conference on Machine Learning*, pages 601–608. Springer, 2005.

[36] Luca Francesco Bertuccelli. *Robust decision-making with model uncertainty in aerospace systems*. PhD thesis, Citeseer, 2008.

[37] Derek Seward, Frank Margrave, Ian Sommerville, and Richard Morrey. Lucie the robot excavator-design for system safety. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 963–968. IEEE, 1996.

[38] Sanjiv Singh. State of the art in automation of earthmoving. *Journal of Aerospace Engineering*, 10(4):179–188, 1997.

[39] Zhuo Wu, Ke Yi Sun, Ming Song, and Min Zheng. Design an autonomous excavation system for hydraulic excavators. *Applied Mechanics and Materials*, 437:471–474, 2013.

[40] Young Bum Kim, Junhyoung Ha, Hyuk Kang, Pan Young Kim, Jinsoo Park, and FC Park. Dynamically optimal trajectories for earthmoving excavators. *Automation in Construction*, 35:568–578, 2013.

[41] Jongwon Seo, Seungsoo Lee, Jeonghwan Kim, and Sung-Keun Kim. Task planner design for an automated excavation system. *Automation in Construction*, 20(7):954–966, 2011.

[42] Jun Gu, James Taylor, and Derek Seward. Proportional-integral-plus control of an intelligent excavator. *Computer-Aided Civil and Infrastructure Engineering*, 19(1):16–27, 2004.

[43] Parvaneh Saeedi, Peter D Lawrence, David G Lowe, Poul Jacobsen, Dejan Kusalovic, Kevin Ardron, and Paul H Sorensen. An autonomous excavator with vision-based track-slippage control. *Control Systems Technology, IEEE Transactions on*, 13(1):67–84, 2005.

[44] Yang Liu, Mohammad Shahidul Hasan, and Hong-Nian Yu. Modelling and remote control of an excavator. *International Journal of Automation and Computing*, 7(3):349–358, 2010.

[45] Yan Jun, Li Bo, Tu Qunzhang, Guo Gang, and Zeng Yonghua. Automatization of excavator and study of its autocontrol. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on*, volume 1, pages 604–609. IEEE, 2011.

[46] Labor Statistics Bureau. Census of fatal occupational injuries (cfoi) - current and revised data, 2013. Retrieved February 3, 2015.

[47] Anca D Dragan and Siddhartha S Srinivasa. A policy-blending formalism for shared control. *The International Journal of Robotics Research*, 32(7):790–805, 2013.

[48] Harshal Maske, Girish Chowdhary, and Prabhakar Pagilla. Intent aware shared control in off-nominal situations. In *Conference on Decisions and Control*, Las Vegas, USA, 2016. IEEE.

[49] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[50] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[51] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[52] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[53] Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. In *Proc. Robot. Sci. Syst.*, pages 33–40, 2010.

[54] Andrew Y Ng, H Jin Kim, Michael I Jordan, Shankar Sastry, and Shiv Ballianda. Autonomous helicopter flight via reinforcement learning. In *NIPS*, volume 16, 2003.

[55] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, page 0278364911428653, 2011.

[56] Jesse Butterfield, Sarah Osentoski, Graylin Jay, and Odest Chadwicke Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 328–333. IEEE, 2010.

[57] Nadia Figueroa and Aude Billard. Transform-invariant non-parametric clustering of covariance matrices and its application to unsupervised joint segmentation and action discovery. *arXiv preprint arXiv:1710.10060*, 2017.

[58] Odest Chadwicke Jenkins and Maja J Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM, 2004.

[59] Kevin R Dixon and Pradeep K Khosla. Trajectory representation using sequenced linear dynamical systems. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3925–3930. IEEE, 2004.

[60] Michael Gienger, Manuel Mühlig, and Jochen J Steil. Imitating object movement skills with robotsa task-level approach exploiting generalization and invariance. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1262–1269. IEEE, 2010.

[61] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. Joint modeling of multiple related time series via the beta process. *arXiv preprint arXiv:1111.4226*, 2011.

[62] Nadia Figueroa and Aude Billard. Learning complex manipulation tasks from heterogeneous and unstructured demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada*, 2017.

[63] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

[64] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[65] Bernard Michini, Thomas J Walsh, Ali-Akbar Agha-Mohammadi, and Jonathan P How. Bayesian nonparametric reward learning from demonstration. *IEEE Transactions on Robotics*, 31(2):369–386, 2015.

[66] T. J. Nokes, C. D. Schunn, and Michelene T. H. Chi. Problem solving and human expertise. In Penelope Peterson, Eva Baker, and Barry McGaw, editors, *International Encyclopedia of Education*, volume 5, pages 265–272. Elsevier, Oxford, 5 edition, 2010.

[67] Michael Harré, Terry Bossomaier, and Allan Snyder. The development of human expertise in a complex environment. *Minds and Machines*, 21(3):449–464, 2011.

[68] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[69] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2007.

[70] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*, 2012.

[71] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

[72] Bernard Michini, Mark Cutler, and Jonathan P How. Scalable reward learning from demonstration. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 303–308. IEEE, 2013.

[73] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 471–477. IEEE, 2015.

[74] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 513–520, 2012.

[75] Christopher K Wikle. A kernel-based spectral approach for spatio-temporal dynamic models. In *Proceedings of the 1st Spanish Workshop on Spatio-Temporal Modelling of Environmental Processes (METMA)*, pages 167–180, 2001.

[76] Christopher K Wikle. A kernel-based spectral model for non-gaussian spatio-temporal processes. *Statistical Modelling*, 2(4):299–314, 2002.

[77] Fernando Pérez-Cruz, Steven Van Vaerenbergh, Juan José Murillo-Fuentes, Miguel Lázaro-Gredilla, and Ignacio Santamaria. Gaussian processes for nonlinear signal processing: An overview of recent advances. *Signal Processing Magazine, IEEE*, 30(4):40–50, 2013.

[78] Girish Chowdhary, Hassan A Kingravi, Jonathan P How, and Patricio A Vela. Bayesian nonparametric adaptive control of time-varying systems using gaussian processes. In *American Control Conference (ACC), 2013*, pages 2655–2661. IEEE, 2013.

[79] David Higdon. A process-convolution approach to modelling temperatures in the north atlantic ocean. *Environmental and Ecological Statistics*, 5(2):173–190, 1998.

[80] C Paciorek and M Schervish. Nonstationary covariance functions for gaussian process regression. *Advances in neural information processing systems*, 16:273–280, 2004.

[81] Amarjeet Singh, Fabio Ramos, H Durrant-Whyte, and William J Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5490–5497. IEEE, 2010.

[82] Tilmann Gneiting, Marc G Genton, and Peter Guttorp. Geostatistical space-time models, stationarity, separability, and full symmetry. *Monographs On Statistics and Applied Probability*, 107:151, 2006.

[83] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[84] David J Nott and William TM Dunsmuir. Estimation of nonstationary spatial covariance structure. *Biometrika*, pages 819–829, 2002.

[85] Roman Garnett, Michael A Osborne, and Stephen J Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2010.

[86] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2009.

[87] Avery N Gilbert, Alan J Fridlund, and Laurie A Lucchina. The color of emotion: A metric for implicit color associations. *Food Quality and Preference*, 52:203–210, 2016.

[88] Rui Gong, Qing Wang, Yan Hai, and Xiaopeng Shao. Investigation on factors to influence color emotion and color preference responses. *Optik-International Journal for Light and Electron Optics*, 136:71–78, 2017.

[89] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE, 2009.

[90] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[91] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. The sticky hdp-hmm: Bayesian nonparametric hidden markov models with persistent states. *Arxiv preprint*, 2007.

[92] Sylvia Fruhwirth-Schnatter. *Finite mixture and Markov switching models*. Springer Science & Business Media, 2006.

[93] Emily Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. Nonparametric bayesian learning of switching linear dynamical systems. In *Advances in Neural Information Processing Systems*, pages 457–464, 2009.

[94] Sang Min Oh, James M Rehg, Tucker Balch, and Frank Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77(1-3):103–124, 2008.

[95] Mica R Endsley. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42(3):462–492, 1999.

[96] Thomas B Sheridan. *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[97] Cristina Urdiales, Alberto Poncela, Isabel Sanchez-Tato, Francesco Galluppi, M Olivetti, and F Sandoval. Efficiency based reactive shared control for collaborative human/robot navigation. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3586–3591. IEEE, 2007.

[98] Blanca Fernández-Espejo, Alberto Poncela, Cristina Urdiales, and F Sandoval. Collaborative emergent navigation based on biometric weighted shared control. In *Computational and Ambient Intelligence*, pages 814–821. Springer, 2007.

[99] Rory A Cooper. Intelligent control of power wheelchairs. *Engineering in Medicine and Biology Magazine, IEEE*, 14(4):423–431, 1995.

[100] Aaron Enes and Wayne Book. Blended shared control of zermelo's navigation problem. In *American Control Conference (ACC), 2010*, pages 4307–4312. IEEE, 2010.

[101] Jingjing Jiang and Alessandro Astolfi. Shared-control for the kinematic model of a mobile robot. In *53rd IEEE Conference on Decision and Control*, pages 62–67. IEEE, 2014.

[102] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[103] Aaron R Enes. *Shared control of hydraulic manipulators to decrease cycle time*. PhD thesis, Georgia Institute of Technology, 2010.

[104] Claire J Tomlin, Ian Mitchell, Alexandre M Bayen, and Meeko Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

[105] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[106] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[107] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, pages 682–688, 2001.

[108] Kemin Zhou, John C. Doyle, and Keith Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, NJ, 1996.

[109] Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, pages 143–145. Springer, 1984.

[110] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[111] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary gaussian process regression using a latent extension of the input space. *URL http://www. kyb. mpg. de/˜ tpfingst*, 2006.

[112] Lehel Csatö and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[113] Lixin Li, Xingyou Zhang, and Reinhard Piltner. A spatiotemporal database for ozone in the conterminous us. In *Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06)*, pages 168–176. IEEE, 2006.

[114] Mitch Allain, Shyam Konduri, Harshal Maske, Praabhakar R. Pagilla, and Girish Chowdhary. Blended shared control of a hydraulic excavator. In *The 20th World Congress of the International Federation of Automatic Control*, Toulouse, France, 2017.