Probabilistic Imitation Learning for Spatiotemporal Human-Robot Interaction

by

Joseph Campbell

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved June 2021 by the
Graduate Supervisory Committee:

Heni Ben Amor, Co-Chair
Georgios Fainekos, Co-Chair
Katsu Yamane
Subbarao Kambhampati

ARIZONA STATE UNIVERSITY

August 2021

ABSTRACT

Imitation learning is a promising methodology for teaching robots how to physically interact and collaborate with human partners. However, successful interaction requires complex coordination in time and space, i.e., knowing what to do as well as when to do it. This dissertation introduces Bayesian Interaction Primitives, a probabilistic imitation learning framework which establishes a conceptual and theoretical relationship between human-robot interaction (HRI) and simultaneous localization and mapping. In particular, it is established that HRI can be viewed through the lens of recursive filtering in time and space. In turn, this relationship allows one to leverage techniques from an existing, mature field and develop a powerful new formulation which enables multimodal spatiotemporal inference in collaborative settings involving two or more agents. Through the development of exact and approximate variations of this method, it is shown in this work that it is possible to learn complex real-world interactions in a wide variety of settings, including tasks such as handshaking, cooperative manipulation, catching, hugging, and more.

ACKNOWLEDGMENTS

My road to this dissertation was a long one, starting eight years ago when I approached my (future) MS advisor Georgios Fainekos inquiring about volunteering in the Cyber-Physical Systems Lab as a freshly admitted graduate student, despite knowing nothing about research nor how to conduct it. I am eternally grateful to him for providing such an incredible lab environment and showing me the ropes; it quite literally changed my life. This same gratitude extends to my PhD advisor, Heni Ben Amor, whom I consider a friend every bit as much as an advisor. With his encouragement and guidance I was able to grow as a researcher, and the existence of this very document is certainly testament to his patience as I learned through my many failures.

I would like to thank my other committee members. Katsu Yamane for being an amazing mentor at Honda Research Institute and for the enormous amount of effort spent on making my project a reality. Subbarao Kambhampati for bringing some much needed perspective to my work and for the insightful discussions.

I also owe a big debt of gratitude to my colleagues at both Interactive Robotics Lab and Cyber-Physical Systems Lab for the many coffees and entertaining discussions over the years. In particular, thank you to Simon Stepputtis for all the hiking, movies, and other fun adventures. Kevin Luck, for the stimulating debates about American and German culture. Shubham Sonawani and Geoff Clark for the great collaborations and ruminations about life. Bardh Hoxha, Erkan Tuncali, Shakiba Yaghoubi, Kangjin Kim, Michael Drolet, and all my other colleagues too numerous to list for all the fun times we shared since the day I started graduate school.

I would like to thank my friends and collaborators at Osaka University – Shuhei Ikemoto, Koh Hosoda, Arne Hitzmann, Hiroaki Masuda, Tsung-Yuan Chen, and many

others – who made every effort to make me feel welcome during my time in Japan; it was an experience I will never forget. A big thank you to Alex Toshev at Google and Sahika Genc at Amazon for being incredible hosts during my internships, I learned so much during my time there.

Finally, I would like to thank my parents, Mary and Kevin Campbell, as well as my siblings, Scott and Holly. You've fully supported my decisions over the years, including the huge leap I took when I left my position as a software engineer to pursue graduate school full-time. Although you may no longer be here to see me graduate, Mom, I'm sure you would be proud.

This material is based upon work supported by the National Science Foundation under EAPSI Fellowship Grant No. 1714060 and IIS-1749783, JSPS under KAKENHI Grant Number 18H01410, and the Honda Research Institute.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Collaborative robots that live and work alongside human partners have the potential to radically transform a variety of application domains including manufacturing, health care, and the service sector. Instead of relying on explicit commands from a human operator, such robots need to autonomously blend their behavior with that of an interaction partner in order to provide seamless human-robot interaction (HRI). This requires the constant monitoring of human behavior in conjunction with the proactive generation of appropriate robot responses. Unfortunately, even for moderately complex human-robot interaction scenarios this approach becomes intractable for traditional programming paradigms, since it involves many possible combinations of states and actions. In recent years, computational approaches to data-driven modeling of human-robot interaction have been put forward [102]. For example, it has been shown that *imitation learning* can be used to extract compact models of interaction dynamics from

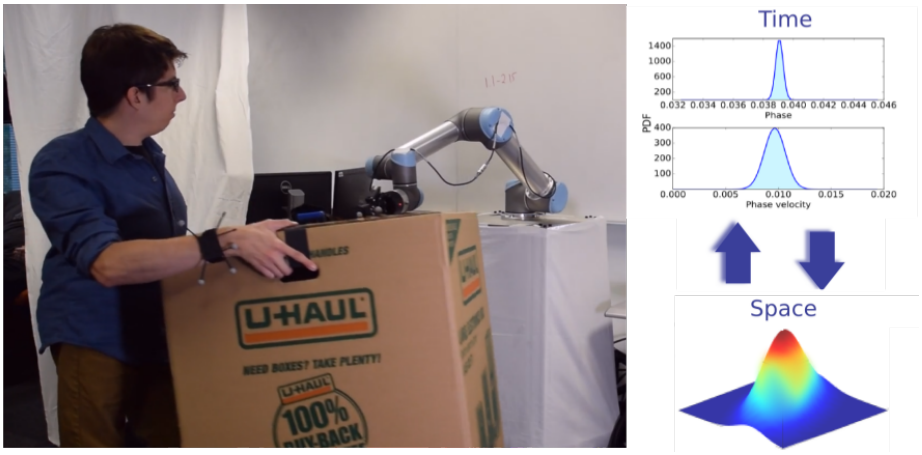


Figure 1: An example cooperative manipulation interaction in which both spatial and temporal components are a necessary prerequisite for success.

a set of human-human or human-robot demonstrations [55]. In turn, these models can be used by a robot at run-time to engage in similar interactions with a human partner.

A prominent approach to modeling such coupled behavior from example demonstrations, known as an Interaction Primitive (IP), has garnered a significant amount of attention [4, 71, 41, 33, 81, 35, 27] due to its relative simplicity and effectiveness. Informally, an IP models a joint probability distribution over the actions of two interaction partners, which is subsequently used to generate robot actions as a function of observed behavior via Bayesian inference. To this point, the original formulation of the Interaction Primitive and its variations all reduce to a probabilistic spatial estimation problem in the state space of the robot. Therein lies the primary drawback of this approach, as a successful human-robot interaction requires more than just spatial adaptation.

Timing is generally considered a critical facet of physical and social human-human interactions [50]. Even small mismatches in timing between interaction partners may severely reduce the likelihood of success in a collaborative task, regardless of the accuracy of the spatial movement. Take for example a hand-over of a food plate between two individuals. In such a scenario, the moment when one partner releases the plate and the other partner receives it is critical to ensuring physical stability during the exchange. Thus, we observe that the spatial and temporal inference problems are necessarily correlated and should be given equal consideration in human-robot interaction.

Despite this, the Interaction Primitive family of methods treats the challenge of timing as an independent variable that is external to the primary inference process; it must be estimated via explicit sequence alignment between observations of the human

and the provided demonstrations before spatial inference can take place. Not only does this approach result in poor computational performance [99], but by failing to leverage the uncertainty in each step, the overall inference accuracy suffers. Intuitively, uncertainty in the temporal estimate affects the uncertainty in the spatial estimate, and conversely uncertainty in the spatial estimate affects uncertainty in the temporal estimate (depicted in Fig. 1).

1.1 Problem Statement

The aim of this dissertation is to investigate the following hypothesis:

The temporal and spatial estimation problems of human-robot interaction are necessarily correlated, and by solving them simultaneously as a joint probabilistic inference problem we improve both the accuracy of our temporal estimate (and by extension, spatial estimate) as well as the robustness to different execution speeds.

Subsequently, I have tackled four separate, yet related research problems through a series of studies. Taken together, this body of work provides a comprehensive investigation of the overall hypothesis and proposes a unified solution to the challenge of data-driven human-robot interaction. These problems, outlined below, will be discussed in the remaining chapters of this dissertation.

1) Joint Spatiotemporal Inference: How can we perform joint spatiotemporal inference? In human-robot interaction, the spatial and temporal components of an interaction are equally important to success. If inference related to either component is inaccurate, the interaction will fail. Consequently, it is important that we leverage the uncertainty in each component in order to make an overall more accurate estimate; for

example, an uncertainty in the spatial component will necessarily affect the uncertainty in the spatial component.

2) Model-free Inference: Can we perform inference without requiring a dynamics model of either the robot or the environment? Often times, acquiring an accurate dynamics model is difficult in practice, requiring either a high-complexity model, many training samples, or both. For some robotic platforms of interest, for example musculoskeletal robots, these dynamics models are impractical to learn from a limited amount of real-world examples. As a result, we require inference which is able to operate in such scenarios without necessitating the use of an explicit dynamics model.

3) Multimodal Inference: How can we utilize sensors of multiple modalities during inference? Robots employ a limited number of sensors, and as a result nearly always have a more limited understanding of the world state than the human partner. To account for this fact, we would like to employ multiple sensors of different modalities that complement each other and help build a more complete state representation, while also providing robustness to sensor failure, e.g., camera occlusion.

4) Hierarchical Inference: Can we utilize inference in a hierarchical framework with other controllers/planners? If we are able to perform probabilistic inference and generate future trajectory distributions, this can be utilized by classical planning algorithms and controllers to generate more complex behaviors. For example, in a scenario involving haptic sensors, we could infer both the expected joint trajectory of the robot as well as the expected contact forces. A classical controller can then utilize these inputs to refine the actual control actions such that both are satisfied.

1.2 Summary of Contributions

Through a series of publications I have introduced a novel imitation learning family of algorithms for human-robot interaction known as Bayesian Interaction Primitives (BIP), which addresses the above-stated research problem. These algorithms present a recursive, probabilistic approach to joint spatiotemporal inference and establish a conceptual link between human-robot interaction and simultaneous localization and mapping – allowing us to draw from a mature field for inspiration. I have introduced both an exact inference formulation, as well as an approximate Monte Carlo-based formulation which relaxes several of the prior, limiting assumptions required for exact inference. These algorithms have been extensively analyzed in several challenging real-world scenarios, including an IRB-approved study with novel interaction partners. Furthermore, I have introduced an open source library known as *IntPrim*, which presents my algorithms and the original Interaction Primitives in an easy-to-use library with extensive support for real-world experiments, analysis, and documentation. This work will be discussed in more detail in the remaining chapters of this dissertation.

Chapter 2:

- ***Joseph Campbell*** *and Heni Ben Amor,* ***"Bayesian Interaction Primitives: A SLAM Approach to Human-Robot Interaction"****, CoRL 2017.* [18]
  This work introduces the original formulation of my novel algorithm, Bayesian Interaction Primitives. The exact formulation is introduced in this paper and the relationship to simultaneous localization and mapping is introduced. We show that BIP produces superior spatial and temporal inference while simultaneously resulting in a reduced computation cost, through both synthetic and real-world

human-robot interaction scenarios. This is the original paper which paves the way for my subsequent work.

- ***Joseph Campbell**, Arne Hitzmann, Simon Stepputtis, Shuhei Ikemoto, Koh Hosoda,and Heni Ben Amor, **"Learning Interactive Behaviors for Musculoskeletal Robots Using Bayesian Interaction Primitives"**, IROS 2019.* [23]

  This study was conducted in collaboration with Osaka University via an NSF grant and further analyzes how robust BIP is in the presence of varying execution speeds with novel interaction partners. We examine a handshake scenario in this paper conducted with participants drawn through an IRB-approved study. This work is notable for demonstrating that we can produce legible interactions even on a pneumatically-actuated musculoskeletal robot for which there is no explicit dynamics model. This work also introduced a novel response elicitation method for collecting demonstrations on a robot which is incapable of employing teleoperation nor kinesthetic teaching.

Chapter 3:

- ***Joseph Campbell**, Simon Stepputtis, and Heni Ben Amor, **"Probabilistic Multimodal Modeling for Human-Robot Interaction Tasks"**, RSS 2019.* [20]

  This work introduces an approximate formulation of BIP, known as Ensemble Bayesian Interaction Primitives (eBIP), by leveraging Monte Carlo sampling. This method was devised in response to a need for an algorithm which was capable of operating at high frequencies while simultaneously relaxing many of the limiting parametric assumptions required by the exact formulation. We demonstrate that these relaxed assumptions allow for increased inference accuracy

when utilizing input data from multiple modalities. The reduced computational complexity allowed us to successfully engage in a catching/throwing task in the real-world, in which the controlled agent was able to accurate infer where a ball would be thrown by directly observing the human partner.

- ***Joseph Campbell*** *and Katsu Yamane,* ***"Learning Whole-Body Human-Robot Haptic Interaction in Social Contexts"****, ICRA 2020.* [22]

  Conducted in collaboration with Honda Research Institute, this paper analyzes how well the approximate version of BIP is able to scale to challenging multimodal social interactions, namely hugging where the input includes full-body haptic sensing. We combine inference in a novel fashion with a lower-level force retargeting controller, allowing us to both predict actions as well as expected contact forces, which improves generalization with novel human partners. We also further analyze interesting spatial and temporal edge cases in order to identify conditions in which BIP is prone to failure.

Chapter 4:

- ***Joseph Campbell****, Simon Stepputtis, and Heni Ben Amor, "****IntPrim: An Interaction Primitives Library****", https://github.com/ir-lab/intprim.*

  This is a release of the open source library *IntPrim*. This library contains Python implementations of Bayesian Interaction Primitives, ensemble Bayesian Interaction Primitives, and several other variants.

- **Joseph Campbell**, Michael Drolet, and Heni Ben Amor, "**IntPrim Framework ROS: A ROS Framework for IntPrim**", https://github.com/ir-lab/intprim_framework_ros.

  This is a release of the open source library *IntPrim Framework ROS*. This is

a mixed C++/Python library which wraps *IntPrim* in a ROS framework such that real-world human-robot interactions can be easily configured and deployed.

Other Contributions:

- *Geoffrey Clark,* ***Joseph Campbell****, Seyed Mostafa Rezayat Sorkhabadi, Wenlong Zhang, and Heni Ben Amor, "****Predictive Modeling of Periodic Behavior for Human-Robot Symbiotic Walking****", ICRA 2020.* [31]
- *Geoffrey Clark,* ***Joseph Campbell****, and Heni Ben Amor, "****Learning Predictive Models for Ergonomic Control of Prosthetic Devices****", CoRL 2020.* [30]
- *Simon Stepputtis,* ***Joseph Campbell****, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor, "****Language-Conditioned Imitation Learning for Robot Manipulation Tasks****", NeurIPS 2020.* [101]
- *Kunal Bagewadi,* ***Joseph Campbell****, and Heni Ben Amor, "****Multimodal Dataset of Human-Robot Hugging Interaction****", AI-HRI 2019.* [8]
- *Kevin Luck,* ***Joseph Campbell****, Michael Andrew Jansen, Daniel M. Aukes, and Heni Ben Amor,* ***"From the Lab to the Desert: Fast Prototyping and Learning of Robot Locomotion"****, RSS 2017.* [68]
- *Andrew Jansen, Kevin Luck,* ***Joseph Campbell****, Heni Ben Amor, and Daniel M. Aukes,* ***"Bio-inspired Robot Design Considering Load-bearing and Kinematic Ontogeny of Chelonioidea Sea Turtles"****, Living Machines 2017.* [56]

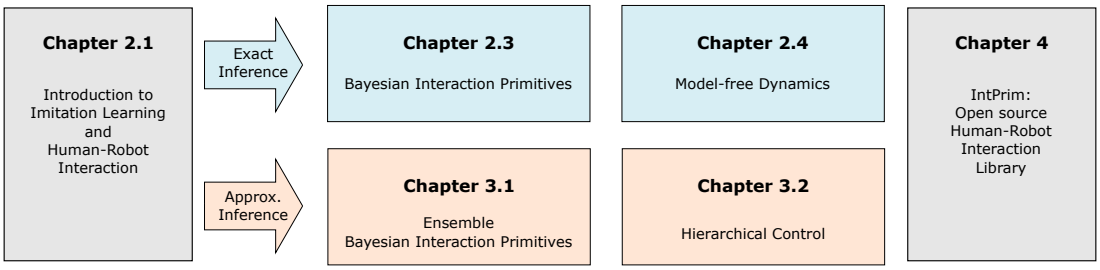


Figure 2: An overview of the chapters in this dissertation.

1.3 Outline

The remaining chapters in this dissertation will introduce probabilistic imitation learning methods for human-robot interaction. Each chapter will address a subset of the research problems introduced in Chapter 1.1, as shown in Fig. 2.

Chapter 2 introduces an exact solution for joint spatiotemporal inference in human-robot interaction and discusses how this can be utilized in a model-free dynamics capacity. This chapter addresses sub-problems **(1)** and **(2)** from Sec. 1.1.

Chapter 3 introduces an approximate solution for joint spatiotemporal inference, derived from the exact solution in Chapter 2. This chapter also describes how a hierarchical framework can be used which is capable of leveraging classical planners and control algorithms. Addresses sub-problems **(3)** and **(4)**.

Chapter 4 introduces an open-source library which simplifies the act of configuring and deploying human-robot interaction scenarios, particularly with the algorithms introduced in this dissertation.

Chapter 5 concludes this dissertation and identifies promising directions for future research.

Chapter 2

EXACT INFERENCE FOR HUMAN-ROBOT INTERACTION

In this chapter, we will introduce Bayesian Interaction Primitives (BIP), which is a human-robot interaction framework which focuses on extracting a probabilistic model of interaction dynamics as presented in a set of example demonstrations. Loosely speaking, human-robot interaction (HRI) is any sort of physical or non-physical interaction in which a human and a robot extend some form of mutual influence or action to each other. On it's own this is a massive field and can encompass interactions ranging from verbal communication to physical activity, and draws from fields such as natural language processing, computer vision, planning, control theory, and many more. Even the social sciences play an important role, as humans have certain expectations when interacting with others, and robots must be aware of and fulfill these expectations in order to result in comfortable interactions. In this dissertation, we restrict ourselves to cooperative physical human-robot interaction (pHRI); in such a scenario the robot and human work together towards a common goal which requires physical interaction with each other and/or the environment. Due to how we capture the dynamics of an interaction, BIP can actually be used to model any temporal multivariate observations which exhibit strong correlations, however, we will focus only on pHRI scenarios in this dissertation.

We define the dynamics of an interaction as how an interaction evolves over time, both in terms of the spatial relationship between interaction partners and the temporal relationship. For example, in a handshake interaction, the hands of each partner converge in physical space and clasp for an amount of time that is not explicitly

communicated, rather these are inherent to the dynamics of an interaction and are understood by both partners. Modeling these dynamics allows us to reproduce the full interaction after only observing a part of it because we know the temporal and spatial relationship over time. Fundamentally, this is what allows us to perform inference with Bayesian Interaction Primitives.

The key question that we are answering with Bayesian Interaction Primitives is: how do we model these dynamics? Deriving an analytical model up front is extremely challenging given the inherent variability in human behavior. Instead, we take a data-driven approach known as imitation learning or learning-from-demonstration, in which we observe example demonstrations of the interaction in question in order to extract the interaction dynamics. This approach resembles how humans and animals learn new skills: through observation.

2.1 Problem Formulation

We define an interaction $\boldsymbol{Y}$ as a time series of D-dimensional sensor observations over time, $\boldsymbol{Y}_{1:T} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T] \in \mathbb{R}^{D \times T}$. Of the D dimensions, D_o of them represent *observed* DoFs from one agent (the human) and D_c of them represent the *controlled* DoFs from the other agent (the robot), such that $D = D_c + D_o$. We will refer to the observed and controlled sensor measurements as $\boldsymbol{Y}^O_{1:T}$ and $\boldsymbol{Y}^C_{1:T}$ respectively. In an interaction, we assume there are both intra- and inter-relationships between the observed and controlled DoFs. Intuitively, for a specific interaction this means that there is an appropriate evolution in each DoF over time for each agent, i.e., a handshake must look like a handshake when viewing a single agent in isolation, and the controlled agent's DoFs must adapt to the observed agent's, i.e., each agent must

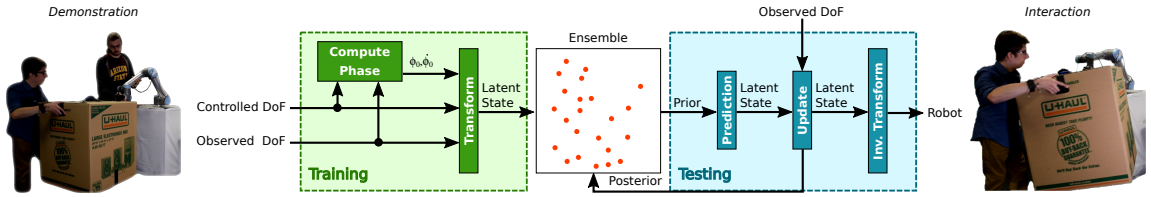


Figure 3: An overview of BIP. Training: training demonstrations are decomposed into a latent space and approximated with a distribution (here, approximately with an ensemble of samples). Testing: observations are collected during a live interaction which is used to perform filtering with the learned distribution and produce a response trajectory.

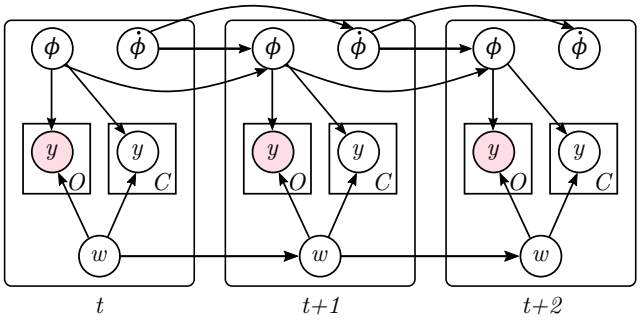
reach each other at the same point in space and time in order to successfully interact. We refer to these relationships as the *interaction dynamics*. In an imitation learning setting, our goal in a human-robot interaction scenario is to infer appropriate actions for the controlled agent's DoFs for current and future time steps of $t \leq T$ in response to observations of the human partner, given a set of example demonstrations which exhibit the desired interaction dynamics to serve as a prior.

Problem Definition: We can now probabilistically formulate our problem in a recursive manner as follows: given a set of example demonstrations $\mathcal{L}$, determine

$$p(\boldsymbol{Y}^C_{t:T}|\boldsymbol{Y}^O_{1:t}, \mathcal{L}) \propto p(\boldsymbol{y}^O_t|\boldsymbol{Y}^C_{t:T}) p(\boldsymbol{Y}^C_{t:T}|\boldsymbol{Y}^O_{1:t-1}, \mathcal{L}) \tag{2.1}$$

under the following assumptions:

1. The example demonstrations $\mathcal{L}$ demonstrate the desired interaction dynamics for the interaction at hand – no generalization to novel interaction scenarios.
2. The controlled and observed agents' task models are equal – no hidden goals or adversarial demonstrations.
3. Interactions are of a finite but variable length.



w	Trajectory shape
y_o	Observed DoFs
y_c	Controlled DoFs
ϕ	Phase
$\dot{\phi}$	Phase velocity

Figure 4: The graphical model for Bayesian Interaction Primitives. At time step t we make observations of each observed DoF y_o which are assumed to be sampled from the normal distribution parameterized by mean w at phase value ϕ.

2.2 Learning Human-Robot Interaction and the Challenge of Timing

In this section we will review the workflow associated with IPs and discuss how the spatial and temporal estimation problems are related. Interaction Primitives are a human-robot interaction framework in which basic building blocks of movement, i.e. primitives, are used to construct physical interactions between a robotic agent and a human. The objective is to learn the spatial and temporal dynamics of an interaction by observing examples of how humans interact. This process can be loosely categorized into two steps: training in which primitives are learned through human-human or human-robot demonstrations of an interaction; and inference in which the learned primitives are dynamically adapted to observations of the human, such that the robot's response is inferred. This workflow is captured in Fig. 3.

Training: (a) We initiate the training process by performing demonstrations of the interaction while capturing sensor trajectories of each degree of freedom (DoF) of interest (such as joint positions in 3D space). These demonstrations can either

be performed by two humans – in which case the correspondence problem must be addressed to provide a human-to-robot mapping – or a human and robot via kinesthetic teaching. (b) The trajectories for each DoF are projected to a lower dimensional latent space via basis function decomposition. While the original trajectories are time-varying, the latent representation is time-invariant such that it represents the entire trajectory of any length at any point in time. (c) The latent representation of each trajectory is used to construct a full joint probability distribution which is the trained model of our interaction. This effectively describes the correlations between the basis representations of all DoFs for both the robot and the human and enables inference given observed (or unobserved) DoFs.

Testing: (d) Having learned a model of the interaction, we now obtain a partially observed trajectory of the human beginning to perform the interaction. This observation is partial as we want the robot to immediately react to the human, as opposed to only responding after the human has finished their full motion. (e) We perform Bayesian inference over the joint probability distribution given the partial trajectory. This yields a posterior distribution over the latent representation of the current interaction with the human based on the observations to this point. (f) From the posterior distribution we extract the basis representations for the robot DoFs, project them back to the full measurement space, and perform the movement. The end result is that we infer the robot's *future* actions, given the partial observation of the human's motion and the prior demonstrations of the interaction. However, as our posterior distribution is a joint density over both the robot and human DoFs, we may also infer the human partner's future actions. This may be useful depending on the application domain, e.g., if ergonomics need to be taken into account.

In the original formulation of Interaction Primitives, step (e) consists of two discrete

steps: temporal estimation and spatial estimation. Informally, this is necessary because the joint probability distribution is in the time-invariant latent space of the interaction, however, the partial observations are in the original measurement space and are time-varying with an unknown length. In order to integrate the measurements into the joint probability distribution, we must project the latent space into the measurement space. This projection requires us to estimate the length of the current interaction based on the (partial) observations, or alternatively, how much of the interaction has already been completed; we define this as the temporal estimation problem. Once this projection has been computed, it is used to integrate the measurements and produce a posterior distribution, which is what we define as spatial estimation.

As previously indicated, in existing formulations of Interaction Primitives the temporal estimate is obtained via time alignment with Dynamic Time Warping. This approach has several undesirable traits: a) it's inherently non-probabilistic and only produces point estimates rather than distributions; b) it is a separate process and does not leverage any of the uncertainty information from the joint distribution; and c) it is computationally inefficient with high-dimensional state spaces, requiring $O(N^2)$ time complexity on average. Furthermore, this can result in discontinuous jumps in time which subsequently cause discontinuous jumps in space.

Our proposed solution is based on the insight that the spatial and temporal estimates are highly correlated in their errors, and as such they should be jointly estimated in a single probability distribution. This is shown mathematically in Sec. 2.3.3 after our proposed method has been formally defined, but for now we can provide an intuitive explanation. The same observations are used to compute both the temporal and spatial estimates, thus inducing a correlation in their uncertainty estimates. Furthermore, the latent state representation of the DoFs all share a

correlated error that itself is correlated with the temporal error. This can be succinctly described as follows: an error in time will necessarily result in an error in space. If we grossly misestimate one of these quantities, it will affect our ability to estimate the other, but by taking the uncertainties of both into account we can greatly improve the overall inference accuracy.

If this seems familiar, it is because a similar problem is encountered in Simultaneous Localization and Mapping [39] (SLAM), which serves as inspiration for our work. Algorithms such as FastSLAM [76] and EKF SLAM [103] effectively estimate a joint model of a robot's pose with respect to a map while simultaneously generating that same map. In such a scenario, the robot's pose serves as a frame of reference for observations of landmarks, and as such, an error in an estimate of the robot's pose will necessarily result in an error in the estimated absolute landmark positions in the map. Intuitively then, there is a similarity between the pose and map in SLAM and the temporal and spatial estimate in the HRI problem we have defined here.

2.3 Spatio-Temporal Inference with Bayesian Interaction Primitives

Having provided the rationale for our proposed solution, we now introduce it in a formal context. We define an interaction $\boldsymbol{Y}$ as a time series of D-dimensional sensor observations over time, $\boldsymbol{Y}_{1:T} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T] \in \mathbb{R}^{D \times T}$. Of the D dimensions, D_o of them represent *observed* DoFs from one agent (the human) and D_c of them represent the *controlled* DoFs from the other agent (the robot), such that $D = D_c + D_o$.

2.3.1 Basis Function Decomposition

Working with the time series directly is impractical due to the fact that the state space dimension would be proportional to the number of observations, so we transform the interaction $\boldsymbol{Y}$ into a latent space via basis function decomposition. Each dimension $d \in D$ of $\boldsymbol{Y}$ is approximated with a weighted linear combination of time-dependent nonlinear basis functions, such that $y_t^d = h^d(\phi(t), \boldsymbol{w}^d) = \Phi_{\phi(t)}^\intercal \boldsymbol{w}^d + \epsilon_y$, where $\Phi_{\phi(t)} \in \mathbb{R}^{1 \times B}$ is a row vector of B^d basis functions, $\boldsymbol{w}^d \in \mathbb{R}^{B \times 1}$, and ϵ_y is i.i.d. Gaussian noise. As this is a linear system with a closed-form solution, the weights $\boldsymbol{w}^d$ can be found through simple linear regression, i.e., least squares. The full latent model is composed of the aggregated weights from each dimension, $\boldsymbol{w} = [\boldsymbol{w}^{1\intercal}, \ldots, \boldsymbol{w}^{D\intercal}] \in \mathbb{R}^{1 \times B}$ where $B = \sum_d^D B^d$ and $\boldsymbol{y}_t = h(\phi(t), \boldsymbol{w})$.

We note that the time-dependence of the basis functions – and the nonlinear function $h(\cdot)$ – is not on the absolute time t, but rather on a relative phase value $\phi(t)$. Consider the basis function decompositions for a motion performed at slow speeds and fast speeds with a fixed measurement rate. If the time-dependence is based on the absolute time t, then the decompositions will be different despite the motion being spatially identical. Thus, we substitute the absolute time t with a linearly interpolated relative phase value, $\phi(t)$, such that $\phi(0) = 0$ and $\phi(T) = 1$. For notational simplicity, from here on we refer to $\phi(t)$ as simply ϕ.

2.3.2 Exact Bayesian Filtering

Given t observations of an interaction, $\boldsymbol{Y}_{1:t}$, the objective in Bayesian Interaction Primitives [18] is to infer the underlying latent model $\boldsymbol{w}$ while taking into account

a prior model $\boldsymbol{w}_0$. We assume that the t observations made so far are of a partial interaction, i.e., $\phi(t) < 1$, and that T is unknown. This requires the simultaneous estimation of the phase, as well as the phase velocity, i.e., how fast we are proceeding through the interaction, alongside the latent model. This joint estimation process is possible since the uncertainty estimates of each weight in the latent model are correlated due to a shared error in the phase estimate. In other words, if we mis-estimate where we are in the interaction in a temporal sense, we will mis-estimate where we are in a physical sense as well. This relationship is described in more detail in Sec. 2.3.3. Probabilistically, we represent this insight with the augmented state vector $\boldsymbol{s} = [\phi, \dot{\phi}, \boldsymbol{w}]$ and the following definition:

$$p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t}, \boldsymbol{s}_0) \propto p(\boldsymbol{y}_t|\boldsymbol{s}_t)p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t-1}, \boldsymbol{s}_0). \tag{2.2}$$

It is important to note that while the weights themselves are time-invariant with respect to an interaction, our estimate of the weights *is* time-varying. That is, every time we integrate a new sensor observation, our estimate of the underlying latent model is updated.

The posterior density in Eq. 2.2 is computed with a recursive linear state space filter, i.e., an extended Kalman filter [38]. Such filters are composed of two steps performed recursively: state prediction in which the state is propagated forward in time according to the system dynamics $p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t-1}, \boldsymbol{s}_0)$, and measurement update in which the latest sensor observation is incorporated in the predicted state $p(\boldsymbol{y}_t|\boldsymbol{s}_t)$. Applying Markov assumptions, the state prediction density can be defined as:

$$\begin{aligned} &p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t-1}, \boldsymbol{s}_0) \\ &= \int p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})p(\boldsymbol{s}_{t-1}|\boldsymbol{Y}_{1:t-1}, \boldsymbol{s}_0)d\boldsymbol{s}_{t-1}. \end{aligned} \tag{2.3}$$

As with all Kalman filters, we assume that all error estimates produced during recursion are normally distributed, i.e., $p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t}, \boldsymbol{s}_0) = \mathcal{N}(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})$ and $p(\boldsymbol{s}_t|\boldsymbol{Y}_{1:t-1}, \boldsymbol{s}_0) = \mathcal{N}(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$. The state evolves according to a linear constant velocity model:

$$\boldsymbol{\mu}_{t|t-1} = \underbrace{\begin{bmatrix} 1 & \Delta t & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}}_{\boldsymbol{G}} \boldsymbol{\mu}_{t-1|t-1}, \tag{2.4}$$

$$\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{G}\boldsymbol{\Sigma}_{t-1|t-1}\boldsymbol{G}^\intercal + \underbrace{\begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\dot{\phi}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}}_{\boldsymbol{Q}_t}, \tag{2.5}$$

where $\boldsymbol{Q}$ is the process noise associated with the state transition update. The noise correlations between phase and phase velocity, $\boldsymbol{\Sigma}_{\phi,\dot{\phi}}$, are determined by a piecewise or continuous first-order white noise model, e.g.,

$$\boldsymbol{\Sigma}_{\phi,\dot{\phi}} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{3} \\ \frac{\Delta t^3}{3} & \Delta t^2 \end{bmatrix} \sigma_\phi^2.$$

The observation function $h(\cdot)$ is nonlinear with respect to the state variable ϕ and must be linearized via Taylor expansion:

$$\begin{aligned} \boldsymbol{H}_t &= \frac{\partial h(\boldsymbol{s}_t)}{\partial s_t} \\ &= \begin{bmatrix} \frac{\partial \Phi_\phi^\intercal \boldsymbol{w}^1}{\partial \phi} & 0 & \Phi_\phi & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_\phi^\intercal \boldsymbol{w}^D}{\partial \phi} & 0 & 0 & \dots & \Phi_\phi \end{bmatrix}. \end{aligned} \tag{2.6}$$

Note that because the augmented state now includes the phase ϕ, the observation function $h(\boldsymbol{s})$ is simply a function of $\boldsymbol{s}$ in order to reduce notational clutter. We can

now integrate the measurement by calculating the innovation covariance as well as the Kalman gain, which dictates how heavily the observation should be weighted:

$$\boldsymbol{S}_t = \boldsymbol{H}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{H}_t^\intercal + \boldsymbol{R}_t, \tag{2.7}$$

$$\boldsymbol{K}_t = \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{H}_t^\intercal \boldsymbol{S}_t^{-1}. \tag{2.8}$$

This enables the calculation of the parameters for the posterior distribution,

$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t - h(\boldsymbol{\mu}_{t|t-1})), \tag{2.9}$$

$$\boldsymbol{\Sigma}_{t|t} = (I - \boldsymbol{K}_t \boldsymbol{H}_t)\boldsymbol{\Sigma}_{t|t-1}, \tag{2.10}$$

where $\boldsymbol{R}_t$ is the Gaussian measurement noise associated with the sensor observation $\boldsymbol{y}_t$.

The prior model $\boldsymbol{s}_0 = [\phi_0, \dot{\phi}_0, \boldsymbol{w}_0]$ is computed from a set of initial demonstrations. That is, given the latent models for N demonstrations, $\boldsymbol{W} = [\boldsymbol{w}_1^\intercal, \ldots, \boldsymbol{w}_N^\intercal]$, we define $\boldsymbol{w}_0$ as simply the arithmetic mean of each DoF:

$$\boldsymbol{w}_0 = \left[\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^1, \ldots, \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^D \right]. \tag{2.11}$$

The initial phase ϕ_0 is set to 0 under the assumption that all interactions start from the beginning. The initial phase velocity $\dot{\phi}_0$ is the arithmetic mean of the phase velocity of each demonstration:

$$\dot{\phi}_0 = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i}, \tag{2.12}$$

where T_i is the length of the i-th demonstration. The prior density is defined as $p(\boldsymbol{s}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ where

$$\boldsymbol{\mu}_0 = \boldsymbol{s}_0, \tag{2.13}$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\phi} & 0 & 0 \\ 0 & \boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}} & 0 \\ 0 & 0 & \boldsymbol{\Sigma}_{\boldsymbol{W},\boldsymbol{W}} \end{bmatrix}, \tag{2.14}$$

and $\boldsymbol{\Sigma}_{\phi,\phi}$ is the sample variance of the phases of the demonstrations, $\boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}}$ is the sample variance of the phase velocities, and $\boldsymbol{\Sigma}_{\boldsymbol{W},\boldsymbol{W}}$ is the sample covariance of the basis weights.

The baseline measurement noise $\boldsymbol{R}$ is also calculated from the set of initial demonstrations with the following closed-form solution:

$$\boldsymbol{R} = \frac{1}{N}\sum_{i}^{N}\frac{1}{T_i}\sum_{t}^{T_i}\left(\boldsymbol{y}_t - h(\phi(t), \boldsymbol{w}_i)\right)^2. \tag{2.15}$$

This value is equivalent to the mean squared error of the regression fit for our basis functions over every demonstration. Intuitively, this represents the variance of the data around the regression and captures both the approximation error and the sensor noise associated with the observations. The full algorithm is given in Fig. 5.

2.3.3 Correlations in Time and Space

With the BIP algorithm formally defined, we can now mathematically show that the spatial and temporal terms are correlated through observations. Recall that the covariance update is determined by Eq. 2.10, which follows from the standard derivation of the Kalman filter [38]:

$$\begin{aligned}
\boldsymbol{\Sigma}_{t|t} &= (I - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{\Sigma}_{t|t-1}, \\
&= (I - \boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t^\intercal\boldsymbol{S}_t^{-1}\boldsymbol{H}_t)\boldsymbol{\Sigma}_{t|t-1}, \\
&= \boldsymbol{\Sigma}_{t|t-1} - \boldsymbol{\Sigma}_{t|t-1}\boldsymbol{H}_t^\intercal\boldsymbol{S}_t^{-1}\boldsymbol{H}_t\boldsymbol{\Sigma}_{t|t-1}.
\end{aligned}$$

The observation matrix, $\boldsymbol{H}_t$, is a $D \times B + 2$ Jacobian which serves as a transformation from the $B + 2$-dimensional latent space to the D-dimensional measurement space. Each row corresponds to an entry in the measurement space and each column to an

Bayesian Interaction Primitives

Input: $\boldsymbol{W} = [\boldsymbol{w}_1^\intercal, \ldots, \boldsymbol{w}_N^\intercal] \in \mathbb{R}^{B \times N}$: set of B basis weights corresponding to N demonstrations, $\boldsymbol{l} = \left[\frac{1}{T_1}, \ldots, \frac{1}{T_N}\right] \in \mathbb{R}^{1 \times N}$: reciprocal lengths of demonstrations, $\boldsymbol{y}_t \in \mathbb{R}^{D \times 1}$: sensor observation at time t.

Output: $\hat{\boldsymbol{y}}_t \in \mathbb{R}^{D \times 1}$: the inferred trajectory at time t.

1. Initialize the prior distribution, $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, according to Eq. 2.12 and Eq. 2.14 such that

$$\boldsymbol{\mu}_0 = \left[0, \dot{\phi}_0, \boldsymbol{w}_0\right], \boldsymbol{\Sigma}_0 = \begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\phi} & 0 & 0 \\ 0 & \boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}} & 0 \\ 0 & 0 & \boldsymbol{\Sigma}_{\boldsymbol{W},\boldsymbol{W}} \end{bmatrix}.$$

2. For time step t, perform state prediction:

$$\begin{aligned} \boldsymbol{\mu}_{t|t-1} &= \boldsymbol{G}\boldsymbol{\mu}_{t-1|t-1}, \\ \boldsymbol{\Sigma}_{t|t-1} &= \boldsymbol{G}\boldsymbol{\Sigma}_{t-1|t-1}\boldsymbol{G}^\intercal + \boldsymbol{Q}_t. \end{aligned}$$

3. If a measurement $\boldsymbol{y}_t$ is available, perform the measurement update step from Eq. 2.10:

$$\begin{aligned} \boldsymbol{\mu}_{t|t} &= \boldsymbol{\mu}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t - h(\boldsymbol{\mu}_{t|t-1})), \\ \boldsymbol{\Sigma}_{t|t} &= (I - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{\Sigma}_{t|t-1}. \end{aligned}$$

4. **Output** the trajectory for each controlled DoF:

$$\hat{\boldsymbol{y}}_t = h(\boldsymbol{\mu}_{t|t}).$$

5. Repeat steps 2-5 until the interaction is concluded.

Figure 5: Bayesian Interaction Primitives

entry in the latent space, thus we can redefine this transformation for the sake of

clarity:

$$\begin{aligned}\boldsymbol{H}_t &= \begin{bmatrix} h^{\phi} & 0 & h^{\boldsymbol{w}^1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h^{\phi} & 0 & 0 & \dots & h^{\boldsymbol{w}^D} \end{bmatrix}, \\ &= \begin{bmatrix} \boldsymbol{H}^{\phi} & 0 & \boldsymbol{H}^{\boldsymbol{w}^1} & \dots & \boldsymbol{H}^{\boldsymbol{w}^D} \end{bmatrix}.\end{aligned}$$

The structure of this transformation reveals two things: 1) each observed DoF is relative with respect to the phase ϕ, resulting in entries for the phase and one of the weight vectors for each row; and 2) each observed DoF corresponds to exactly one weight vector. For the remainder of this proof, we assume that the $B^1 = \cdots = B^D = 1$ and $B = D$ without loss of generality.

Proposition 1: Assume that the n-th DoF is observed at time t with a finite observation noise $\boldsymbol{R}_t$, which yields the following observation matrix,

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{H}^{\phi} & 0 & 0 & \dots & \boldsymbol{H}^{\boldsymbol{w}^n} & \dots & 0 \end{bmatrix}.$$

Then this induces a correlation between the n-th DoF and the temporal state ϕ in the updated covariance matrix.

Proof: following the standard rules of matrix multiplication, the updated covariance entry takes the following form if there is no prior correlation at $t-1$:

$$\boldsymbol{\Sigma}_{t|t}^{\phi,\boldsymbol{w}^n} = -\boldsymbol{\Sigma}_{t|t-1}^{\phi,\phi} \boldsymbol{H}_t^{\phi\intercal} \boldsymbol{S}_t^{-1} \boldsymbol{H}_t^{\boldsymbol{w}^n} \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{w}^n,\boldsymbol{w}^n},$$

and the following form if there is a prior correlation at $t-1$:

$$\begin{aligned}\boldsymbol{\Sigma}_{t|t}^{\phi,\boldsymbol{w}^n} &= \boldsymbol{\Sigma}_{t|t-1}^{\phi,\boldsymbol{w}^n} \\ &- (\boldsymbol{\Sigma}_{t|t-1}^{\phi,\phi} \boldsymbol{H}_t^{\phi\intercal} + \boldsymbol{\Sigma}_{t|t-1}^{\phi,w^n} \boldsymbol{H}_t^{w^n\intercal}) \boldsymbol{S}_t^{-1} \boldsymbol{H}^{\phi} \boldsymbol{\Sigma}_{t|t-1}^{\phi,w^n} \\ &+ (\boldsymbol{\Sigma}_{t|t-1}^{\phi,\phi} \boldsymbol{H}_t^{\phi\intercal} + \boldsymbol{\Sigma}_{t|t-1}^{\phi,w^n} \boldsymbol{H}_t^{w^n\intercal}) \boldsymbol{S}_t^{-1} \boldsymbol{H}^{w^n} \boldsymbol{\Sigma}_{t|t-1}^{w^n,w^n}.\end{aligned}$$

In both cases, the covariance innovation $\boldsymbol{S}_t$ becomes correlated with both the n-th DoF, $\boldsymbol{w}^n$, as well as the temporal phase, ϕ, due to the fact that the observation is relative to the phase.

Proposition 2: Assume that the n-th DoF is observed at time t with a finite observation noise $\boldsymbol{R}_t$, and there is a correlation at time $t-1$ between the temporal state ϕ and the k-th DoF, where $k \neq n$, and

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{H}^{\phi} & 0 & 0 & \dots & \boldsymbol{H}^{\boldsymbol{w}^n} & \dots & 0 \end{bmatrix}.$$

Then this induces a correlation between the k-th DoF and the n-th DoF in the updated covariance matrix.

Proof: following the standard rules of matrix multiplication, the updated covariance entry takes the following form if there is no prior correlation at $t-1$:

$$\boldsymbol{\Sigma}_{t|t}^{\boldsymbol{w}^k,\boldsymbol{w}^n} = -\boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{w}^k,\phi} \boldsymbol{H}_t^{\phi\intercal} \boldsymbol{S}_t^{-1} \boldsymbol{H}_t^{\boldsymbol{w}^n} \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{w}^n,\boldsymbol{w}^n},$$

and the following form if there is a prior correlation at $t-1$:

$$\begin{aligned} \boldsymbol{\Sigma}_{t|t}^{\boldsymbol{w}^k,\boldsymbol{w}^n} &= \boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{w}^k,\boldsymbol{w}^n} \\ &- (\boldsymbol{\Sigma}_{t|t-1}^{\boldsymbol{w}^k,\phi} \boldsymbol{H}_t^{\phi\intercal} + \boldsymbol{\Sigma}_{t|t-1}^{w^k,w^n} \boldsymbol{H}_t^{w^n\intercal}) \boldsymbol{S}_t^{-1} \boldsymbol{H}^{w^n} \boldsymbol{\Sigma}_{t|t-1}^{w^n,w^n}. \end{aligned}$$

The covariance innovation $\boldsymbol{S}_t$ becomes correlated with both the n-th DoF, $\boldsymbol{w}^n$, as well as the k-th Dof, $\boldsymbol{w}^k$, due to the presence of an existing correlation between the $\boldsymbol{w}^k$ and ϕ.

Theorem 1: Assume that there are initially no correlations between the the DoFs,

such that the initial covariance matrix is,

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} \boldsymbol{\Sigma}^{\phi,\phi} & 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{\Sigma}^{\dot{\phi},\dot{\phi}} & 0 & \dots & 0 \\ 0 & 0 & \boldsymbol{\Sigma}^{\boldsymbol{w}^1,\boldsymbol{w}^1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \boldsymbol{\Sigma}^{\boldsymbol{w}^D,\boldsymbol{w}^D} \end{bmatrix}.$$

The phase ϕ and phase velocity $\dot{\phi}$ will become correlated with each other as well as any observed DoFs. Furthermore, each observed DoF will become correlated with each other observed DoF.

Proof: Upon state prediction at $t = 1$, covariance values between the phase and phase velocity are induced due to the system dynamics as defined in the state transition matrix $\boldsymbol{G}$ in Eq. 2.5:

$$\boldsymbol{\Sigma}_{1|0} = \begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\phi} & \boldsymbol{\Sigma}_{\phi,\dot{\phi}} & 0 & \dots & 0 \\ \boldsymbol{\Sigma}_{\dot{\phi},\phi} & \boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}} & 0 & \dots & 0 \\ 0 & 0 & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\boldsymbol{w}^1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\boldsymbol{w}^D} \end{bmatrix}.$$

Let the observation at $t = 1$ contain an observation of the first DoF, $\boldsymbol{w}^1$. By Proposition 1, this yields correlations between ϕ and $\boldsymbol{w}^1$, and by Proposition 2 correlations between $\dot{\phi}$ and $\boldsymbol{w}^1$. This results in the following posterior covariance matrix:

$$\boldsymbol{\Sigma}_{1|1} = \begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\phi} & \boldsymbol{\Sigma}_{\phi,\dot{\phi}} & \boldsymbol{\Sigma}_{\phi,\boldsymbol{w}^1} & \dots & 0 \\ \boldsymbol{\Sigma}_{\dot{\phi},\phi} & \boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}} & \boldsymbol{\Sigma}_{\dot{\phi},\boldsymbol{w}^1} & \dots & 0 \\ \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\phi} & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\dot{\phi}} & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\boldsymbol{w}^1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\boldsymbol{w}^D} \end{bmatrix}.$$

Now let the observation at $t = 2$ contain an observation of the last DoF, $\boldsymbol{w}^D$. By Proposition 1 this yields correlations between ϕ and $\boldsymbol{w}^D$, and by Proposition 2 correlations between $\dot{\phi}$ and $\boldsymbol{w}^D$, as well as between $\boldsymbol{w}^1$ and $\boldsymbol{w}^D$.

$$\boldsymbol{\Sigma}_{2|2} = \begin{bmatrix} \boldsymbol{\Sigma}_{\phi,\phi} & \boldsymbol{\Sigma}_{\phi,\dot{\phi}} & \boldsymbol{\Sigma}_{\phi,\boldsymbol{w}^1} & \dots & \boldsymbol{\Sigma}_{\phi,\boldsymbol{w}^D} \\ \boldsymbol{\Sigma}_{\dot{\phi},\phi} & \boldsymbol{\Sigma}_{\dot{\phi},\dot{\phi}} & \boldsymbol{\Sigma}_{\dot{\phi},\boldsymbol{w}^1} & \dots & \boldsymbol{\Sigma}_{\dot{\phi},\boldsymbol{w}^D} \\ \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\phi} & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\dot{\phi}} & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\boldsymbol{w}^1} & \dots & \boldsymbol{\Sigma}_{\boldsymbol{w}^1,\boldsymbol{w}^D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\phi} & \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\dot{\phi}} & \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\boldsymbol{w}^1} & \dots & \boldsymbol{\Sigma}_{\boldsymbol{w}^D,\boldsymbol{w}^D} \end{bmatrix}.$$

Thus we have established correlations between ϕ, $\dot{\phi}$, and between all observed DoFs. As this is a recursive algorithm, this holds for each observation at each time step. Furthermore, Proposition 1 and Proposition 2 hold when multiple DoFs are observed in a single time step and when the DoFs are initially correlated, as is the case in both BIP and eBIP. Note that this proof follows a similar structure to proofs encountered in EKF SLAM formulations when unobserved state variables are present.

2.4 Model-Free Application to Musculoskeletal Robots

An important property of the Bayesian Interaction Primitive formulation is that an explicit model of the robot's dynamics are not considered. This is both a benefit and a detriment; on the negative side this means that the algorithm alone cannot account for physical constraints resulting from either the robotic system or the environment, but accordingly this allows us to apply Bayesian Interaction Primitives to systems for which there is no known dynamics model. An interesting use case which is examined in this work is human-robot interaction with musculoskeletal robots.

To ensure that robotic interactions with humans are safe and productive, robots

need to be both mechanically and behaviorally responsive to their human counterparts. Mechanical responsiveness and compliance guarantees that no physical contact or force-exchange is harmful to the human. Traditional robotic systems are typically composed of non-compliant, rigid limbs that do not yield when contact with an opposing body occurs. In contrast to that, various musculoskeletal robots have been proposed which are based on McKibben pneumatic actuators [61]. These biologically-inspired systems mimic the behavior of human muscles and tendons, and are capable of providing a significant amount of force while remaining inherently safe due to their compliant, back-drivable nature. Furthermore, when arranged in a musculoskeletal structure which mimics human anatomy they tend to produce predictable, legible [36] motions. This is important in interactions with humans as unexpected movements may result in injury or unsafe situations. Unfortunately, due to the uncertainty and nonlinearity underlying pneumatic actuation, controlling such systems can pose major difficulties to the control framework [104, 108]. Often, getting a complex robot to perform smooth, generalizable actions on its own can be extremely challenging, let alone react to a human interaction partner.

Specifically, once external forces are applied to the muscles, it becomes difficult to accurately determine from the actuation pressure where in space the actuated components are. This limitation complicates control during interaction phases in which the robot experiences external forces, but it also introduces difficulties when training the robot. In learning from demonstration algorithms, a common technique to train the robot relies on kinesthetic guidance, *i.e.* the robot is physically moved along a desired trajectory and the internal states of the actuators are recorded. While it is possible to kinesthetically teach musculoskeletal robots actuated with PAMs [53], this requires a specific design of the robot that is not always feasible. In the case of the

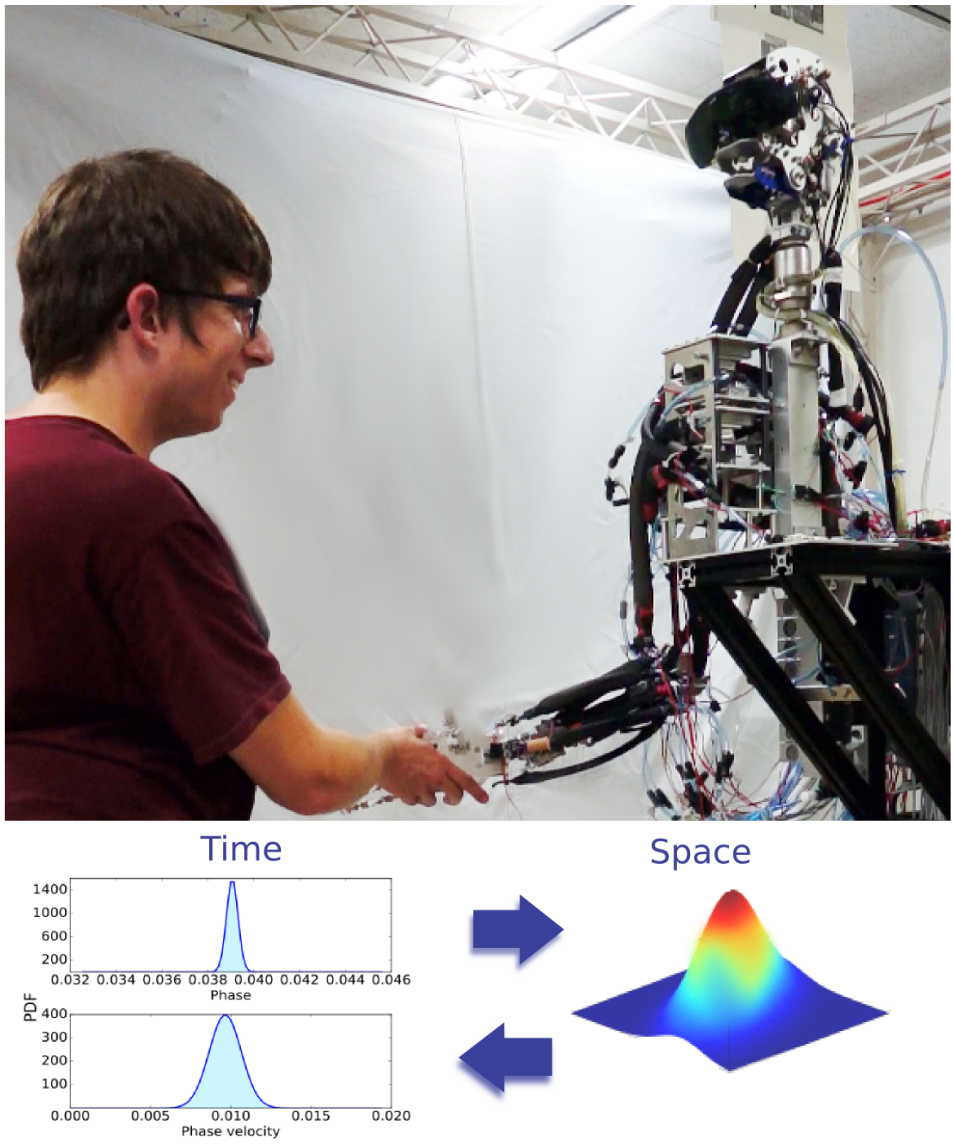


Figure 6: A musculoskeletal robot learning to shake the hand of a human partner. Bayesian Interaction Primitives are used to determine *when* and *how* to interact.

robot used in this work, this approach was not possible since the state of pneumatic actuators is different when the robot is subject to external forces, *i.e.* undergoing kinesthetic teaching, compared to when it is moving autonomously.

However, with BIP, only the correlations between the trajectories of the human

and the robot need to be captured. As such, during training, instead of adapting the robot's trajectory to the human's, we introduce a teaching method in which the human adapts to the robot while it is executing an open-loop policy. More specifically, we created hand-crafted trajectories of the robot's desired action which are executed independently of the human during training, while the human produces an appropriate response. In this way, we do not apply any external forces to the robot and are still able to accurately capture the relationship between the trajectories despite being unable to kinesthetically teach the robot.

Another consideration is that of the execution of the generated response trajectories. The BIP algorithm updates at a given frequency and at each iteration generates a new response trajectory from the current point in the interaction (as estimated by ϕ) to the end of the interaction; this is trivial to calculate with the estimated latent state representation. This is preferable to generating only the next state, since depending on the size of the state dimension it may not be possible to calculate the next state in real-time. However, this can produce discontinuous trajectories where the robot will need to make a large adjustment in position when a new response trajectory is generated; this is unsafe in human-robot interaction. Therefore, an additional alpha-beta filter is employed to smooth the transition between the trajectories generated by BIP.

2.5 Experimental Design

In order to analyze the effectiveness of Bayesian Interaction Primitives, we test the accuracy, robustness, and computational efficiency against standard Interaction Primitives in three scenarios: one involving synthetic benchmark data; one involving

a real-world complex, multimodal, cooperative manipulation task; and the last with novel partners in a handshaking scenario with a musculoskeletal robot. In the synthetic and cooperative manipulation experiments, the BIP algorithm as presented here is compared against a standard IP implementation utilizing DTW for phase estimation. Our IP implementation performs a DTW search every 5 measurement updates so as to improve computational efficiency with the reference trajectory projected from the most recently conditioned weights.

2.5.1 Synthetic Benchmark

The synthetic benchmark experiment utilizes a sample of two-dimensional handwriting data, as shown in Fig. 7. The original trajectory consists of 100 measured states, $y_t = [x, y]^T$, from which new trajectories are created via translation, interpolation, and truncation. Accuracy is measured in terms of the mean absolute error of both dimensions over all time steps.

The effectiveness of BIP is first compared to IP in terms of temporal robustness: 50 training trajectories are generated in which each state $y_t \in y$ is transformed by a linear translation drawn from the Gaussian distribution $\mathcal{N}(0, 5.0)$ with i.i.d. Gaussian noise $\mathcal{N}(0, 0.1)$ applied to each state. Test trajectories are generated with varying lengths by sampling from the trained linear basis model and applying a linear transformation drawn from the same distribution as applied to the training demonstrations. The sampled trajectories vary from 25 to 400 samples (20 trajectories for each domain value), equivalent to a four-fold decrease in temporal speed up to a four-fold increase, assuming a constant sampling frequency. Next we analyze the spatial robustness of BIP by applying different magnitudes of linear translations. The training and

test trajectories are held constant at 100 samples, however, the translation is now sampled from a uniform distribution $\mathcal{U}(-a, a)$ where $a \in [1, 15]$. Lastly, we analyze the robustness of BIP to partially observed trajectories. Training and test trajectories are generated via translations drawn from the Gaussian distribution $\mathcal{N}(0, 5.0)$ with i.i.d. Gaussian noise $\mathcal{N}(0, 0.1)$ with 100 samples per trajectory. However, now the test trajectories are truncated from the end such that only the first $10b$ samples remain and the phase value for the last observed measurement is b, with $b \in [0.1, 1.0]$.

2.5.2 Cooperative Manipulation

The cooperative manipulation experiment consists of two separate human-robot interaction scenarios between a human outfitted with wearable sensors of different modalities and a Universal Robotics UR5 6-DOF robotic arm equipped with a three-finger Robotiq Adaptive Gripper. In both experiments, the trajectories consist of 6 separate sensor modalities for a total of 74 DOF. These include: Kinect skeleton tracking for 13 joints, 6-axis inertial measurement unit (IMU) sensor readings on each forearm, electromyography (EMG) readings from each forearm, pressure sensor readings from each foot, and joint angles for each joint in the UR5 and the gripper. Sensor readings are synchronized online and collected at a frequency of 30 Hz. Training trajectories are obtained via kinesthetic teaching, *i.e.*, the robot is manually manipulated in tandem with the human to compose demonstrations. Testing is performed via leave-one-out cross validation, where the UR5 trajectories are intentionally excluded such that we can measure the error between the generated UR5 trajectory and the expected trajectory. In order to test temporal robustness, test trajectories

are truncated and expanded via non-linear deletion and duplication of measurement states drawn from a uniform distribution.

In the first experiment, the human attempts to reach for an object placed on a table out of reach, as shown in Fig. 7. The robot, moving in response to the human, reaches for the object, grasps it, and performs a hand-over maneuver such that the trajectory ends with the human holding the object. It should be noted that no visual detection is used in this experiment; the location of the object is inferred purely from the human's trajectory. The purpose of this experiment is to demonstrate a robotic response to human ergonomics and safety.

The second experiment involves the human and UR5 cooperatively manipulating an object, as shown in Fig. 7. The human grasps the box from one side and the UR5 from the other and together they cooperatively lift the box such that it is placed on a nearby table. In this scenario, we demonstrate the importance of utilizing multiple sensor modalities. The object being manipulated is a large box which obstructs visual skeleton tracking by the Kinect, rendering it all but useless for trajectory prediction. This is an extremely common scenario in everyday life and it is important for human-robot interaction algorithms to be capable of accommodating it.

2.5.3 Musculoskeletal Handshake

The musculoskeletal robot [49] employed in this work, shown in Fig. 6, contains 10 kinematic degrees of freedom: 7 in the arm linkage and 3 in the shoulder mechanism. These 10 degrees of freedom are actuated with 27 PAMs in an anatomical structure similar to that of humans. Due to the prevalence of the spherical joints required to create a complex biomimetic structure, the robot does not contain conventional joint

angle sensors. Rather, each PAM was equipped with a tension sensor and a pressure sensor to capture the state of the actuators, however, only the pressure sensors were used in this experiment. The PAMs themselves are connected to proportional valves which are controlled via PID controllers operating at 500 Hz with a pressure reference signal. The values reported in this work are measured in mPa as a difference from atmospheric pressure. Human subjects were tracked with 3 degrees of freedom using skeleton tracking running on a Kinect v2 camera. The degrees of freedom correspond to the x-, y-, and z-position of the right hand, with the camera at an angle such that both the x- and y-axes indicate the distance and direction (left/right) of the handshake, while the z-axis indicates the height. Thus, the total number of degrees of freedom in this experiment was 30 and sampling was performed at a rate of 30 Hz.

During training, the robot executed 12 manually-crafted trajectories with no feedback, i.e., in an open-loop fashion, as explained in Sec. 2.4. These trajectories were constructed via linear interpolation from a start pressure value and an end pressure value for each of the 27 PAMs in the robot; for some of the PAMs, the start and end values were equal thus producing no movement for that DOF. The end pressure values were chosen to produce handshake end points over the entire range of the robot in 3-D space. The human participants who assisted in training were instructed to shake the hand of the robot once for each executed trajectory over a time window of 10 seconds. Three participants contributed training demonstrations with 3 repetitions of each trajectory, resulting in a total of 108 total training demonstrations. During testing, the three participants from training as well as five additional participants were asked to shake the hand of the robot in eight different scenarios. In four of the scenarios, the robot once again executed a manually-crafted trajectory as in the training demonstrations, however, this time with different end points. In the remaining

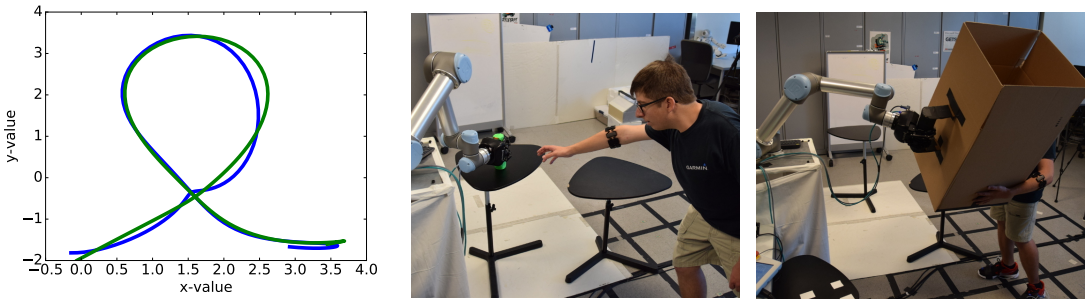


Figure 7: Experimental setup for 1 (left), 2 (middle), and 3 (right).

four scenarios, the BIP algorithm was employed with the robot generating response trajectories based on the human's hand movement. The participants were asked to move their hand to an arbitrary location and shake the hand of the robot while moving their hand at a requested speed: fast, normal (as in the same speed as used in the demonstrations), slow, and a special case of no movement at all. The speed definition was purposely vague and left up to the determination of each participant so as to adequately test the temporal robustness of the BIP algorithm. Each test participant executed each scenario 2 times, for a total of 128 test trajectories. A response trajectory was generated by BIP at a frequency of 3 Hz (for computational reasons) using 15 basis degrees for a total state dimension of 450. This trajectory was executed by the robot at 10 Hz and consists solely of pressure values for each of the robot's degrees of freedom, no inverse kinematics or dynamics were used at any point as such models were unavailable.

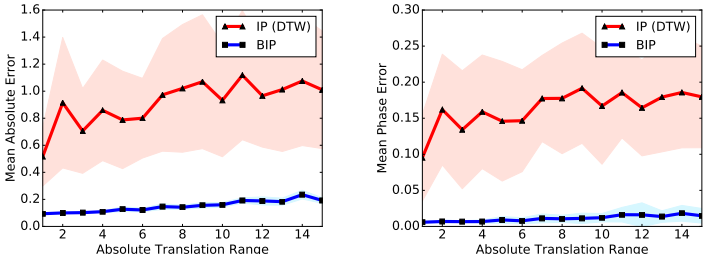


Figure 8: The results for the spatial robustness test for Experiment 1. The x-axis indicates the value of a for the uniform distribution from which translations were drawn.

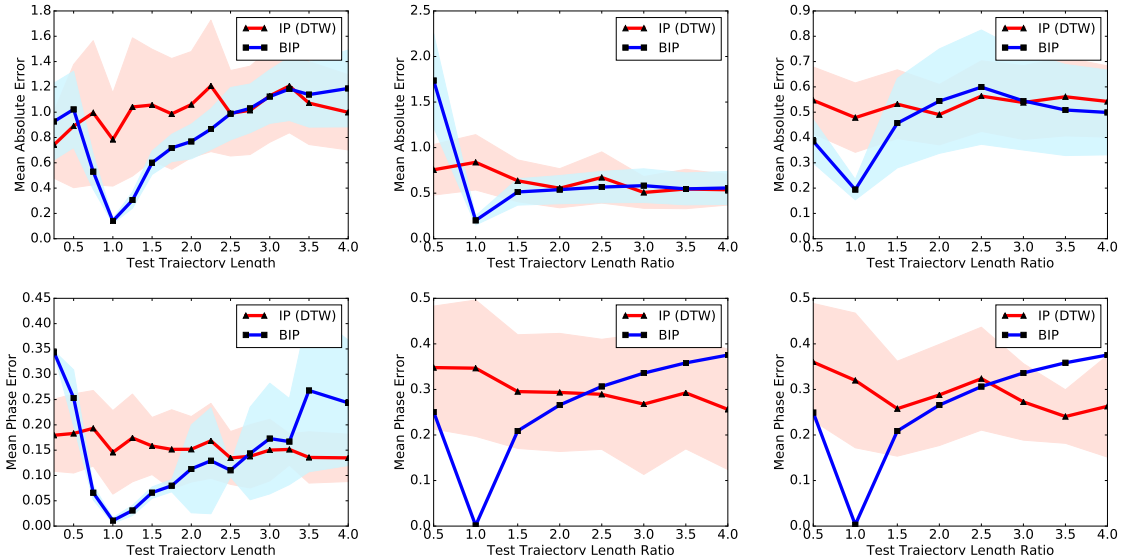


Figure 9: The mean absolute error (top) and mean phase error (bottom) for the temporal robustness test for Experiment 1 (left), Experiment 2 (middle), and Experiment 3 (right). A trajectory length of 1.0 is normal speed, 2.0 twice as fast, etc. The lines indicate the mean result and the shaded regions indicate the standard deviation.

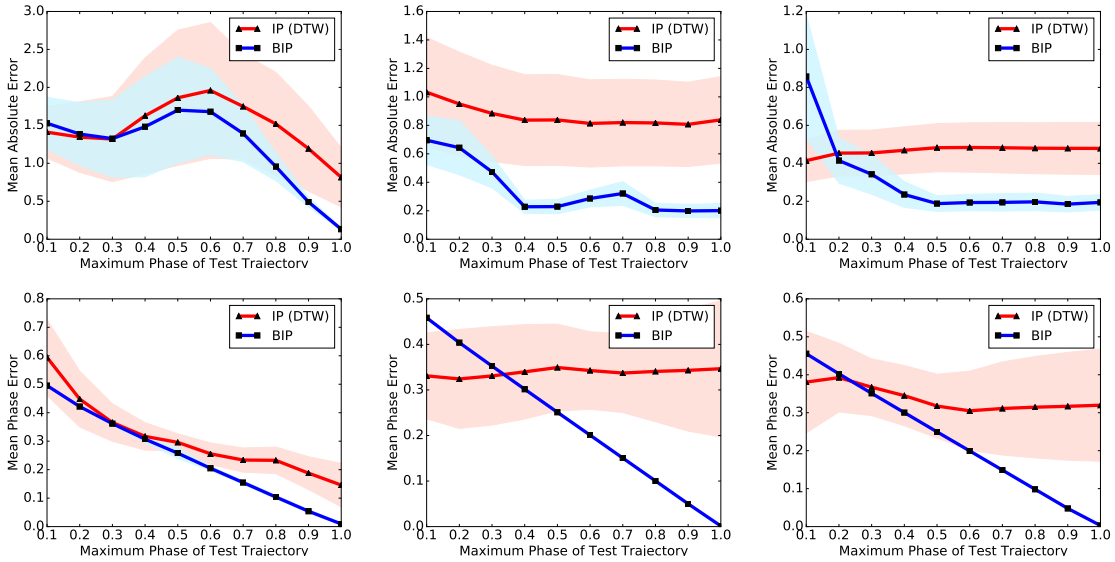


Figure 10: The mean absolute error (top) and mean phase error (bottom) for the partial visibility robustness test for Experiment 1 (left), Experiment 2 (middle), and Experiment 3 (right). The length of the test trajectories are given as a ratio of the observed trajectory to the full trajectory along the x-axis.

2.6 Results

2.6.1 Synthetic Benchmark

The results from the synthetic benchmark experiment indicate clear advantages in the BIP framework over standard IP. We first note that while BIP performed slightly worse than IP in terms of temporal robustness (Fig. 9) for very slow and very fast motions ($< 50\%$ or $> 300\%$ movement speed), the generated trajectories in between these extremes are more accurate as well as more consistent, *i.e.*, a smaller standard deviation. Particularly noteworthy is the excellent inference accuracy when

the test trajectory is close to the same temporal speed as the training demonstrations; this is a common scenario so accuracy is vital. At the same time, BIP significantly outperformed IP in the spatial robustness test which is shown in Fig. 8, indicating that BIP is quite robust with regard to spatial uncertainty. In terms of the partial visibility robustness results shown in Fig. 10, the results between the two methods are closer; however, BIP again outperforms IP in both mean absolute error and mean phase error at nearly every length trajectory, only lagging 20% or less of the test trajectory is observed.

2.6.2 Cooperative Manipulation

These same trends only become more evident for the more complicated cooperative manipulation scenarios. It is clear from Fig. 9 and Fig. 10 that the standard IP algorithm struggles to estimate the correct phase; even with 100% of the trajectory visible there is still a mean phase error of 0.34 and 0.32 in Experiments 2 and 3 respectively. As a result, the inference accuracy does not display any noticeable improvement as more of the observed trajectory is made available. BIP, by contrast, exhibits a constant increase in phase estimation accuracy which leads to a significant improvement in the predicted trajectory error. This is the same type of behavior that is seen in the partial visibility test of Experiment 1. We also observe that the choice of process and measurement noise values becomes extremely important with BIP; a poor choice of noise can cause severe effects in the accuracy of the Bayesian inference.

What makes these results all the more striking is the fact that BIP enjoys a significant computational advantage over IP in every experiment, as shown in Fig. 11. While computational times are difficult to estimate and highly dependent on algorithm

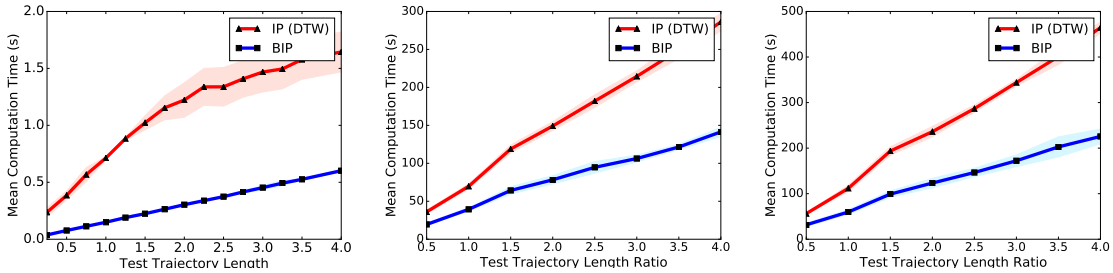


Figure 11: The computational time required to process trajectories of varying lengths for Experiment 1 (left), Experiment 2 (middle), and Experiment 3 (right).

implementation, there is the simple fact that the standard IP framework must spend upwards of $O(N^2)$ computational cycles performing DTW at each measurement update step. This is an overhead that BIP simply does not have, and as a result achieves upwards of a 66% reduction in computation time for fully observed trajectories regardless of the dimensionality of the state vector.

2.6.3 Musculosekeletal Handshake

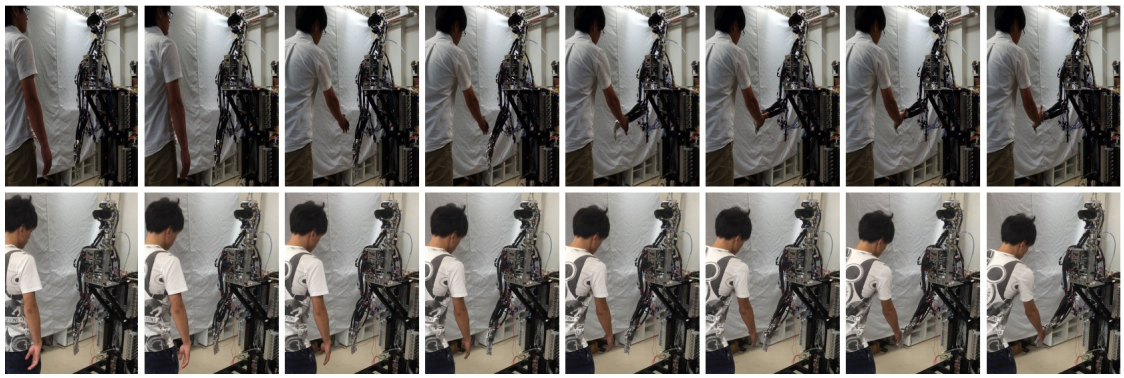


Figure 12: The motion of the human and robot over time when shaking hands at arbitrary end points with fast movement (top) and slow movement (bottom). Each sequence begins at the start of the interaction and each image is sampled at the same rate.

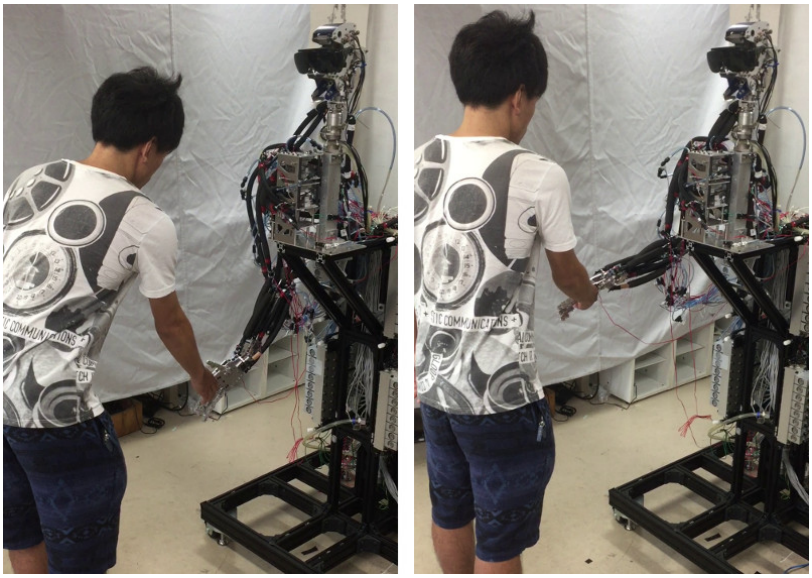


Figure 13: Different test interactions emphasizing BIP's spatial generalization. The left image shows a handshake low and closer to the robot, while the right image shows a handshake high and closer to the human.

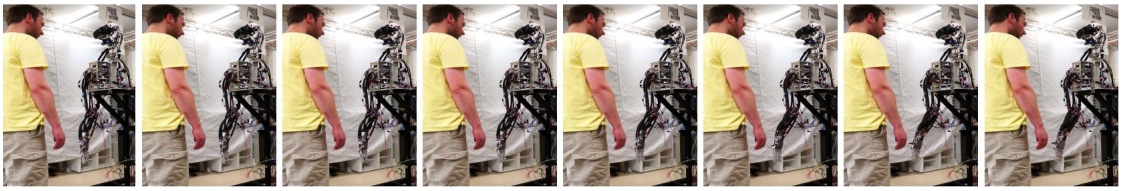


Figure 14: The motion of the human and robot over time when the human partner does not move their arm. The robot does not engage in a hand shake.

2.6.3.1 Qualitative Analysis

A sequence of images showing two different test interactions over time for different participants is shown in Fig. 12. Qualitatively, these sequences simultaneously demonstrate three things: a) the robot learned to reproduce surprisingly human-like handshake motions despite the lack of access to any sort of analytical model, b) BIP is capable of generalizing this motion across both space *and* time, and c) the algorithm is able to generalize across different human participants. The temporal differences

	Mean (All)	Var (All)	Mean (T)	Var (T)	Mean (NT)	Var (NT)
BIP (Fast)	0.2640	0.0013	0.2479	0.0001	0.2752	0.0019
BIP (Normal)	0.3094	0.0064	0.3148	0.0089	0.3062	0.0049
BIP (Slow)	0.3824	0.0138	0.4049	0.0164	0.3689	0.0117
Static (1)	0.3272	0.0055	0.2950	0.0012	0.3465	0.0070
Static (2)	0.3009	0.0109	0.2450	0.0018	0.3345	0.0134
Static (3)	0.3387	0.0057	0.3430	0.0060	0.3365	0.0056
Static (4)	0.2914	0.0017	0.2642	0.0030	0.3078	0.0002

Table 1: The mean Time-to-Completion (as a ratio of trajectory length, lower is better) and variance for all test participants (All), test participants who trained the model (T), and test participants who did not train the model (NT). BIP refers to our approach while static refers to an open-loop trajectory. Green cells indicate the scenarios with the smallest mean values while gray cells indicate scenarios that are not significantly different, as calculated with the Mann-Whitney U test with a p-value < 0.05.

in particular can be clearly seen in Fig. 12; in the fast (top) sequence, the human and robot have reached each other by the 5th frame, whereas in the slow (bottom) sequence, this does not occur until the 8th (last) frame. While spatial differences are visible in these sequences, they can be more clearly seen in Fig. 13. In the first image, the human chooses an end point for the handshake that is low and near the robot, while in the second image the same human participant chooses an end point that is higher and closer to the human. Furthermore, an image sequence for an extreme edge case is shown in Fig. 14 in which the human participant does not move their hand at all, thus, never beginning the interaction. In response, the robot similarly does not proceed with the interaction and produces only minute movements.

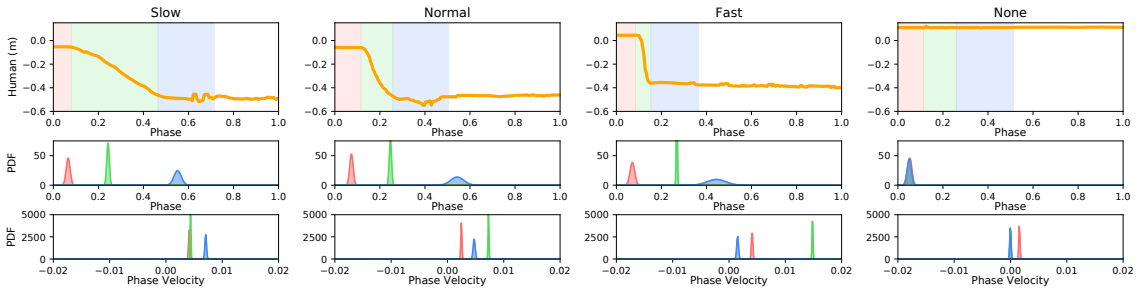


Figure 15: Top: the trajectory of the human's hand along the x-axis, which approximately corresponds to the distance to the robot, during different test interactions. The trajectory region shaded in red indicates the beginning portion of the interaction in which the participant has yet to move. The green region indicates the period in which the participant actively moves to shake the robot's hand. The blue region indicates the period after which the handshake is completed. Middle: the probability density corresponding to the estimated phase at the end of each aforementioned period (red, green, blue). Bottom: the probability density corresponding to the estimated phase velocity for each region.

2.6.3.2 Temporal Analysis

An analysis of the inferred phase at different points in the interaction for each movement speed is shown in Fig. 15. The trajectories shown in the top plots represent the movement of the human's hand and are broken up into three periods: the period before the human has moved (red), the period during which the human is moving (green), and the period after the human has stopped moving (blue). The phase and phase velocity predictions at the end of each period are shown such that the portion of the trajectory falling inside the corresponding shaded region has been observed. These plots yield an interesting insight: the estimated phase for each movement speed is approximately the same as in the normal speed case. That is, no matter how quickly the interaction proceeds in real time, the phase of the interaction is the same immediately before movement begins and immediately after. Instead, it is the phase velocity that differs, particularly in the region during which movement occurs (the

green region). The slow speed interaction has the smallest phase velocity at the end of the movement period while the fast speed interaction has the greatest velocity. The practical implication of this is that the amount of absolute time required to reach each point in the interaction differs due to different phase velocities, even if the phase value at each point is the same. Since the state (Eq. 2.4) evolves according to a linear velocity model, a slower phase velocity requires more state updates to reach the same point in phase.

Figure 16 depicts the distribution of estimated phases and phase velocities for all interactions, and shows that this trend holds for all cases. While the phase estimates exhibit little difference between the movement speed cases, the velocity estimates show a positive relationship between movement speed and phase velocity. Analytically, this is due to our choice of initial uncertainty in the phase and phase velocity in Eq. 2.14; the uncertainty in phase is set much lower than that of phase velocity because we are confident that the initial phase is 0. Phase and phase velocity have a non-zero covariance (an uncertainty in velocity affects the uncertainty in phase), but due to the different magnitudes in the initial uncertainty the phase velocity experiences more significant updates during inference. The other observation we can make from Fig. 16 is that the larger phase velocity estimates exhibit greater variance. This is due to the increased uncertainty corresponding to the larger estimates, which can also be seen in the green velocity plots of Fig. 15. As a consequence, this impacts the uncertainty in the phase estimate for the blue region, as inference has already taken into account the velocities (and uncertainties) of the green region. This is why the uncertainty for the blue region phase estimate increases with interaction speed in Fig. 15.

Lastly, we note that BIP handles the special case of no movement speed by reducing the phase velocity to 0, thus stopping the temporal progression of the interaction. This

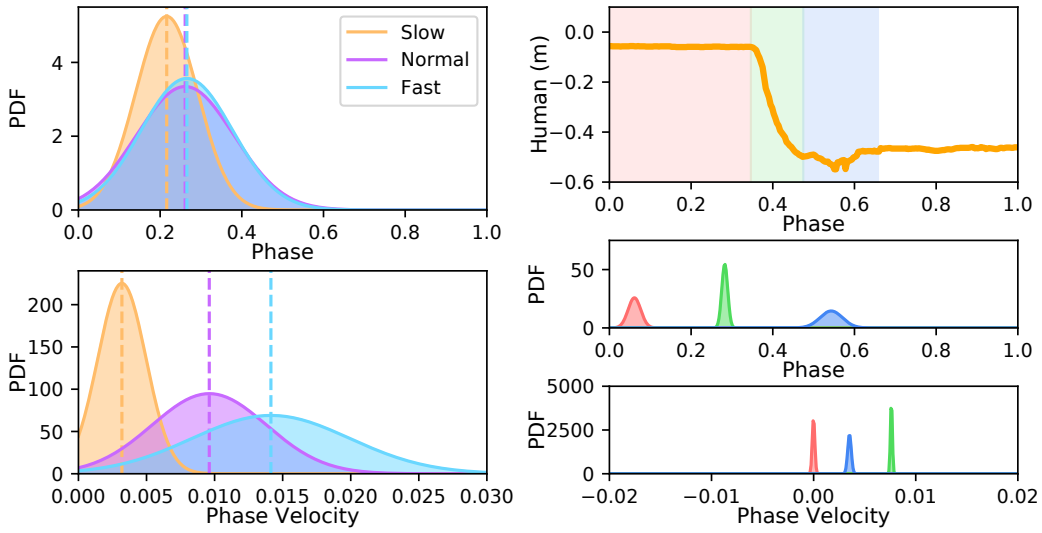


Figure 16: Left: the distribution of estimated phases (top) and phase velocities (bottom) after the participant moves to shake the robot's hand, for all tested slow, normal, and fast interactions. This corresponds to the green region shown in Fig. 15. Right: the same type of plot as in Fig. 15 for the same normal speed interaction with the addition of an artificial pause at the beginning.

is visualized in the last column of Fig. 15. However, this elimination of the temporal velocity is flexible and can be recovered from, as can be seen with the introduction of an artificial pause at the beginning of the interaction as shown in the right column of Fig. 16. Although the phase velocity has been reduced to 0 at the end of the extended "no movement" (red) period, it quickly progresses when the human begins moving and yields phase estimates consistent with what we expect. This indicates that not only is BIP robust to the movement speed of the interaction, but it is also robust to variations in *when* the interaction begins.

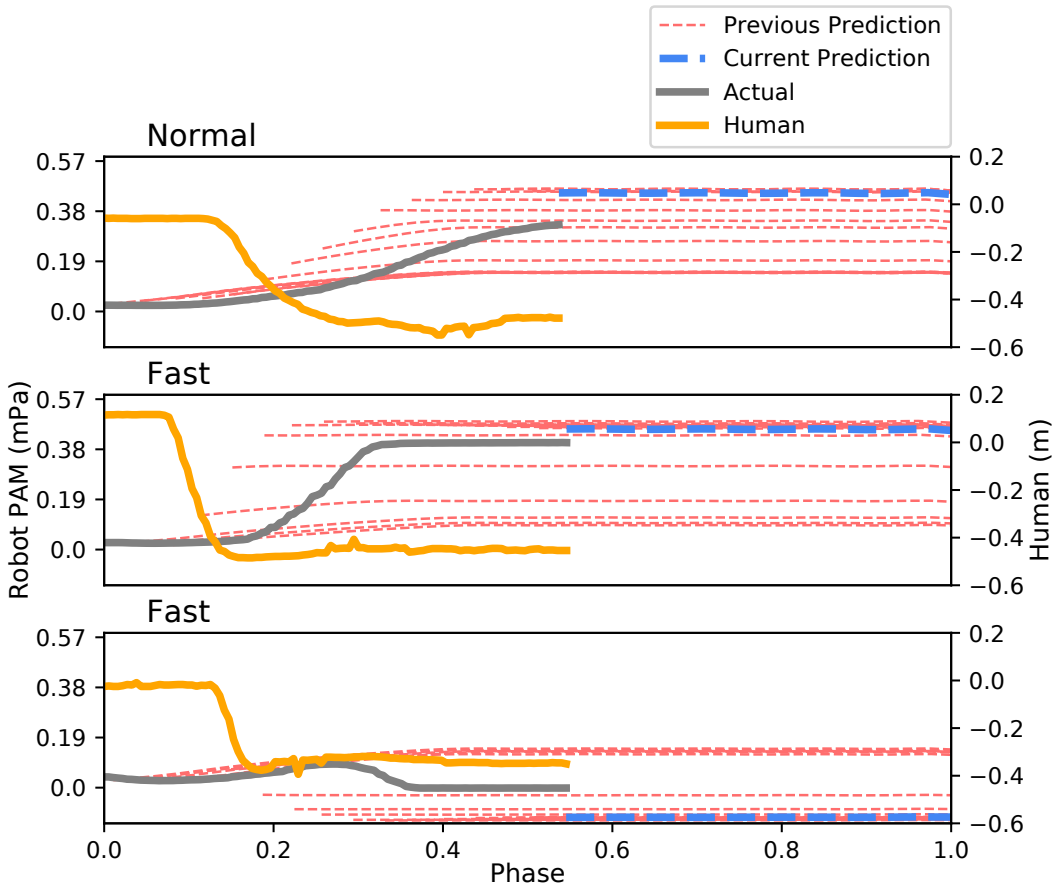


Figure 17: The predicted pressure trajectory for one of the robot's PAMs during normal (top) and fast (middle and bottom) test interactions. In all cases, the current prediction (blue) is from approximately 50% through the interaction (the 17th inference step), with the predicted trajectories (red) from the previous 15 inference steps shown for reference. These trajectories are given to the robot's PID controller with the resulting actual pressure values shown in gray. The predicted and actual values do not directly coincide due to physical restrictions inherent to the pneumatic system that the learned model is unaware of. The corresponding human observations along the x-axis are shown in orange.

2.6.3.3 Spatial Analysis

While we have demonstrated that the BIP framework is robust to temporal variations, our results also show that is also robust to spatial variations. As Fig. 17 shows, the predicted robot trajectory is dependent both on the value of the human observations (spatial variation) as well as the rate of change (temporal variation). The top and middle figures demonstrate spatial generalization through the pattern of increasing pressure values associated with the previous predictions (red dashed lines); as more human observations become available, BIP continues to refine the prediction. However, because these two interactions have similar end points, the effect of movement speed becomes evident on the predictions. While both predictions ultimately arrive at approximately the same pressure value for this particular PAM, BIP predicts these values for the fast interaction much earlier in response to the high rate of change of the human movement. In contrast, the bottom figure is associated with a fast interaction at a different end point and the predicted trajectory is significantly different than the others, demonstrating the ability to spatially generalize to different end points. These figures also highlight limitations of the BIP framework: namely, lack of knowledge of control lag and physical constraints. The control lag can be observed via the phase shift between the predicted pressure values and when the actual robot actually reaches those values (gray line). Similarly, BIP yields inferred pressure values that the system is incapable of reaching due to physical and/or mechanical limitations. This is visible in the offset between the current prediction (blue dashed line) and the actual robot.

As a result of spatial generalization, empirically we hypothesize that the robot and human positions converge to the same physical location faster when both participants

are active in the interaction (BIP scenarios), as opposed to only the human (static scenarios). However, for technical reasons, we lack the position of the robot end-effector in 3D space and, therefore, measure the length of an interaction by how long it takes the human and robot degrees of freedom to converge to steady-state values, which we are referring to as Time-to-Completion. To this end, the variance of each DoF was calculated over a sliding window of 2 seconds. If the variance of all DoFs within the window falls below a threshold – 0.001 m for the human and 0.001 mPa for the robot – then the start time of the sliding window is taken to be the Time-to-Completion. These thresholds are chosen such that all scenarios yield a completion time.

The results, shown in Table 1, support our hypothesis. The mean Time-to-Completion values for all test participants are smaller when the BIP algorithm is employed compared to a static handshake trajectory. The one exception is in the case of test participants who also trained the model (the T subset), in which case the BIP interactions were not statistically different than the second static trajectory. This is an interesting observation, since it suggests that participants who had already participated in training the model were better able to predict where the robot would go when compared to new participants, despite new handshake end points. Thus, their interactions resulted in lower Time-to-Completion times than the participants who hadn't trained the model (the NT subset). However, despite their familiarity with the experiment, BIP was still able to produce similar completion values; in the case of new participants who hadn't trained the model, BIP resulted in more efficient interactions with smaller completion values.

The results in Table 1 also indicate that the variance in Time-to-Completion is smaller for the static trajectories when compared to the BIP ones, particularly slow speed interactions. This is expected considering that the test participants are free to

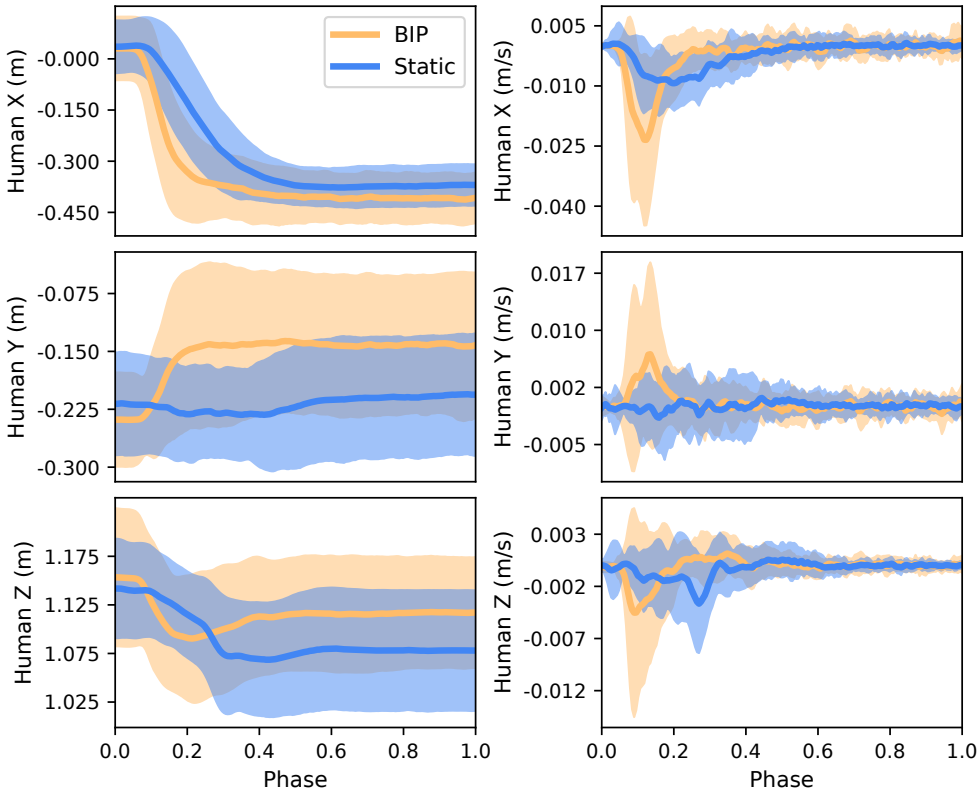


Figure 18: Left: the position distribution (mean and std dev) for all static and BIP scenarios for the human's hand along the x-axis (top), y-axis (middle), and z-axis (bottom). Right: the corresponding velocity distributions.

choose their own handshake end point when testing with BIP as opposed to the static trajectories which result in an identical handshake end point for every participant. This is especially clear when looking at the position and velocity distributions of the human as shown in Fig. 18. Likewise, the human begins moving earlier in BIP scenarios than static ones, as they are dictating the end points rather than the robot – visualized in the earlier peak of the velocity distributions. The shape of the velocity distributions are also quite illuminating, as the BIP distributions closely resemble the typical bell-shaped velocity profiles of point-to-point movements observed in humans [43].

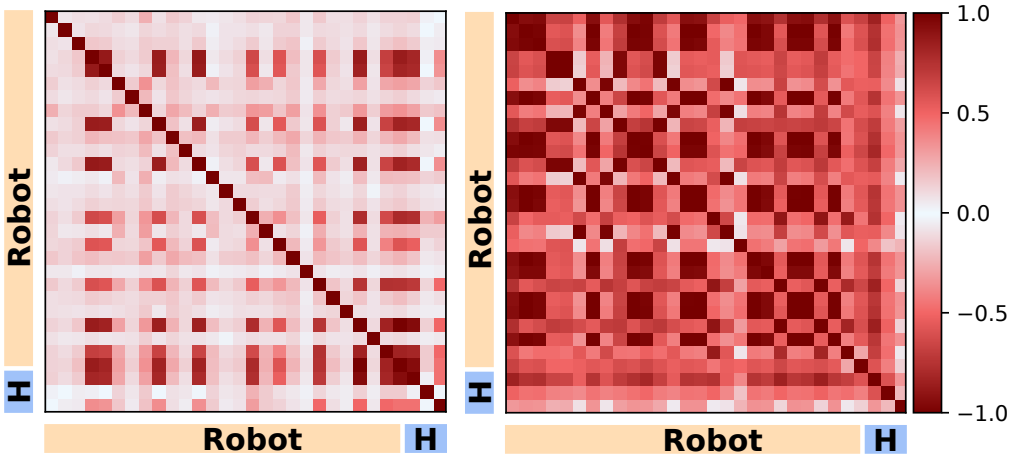


Figure 19: Mean Pearson correlation coefficients for static trajectories (left) and BIP (right). The color of the squares indicates the magnitude of the correlation. The first 27 rows and columns represent the coefficients for the robot degrees of freedom. The last 3 rows and columns represent the human. The mean correlations are calculated for all participants in all static scenarios and all BIP scenarios.

The last result highlighting spatial generalization is that the Pearson correlation coefficients differ between the human and robot trajectories for BIP and static handshakes, as shown in Fig. 19. Unsurprisingly, the only correlations above a magnitude of 0.5 in the static scenarios are between the human degrees of freedom and the actuated robot degrees; the un-actuated degrees exhibit no significant correlations. By contrast, all degrees of freedom which were actuated in the training process yielded significant correlations when BIP is used, indicating that BIP is effectively exploiting the model learned from demonstrations. This is also supported by the histogram that is generated from a non-actuated degree of freedom (in the static scenarios) in Fig. 20. The histogram for the static scenarios is approximately Gaussian with a mean of 0, in other words, Gaussian noise. However, the histogram for the BIP scenarios is bi-modal with peaks at -1 and 1, demonstrating strong positive and negative correlations.

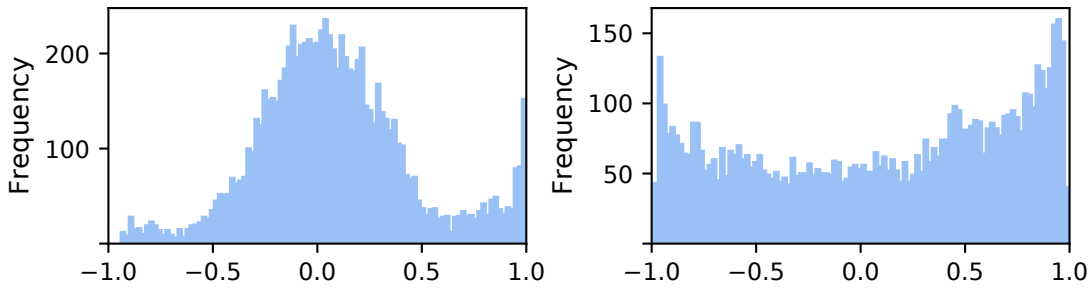


Figure 20: Histogram for the Pearson correlation coefficient generated from a sliding window over static trajectories (left) and BIP (right). This correlation is for an un-actuated PAM and hand position along the y-axis.

2.7 Related Work

2.7.1 Imitation Learning

Imparting new skills to a robot is a challenging task given the complexity of modern robots, the environments they operate in, and limited observability. Brute-force methods such as hand-crafting new skills are inefficient, require deep domain knowledge, and typically do not generalize well to new scenarios, given the sheer size of the state space [10]. Inspired by the learning processes employed by humans and animals [77], *imitation learning* – also known as *learning from demonstration* [7] and *programming by demonstration* [10, 96] – has become a prominent technique for teaching robots new actions. In imitation learning, a knowledgeable teacher provides a successful (but not necessarily optimal) demonstration of the action that is to be learned. This process serves two purposes: it constrains the state space to the area of a successful demonstration; and it precludes the teacher from having to manually define a mathematical model of the action, instead leveraging statistical learning techniques to identify what an action is comprised of as well as when it applies. Demonstrations

can be provided either directly by the human – in which case the correspondence problem must be addressed [77] and motions must be segmented [63] – or through direct manipulation of the robot itself via kinesthetic teaching [15, 2, 5].

Dynamic Movement Primitives (DMPs) [97] are a specific approach to imitation learning in which time-varying sensor observations are used to create a parameterized dynamical system. The nonlinear forcing function of the system is learned directly from the observations of each degree of freedom (DoF) – typically through regression – and defines the trajectory of the system, while the start and goal states define the attractor and can be specified so as to allow for generalization to new situations [52]. Recent work on DMPs has extended the approach to domains that require high-dimensional control signals [3], singularity-free trajectory representations of orientations [107], or non-Euclidean distance functions [37].

2.7.2 Human-Robot Interaction

An extension to collaborative interactions between two agents, i.e., a human and a robot, led to the introduction of Interaction Primitives (IP) [4]. Fundamentally, an IP creates a dynamical system for the human and a dynamical system for the robot which individually model the motion of each during an interaction. The relationship between these systems is captured in a joint probability distribution over the regressed parameters of the forcing functions of both systems, thus capturing the correlation between the trajectories. This distribution can then be used in a Bayesian inference sense to infer what the robot's DMP parameters should be given an observation of the human. The approach allows for the probabilistic imitation of observed human-human interactions by a learning robot. The human-human demonstrations provide training

information about the typical dynamics and spatio-temporal relationships involved in a specific human interaction. For example, a small set of demonstrations of a 'high-five' interaction allow the robot to infer that the two hands of the interaction partners need to touch each other [4].

An alternative to the dynamical system approach of the DMP is that of the Probabilistic Movement Primitive (ProMP) [82]. With ProMPs, the dynamical system is replaced with a full probability distribution over parameterized approximations of the system degrees of freedom, which allows for Bayesian inference of desired degrees. As with DMPs, this approach has been extended to HRI scenarios with Interaction Primitives based on ProMPs [71]. While the convergence and stability guarantees of DMP-based systems are lost, the fully probabilistic nature allows for several important properties, such as the combination of and transition between multiple ProMPs.

Interaction Primitives have since been extended in various ways in order to improve interaction with human partners [26, 41]. Anticipative capabilities have been introduced such that the robot's response is based on the most likely sequence of future actions, reducing the delay in the action-reaction cycle [70]. Environmental conditions have also been considered, such that a relationship between external factors and primitives are established such that the robot's response not only considers the human partner's behavior, but also environmental variables [33].

Many approaches [110, 46, 71, 4] resolve the challenge of timing and phase estimation using time alignment algorithms such as DTW. This is undesirable for several reasons: time alignment is performed independently and does not leverage any information learned from the demonstrations, DTW is inherently non-probabilistic [9], and DTW performs poorly in high-dimensional spaces [99]. While other algorithms such as the Monte Carlo-based particle filter can yield probabilistic alignments, they

also scale exponentially with the state dimension [90] and fail to address the issue that time alignment is still independent of the learning process.

Fundamentally, IPs can be viewed as a representation of Markov models with continuous state spaces and Gaussian assumptions [93]. Although standard hidden Markov models with discrete state spaces have long been used in imitation learning [88, 51, 66, 63, 94], observations must be mapped to discretized symbols before the HMMs can be used for further processing. This is complicated by the fact that inference algorithms for HMMs scale poorly in high dimensional spaces [42]. Even Gaussian mixture model-based methods [16] rely on independently time aligning the observations in a latent space. Time-delay neural networks have shown promising results in imitation learning [78] without requiring explicit time alignment, but the motion is no longer modeled as a primitive and lacks the well-understood mathematical model. In particular, it is unclear how a learned primitive will behave or generalize to a new situation. In addition, training neural network models typically requires an extensive number of training samples and cannot be effectively performed using a small number of demonstrations.

2.7.3 Musculoskeletal Robots

Robots with pneumatic artificial muscles (PAMs) and compliant limbs have been shown to be desirable for human-robot interaction scenarios [80, 106]. When configured in an anthropomorphic musculoskeletal structure, such robots provide an intriguing platform for human-robot interaction (HRI) [102] due to their potential to generate human-like motions while offering a degree of safety as a result of their compliance when confronted with an external force – such as contact with a human. Recent

work [49] has shown the value of utilizing McKibben actuators in the design of these robots, due to their inherent compliance and inexpensive material cost. However, while analytical kinematics models are in theory possible [29, 53], they are not always practical due to the effects of friction and the deterioration of mechanical elements, which are difficult to account for (although some gains have been made in this area [48]). Subsequently, this motivates our use of a data-driven approach as proposed in this work.

2.8 Conclusions

In this chapter we have introduced a fully Bayesian generalization of Interaction Primitives in which phase estimation and trajectory conditioning are coupled together in a joint model of both time and space. This generalization follows from a connection we have established between SLAM and human-robot interaction; this is advantageous in that the vast body of knowledge regarding localization and mapping may now be applicable to interaction. We have shown that this enables greater phase estimation accuracy, and consequently inference accuracy, along with reduced computational complexity in human-robot interaction scenarios. Most importantly, this allows us to tackle far more complex scenarios than is possible with normal IP. Despite interacting with a robot in a multimodal, high-dimensional environment, Bayesian Interaction Primitives are capable of accurately performing trajectory inference, thus opening the door to further work in such complex scenarios. One such application has been explored in some detail here, namely that of interaction with a musculoskeletal humanoid robot. Despite the challenges inherent to pneumatic artificial muscles – nonlinear dynamics and lack of kinesthetic teaching – BIPs were able to learn strong

spatiotemporal relationships between the movement trajectories of the robot and human test participants. Furthermore, these relationships were generalizable to new human partners who did not take part in training the model, as well as significant temporal variations including an edge case in which the human does not interact at all.

At the same time, this work has also demonstrated some of the limitations in BIP. Most significantly among them, is that we do not directly model the dynamics of neither the robot nor the environment. As a result, we are unable to model mechanical constraints which are present in many scenarios, including the musculoskeletal experiments detailed here. Other factors which pose challenges in real-time, physical interactions such as control lag also remain unaddressed, although a proposed solution is explored in Chapter 3

Chapter 3

APPROXIMATE INFERENCE FOR HUMAN-ROBOT INTERACTION

Human-robot interaction (HRI) requires constant monitoring of human behavior in conjunction with proactive generation of appropriate robot responses. This decision-making process must often contend with high levels of uncertainty due to partial observability, noisy sensor measurements, visual occlusions, ambiguous human intentions, and a number of other factors. The inclusion of sensor measurements from a variety of separate modalities, e.g., cameras, inertial measurement units, and force sensors, may provide complementary pieces of information regarding the actions and intentions of a human partner, while also increasing the robustness and safety of the interaction. Even in situations in which a complete sensor modality becomes temporarily unavailable, i.e., due to a hardware failure, other available modalities may ensure graceful degradation of the system behavior. Hence, it is critical to support decision-making in HRI with inference and control methods that can deal with a variable number of data sources, each of which may have distinctive numerical and statistical characteristics and limitations.

In this chapter, we investigate how multimodal models of human-robot interaction can be efficiently learned from demonstrations and, later, used to perform reasoning, inference, and control from a collection of data sources. Fig. 21 depicts a motivating example – a robot arm catching a ball. In this example, the position of the ball can be continuously tracked using motion capture markers, but is occluded from view while in the human's hand. Yet, even before the ball is released from the hand, the robot may already intuit the moment of release and travel direction by reading pressure

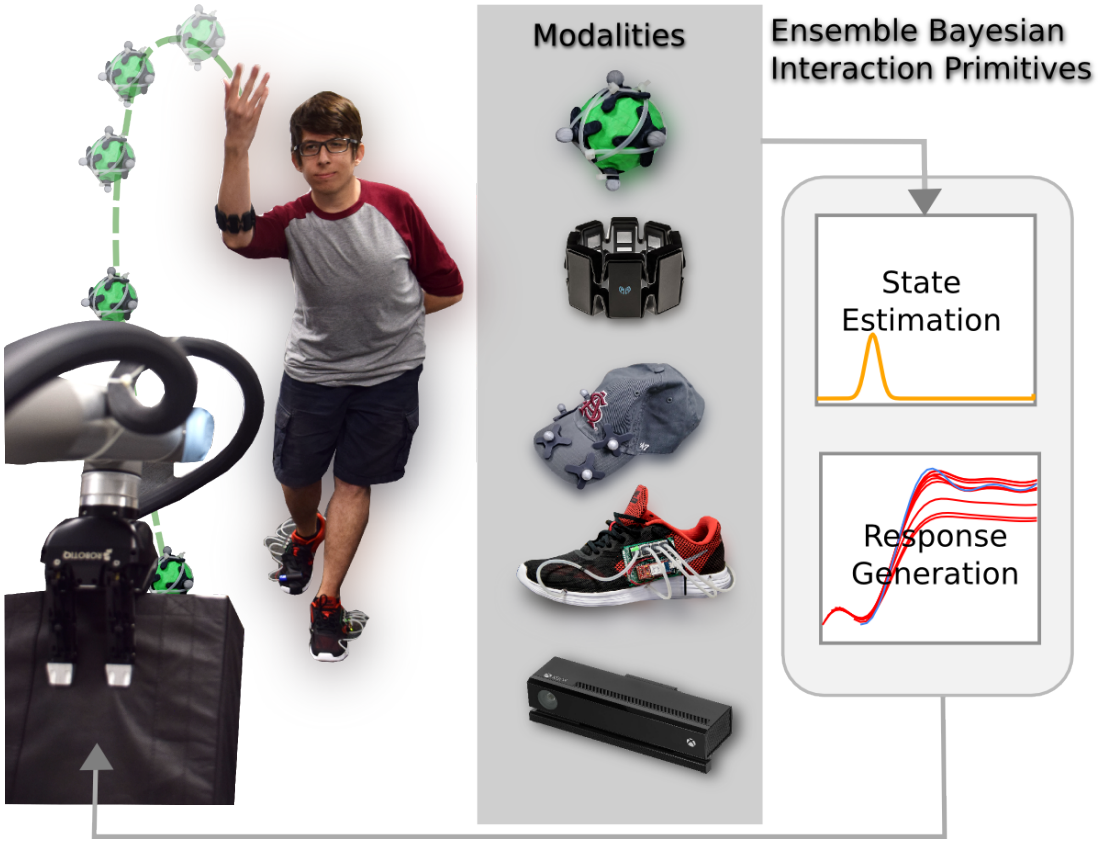


Figure 21: A robot learning to catch a thrown ball by combining information from different modalities.

information from a smart shoe, from inertial measurements on the throwing arm, or from human pose data acquired via skeletal tracking. Integrating these pieces of information together, we would expect the robot to generate earlier and better predictions of the ball, as well as better estimates of necessary control signals to intercept it.

Few probabilistic inference methods for HRI have examined reasoning across multiple modalities as in the above example, with many instead opting to construct models relating only two modalities, e.g., a single observed modality to a single controlled modality. In the case of Bayesian Interaction Primitives (BIP) introduced in Chapter 2, demonstrations of two (human) agents interacting with each other are used to form a

joint probability distribution among all degrees of freedom (DoF) and all modalities. During inference, this distribution is used as the prior for Bayesian filtering, which is then refined through sensor observations of the observed modalities and subsequently used to infer the controlled DoFs. However, when multiple sensing modalities are employed, several challenges quickly arise. First, in order to maintain computational tractability of the filtering process, limiting assumptions are made both about the form of the joint probability distribution, i.e., unimodal and Gaussian, as well as the linearity of the system as a whole. As the number of sensing modalities increases, each with their own unique statistical characteristics, these assumptions begin to negatively impact inference accuracy. Second, expanding the sensing modalities translates to an increased number of degrees of freedom which magnifies the computational burden and jeopardizes real-time performance – a vital property in HRI contexts.

In this chapter, we propose ensemble Bayesian Interaction Primitives (eBIP) for human-robot interaction scenarios, an alternative formulation of Bayesian Interaction Primitives that is particularly well-suited for inference and control in the presence of many input modalities. This approach is an ensemble-based approach to Bayesian inference for HRI, which combines the advantages of parametric and non-parametric methods and requires neither an explicit covariance matrix nor a measurement model. As a result, this allows for fast and efficient inference in high-dimensional nonlinear interactive systems, while avoiding typical inaccuracies due to either linearization errors, the parametric family of the prior, or the underlying state dimensionality. The non-parametric nature of this prior avoids computational overheads and inaccuracies as found, for example, when fitting a mixture model.

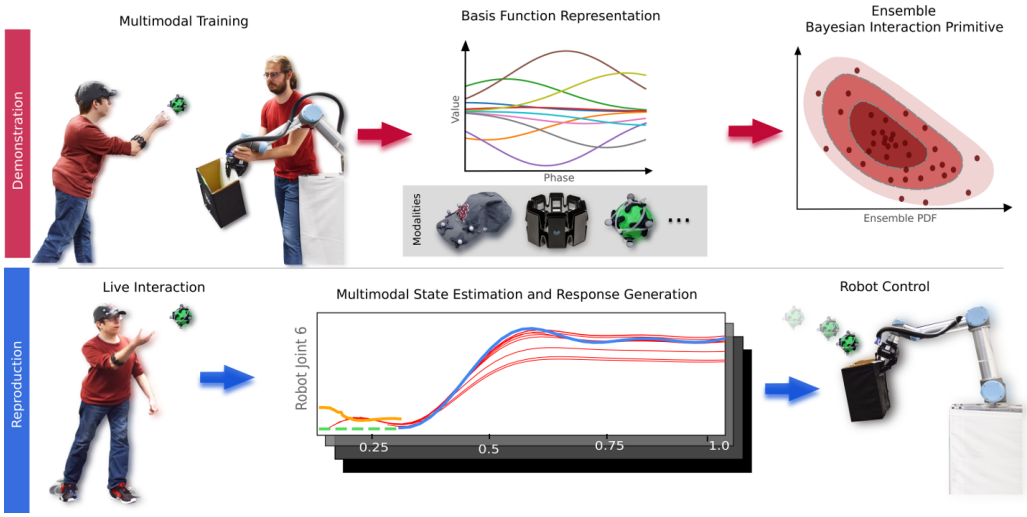


Figure 22: An overview of eBIP. Top: training demonstrations (left) are decomposed into a latent space (middle) and transformed into an ensemble of samples (right). Bottom: observations are collected during a live interaction (left) which is used to perform filtering with the learned ensemble (middle) and produce a response trajectory (right).

3.1 Ensemble Bayesian Interaction Primitives

While Bayesian Interaction Primitives is an analytical method to compute the exact solution to the simultaneous temporal and spatial estimation problem, it suffers from several drawbacks in practice: 1) the prior distribution is assumed to be Gaussian, which is unlikely to be the case; 2) the first-order Taylor approximation introduces (potentially significant) linearization errors; and 3) the integration of the covariance matrix is computationally prohibitive for large state dimensions. Therefore, in contrast to the exact solution yielded by Bayesian Interaction Primitives, we present a Monte Carlo-based method known as ensemble Bayesian Interaction Primitives (eBIP). Originally motivated as a solution to multimodal applications which amplify the above problems, this method also yields improvements in inference accuracy and computational performance in the general case and so we present it here.

Non-Gaussian Uncertainties: In general, the extended Kalman filter employed for recursive filtering in BIP relies on the assumption that uncertainty in the state prediction is approximately Gaussian. When this is not the case, the estimated state can diverge rapidly from the true state. One potential source of non-normality in the uncertainty is the nonlinear state transition or observation function in the dynamical system. The original formulation of BIP addresses this challenge by linearizing these functions about the estimated state via first-order Taylor approximation, which is performed in Eq. 2.6 for the nonlinear observation function $h(\cdot)$. Unfortunately, this produces linearization errors resulting from the loss of information related to higher-order moments. In strongly nonlinear systems this can result in poor state estimates and in the worst case cause divergence from the true state [73].

We follow an ensemble-based filtering methodology [40] which avoids the Taylor series approximation and hence the associated linearization errors. Fundamentally, we approximate the state prediction with a Monte Carlo approximation where the sample mean of the ensemble models the mean $\boldsymbol{\mu}$ and the sample covariance the covariance $\boldsymbol{\Sigma}$. Thus, rather than calculating these values explicitly during state prediction at time t as in Eq. 2.10, we instead start with an ensemble of E members sampled from the prior distribution $\mathcal{N}(\boldsymbol{\mu}_{t-1|t-1}, \boldsymbol{\Sigma}_{t-1|t-1})$ such that $\boldsymbol{X}_{t-1|t-1} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^E]$. Each member is propagated forward in time using the state evolution model with an additional perturbation sampled from the process noise,

$$\boldsymbol{x}^j_{t|t-1} = \boldsymbol{G}\boldsymbol{x}^j_{t-1|t-1} + \mathcal{N}\left(0, \boldsymbol{Q}_t\right), \quad 1 \leq j \leq E. \tag{3.1}$$

As E approaches infinity, the ensemble effectively models the full covariance calculated in Eq. 2.5 [40]. We note that in BIP the state transition function is linear, however, when this is not the case the nonlinear function $g(\cdot)$ is used directly.

During the measurement update step, we calculate the innovation covariance $\boldsymbol{S}$

and the Kalman gain $\boldsymbol{K}$ directly from the ensemble, with no need to specifically maintain a covariance matrix. We begin by calculating the transformation of the ensemble to the measurement space, via the nonlinear observation function $h(\cdot)$, along with the deviation of each ensemble member from the sample mean:

$$\boldsymbol{H}_t\boldsymbol{X}_{t|t-1} = \left[h(\boldsymbol{x}^1_{t|t-1}), \ldots, h(\boldsymbol{x}^E_{t|t-1})\right]^\intercal, \tag{3.2}$$

$$\begin{aligned}\boldsymbol{H}_t\boldsymbol{A}_t = \boldsymbol{H}_t\boldsymbol{X}_{t|t-1} \\ - \left[\frac{1}{E}\sum_{j=1}^{E} h(\boldsymbol{x}^j_{t|t-1}), \ldots, \frac{1}{E}\sum_{j=1}^{E} h(\boldsymbol{x}^j_{t|t-1})\right].\end{aligned} \tag{3.3}$$

The innovation covariance can now be found with

$$\boldsymbol{S}_t = \frac{1}{E-1}(\boldsymbol{H}_t\boldsymbol{A}_t)(\boldsymbol{H}_t\boldsymbol{A}_t)^\intercal + \boldsymbol{R}_t, \tag{3.4}$$

which is then used to compute the Kalman gain as

$$\boldsymbol{A}_t = \boldsymbol{X}_{t|t-1} - \frac{1}{E}\sum_{j=1}^{E} \boldsymbol{x}^j_{t|t-1}, \tag{3.5}$$

$$\boldsymbol{K}_t = \frac{1}{E-1}\boldsymbol{A}_t(\boldsymbol{H}_t\boldsymbol{A}_t)^\intercal\boldsymbol{S}_t^{-1}. \tag{3.6}$$

With this information, the ensemble can be updated to incorporate the new measurement perturbed by stochastic noise:

$$\begin{aligned}\tilde{\boldsymbol{y}}_t &= \left[\boldsymbol{y}_t + \epsilon^1_y, \ldots, \boldsymbol{y}_t + \epsilon^E_y\right], \\ \boldsymbol{X}_{t|t} &= \boldsymbol{X}_{t|t-1} + \boldsymbol{K}(\tilde{\boldsymbol{y}}_t - \boldsymbol{H}_t\boldsymbol{X}_{t|t-1}).\end{aligned} \tag{3.7}$$

It has been shown that when $\epsilon_y \sim \mathcal{N}(0, \boldsymbol{R}_t)$, the measurements are treated as random variables and the ensemble accurately reflects the error covariance of the best state estimate [13].

One of the advantages of this algorithm is the elimination of linearization errors through the use of the nonlinear functions. While this introduces non-normality

into the state uncertainties, it has been shown that the stochastic noise added to the measurements pushes the updated ensemble towards normality, thereby reducing the effects of higher-order moments [65, 67] and improving robustness in nonlinear scenarios. The full algorithm is given in Fig. 23.

Non-Gaussian Prior: Another source of non-Gaussian uncertainty is from the initial estimate (the prior) itself. In BIP, our prior is given by a set of demonstrations which indicate where we believe a successful demonstration would lie in the state space. As we have yet to assimilate any observations of a new interaction, the (unknown) true distribution from which the demonstrations are sampled represents our best initial estimate of what it may be. However, given that these are real-world demonstrations, they are highly unlikely to be normally distributed. As such, two options are available in this case: we can either use the demonstrations directly as samples from the non-Gaussian prior distribution or approximate the true distribution with a Gaussian and sample from it. The latter approach is used by BIP in Eq. 2.13 and Eq. 2.14, however, this comes with its own risk as a poor initial estimate can lead to poor state estimates [47]. Given that the ensemble-based filtering proposed here provides a degree of robustness to non-Gaussian uncertainties, we choose to use samples from the non-Gaussian prior directly in the eBIP algorithm, with the knowledge that the ensemble will be pushed towards normality. In the event that the number of ensemble members E is greater than the number of available demonstrations, then the density of the true interaction distribution will need to be estimated given the observed demonstrations. This can be accomplished using any density estimation technique, however, we employ a Gaussian mixture model here and denote the resulting algorithm as eBIP^{-}.

Computational Performance: Many HRI applications require the use of mul-

Ensemble Bayesian Interaction Primitives

Input: $\boldsymbol{W} = [\boldsymbol{w}_1^\intercal, \ldots, \boldsymbol{w}_N^\intercal] \in \mathbb{R}^{B \times N}$: set of B basis weights corresponding to N demonstrations, $\boldsymbol{l} = \left[\frac{1}{T_1}, \ldots, \frac{1}{T_N}\right] \in \mathbb{R}^{1 \times N}$: reciprocal lengths of demonstrations, $\boldsymbol{y}_t \in \mathbb{R}^{D \times 1}$: sensor observation at time t.

Output: $\hat{\boldsymbol{y}}_t \in \mathbb{R}^{D \times 1}$: the inferred trajectory at time t.

1. Create the initial ensemble $\boldsymbol{X}_0$ such that

$$\boldsymbol{x}_0^j = \left[0, \dot{\phi}^j, \boldsymbol{w}^j\right], \quad 1 \leq j \leq E,$$

   eBIP: $i \sim \mathcal{U}\{1, N\}$, $\boldsymbol{w}^j = \boldsymbol{w}_i$, $\dot{\phi}^j = \frac{1}{T_i}$.
2. For time step t, propagate the ensemble forward in time as in Eq. 3.1:

$$\boldsymbol{x}_{t|t-1}^j = \boldsymbol{G}\boldsymbol{x}_{t-1|t-1}^j + \mathcal{N}\left(0, \boldsymbol{Q}_t\right), \quad 1 \leq j \leq E.$$

3. If a measurement $\boldsymbol{y}_t$ is available, perform the measurement update step from Eq. 3.7:

$$\boldsymbol{X}_{t|t} = \boldsymbol{X}_{t|t-1} + \boldsymbol{K}(\tilde{\boldsymbol{y}}_t - \boldsymbol{H}_t \boldsymbol{X}_{t|t-1})$$

4. Extract the estimated state and uncertainty from the ensemble:

$$\boldsymbol{\mu}_{t|t} = \frac{1}{E}\sum_{j=1}^{E} \boldsymbol{x}_{t|t-1}^j, \quad \boldsymbol{\Sigma}_{t|t} = \frac{1}{E-1}\boldsymbol{A}_t \boldsymbol{A}_t^\intercal$$

5. **Output** the trajectory for each controlled DoF:

$$\hat{\boldsymbol{y}}_t = h(\boldsymbol{\mu}_{t|t})$$

6. Repeat steps 2-5 until the interaction is concluded.

Figure 23: Ensemble Bayesian Interaction Primitives

tiple sensors, which increases the size of the latent space dimension and results in undesirable increases in computation time in the BIP algorithm. This is due to the necessary covariance matrix operations defined in Eq. 2.10, which causes BIP to yield an asymptotic computational complexity of approximately $O(n^3)$ – with the state of

the art lower bounded at approximately $O(n^{2.4})$ – where n is the state dimension [103]. However, as eBIP is ensemble-based, we no longer explicitly maintain a covariance matrix; this information is implicitly captured by the ensemble. As a result, the computational complexity for eBIP is approximately $O(E^2 n)$, where E is the ensemble size and n is the state dimension [72]. Since the ensemble size is typically much smaller than the state dimension, this results in a performance increase when compared to BIP. Furthermore, the formulation presented in this work also obviates the need to explicitly construct the observation matrix $\boldsymbol{H}$. The creation of the observation matrix introduces an additional overhead for BIP as it must be initialized at each time step due to the phase-dependence, a process which is unnecessary in eBIP.

Ensemble Bayesian Interaction Primitives also benefit from the computational performance-accuracy trade off inherent to all sample-based methods. Inference accuracy can be sacrificed for computational performance by lowering the number of ensemble members when called for. While this is also true for particle filters, they generally scale poorly to higher state dimensions due to sample degeneracy. In particle filtering, ensemble members are re-sampled according to their weight in a scheme known as importance sampling. However, in large state spaces it is likely that only a small number of ensemble members will have high weights, thus eventually causing all members to be re-sampled from only a few. In our proposed method this is not the case, as all members are treated as if they have equal weight, thus lending itself well to high-dimensional state spaces.

3.2 Full-Body Haptic Social Interactions

The proposed algorithm, ensemble Bayesian Interaction Primitives, excels at performing inference in high-dimensional multimodal interactions. One type of interaction which is of particular interest is a full-body haptic social interaction. Physical human-robot interaction in social contexts requires the robot to be cognizant of the safety and comfort of the human partner. Whole-body haptic information, consisting of both kinesthetic and tactile information, is critical in intimate haptic interactions as the arms are often occluded and touch is the only valid source of information on the timing and intensity of contact. In hugging, for example, each person feels the other's hugging force at the chest and back and adjusts their own exerted force in order to reciprocate.

While whole-body haptic modalities are starting to see usage in pHRI scenarios, they have thus far been largely limited to state classification and behavior selection using a snapshot of tactile information [6]. The main difficulties in using continuous-time tactile information for learning haptic interactions are twofold: 1) high data dimensionality and 2) temporal and spatial sparsity. Whole-body tactile sensing requires tens or even hundreds of sensors to cover a wide area while preventing gaps in coverage. The high dimensionality of tactile data not only increases the computational complexity but also makes the effect of tactile information on the model larger than that of kinesthetic information. Another negative effect of dense sensor placement is that the data of individual sensors tends to be both temporally and spatially sparse because tactile information is relevant only when the robot and human partner are in contact, and many of the sensors may not see any contact during a specific interaction.

In this section, we will show that ensemble Bayesian Interaction Primitives can be

successfully employed in a hierarchical manner with a force-based joint retargeting controller in order to engage in pHRI in a whole-body haptic social interaction. The eBIP model is capable of predicting both an appropriate robotic response (consisting of joint trajectories) as well as contact forces that should be exerted by the robot, given observations of the partner's pose and the force currently exerted by the partner onto the robot. The predicted joint trajectories and contact forces are then sent to a motion retargeting controller [58], which is capable of accounting for variances in the partner's body shape and size.

While other non-social experiments in this chapter include haptic interaction, the tactile information is dense, low-dimensional, and only used as an input observation. In order to tackle whole-body sparse tactile information, we develop three techniques for reducing the dimensionality of tactile information for learning haptic pHRI through the eBIP framework. Firstly, we take advantage of temporal sparsity by employing a non-uniform distribution of basis functions to reduce the dimensionality of the latent space that represents the demonstrated interactions. The second technique leverages spatial sparsity to choose the relevant tactile sensors by maximizing the mutual information between the input and output force measurements, effectively choosing the sensors which exhibit the highest mutual dependency. An issue with this method is that due to variations in human motion and body size, the chosen sensors may not experience contact during actual interactions. To remedy this, our third technique uses hand-crafted features for inference; specifically, the maximum force value within a predefined (local) tactile sensor group. This method is motivated by the intuition that there will be no significant difference in perception regardless of where contact occurs in a densely packed sensor group, i.e. the force corresponding to a general spatial location is more important than a specific one.

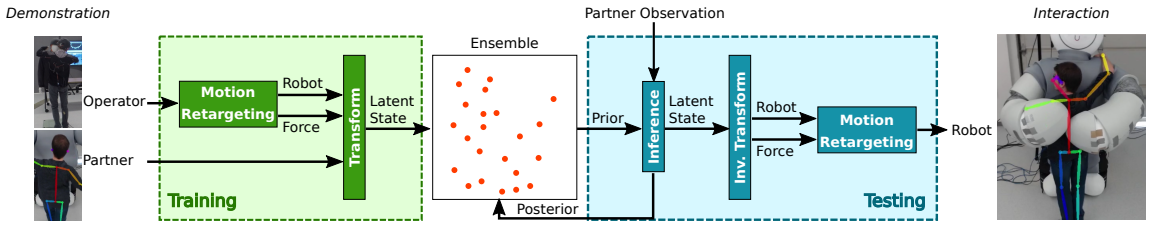


Figure 24: Framework overview. Training demonstrations are used to populate the initial ensemble, which is then updated recursively in testing.

3.2.1 Latent State Representation

We explicitly model the joint spatiotemporal relationship of the poses and forces between the robot and the partner with ensemble Bayesian Interaction Primitives. This relationship is learned from a set of training demonstrations and acts as a source of prior knowledge for the interaction, as depicted in the *Training* block of Fig. 24. At run-time, the robot is presented with observations of a human partner and, utilizing both the prior knowledge and the current observations, infers (a) the next partner actions, and (b) the appropriate robot response. This is shown in the *Testing* block of Fig. 24. The spatiotemporal relationship we model allows us to infer both the robot's poses and forces from those of the partner, allowing responsive behavior to not only discernable movements but also indiscernable ones, such as the strength of the hug.

The latent state representation consists of the inferred joint positions as well as the contact forces along the body of the robot. After each update step, rather than sending the inferred joint positions directly to a position controller for execution, we instead send it to a retargeting controller [58] as a reference signal along with the expected contact forces. The retargeting controller then modifies the joint positions until the desired contact forces are achieved.

3.2.2 Feature Selection for Sparse Contact Forces

The robot employed in the social interactions in this work is equipped with 61 force sensors over its whole body as discussed in Sec. 3.3.2 and it is expected that not all of them will be in contact with the partner during an interaction. One difficulty in working with full-body haptic modalities is that the dimension of the state space tends to increase dramatically. A state space that is *too* large is undesirable for several reasons 3.1, most significantly that it can a) increase the error in higher-order statistics that are not tracked and b) increase the number of ensemble members required to track the true distribution, which negatively impacts computational performance and increases the minimum number of training demonstrations that must be provided.

However, we can exploit the fact that haptic modalities are sparse, both temporally and spatially, in order to reduce the state dimension. In the case of temporal sparsity, the force sensors will only relay useful information at the time of contact; the time periods before and after are not informative and do not need to be approximated. In terms of spatial sparsity, some sensors may not experience contact at all depending on the specific interaction. For example, a hugging motion where both of the partner's arms wrap under the robot's arms will register different contact forces than if the arms wrap over. Furthermore, this can vary between person to person as physical characteristics such as height influence contact location and strength.

To utilize temporal sparsity, we employ a non-uniform distribution of basis functions across a subregion of the phase domain, unlike in prior works [18, 20] where a uniform distribution over the full domain of $[0, 1]$ was utilized. The basis space is found via Orthogonal Least Squares (OLS) [28], as our basis decomposition shares many similarities with a 1-layer Radial Basis Function network. This allows us to reduce

the number of required basis functions while still adequately covering the informative interaction periods.

Spatial sparsity, however, requires a different approach as we cannot know which sensors will be useful for a given interaction in advance. We therefore employ a feature selection method based on mutual information [84, 109] to maximize the dependency between the input and output force sensor DoFs. Some care must be taken as there are a limited set of training demonstrations available and we are selecting based on continuous state variables (basis function coefficients), however, this can be overcome through binning [62], K-nearest neighbors [92], or Parzen windows [64]. In our case, we employ mutual information estimation based on binning and incrementally select features until there is no significant improvement in mutual information, based on a desired threshold (> 0.07). We opt to use mutual information rather than other standard measures as some of the force sensors experience false positives due to deformation caused by the robot's own movements.

By considering the physical layout of the robot, we can further reduce the state dimension. In general, tactile sensors have a limited sensing range and are used in large groups to provide greater surface coverage. However, when a human exerts a contact force to this region we do not care whether it activated the first or the tenth sensor of a group, we simply care that force was exerted on this region. This allows us to reduce an entire group of force sensors to a single DoF by simply taking the maximum force value of all the sensors in that group at any given point in time.

3.3 Experimental Design

We evaluate the effectiveness of ensemble Bayesian Interaction Primitives in two real-world multimodal experiments: one in which a human participant outfitted with a variety of sensors tosses a ball which is caught by a UR5 [91] arm, and another which a robot equipped with whole-body haptic sensors hugs a human partner.

3.3.1 Ball Catching

The sensors can be broadly grouped into two categories: modalities that observe the human and modalities that observe the ball. Observations of the ball are unavailable while it is grasped by the human, due to occlusions, and do not become available until the ball is thrown. In empirical tests, we have observed that it is not possible for the robot to catch the ball using a purely reactive strategy given the limited time to react, kinematic constraints, phase lag, etc. Hence, we leverage the observation modalities of the human to predict how the robot should react while the human is still in the preparatory phase, i.e., the "wind up" for the throw. This strategy is fundamentally built upon a predictive approach – we can begin reacting as early as possible and refine our predictions as more detailed observations become available.

The experiment is designed to emphasize the advantages of a multimodal observation set by having different sensors reveal different information about the true environment state at different points in time. However, throwing and catching are fast-paced actions requiring a high frequency observation rate and appropriately low computation times for inference; without these properties an HRI algorithm will likely fail at catching the ball in real experiments. We utilize sensor observations of 8 objects

from 5 modalities: the positions of the human participant's hands and feet, inertial measurements of the throwing arm, pressure measurements from the soles of both left and right feet, the orientation of the head, the position of the ball being thrown, and the joint positions of the robot. The observations were synchronized and collected at a frequency of 60Hz. The basis decomposition for each sensor object was chosen from a set of candidate basis spaces comprised of Polynomial, Gaussian, and Sigmoid functions – standard choices in this type of application [11] – using the Bayesian Information Criterion, yielding a total state space dimension of 559 dimensions.

During training, the ball was thrown from a distance of approx. 3.7m and was caught within a box grasped by the robot (the end effector used in this experiment actuates too slowly for in-hand interception). An initial set of 221 demonstrations was provided via kinesthetic teaching in which the robot was manually operated by a human (top left of Fig. 24) in order to catch the ball while joint positions were recorded. These demonstrations provided the only source of prior knowledge for the interaction (for both state estimation and control); no inverse kinematics or other models were employed at any point in time. We compare our algorithm, to the original BIP formulation, as well as particle filtering (PF). In all cases, the PF model used the same number of ensemble members as in eBIP and employed a systematic resampling scheme when the effective number of members was less than $E/2$.

3.3.2 Whole-Body Haptic Hugging

In this work, we collect human-robot demonstrations with a teleoperated bimanual robot as it is difficult to demonstrate intimate interactions through direct kinesthetic teaching. Compared to obtaining data from human-human demonstrations, this

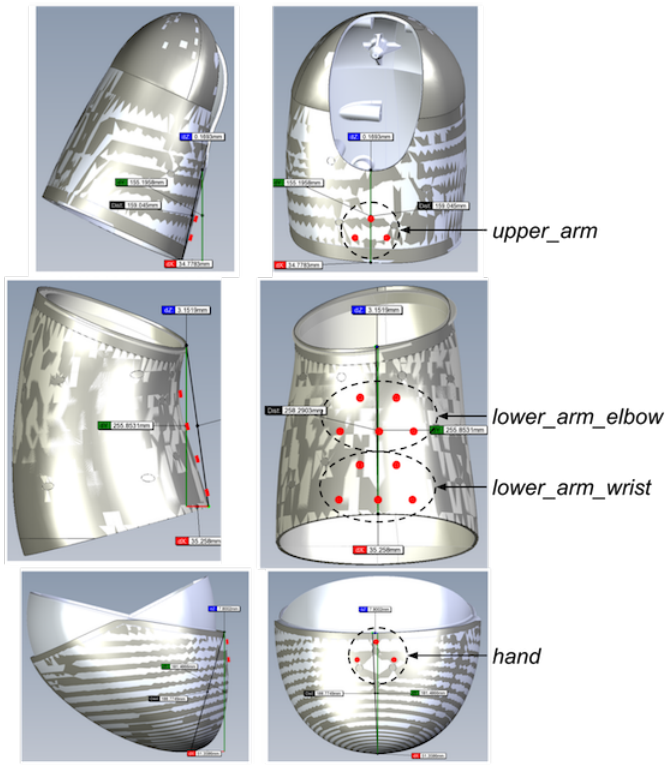


Figure 25: Force sensor placement on the robot arm. The dotted circles denote the grouping corresponding to the force sensors attached to the operator.

method has the advantage of directly modeling the motion and force which the robot is physically capable of as well as genuine human reactions.

3.3.2.1 Robot Hardware and Teleoperation System

We use a teleoperated bimanual robot consisting of two torque-controlled Franka Emika 7-DOF arms [44] wrapped in a soft, padded exterior. A set of 61 force sensors [105] are embedded into the padded exterior of the robot, providing tactile information on the robot's arms (Fig. 25), chest (Fig. 26 left), and back (Fig. 26 right).

During teleoperation, the operator hugs a static mannequin while wearing a suit

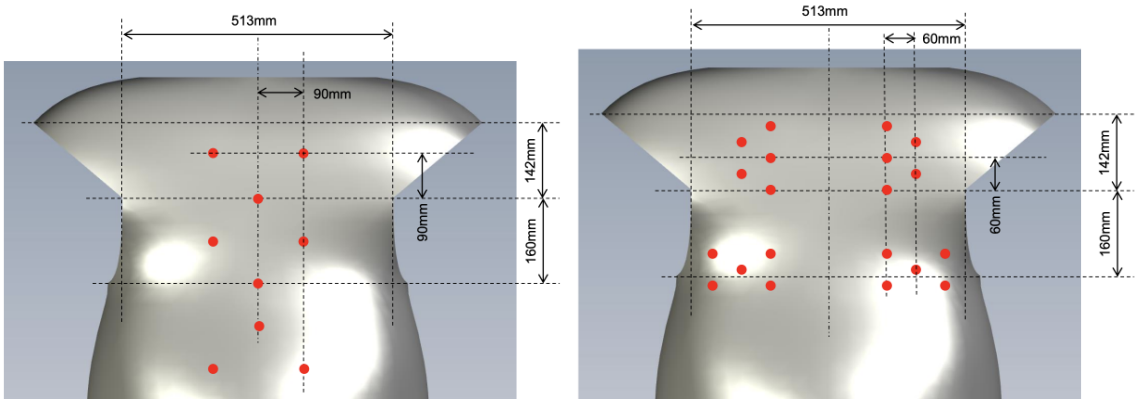


Figure 26: Force sensor placement on the robot's chest (left) and back (right).

equipped with 8 IMUs [54] (2 on the back and 1 on each hand, forearm, and upper arm) as well as 8 force sensors (1 on each hand, 2 on each forearm, and 1 on each upper arm). Because the robot's arms are much larger than the human's, each force sensor on the operator corresponds to a group of 3–5 sensors placed on a topologically similar location on the robot arm, as shown in Fig. 25. This placement ensures that contact forces are detected even with variation in motion and body shape. We use the raw output of the force sensors (via an 8-bit A/D converter) directly as a reference signal, possible only because the human and robot use the same type of sensor.

A motion retargeting controller has been developed [58] to adapt the operator motion such that the contact timing, states, and magnitudes on the robot approximately match those of the operator, thereby accounting for variations in body sizes and shapes. This retargeting controller is used in both training, in which the operator's force on the mannequin is matched, and in testing, in which the inferred contact force generated by BIP is matched.

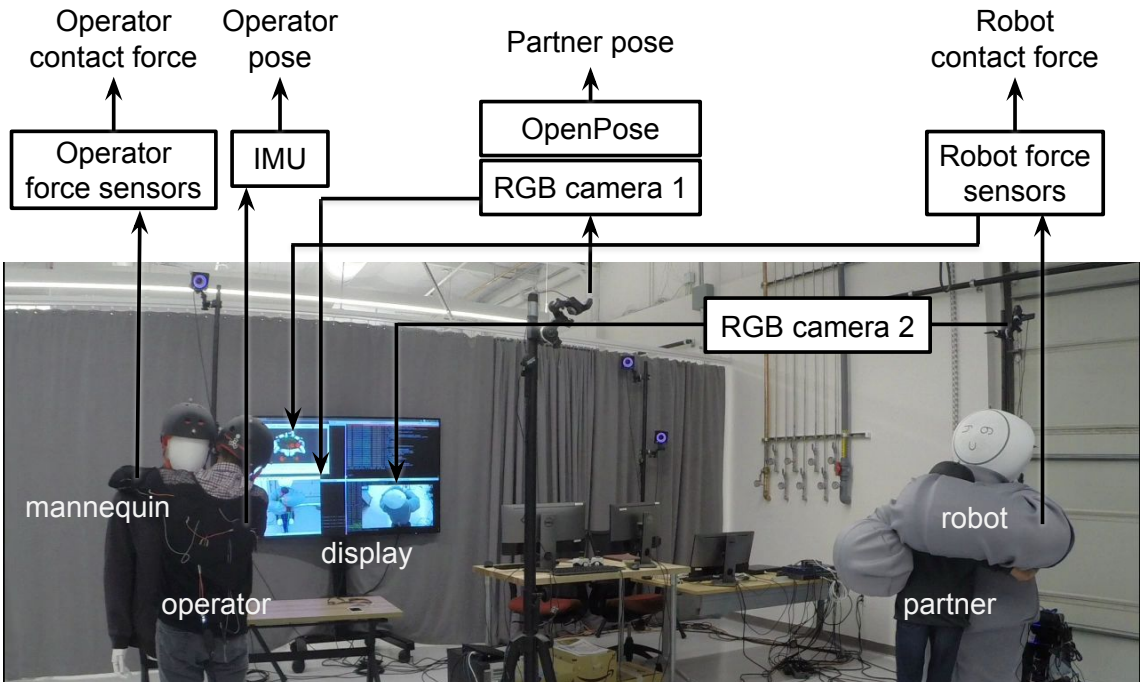


Figure 27: Demonstration data collection through teleoperation.

3.3.2.2 Demonstration Data Collection

Figure 27 shows the setup for collecting learning data using the teleoperated robot. In addition to the sensors attached to the operator and robot, we use two RGB cameras. The first is mounted at a fixed point above and behind the partner and used to estimate the partner's 2D pose at 30Hz using the OpenPose library [24]. To help the operator decide the motion and timing, the second camera is fixed behind the robot and allows the operator to view the interaction with the partner. The pose estimate of the partner is visualized for the operator's benefit, and the force sensor readings are overlaid as spheres on a model of the robot. Their size and color are dictated by the magnitude of the current reading, such that the operator can compensate for the lack of haptic feedback indicating the force exerted by the partner. The image

from the RGB camera used for skeleton tracking of the partner is also shown to the operator. Another RGB camera placed behind the robot provides a birds-eye view from the front of the partner. The data from robot force sensors are also visualized as the size and color of spheres along with a 3D model of the robot to supplement the lack of haptic feedback of the force exerted by the partner on the robot.

In summary, we collect the following data for training:

- Operator: motion (8 IMUs) and forces (8 sensors),
- Robot: 12 joint position commands generated by motion retargeting and contact forces (61 sensors), and
- Partner: pose (RGB camera and OpenPose).

Although we also measure the positions of the robot, partner, operator, and mannequin using an optical motion capture system, we ultimately did not use this data as the partner's distance from the robot was indirectly inferred based on the partner's vertical location in the image.

3.3.2.3 Training and Testing

We recruited 6 total participants to hug the robot, 4 of whom participated in training and all 6 in testing (a combination of offline and online); this allows us to test generalization capabilities as 2 of the testing participants were not previously seen. During training, the participants stood approximately 2m away from the robot and were instructed to step forward and initiate a hug by raising their arms. The robot, controlled by another person via teleoperation, responded by raising its arms so that the hug may proceed. The participants were also instructed to take the lead in disengaging from the hug. That is, when the participant ceased to apply pressure

to the robot, indicating that the hug was over, the operator lowered the robot's arms such that the participant could step back and conclude the hug. Furthermore, the participants were given instructions to attempt to match the strength of the robot's hug. This instruction was left intentionally vague such that each participant would naturally apply an amount of force appropriate to their own preferences in response to the force exerted by the robot. This was repeated 25 times for each participant resulting in a total of 150 training demonstrations, although we only used 121 of these as the remainder did not experience significant contact forces. The same operator controlled the robot for all participants and the BIP prior was trained using all 121 training demonstrations. The online tests were conducted with a model that utilized both force sensor grouping (spatial sparsity) and a non-uniform basis space selected with OLS (temporal sparsity).

During online testing, the participants were instructed to initiate and conclude the hug, as in training, except now with no teleoperation and no instruction to match the force of the robot. Each participant performed six long hugs, six short hugs, and two hugs each for the following edge cases: doing nothing, delaying before hugging, delaying after raising arms, hugging the air without moving, and hugging the robot without making contact. Online testing was performed with 3 participants – 2 of whom were novel and did not participate in training, and 1 who had participated in training – for a total of 66 hugs.

In addition to online testing we also conducted offline testing in which the BIP model's inference accuracy was compared against the ground truth data from demonstrations of all 4 training participants. This analysis was conducted using 10-fold cross validation such that the BIP was trained using 108 demonstrations at a time. Four model variations were tested: one which used all input force DoFs (All), one with

input force DoFs selected through mutual information feature selection (MIFS), one with force sensor grouping without MIFS (Group), and one with both force sensor grouping and non-uniform basis space (Group + OLS).

3.4 Results

3.4.1 Ball Catching

Robot Joint Error

		Ball	Shoe, IMU	Shoe, IMU Head	Shoe, IMU Head, Ball	All
43%	BIP	-	3.91×10^{15}	1.48×10^{14}	9.56×10^{14}	3.97×10^{16}
	PF	-	0.37	0.28	0.28	0.26
	eBIP−	-	5.62×10^{6}	9.98×10^{6}	1.08×10^{7}	6.00×10^{7}
	eBIP	-	0.18	0.19	0.18	0.14
82%	BIP	1.04×10^{15}	1.59×10^{17}	4.22×10^{15}	3.66×10^{15}	6.52×10^{17}
	PF	0.11	0.37	0.28	0.28	0.26
	eBIP−	10.52	9.46×10^{6}	1.36×10^{7}	1.68×10^{7}	7.66×10^{7}
	eBIP	0.05	0.20	0.19	0.11	0.09

Table 2: Indicates the mean squared error values for the first three joints of the robot at the time the ball is caught. A green box represents the best method and a gray box represents methods which are not statistically worse than the best method (Mann-Whitney U, $p < 0.05$). The values 43% and 82% indicate inference is performed after 43% of the interaction is observed (corresponding to before the ball is thrown) and 82% is observed (the ball is partway through its trajectory). The ball itself is not visible for the first 43% as this is when it is occluded by the participant's hand. The standard error for eBIP is less than ± 0.01 in all cases for the joint MSE and ± 0.015 for the ball MAE.

The inference errors for the robot joints are shown in Table 2, along with the errors for the estimation of the location of the ball at the time of interception in Table 3. For each data category, e.g., {Shoe, IMU}, only the indicated subset of sensor modalities

Ball Position Error

		Ball	Shoe, IMU Head, Ball	All
43%	BIP	-	116.93	1.35×10^3
	PF	-	0.61	0.61
	eBIP−	-	7.11×10^3	8.91×10^3
	eBIP	-	0.61	0.59
82%	BIP	6.67	205.04	2.74×10^3
	PF	0.20	0.26	0.28
	eBIP−	5.38	8.44×10^3	9.46×10^3
	eBIP	0.13	0.23	0.24

Table 3: Indicates the mean absolute error for the inferred ball position. A green box represents the best method and a gray box represents methods which are not statistically worse than the best method (Mann-Whitney U, $p < 0.05$). The values 43% and 82% indicate inference is performed after 43% of the interaction is observed (corresponding to before the ball is thrown) and 82% is observed (the ball is partway through its trajectory). The ball itself is not visible for the first 43% as this is when it is occluded by the participant's hand. The standard error for eBIP is less than ± 0.01 in all cases for the joint MSE and ± 0.015 for the ball MAE.

is observed. We evaluate the prediction capabilities by observing a partial trajectory of sensor measurements and inferring the robot joint positions, as well as the position of the ball at the time of interception. We divide this into two categories: 43% of the trajectory corresponds to the period of time in which the ball is in the human partner's hand and has yet to be thrown while 82% corresponds to when the ball is still in the air and has yet to be caught. The errors are listed in terms of the mean squared error for the first 3 joints of the robot (the wrist joints are less important in this scenario) and the mean absolute error of the ball prediction. Errors are computed via 10-fold cross validation over the randomly shuffled set of demonstrations, which also limits the maximum number of ensemble members used in both the eBIP and PF models to 198.

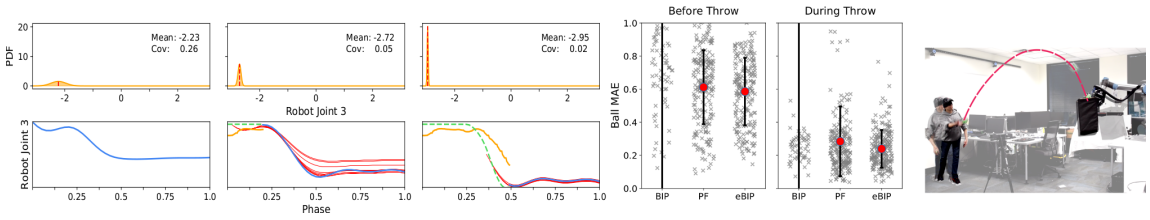


Figure 28: (**Left**) A sequence of frames from different time points during an interaction. Top: the PDF of the third robot joint; the initial uncertainty is high and decreases over time. Bottom: the inferred trajectory for the robot joint. The blue line indicates the current prediction while the red lines indicate the predictions for the past 10 time steps. The yellow line is the actual response from the robot while it attempts to follow the inferred trajectory, and the dashed green line indicates the expected trajectory from the demonstration. (**Center**) The ball MAE of the {All} subset. While overall error decreases for both PF and eBIP over time, only eBIP experiences a reduced variance. (**Right**) A blindfolded user throws a ball which is, in turn, caught by the robot.

Prior Approximation Errors: Results in Table I show that attempting to model the demonstrations with a parametric Gaussian model yields a poor approximation and leads to an incorrect estimate of the initial uncertainty. This is supported by the fact that both BIP (Gaussian prior) and eBIP$^-$ (mixture model prior) produce predictions that are many orders of magnitude from the true state. In the case of eBIP$^-$, expectation maximization regularly produced non-positive semi-definite covariance matrices (using 1 component as determined by BIC), indicating a poor fit to the data set. As a result, we were forced to use the sample mean and covariance of the demonstrations for the Gaussian prior as in BIP, from which the initial ensemble is sampled. The PF and eBIP methods, on the other hand, were initialized directly from the demonstrations without making an assumption about the parametric family of the true (unknown) distribution. As a result both methods fared much better, however, eBIP significantly more so as it achieved the best result in every category (see green box).

Linearization Errors: We can also observe that BIP certainly suffers from

linearization errors. Since eBIP$^-$ models the prior distribution with the same unimodal Gaussian as BIP, we expect it to suffer from the same prior approximation errors. However, we see from Table 2 that BIP yields a 1.04×10^{15} joint prediction MSE error when 82% of the ball trajectory is observed while eBIP$^-$ only yields a joint prediction MSE of 10.52. The remainder of this error is due to the different update methods and linearization errors inherent to BIP.

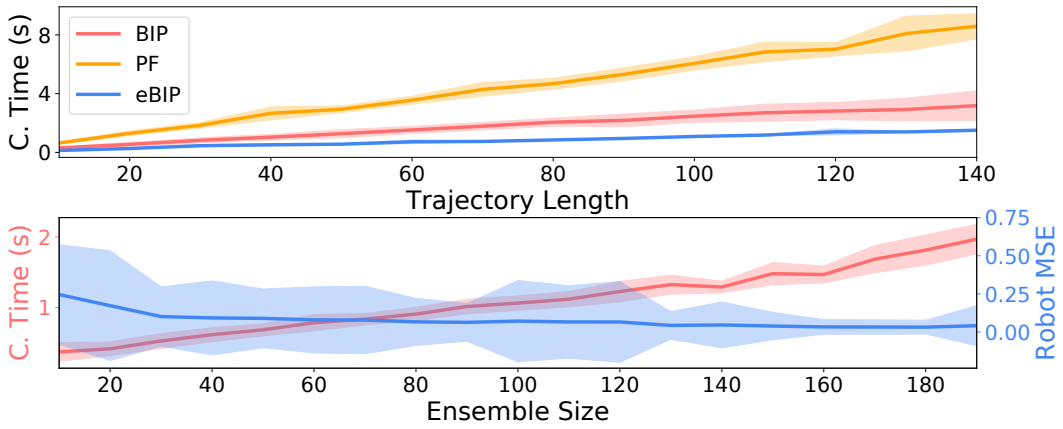


Figure 29: Top: computation time required for filtering observations of varying lengths. Bottom: the computation time-accuracy-ensemble size trade-off for eBIP.

Errors Resulting from Increased State Dimension: These results also show that both the PF and eBIP$^-$ produce worse inference results as the number of active modalities increases. For example, the PF predictions result in a joint MSE of 0.11 when only the trajectory of the ball was observed, but a MSE of 0.26 when all modalities were observed. We can rule out both prior approximation error and linearization errors, since PF utilizes the demonstrations and nonlinear system functions directly as in eBIP. Therefore, we conclude that the number of ensemble members is simply too low to provide accurate coverage of the state space leading

to sample degeneracy as a result of importance sampling. In the case of eBIP$^-$, the increasing error is due to the approximation errors stemming from the prior distribution, since otherwise the algorithm is identical to eBIP.

Errors Resulting from Additional Modalities: Lastly, we observe that the introduction of additional modalities does not always yield an increase in inference accuracy, although it may provide other benefits depending on the modalities. This is evident when comparing the joint MSE prediction errors of eBIP for the {Shoe, IMU} data set and that of the {Shoe, IMU, Head} data set. The introduction of the head modality actually *increases* the inference error when 43% of the trajectory is observed, which is when the head modality is most relevant. This becomes particularly evident when looking at the MAE results of the ball; adding additional sensor modalities increases the inference error of the final ball position by a factor of 2. However, we also gain the ability to initially predict the ball position much earlier in the interaction, before the ball is visible. This process is visualized in Fig. 28 through the uncertainty in the inferred joint trajectories. The width of the catchable region is approximately 180cm. Hence, the fact that we can predict the interception point to within about 60cm, or 1/3 of our operating region, *before the ball is visible* is quite significant. By the time the ball is in the air (82% of the trajectory is observed), we have further narrowed the prediction down to 23cm with additional modalities. While this error is still significantly higher than the 13cm error produced by incorporating only the ball, we observed empirically that a 23cm error still results in a catch in most cases and justifies the inclusion of additional modalities. Given that the radius of the ball itself is 4.5cm and the width of the box is 32cm, this amount of error is adequate and is offset by the proactive behavior of the robot in this setting. Still, the above results

suggests that there may be substantial benefit to the ability of the inference process to switch modalities on and off in real-time according to the context.

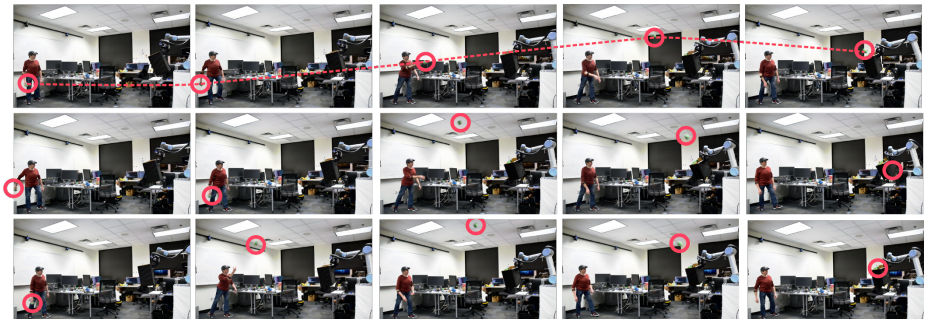


Figure 30: A sequence of images from three live interactions. The robot is already reacting to the human by the second image in each sequence and catches the ball in different poses due to the different ball trajectories.

Real-time interactions: Results of the real-time interaction and reproduction with the robot can be seen in Fig. 30. To minimize the computation time required for inference, the number of ensemble members was limited to 80. This setup ensured a reasonable trade-off between accuracy and computational performance as shown in Fig. 29. Three observation can be made from the image sequences in Fig. 30: the throws all have significantly different trajectories (the top throw has a low apex and fast velocity while the bottom throw has a high apex and low velocity), the robot is already moving into position before the ball is thrown (second image in each sequence), and the robot catches the ball in a different pose for each throw (last image in each sequence). Recordings of a variety of throwing experiments can be found in the accompanying video. To avoid habituation or any unconscious effort to throw the ball directly at the robot, we also performed a set of experiments in which the user threw the ball while in a blindfolded condition, see Fig. 28 (right) for an example. Even under this condition, the ball was successfully caught 12 out of 20 times, for a

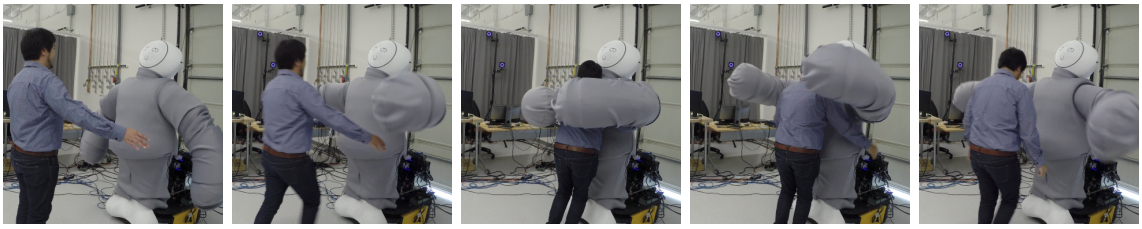


Figure 31: Snapshots from a hug interaction.

Table 4: The mean absolute error (MAE) values for the predicted joint (radians), left arm force (raw), and right arm force (raw) values for a phase look-ahead of 0.0, 0.05, and 0.1 computed using 10-fold cross validation. A green box represents the best method and a red box any method which *is* statistically worse than the best method (Mann-Whitney U, $p < 0.05$).

		All	MIFS	Group	Group + OLS
	Dimension	664	446	392	378
0.00	Joints	0.191	0.180	0.183	0.180
	Left	7.224	7.580	8.011	7.937
	Right	3.760	4.163	3.936	3.868
0.05	Joints	0.213	0.201	0.209	0.205
	Left	7.992	7.965	8.257	8.077
	Right	4.058	4.110	3.897	3.850
0.10	Joints	0.250	0.238	0.247	0.243
	Left	9.03	8.378	8.257	8.180
	Right	4.355	4.061	3.921	3.867

success rate of 60%. We noticed, however, that in this condition the user frequently threw the ball outside the robot's reach.

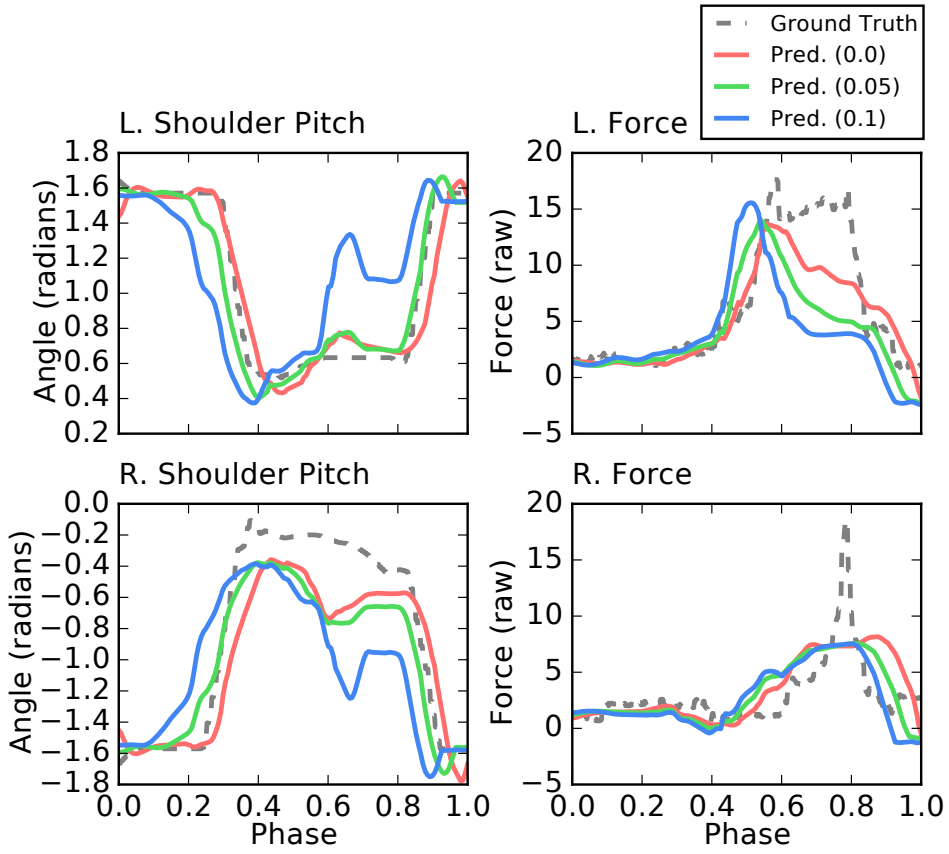


Figure 32: Left: shoulder pitch joint angle in the robot's left (top) and right (bottom) arms. The dashed gray line indicates the ground truth, while the solid red line indicates the prediction with a phase look-ahead of 0.0, green with 0.05, and blue with 0.1. Right: the force ground truth and predicted values for the mean wrist force sensors in the left and right arms.

3.4.2 Whole-Body Haptic Hugging

Figure 31 shows snapshots from a successful hug. The supplemental video shows select demonstrations and online interactions including the edge cases. During online

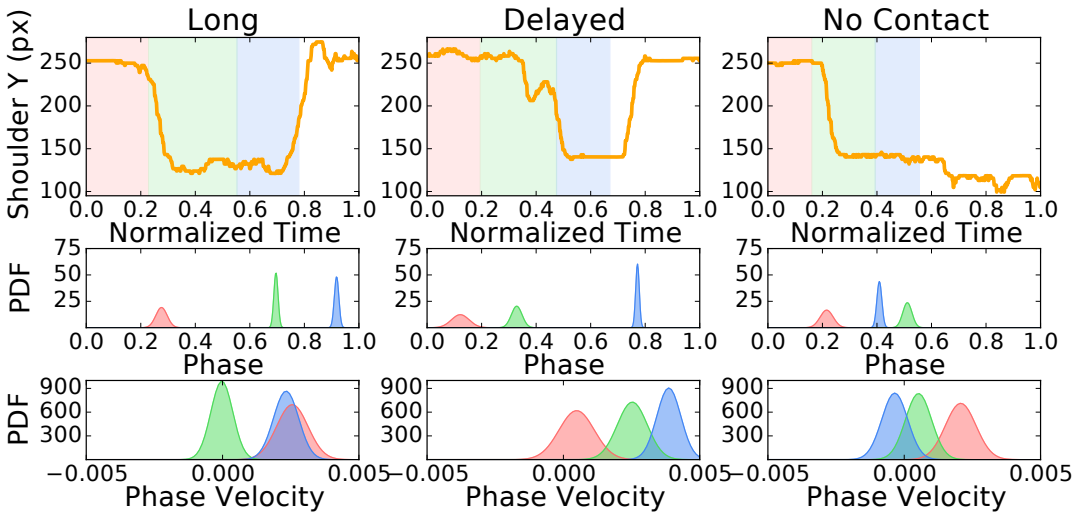


Figure 33: Top: the position of the human's right shoulder along the y-axis as determined by pose tracking. The red shaded region indicates the portion of the interaction before movement has begun, the green region the portion after contact has been made, and the blue region the portion where the human is withdrawing from the hug. Middle: the phase distribution corresponding to the end of each region. Bottom: the phase velocity distribution.

testing, we achieved an approximately 82% (54/66) qualitative success rate which we define as the robot hugging the human participant and responding to their cues; this is by definition a subjective metric but it is a useful frame of reference. Breaking this down further, novel participants were hugged successfully ∼80% (35/44) of the time while the sole participant who also trained was successful ∼86% (19/22) of the time.

Table 4 shows the mean absolute error results for the inferred joint positions and contact forces in the left and right arms in offline testing. We show that despite our proposed method (Group + OLS) having the smallest state dimension, it never performs significantly worse than any other method. This indicates that we can leverage the sparsity of contact forces in order to reduce the state dimension without negatively impacting inference accuracy. We note that the smaller MAE values for

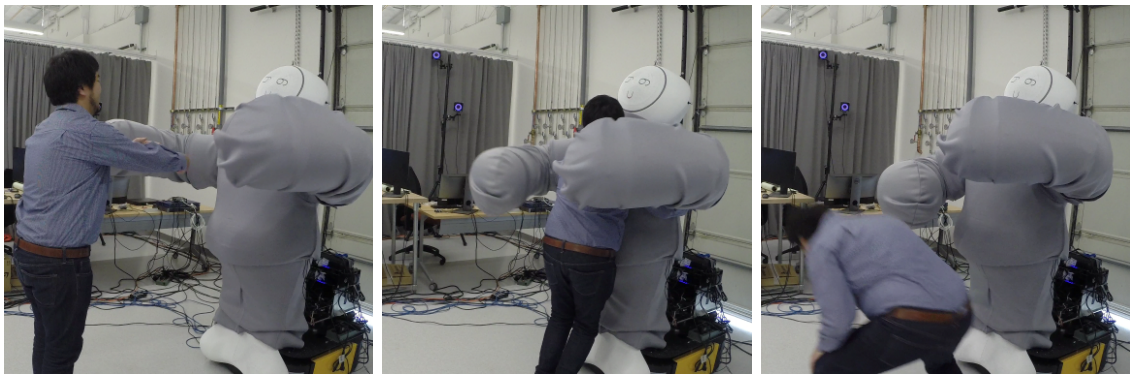


Figure 34: Edge cases. Left: hugging the air; the hug did not complete because the phase did not proceed further. Center: delay before hugging; hug was successful because the model correctly recognized the beginning of a hug. Right: hugging without making contact; hug failed because the model received conflicting information.

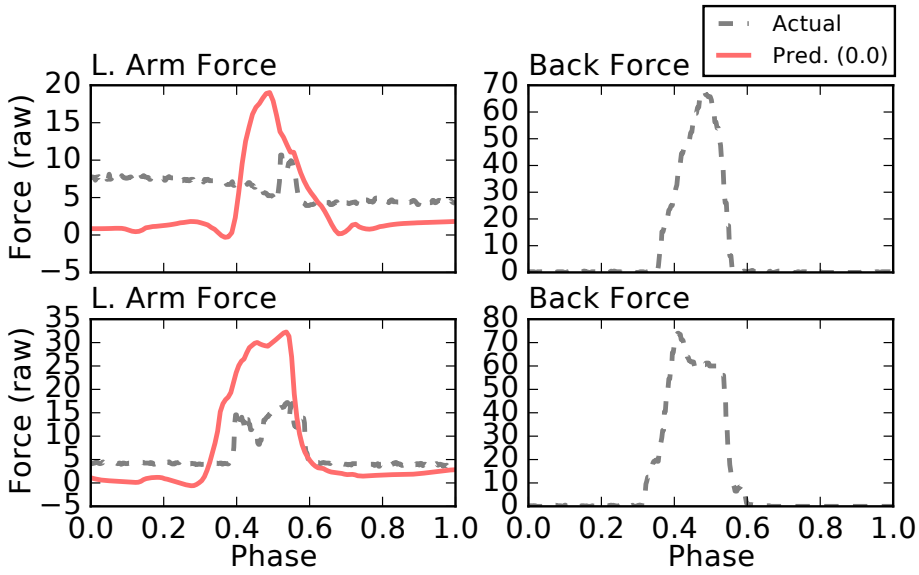


Figure 35: Top: the contact forces for the mean of the wrist force sensors in the left (left) and a back sensor (right) for a single interaction. The dashed gray line indicates the actual force and the solid red line the predicted. Bottom: the contact forces for a different interaction.

the right arm forces do not indicate a better prediction, simply that they experienced less pressure; we emphasize the comparison between methods and not output DoFs.

We also introduce the notion of a phase look-ahead value in these results, which is a non-negative offset applied to the inferred phase. A phase look-ahead of 0.0

means we predict values for the currently estimated phase; a look-ahead of 0.1 means we are predicting values 10% (relative to phase) into the future. This look-ahead is vital for natural interactions, as it allows us to overcome phase lag and produce more responsive robot actions. However, as Table 4 demonstrates, a higher look-ahead value results in higher inference error. This is further demonstrated in Fig. 32. While all look-ahead values result in accurate responses, higher look-ahead values result in temporally earlier responses at the cost of larger errors (especially noticeable with 0.1 look-ahead error).

Figure 33 shows the phase estimate distributions for various edge cases conducted during online testing. The BIP model used in this work is robust to temporal variance which is exhibited in the phase and phase velocity adaptation here. Most significantly, we show that when there is a long hug duration the phase velocity drops to 0 and the interaction essentially halts (left column, green), until the human partner withdraws their arms (left column, blue). Similarly, a delay at the beginning of the interaction results in a phase velocity of 0 (middle, red), ceasing the temporal progression of the interaction until the human begins to move (middle, green). In the case of no contact, the interaction progresses until the point where contact pressure is expected (right, green). As this pressure never comes, the interaction halts indefinitely until cancelled (right, blue). Snapshots from some edge cases are shown in Fig. 34.

Figure 35 demonstrates a weakness in the BIP interaction model: it does not account for external physical limitations. In this case, two different interactions are shown in which the inferred contact force for the left arm is driven by a force on the back, however, the actual resulting left contact force is unable to realize the inferred value. This can be due to multiple reasons, including joint angle limitations, limited

force sensor coverage, collision between hands, and unexpected movement from the human.

3.5 Related Work

In the following section, we will review relevant work on probabilistic modeling of joint actions and multimodal modeling. For a detailed discussion of computational techniques in the HRI domain, see the excellent surveys in [55, 102].

3.5.1 Probabilistic Modeling of Joint Actions

Early work on modeling HRI scenarios using probabilistic representations focused on HMMs [88] as a method of choice, see for instance the works in [66, 94]. The ability of HMMs to perform inference in both time and space, makes them particularly interesting for collaborative and interactive tasks. However, these advantages come at a cost – HMMs require a discretization of the state space and do not scale well in high-dimensional spaces. The concept of Interaction Primitives (IP) was first proposed in [4] as an alternative approach for *learning from demonstration*. Intuitively, an IP models the actions of one or more agents as time dependent trajectories for each measured degree of freedom. The approach has gained popularity in HRI and has been applied to a number of tasks [34, 18, 81, 35, 26, 41].

Most recently, in [18] a fully Bayesian reformulation of IPs called *Bayesian Interaction Primitives* (BIP) was introduced. Most importantly, this work establishes a theoretical link between HRI and joint optimization frameworks as found in the Simultaneous Localization and Mapping (SLAM) literature [103]. The resulting

inference framework for BIPs was shown to produce superior space-time inference performance when compared to previous IP formulations.

3.5.2 Multimodal Modeling and Inference

Multimodal integration, inference and reasoning has been a longstanding and challenging problem of artificial intelligence [32] and signal processing [69]. Following the principles formulated by Piaget [86], many existing multimodal systems separately process incoming data streams of different types, deferring the integration step to later stages of the processing pipeline. In [14], Calinon and colleagues present a fully probabilistic framework in which social cues from the gaze direction and speech patterns of a human partner are incorporated into the robot movement generation process. The multimodal inference process is achieved by modeling such social cues as prior probability distributions. In a similar vein, the work by Dermy et al. [35] uses joint probability distributions over human-robot joint actions, in order to infer robot responses to human visual or physical guidance. However, the approach assumes a fixed user position at training and test time and models the phase variable according to a predetermined relationship to the execution speed. More recently, deep learning approaches for multimodal representations have gained considerable attention. A prominent methodology is to process each data modality with separate sub-networks, which are integrated at a shared final layer [78, 89]. However, such neural network approaches are not well-suited for probabilistic data integration and do not provide an estimate of the uncertainty inherent to observations or outputs. Also, such approaches cannot cope with missing inputs or changing query types, i.e., any change to the number or type of inputs requires a complete retraining of the network.

3.5.3 Social and Physical Human-Robot Interaction

Intimate, social pHRI such as hugging has been found to have positive effects on the human emotional state [111, 95, 98]. However, the robot platforms used in these studies all had limited physical capabilities, making it impossible for the robots to provide the human with reciprocal forces.

Block et al. [12], on the other hand, developed a robot based on the PR-2 that can hug a person with significant force. Their study suggests that hug duration and strength are important factors to realize comfortable robotic hugs. The caveat is that the robot largely acted independently of the human during experiments. While the release timing of the hug was adapted in response to user contact, the hug itself was initiated by the robot and the strength was predetermined from three discrete levels.

3.5.4 Learning Haptic pHRI

Recently, robots with whole-body tactile sensing capabilities have been developed for pHRI, in particular, collision response, human-robot communication, and robot behavior development [6]. Although social pHRI can be considered a form of human-robot communication, most work thus far uses a snapshot [75] or statistic [79, 57] of tactile information for classifying human intention or behavior. In contrast, time-series tactile information has largely been limited to non-HRI tasks such as whole-body grasping [74] and locomotion [45], although recent studies aim to change this [8]. Advances in sensing may also further this area, as Kim et al. [60] proposed to cover a robot with large, airtight, pressure-sensing cavities (in contrast to densely placed small tactile sensors), although this has yet to be applied to pHRI.

Using kinesthetic teaching or traditional haptic devices, one can apply LfD to HRI tasks involving haptic interactions at the end effectors or through an object. Calinon et al. [17] presented an LfD framework for teaching a collaborative lifting task by operating a humanoid robot and feeling the interaction force through a PHANToM device. Peternel et al. [85] developed a dedicated haptic interface to demonstrate compliant responses to human push and pull of a standing humanoid robot.

3.6 Conclusions

In this chapter, we introduced ensemble Bayesian Interaction Primitives and discussed their application to state estimation, inference, and control in challenging, fast-paced HRI tasks with many data sources. We discussed an ensemble-based approach to Bayesian inference in eBIP, which requires neither an explicit formation of a covariance matrix, nor a measurement model, resulting in significant computational speed-ups. The approach allows for fast inference from high-dimensional, probabilistic models and avoids typical sources for inaccuracies, e.g., linearization and Gaussian priors. In our real-robot experiments, a relatively small number of ensemble members produced a reasonable trade-off between accuracy and computational performance. We have also extended this approach to allow high-dimensional, sparse tactile data seen in intimate physical interactions, where contact forces are relevant only in a part of the interaction and the set of active sensors may be different among demonstrations. When combined with a motion retargeting controller, we can realize a predicted motion and force across different human body shapes and sizes which was validated in a real-world user study with novel partners.

However, our results also indicate that the uncontrolled inclusion of many data

sources is not always beneficial. Some modalities may introduce spurious correlations or significant amounts of noise into the filtering process, thereby harming the accuracy of predictions. These challenges may be overcome by incorporating feature selection mechanisms, or by switching individual modalities on and off according to context.

Chapter 4

A GENERAL LIBRARY FOR INTERACTION PRIMITIVES

An argument can be made that the current popularity of AI and machine learning research is at least partly due to the availability of open-source software libraries such as Tensorflow [1] and PyTorch [83]. These tools democratize access to the development and deployment of AI models by lowering the barrier of entry. The fact that these libraries are readily available and simple to use, combined with an extensive amount of open-access tutorials, allows users to enter the field without prior experience, and more importantly, encourages them to stay by giving them the ability to quickly build and deploy working models. As such, the success of these and other related tools provide an encouraging path forward for similar fields, namely, human-robot interaction.

Human-robot interaction can be considered a subset of AI, machine learning, and robotics, and therefore already benefits from the availability of the aforementioned software libraries in addition to others such as the Robot Operating System (ROS) [87]. However, there are many unique technical challenges involved *because* of the fact

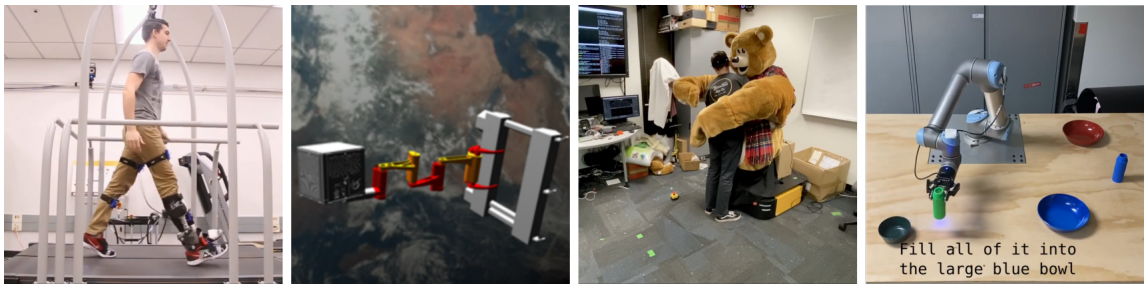


Figure 36: A selection of some of the projects the IntPrim library has been used in. From left to right: prosthetic control for ergonomic walking, in-orbit robotic assembly, multimodal hugging, and language-conditioned interactions.

that it spans these fields. For example, if we consider a typical real-world human-robot interaction scenario such as those in Fig. 36, we must be able to: a) interface with multiple sensors of different modalities and sampling frequencies, b) interface with robots of varying specifications and control requirements, c) collect and pre-process large amounts of high-dimensional data in real-time, d) perform high-frequency inference with a variety of models and algorithms, e) execute control actions based on the results of inference while ensuring safety of the human participant, and f) provide proper support for offline-offline testing, debugging, and visualization. That is not to say that any one of these challenges is unique to human-robot interaction, they are certainly not as any real-world robotics problem must also contend with several of them, but the combination of all these challenges results in a field with a rather high barrier of entry. We therefore introduce an open-source software framework that implements these capabilities and thus simplifies the process of running real-world HRI experiments, particularly with the BIP algorithms.

4.1 IntPrim: An Overview

In order to facilitate the process of deploying real-world experiments with BIP, we introduce two open-source software libraries: 1) a Python library known as *IntPrim* [21] which implements BIP, eBIP, and other standard IP-related algorithms, and 2) a Python/C++ framework, *IntPrim Framework ROS* [19] which integrates both ROS and IntPrim in order to enable experiments in a wide-variety of settings. Such a system is outlined in Fig. 37.

The IntPrim library provides optimized versions of the BIP family of algorithms and supports the following feature set: training a model from demonstration data;

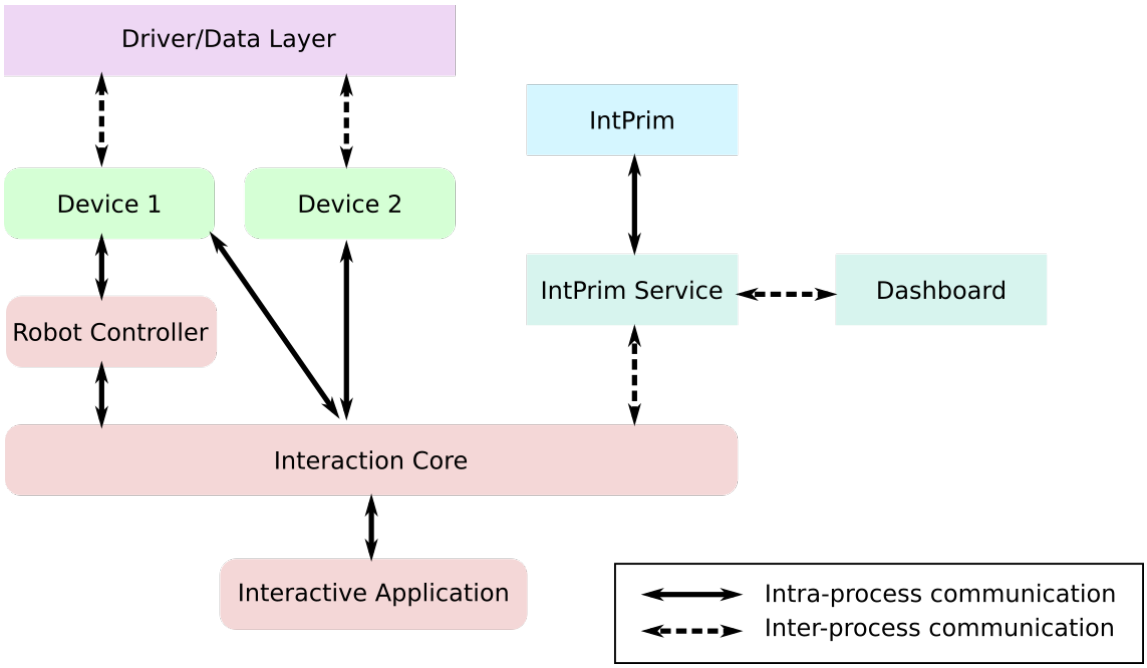


Figure 37: An overview of the IntPrim library and associated ROS framework.

performing recursive inference and generating future states; automatic basis space selection with support for Gaussian; Sigmoidal, and Polynomial basis functions; automatic computation of observation noise; and a comprehensive suite of interactive visualization and analysis tools as shown in Fig. 38. This library solves many common engineering-related challenges that are present when using BIP with real data, such as computing an appropriate basis space, but also abstracts out the algorithm from any robotics-specific tasks such that it can be applied in non-robotics settings. In addition, the library introduces several theoretical advances such as the computation of an appropriate observation noise from demonstration data and a refactorization of some of the primary equations to enable high-speed computations.

The IntPrim Framework ROS library is intended to support the usage of BIP in robotics settings. As such, it utilizes the ROS communication stack and supports the following features: automates the capture and management of training data;

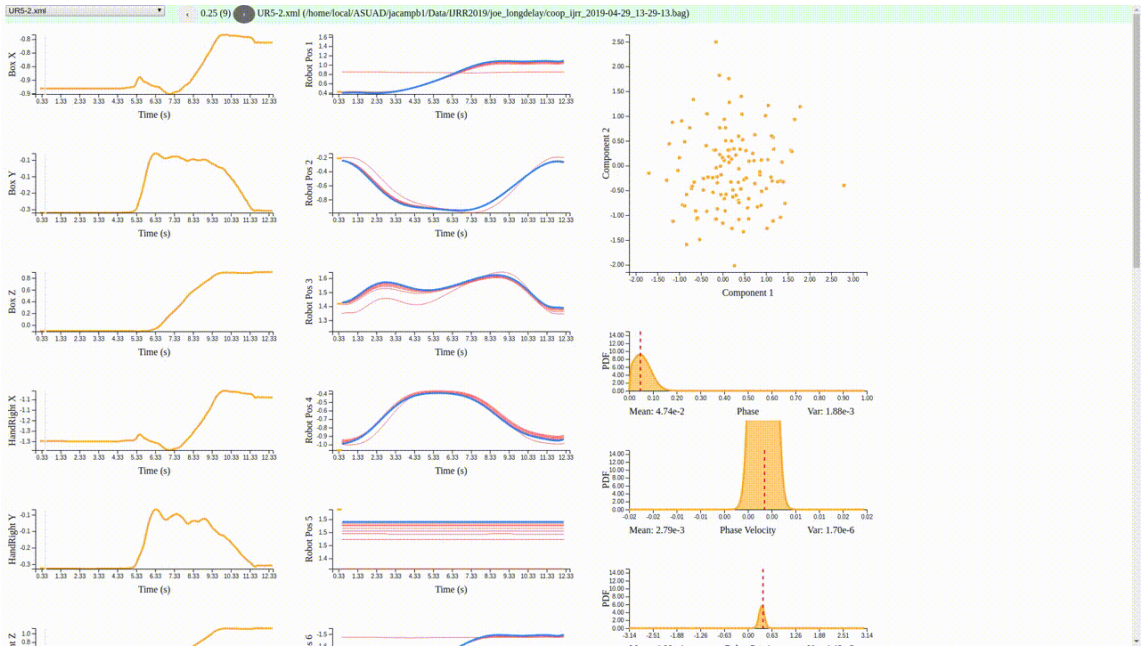


Figure 38: A selection of some of the analysis tools available in IntPrim.

synchronizes data from multiple input sources while accounting for differences in sampling frequency; automates the generation of response actions via the IntPrim library; provides an extensive mechanism for transmitting the response actions to the appropriate robot controllers; seamlessly operates over both offline (pre-recorded) and online (live) data sources; provides graphical utilities for training models, selecting hyper-parameters, and analyzing performance. As with IntPrim, this framework is intended to solve many common engineering challenges that are encountered when running real-world robotics experiments, especially human-robot interaction. Qualitatively, this framework is particularly successful in that users have been able to spin up experiments in as little as a day, as opposed to the many months of effort it would take to implement such software mechanisms from scratch. These libraries have been successfully utilized in works from [30, 31, 8, 100, 101], some of which are shown in Fig. 36.

4.2 IntPrim: In-Depth Tutorial

This section provides a more in-depth tutorial compared to Section 2, and will discuss the intuition behind the different features of the IntPrim library. One of the main strengths of this library is that very few parameters must be manually specified in order to perform inference, so we will also cover the utilities and quality-of-life features that enable this data-driven approach.

4.2.1 Basis Spaces

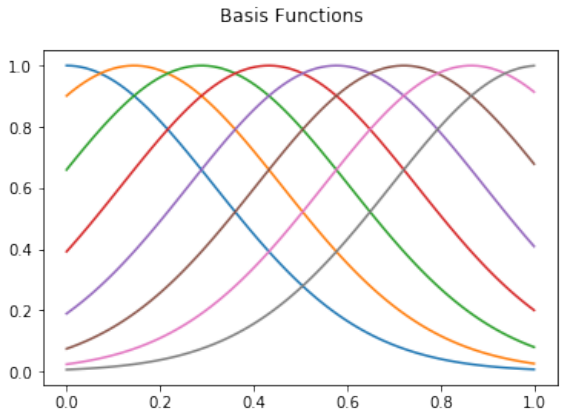
The IntPrim library natively supports three different types of basis functions:

- Gaussian radial basis functions
- Logistic sigmoid radial basis functions
- Polynomial basis functions

Example 1 Fitting a Gaussian basis model to synthetic handwriting data.

```
import matplotlib.pyplot as plt
import numpy as np

# Import the library.
import intprim

# Set a seed for reproducibility
np.random.seed(213413414)

# Define the data axis names.
dof_names = np.array(["X", "Y"])
```

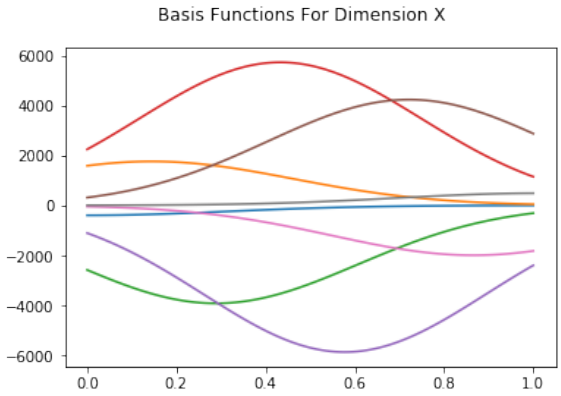
```
# Decompose the handwriting trajectories to a basis space with 8
    uniformly distributed Gaussian functions and a variance of 0.1.
basis_model_gaussian = intprim.basis.GaussianModel(8, 0.1, dof_names
    )

basis_model_gaussian.plot()
```



The Gaussian basis space can be seen in the above plot. In this case, it consists of 8 functions uniformly distributed from 0.0 to 1.0 phase with a variance of 0.1. We can fit a sample trajectory to this basis space and plot the results.

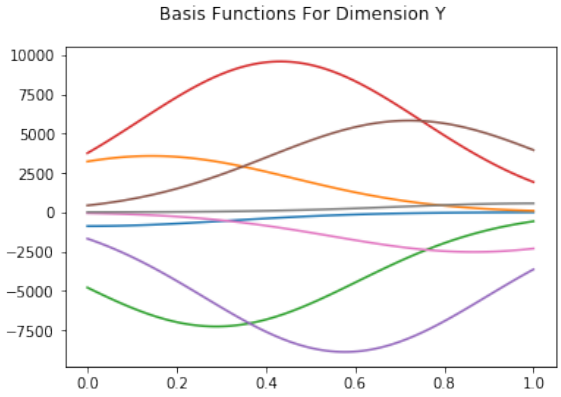
```
# Define some parameters used when generating synthetic data.
num_train_trajectories = 50
train_translation_mean = 0.0
train_translation_std = 5.0
train_noise_std = 0.01
train_length_mean = 95
train_length_std = 30

# Generate some synthetic handwriting trajectories.
training_trajectories = intprim.examples.create_2d_handwriting_data(
```

```
    num_train_trajectories,
    train_translation_mean,
    train_translation_std,
    train_noise_std,
    train_length_mean,
    train_length_std)

domain = np.linspace(0, 1, training_trajectories[0].shape[1], dtype
   = intprim.constants.DTYPE)
weights = basis_model_gaussian.
   fit_basis_functions_linear_closed_form(domain,
   training_trajectories[0].T).reshape(2, 8)
basis_model_gaussian.plot_weighted(weights, dof_names)
```

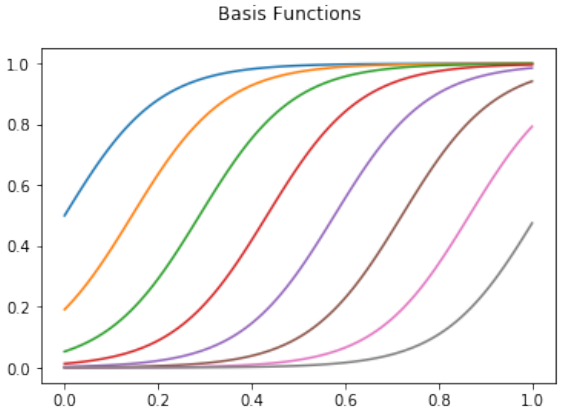


The process of fitting a basis model to a demonstration is normally handled implicitly by the `add_demonstration()` method in `BayesianInteractionPrimitive`, but we do it manually here to illustrate what is happening. The `GaussianModel` is a good all-around choice, but it's not always the most appropriate. For this reason, we also provide the `SigmoidalModel`,

```
basis_model_sigmoidal = intprim.basis.SigmoidalModel(8, 0.1,
   dof_names)
```




```
basis_model_sigmoidal.plot()
```



These models all satisfy the same API and can be used interchangeably, with the exception that PolynomialModel does not take a scale parameter.

4.2.2 Choosing an Appropriate Basis Space

With all of these basis functions and parameters to choose from, how do we choose an appropriate basis space? We provide such functionality in IntPrim through the Selection class. This performs a grid search over a set of basis functions and parameters and chooses the one with the lowest Bayesian Information Criterion as

well as Aikake Information Criterion. The grid parameters can be modified within the class depending on the use-case.

```
selection = intprim.basis.Selection(dof_names)

for trajectory in training_trajectories:
    selection.add_demonstration(trajectory)

aic, bic = selection.get_information_criteria(np.array([0, 1], dtype
    = np.int32))
selection.get_best_model(aic, bic)
```

In this case, the AIC and BIC both chose the `GaussianModel` with degree 9 and variance 0.1 as the best model. This is a very reasonable selection since the synthetic handwriting data is actually generated from a `GaussianModel` with 8 functions and 0.1 variance (see `create_2d_handwriting_data` in *examples/tutorial.py*). Note that the selection is influenced by the noise in the data as well as the model noise used by Selection when calculating the log-likelihood (by default that is set to 0.05).

4.2.3 Using Multiple Basis Spaces

Now that we can leverage multiple basis spaces and choose the best one, the question is what if we want to use a different basis space for each degree of freedom? In the above case, we performed selection over both degrees of freedom at once (by specifying np.array([0, 1]) as a parameter), but we could just as easily have performed it separately. In this case, it is likely that each DoF would be better approximated by a different basis space. IntPrim can accommodate this through the use of a MixtureModel (not to be confused with mixture models as used for density

estimation). For example, let's assume that we want to use a GaussianModel with 8 functions and 0.1 variance for X and a SigmoidalModel with 8 functions and 0.1 variance for Y.

```
basis_model_gaussian = intprim.basis.GaussianModel(8, 0.1, ["X"])
basis_model_sigmoidal = intprim.basis.SigmoidalModel(8, 0.1, ["Y"])
models = [basis_model_gaussian, basis_model_sigmoidal]


basis_model_mixture = intprim.basis.MixtureModel(models)
```

This model can now be directly passed to filters just like a regular basis model would be. Note that the order of the models in the list matters! If we reversed them and passed `[basis_model_sigmoidal, basis_model_gaussian]` then we would effctively reverse the mapping of the degrees of freedom.

4.2.4 Computing Observation Noise

There is one more aspect that we can use Selection for, and that is to calculate the observation noise from a set of training demonstrations. Functionally, we calculate the observation noise as simply the mean squared error of the regression fit of our basis spaces to the data. Intuitively, this represents the variance of the data around the regression and captures both the approximation error and the sensor noise associated with the observations.

```
observation_noise = np.diag(selection.get_model_mse(
    basis_model_mixture, np.array([0, 1])))
```

The array returned by `get_model_mse` is the diagonal of the covariance matrix for the noise, but since we assume that the errors are not correlated it is a trivial matter to expand this into a full covariance matrix. It is interesting to observe here

the impact that the basis space selection has on the observation error. The synthetic data is generated from a `GaussianModel` with 8 functions and 0.1 variance, which produces an MSE of $9e-5$ on that same `GaussianModel` for the x-axis, which is approximately equal to `train_noise_std ** 2` as we would expect. However, for the `SigmoidalModel` on the y-axis it yields a significantly larger MSE of $1e-2$. This reinforces how important it is to select a proper basis space in order to minimize approximation errors.

4.2.5 Scaling

Lastly, we will show how scaling can be used with basis models.

```
# Define a scaling group where both DoFs are scaled together.
scaling_groups_together = [
    [0, 1]
]
# Define a scaling group where the DoFs are scaled separately.
scaling_groups_separate = [
    [0],
    [1]
]

# Initialize a BIP instance.
primitive = intprim.BayesianInteractionPrimitive(basis_model_mixture
   , scaling_groups = scaling_groups_together)

# First compute the scaling
for trajectory in training_trajectories:
    primitive.compute_standardization(trajectory)
```

```
# Then add the demonstrations
for trajectory in training_trajectories:
    primitive.add_demonstration(trajectory)
```

Scaling applies a MinMax scaling such that the maximum value in the training demonstrations is scaled to 1 and the minimum value to 0. Scaling can be applied to individual DoFs or groups; it doesn't make sense to scale gyroscope and actuator measurements together, for example. This is easy to define: the scaling groups is simply a list of lists where the sub-lists specify data indices that should be scaled together as a group. No matter which grouping is used, however, the scalers must first be initialized over the entire data set before training any demonstrations! This is because the weights that are extracted when adding a demonstration are dependent on the scaling that is used, and this cannot be re-computed after the fact if the scaling changes.

Important note: if scaling is used it is very important to compute the observation error using the Selection class above. This is because scaling will significantly affect the magnitude of the observation error and will impact inference performance if scaling is turned on without adjusting the noise. This can be seen by the much smaller observation error produced below.

```
selection = intprim.basis.Selection(dof_names, scaling_groups =
   scaling_groups_together)

for trajectory in training_trajectories:
    selection.add_demonstration(trajectory)

observation_noise = np.diag(selection.get_model_mse(
   basis_model_mixture, np.array([0, 1])))
```

4.2.6 Export and Import

Once a primitive has been trained, it is a simple matter to export and import it such that training doesn't have to be repeated.

4.2.6.1 Export

Exporting can be accomplished simply by calling `export_data` and passing a file name.

```
primitive.export_data("some_file_name.bip")
```

4.2.6.2 Import

And importing is just as simple.

```
primitive.import_data("some_file_name.bip")
```

4.2.7 Filters

IntPrim supports three different spatiotemporal filters and one spatial filter. These filters affect both the accuracy and computational performance of inference. Mathematical details on how these filters function can be found in Section 4.

4.2.7.1 Spatiotemporal Filters

IntPrim supports three types of spatiotemporal nonlinear filters:

- ensemble Kalman filter as used in Ensemble Bayesian Interaction Primitives
- extended Kalman filter as used in Bayesian Interaction Primitives
- particle filter

All of these filters produce a probabilistic estimate of the state of the interaction, where we define the state as consisting of the phase (temporal state) and the weights corresponding to the basis model (spatial state). The objective is to simultaneously infer both the spatial and temporal state given a sequence of observations from an interaction of unknown length.

As with any state space-based filter, we must define a transition function which describes how we predict the state will evolve at the beginning of each time step; this occurs in the aptly named prediction step. While the basis weights in the state space are time invariant and do not have a meaningful state transition, the phase value is time varying. We assume that the phase follows a linear constant velocity model, that is, we estimate both the phase and phase velocity and assume that the velocity remains constant between any two consecutive time steps (more details on this in Section 4). The state transition matrix then is well-defined and does not need to be specifically provided to the filters, however, the process noise does need to be defined as it may vary from scenario to scenario depending on how quickly observations were collected and how confident we are that the constant velocity model accurately represents the interaction. From experience, a good rule of thumb is a process noise value of 1e-8 for observation frequencies between 30-60 Hz.

In addition, before we can initialize any filter we must determine the initial values of the state. The initial temporal state is computed from the length of the training demonstrations and for all spatiotemporal filters we model it with a Gaussian distribution, thus we need to estimate the mean and covariance parameters.

```
# Compute the phase mean and phase velocities from the
   demonstrations.
phase_velocity_mean, phase_velocity_var = intprim.examples.
   get_phase_stats(training_trajectories)
phase_mean = 0.0
phase_var = 1e-4

process_var = 1e-8

# Define the initial mean/variance of the temporal state.
initial_phase_mean = [phase_mean, phase_velocity_mean]
initial_phase_var = [phase_var, phase_velocity_var]
```

In the above code, we compute the initial mean and variance of both the phase and phase velocity that corresponds to a 1st order linear constant velocity system. We assume that there is no initial correlation between the phase and the velocity, so only the diagonal of the covariance matrix is specified in `initial_phase_var`. Note that we have assumed that the phase always begins at 0.0 with a large degree of confidence (uncertainty is $1e-4$). This does not necessarily need to be the case; for example if we're not sure whether we've observed an interaction from the beginning we could adjust the mean and increase the variance and allow the filter to attempt to correct. Furthermore, while a 1st order system is the default model, IntPrim also supports 0th order (phase only) and 2nd order (phase, velocity, acceleration) systems:

```
# Define the initial mean/variance for a 0th order system.
initial_phase_mean = [phase_mean]
initial_phase_var = [phase_var]

# Arbitrarily choose acceleration values
```

```
phase_accel_mean = 1e-4
phase_accel_var = 1e-10

# Define the initial mean/variance for a 2nd order system.
initial_phase_mean = [phase_mean, phase_velocity_mean,
   phase_accel_mean]
initial_phase_var = [phase_var, phase_velocity_var, phase_accel_var]
```

Of course, if a 2nd order system is used in practice the acceleration values should be chosen with some care; the ones used here are just for demonstration. Lastly, we can put everything together and initialize our filters.

Ensemble Bayesian Interaction Primitives (ensemble Kalman filter) There are two ways to initialize an ensemble Kalman filter. The first (and preferred) method is to initialize the ensemble directly from a set of training demonstrations.

```
# Initialize an ensemble Kalman filter
filter_enkf = intprim.filter.spatiotemporal.EnsembleKalmanFilter(
    basis_model = basis_model_mixture,
    initial_phase_mean = initial_phase_mean,
    initial_phase_var = initial_phase_var,
    proc_var = process_var,
    initial_ensemble = primitive.basis_weights)
```

Here we've directly provided the basis weights with `initial_ensemble` that will serve as the initial spatial states for the ensemble. In this method as well as the particle filter, the temporal states of the ensemble will be sampled from a Gaussian distribution with the parameters specified in `initial_phase_mean` and `initial_phase_var`.

The second method is to sample the initial ensemble from some probability distribution that approximates the true unknown distribution:

```
import sklearn.mixture
```

```
ensemble_sampler = sklearn.mixture.GaussianMixture(n_components = 2,
    reg_covar = 1e-4)
ensemble_sampler.fit(primitive.basis_weights)

sampled_ensemble = ensemble_sampler.sample(num_train_trajectories)
    [0]

# Initialize an ensemble Kalman filter
filter_enkf = intprim.filter.spatiotemporal.EnsembleKalmanFilter(
    basis_model = basis_model_mixture,
    initial_phase_mean = initial_phase_mean,
    initial_phase_var = initial_phase_var,
    proc_var = 1e-8,
    initial_ensemble = sampled_ensemble)
```

In general, the first method where the demonstrations are used directly results in increased inference accuracy since it does not accumulate any density approximation errors over the prior. However, if the number of available training demonstrations is less than the number of required ensemble members, then the second option may be employed.

Bayesian Interaction Primitives (extended Kalman filter) Unlike the ensemble Kalman filter, the extended Kalman filter as used in Bayesian Interaction Primitives does not use Monte Carlo approximation and instead computes the state posterior probability distribution analytically. As a result, the initialization procedure differs from the ensemble Kalman filter in that we compute the sample mean and covariance of the training demonstrations and pass those, rather than the initial ensemble directly.

```
# Calculate sample mean and covariance of demonstrations
mean, cov = primitive.get_basis_weight_parameters()

# Initialize an extended Kalman filter
filter_ekf = intprim.filter.spatiotemporal.ExtendedKalmanFilter(
    basis_model = basis_model_mixture,
    initial_phase_mean = initial_phase_mean,
    initial_phase_var = initial_phase_var,
    proc_var = 1e-8,
    mean_basis_weights = mean,
    cov_basis_weights = cov)
```

Particle filter The particle filter is another ensemble-based nonlinear filter, much like the ensemble Kalman filter, and subsequently is initialized in a very similar fashion. The only additional parameter is the introduction of a ratio which is used as a threshold to trigger the resampling strategy. If the number of effective samples falls below this ratio, then resampling is performed.

```
# Initialize a particle filter
filter_pf = intprim.filter.spatiotemporal.ParticleFilter(
    basis_model = basis_model_mixture,
    initial_phase_mean = initial_phase_mean,
    initial_phase_var = initial_phase_var,
    proc_var = process_var,
    initial_ensemble = primitive.basis_weights,
    num_effective_ratio = 0.5)
```

4.2.7.2 Spatial Filters

All of these filters produce a probabilistic estimate of the state of the interaction, where we define the state as consisting of the phase (temporal state) and the weights corresponding to the basis model (spatial state). The objective is to simultaneously infer both the spatial and temporal state given a sequence of observations from an interaction of unknown length.

In contrast to the spatiotemporal filters, the spatial filters (of which there is currently only one) produce a probabilistic state estimate which consists of only the basis model – the spatial state. The temporal state is inferred separately from the filter through an explicit time alignment algorithm such as Dynamic Time Warping.

Probabilistic Movement Primitives (Kalman filter)

```
# Initialize a Kalman filter
filter_kf = intprim.filter.KalmanFilter(
    basis_model = basis_model_mixture,
    mean_basis_weights = mean,
    cov_basis_weights = cov,
    align_func = intprim.filter.align.dtw.fastdtw,
    iterative_alignment = False)
```

The only additional parameters we need to worry about in this case are `align_func` and `iterative_alignment`. Currently, only one alignment function is included natively – Dynamic Time Warping – but others can easily be utilized by writing a custom function which returns the index corresponding to the currently estimated phase. The `iterative_alignment` flag indicates whether alignment should be performed over the entire trajectory each time, meaning on every iteration it aligns from phase 0 to phase 1; or whether alignment is performed recursively from last iteration, meaning

it aligns from the currently estimated phase to phase 1. In practice, we have found using iterative alignment introduces a significant amount of accumulated error over many time steps and so we do not recommend setting this to True.

4.2.7.3 Cyclical Filtering

One option that is common to (currently only spatiotemporal) filters is the ability to perform cyclical filtering. This is useful for performing inference with periodic or rhythmic interactions where we want the phase of the interaction to "roll over" back to 0 once it reaches 1, thus resetting the temporal progress and allowing the interaction to continue indefinitely. To perform cyclical filtering simply set the cyclical flag to True; an example is shown below.

```
# Initialize an ensemble Kalman filter
filter_enkf = intprim.filter.spatiotemporal.EnsembleKalmanFilter(
    basis_model = basis_model_mixture,
    initial_phase_mean = initial_phase_mean,
    initial_phase_var = initial_phase_var,
    proc_var = process_var,
    initial_ensemble = primitive.basis_weights,
    cyclical = True)
```

4.2.8 Inference

We will now examine some of the different ways in which we can perform inference over a sequence of observations. The example given in Section 2 is indicative of a general method in which we want to infer the trajectory from the currently estimated

phase to the end of the interaction. We re-state it here using everything we've introduced so far in this tutorial:

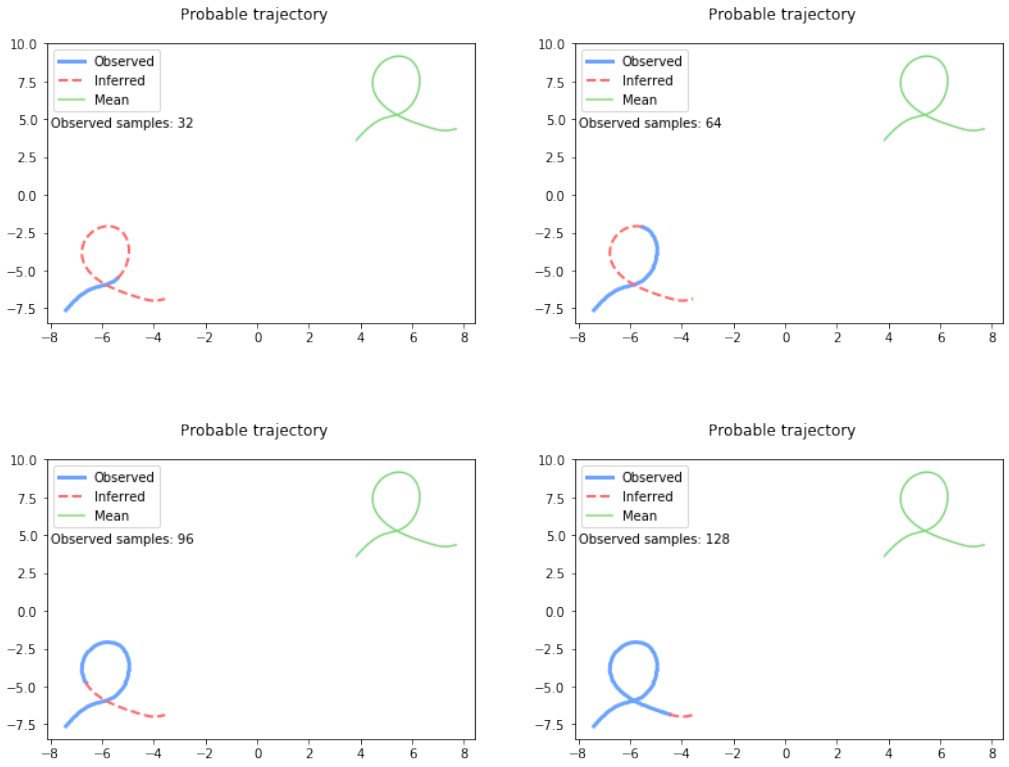
```
import copy

################  Initialize the BIP instance ################

# Define a scaling group where both DoFs are scaled together.
scaling_groups_together = [
    [0, 1]
]

# Use the model that we found during selection earlier.
basis_model_gaussian = intprim.basis.GaussianModel(9, 0.1, dof_names
    )

# Initialize a BIP instance.
primitive = intprim.BayesianInteractionPrimitive(
    basis_model_gaussian, scaling_groups = scaling_groups_together)

# First compute the scaling.
for trajectory in training_trajectories:
    primitive.compute_standardization(trajectory)

# Then add the demonstrations.
for trajectory in training_trajectories:
    primitive.add_demonstration(trajectory)

################  Determine the observation noise ################
```

```
selection = intprim.basis.Selection(dof_names, scaling_groups =
    scaling_groups_together)

for trajectory in training_trajectories:
    selection.add_demonstration(trajectory)

# Get the observation noise. Note that it is much smaller than
    train_noise_std^2 because we have scaled the values between [0,
    1].
observation_noise = np.diag(selection.get_model_mse(
    basis_model_gaussian, np.array([0, 1])))

################ Create a test trajectory ################

num_test_trajectories = 1
test_translation_mean = -10.0
test_translation_std = 1e-5
test_noise_std = 0.01
test_length_mean = 145
test_length_std = 1e-5

# Create test trajectories.
test_trajectories = intprim.examples.create_2d_handwriting_data(
    num_test_trajectories, test_translation_mean,
    test_translation_std, test_noise_std, test_length_mean,
    test_length_std)

# Explicitly zero out the x-axis values to illustrate that they are
    not being used.
```

```
test_trajectory_partial = np.array(test_trajectories[0], copy = True
    )
test_trajectory_partial[0, :] = 0.0

# We must make sure to inflate the corresponding observation noise
    entry as well so we ignore the zero values.
observation_noise[0, 0] = 10000.0

# Define the active DoF to only be the y-axis.
active_dofs = np.array([1])

################ Create a filter for inference ################

# Compute the phase mean and phase velocities from the
    demonstrations.
phase_velocity_mean, phase_velocity_var = intprim.examples.
    get_phase_stats(training_trajectories)
phase_mean = 0.0
phase_var = 1e-4

process_var = 1e-8

# Define the initial mean/variance of the temporal state.
initial_phase_mean = np.array([phase_mean, phase_velocity_mean])
initial_phase_var = np.array([phase_var, phase_velocity_var])

# Initialize an ensemble Kalman filter.
filter_enkf = intprim.filter.spatiotemporal.EnsembleKalmanFilter(
    basis_model = basis_model_gaussian,
    initial_phase_mean = initial_phase_mean,
```

```
    initial_phase_var = initial_phase_var,
    proc_var = process_var,
    initial_ensemble = primitive.basis_weights)

######## Initialize the BIP instance for inference ########

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

######## Perform inference and plot the results ########

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(32, test_trajectory_partial.shape[1],
   32):
    # Perform inference over the test trajectory in batches of 32
   samples, just so we can clearly plot how the inference changes
   over time.
    # In practice, you can pass as many or as few observations at a
   time as desired.
    inferred_trajectory, phase, mean, var = primitive.
   generate_probable_trajectory_recursive(
        test_trajectory_partial[:, prev_observed_index:
   observed_index],
        observation_noise,
```

```
        active_dofs,
        num_samples = test_trajectory_partial.shape[1] -
observed_index)

 prev_observed_index = observed_index

 # Plot the results as we go.
 intprim.util.visualization.plot_partial_trajectory(
inferred_trajectory, test_trajectories[0][:, :observed_index],
mean_trajectory)
```



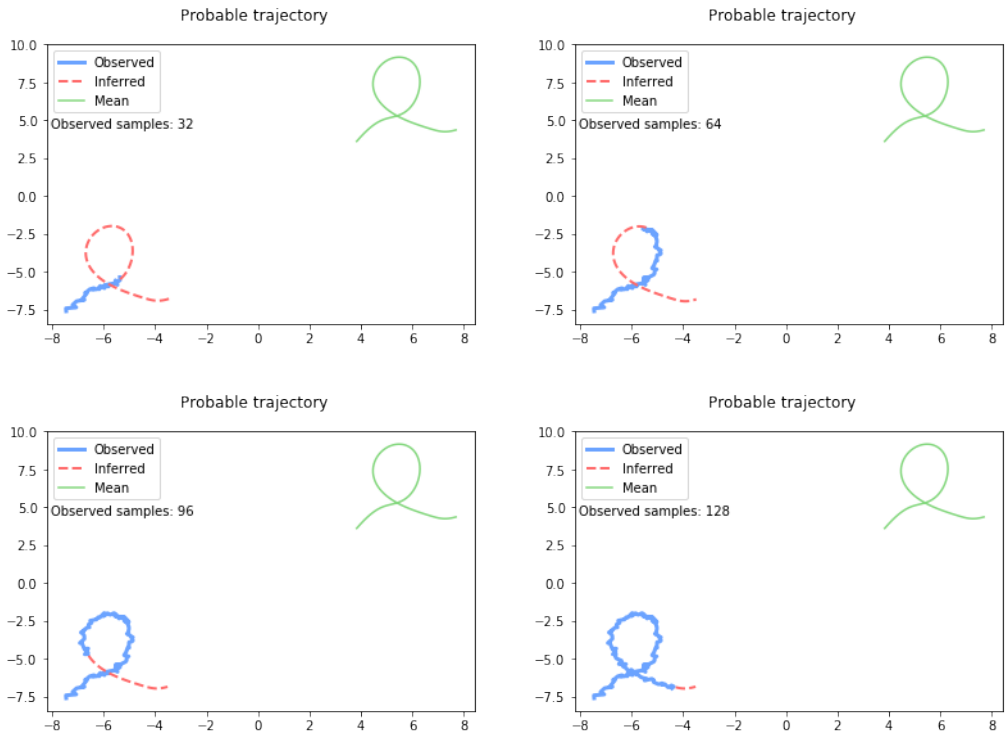
As we can see above, the filter does an exceptional job tracking both the spatial and temporal states, despite the large translation and increased length of the test sequence. This system is fairly robust, so play around with the parameters used to generate the test trajectory and observe how it performs in different scenarios. This

remains true even if we significantly increase the noise of the test trajectory as shown below.

```
################ Create a test trajectory ################

num_test_trajectories = 1
test_translation_mean = -10.0
test_translation_std = 1e-5
test_noise_std = 0.08
test_length_mean = 145
test_length_std = 1e-5

# Create test trajectories.
test_trajectories = intprim.examples.create_2d_handwriting_data(
    num_test_trajectories, test_translation_mean,
    test_translation_std, test_noise_std, test_length_mean,
    test_length_std)

# Explicitly zero out the x-axis values to illustrate that they are
    not being used.
test_trajectory_partial = np.array(test_trajectories[0], copy = True
    )
test_trajectory_partial[0, :] = 0.0

############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))
```

```
############ Perform inference and plot the results ############

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(32, test_trajectory_partial.shape[1],
    32):
     # Perform inference over the test trajectory in batches of 32
    samples, just so we can clearly plot how the inference changes
    over time.
     # In practice, you can pass as many or as few observations at a
    time as desired.
     inferred_trajectory, phase, mean, var = primitive.
    generate_probable_trajectory_recursive(
          test_trajectory_partial[:, prev_observed_index:
    observed_index],
          observation_noise,
          active_dofs,
          num_samples = test_trajectory_partial.shape[1] -
    observed_index)

     prev_observed_index = observed_index

     # Plot the results as we go.
     intprim.util.visualization.plot_partial_trajectory(
    inferred_trajectory, test_trajectories[0][:, :observed_index],
    mean_trajectory)
```

However, there is a limit to the amount of simultaneous generalization that can



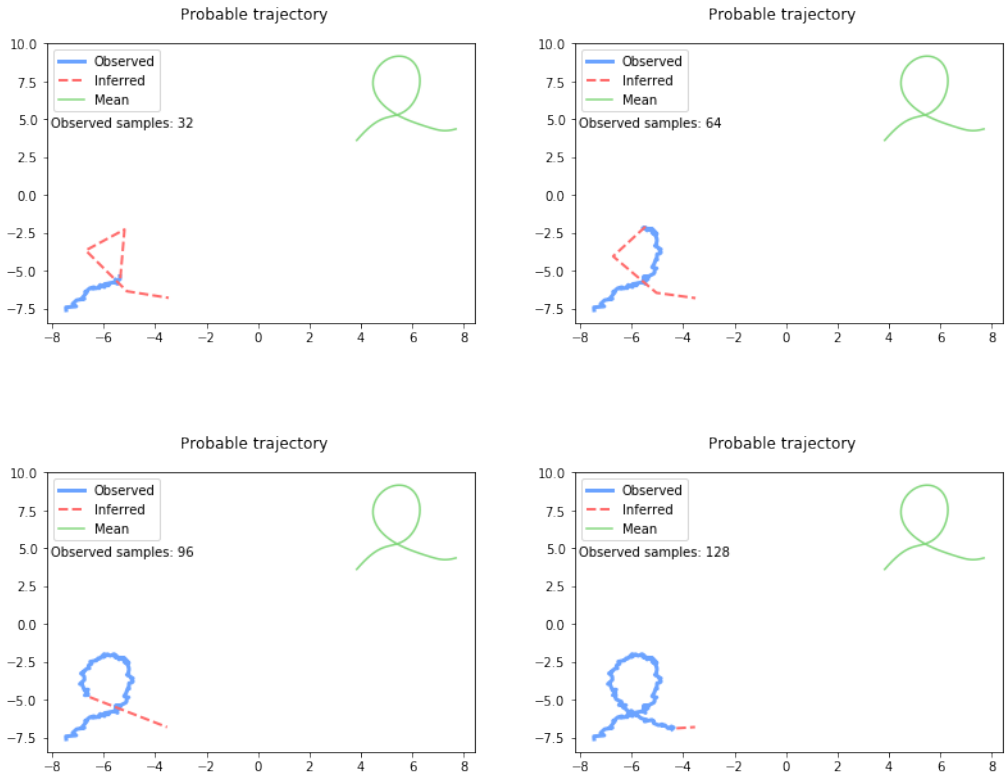
occur if the observations are especially noisy in both space and time. This is covered in more detail in Section 4, but the intuitive explanation is that if we make an observation and are very uncertain about what is happening and when it is happening, then it is difficult to adjust the state estimate because the individual error contributions from the spatial and temporal estimates cannot be determined accurately.

4.2.8.1 Inferred Trajectory Length

The length of the inferred trajectory can be adjusted through the `num_samples` parameter of `generate_probable_trajectory_recursive`. While at first glance this may seem trivial, it has a significant impact on the smoothness of the resulting trajectory. For example, if we set it arbitrarily low we end up with something like this.

```
############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

############ Perform inference and plot the results ############

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(32, test_trajectory_partial.shape[1],
   32):
    num_samples = np.max([2, (test_trajectory_partial.shape[1] -
   observed_index) / 20])
    # Perform inference over the test trajectory in batches of 32
   samples, just so we can clearly plot how the inference changes
   over time.
    # In practice, you can pass as many or as few observations at a
   time as desired.
    inferred_trajectory, phase, mean, var = primitive.
   generate_probable_trajectory_recursive(
        test_trajectory_partial[:, prev_observed_index:
   observed_index],
        observation_noise,
        active_dofs,
        num_samples = num_samples)
```

```
 prev_observed_index = observed_index

 # Plot the results as we go.
 intprim.util.visualization.plot_partial_trajectory(
inferred_trajectory, test_trajectories[0][:, :observed_index],
mean_trajectory)
```



Often, it is useful in real experiments to only predict a single value for the return trajectory. In interactions where inference is performed at a high frequency, we typically only need to execute the next inferred state and by setting `num_samples` = 1 we save on the computation cost (as small as it is) of generating the rest of the trajectory.

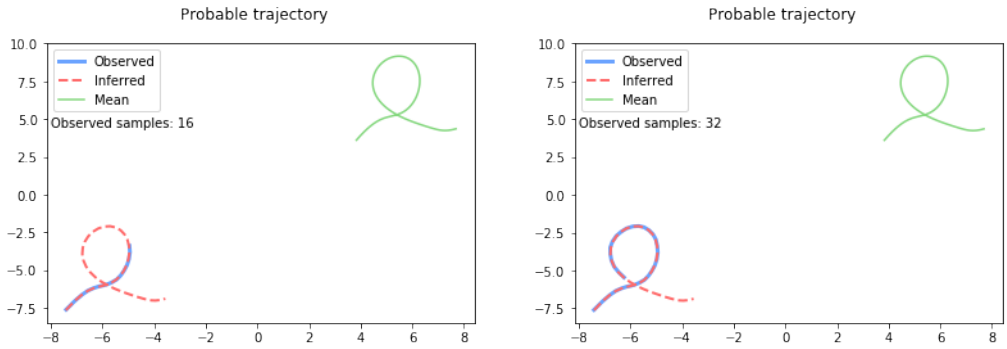
4.2.8.2 Starting Phase

Thus far, we have always inferred a trajectory which starts at the currently estimated phase. However, in some cases it is useful to estimate a trajectory (or a single state as just mentioned above) at a specific point in time. This is accomplished with the `starting_phase` parameter of `generate_probable_trajectory_recursive`. If this parameter is set to None (which is the default), then it will use the currently estimated phase as the starting point, but if we set this to a different phase value in the range [0, 1] then it will use this instead. For example, if we always want to generate a trajectory from the beginning, we can set `starting_phase` = 0.0.

```
################ Create a test trajectory ################

num_test_trajectories = 1
test_translation_mean = -10.0
test_translation_std = 1e-5
test_noise_std = 0.01
test_length_mean = 45
test_length_std = 1e-5

# Create test trajectories.
test_trajectories = intprim.examples.create_2d_handwriting_data(
    num_test_trajectories, test_translation_mean,
    test_translation_std, test_noise_std, test_length_mean,
    test_length_std)

# Explicitly zero out the x-axis values to illustrate that they are
    not being used.
```

```
test_trajectory_partial = np.array(test_trajectories[0], copy = True
    )
test_trajectory_partial[0, :] = 0.0

############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

############ Perform inference and plot the results ############

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(16, test_trajectory_partial.shape[1],
    16):
    # Perform inference over the test trajectory in batches of 16
    samples, just so we can clearly plot how the inference changes
    over time.
    # In practice, you can pass as many or as few observations at a
    time as desired.
    inferred_trajectory, phase, mean, var = primitive.
    generate_probable_trajectory_recursive(
        test_trajectory_partial[:, prev_observed_index:
    observed_index],
        observation_noise,
```

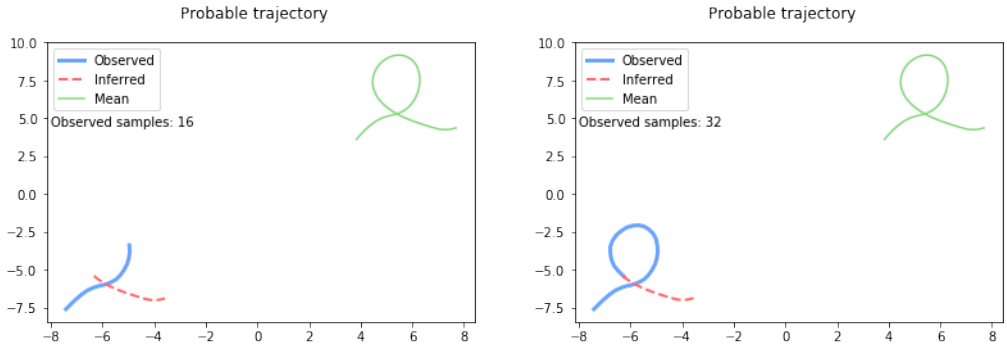
```
        active_dofs,
        num_samples = test_trajectory_partial.shape[1],
        starting_phase = 0.0)

    prev_observed_index = observed_index

    # Plot the results as we go.
    intprim.util.visualization.plot_partial_trajectory(
   inferred_trajectory, test_trajectories[0][:, :observed_index],
   mean_trajectory)
```



Or alternatively, if we want to start inferring from 70% of the way through the interaction, `starting_phase` = 0.7:

```
############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

############ Perform inference and plot the results ############
```

```
# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(16, test_trajectory_partial.shape[1],
    16):
     # Perform inference over the test trajectory in batches of 16
    samples, just so we can clearly plot how the inference changes
    over time.
     # In practice, you can pass as many or as few observations at a
    time as desired.
     inferred_trajectory, phase, mean, var = primitive.
    generate_probable_trajectory_recursive(
          test_trajectory_partial[:, prev_observed_index:
    observed_index],
          observation_noise,
          active_dofs,
          num_samples = test_trajectory_partial.shape[1],
          starting_phase = 0.7)

     prev_observed_index = observed_index

     # Plot the results as we go.
     intprim.util.visualization.plot_partial_trajectory(
    inferred_trajectory, test_trajectories[0][:, :observed_index],
    mean_trajectory)
```



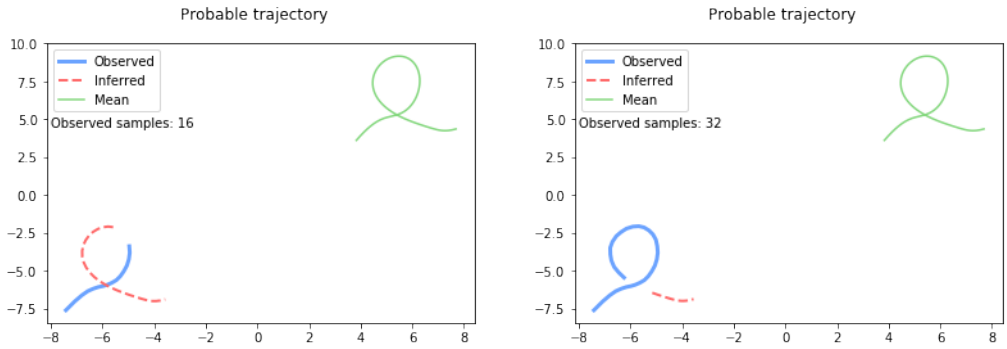
4.2.8.3 Phase Lookahead

In human-robot interaction, it is very useful to be able to infer about what will happen in the future, which is something that is provided with Bayesian Interaction Primitives and the IntPrim library. Physical experiments often have to contend with many types of delays due to real world constraints; it takes time to collect observations, perform inference, and execute a control trajectory. The end effect is that the robot often responds later than we desire, leaving it feeling "sluggish" and non-responsive to the human. With IntPrim, we provide the ability to return an inferred trajectory beginning at some phase offset into the future, which we denote as a `phase_lookahead`. In order for this to be utilized, `starting_phase` must be set to None and `phase_lookahead` needs to be specified to a value within the range of [0, 1]. For example, if we want to infer our trajectory beginning 10% into the future such that the robot is more responsive, we would perform inference with `phase_lookahead` = 0.1.

```
############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
```

```
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

############ Perform inference and plot the results ############

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory()

prev_observed_index = 0
for observed_index in range(16, test_trajectory_partial.shape[1],
    16):
     # Perform inference over the test trajectory in batches of 16
    samples, just so we can clearly plot how the inference changes
    over time.
     # In practice, you can pass as many or as few observations at a
    time as desired.
     inferred_trajectory, phase, mean, var = primitive.
    generate_probable_trajectory_recursive(
          test_trajectory_partial[:, prev_observed_index:
    observed_index],
          observation_noise,
          active_dofs,
          num_samples = test_trajectory_partial.shape[1] -
    observed_index,
          phase_lookahead = 0.1)

     prev_observed_index = observed_index

     # Plot the results as we go.
```

```
intprim.util.visualization.plot_partial_trajectory(
inferred_trajectory, test_trajectories[0][:, :observed_index],
mean_trajectory)
```



The caveat is that in more complicated scenarios inference accuracy typically has an inverse relationship with the phase lookahead. That is, the further into the future that we look the less accurate our predictions are.

4.2.9 Analysis

IntPrim supports the creation of an analysis file that exports the internal state of the filter to XML format, along with observations and inferred trajectories. This is useful from an interpretability perspective; if something does not work as intended or we want to see the exact probability distributions at a given point in time we can visualize this with the use of an analysis file and the provided visualization tool.

4.2.9.1 Exporting Analysis File

In order to export an analysis file, we use the `StatCollector` class. Upon each call to `generate_probable_trajectory_recursive()`, we will add an additional

call to the collect() method of `StatCollector` and pass it the relevant information. Upon completion of the interaction, an analysis file can be created with the `export()` method.

```
############ Initialize the BIP instance for inference ############

# Set the filter using a deep copy that way we can re-use the filter
    for inference multiple times without having to re-create it from
    scratch.
# This is simply for convenience.
primitive.set_filter(copy.deepcopy(filter_enkf))

################ Initialize StatCollector ################

# We need to specify which DoFs we want to infer trajectories for...
generated_indices = np.array([0])
# ...and which we are observing but not inferring. In this example,
   there is no difference.
observed_indices = np.array([1])

stat_collector = intprim.util.StatCollector(primitive,
   generated_indices, observed_indices)

########## Perform inference and export the analysis file ##########

# Calculate the mean trajectory for plotting.
mean_trajectory = primitive.get_mean_trajectory(num_samples =
   test_trajectory_partial.shape[1])

# Call collect once before inference starts, to capture the initial
   state of the filter.
```

```
# It is important that timestamp = None here and the observed
   trajectory is empty for visualization to properly function.
# Note that we must transpose the trajectories that are collected.
stat_collector.collect(
    primitive,
    np.array([[] for _ in range(mean_trajectory.shape[0])]),
    mean_trajectory.T,
    timestamp = None)

prev_observed_index = 0
for observed_index in range(1, test_trajectory_partial.shape[1], 1):
    # Perform inference over the test trajectory in batches of 1
   samples, just so we can clearly plot how the inference changes
   over time.
    # In practice, you can pass as many or as few observations at a
   time as desired.
    inferred_trajectory, phase, mean, var = primitive.
   generate_probable_trajectory_recursive(
        test_trajectory_partial[:, prev_observed_index:
   observed_index],
        observation_noise,
        active_dofs,
        num_samples = test_trajectory_partial.shape[1] -
   observed_index)

    # Pass stat collector the necessary information. Note that the
   timestamp will be used to sequence calls.
    # In real experimens, this is typically a clock timestamp.
    stat_collector.collect(
        primitive,
```

```
        test_trajectories[0][:, prev_observed_index:observed_index].
   T,
        inferred_trajectory.T,
        timestamp = observed_index)

    prev_observed_index = observed_index

# Export the analysis file
stat_collector.export(
    primitive,
    export_dir_path = ".", # The directory in which to save the
   analysis file
    debug_bag_file = "", # If experiment is conducted from a rosbag
   file, this can be indicated here. Discussed more in
   intprim_ros_framework.
    response_length = test_trajectory_partial.shape[1], # The
   maximum length of the inferrred trajectory
    use_spt = False, # True if num_samples has been set to 1 to only
    generate a single state
    spt_phase = None # If use_spt is True, the starting_phase.
)
```

4.2.9.2 Visualization with HTML/JavaScript

IntPrim includes a custom dynamic HTML/JavaScript script which can be run and displayed locally to visualize the information contained in analysis files. This script is named index.html and is located under the utils directory in the IntPrim root; simply run it in a web browser (Firefox and Chrome have been tested). For the

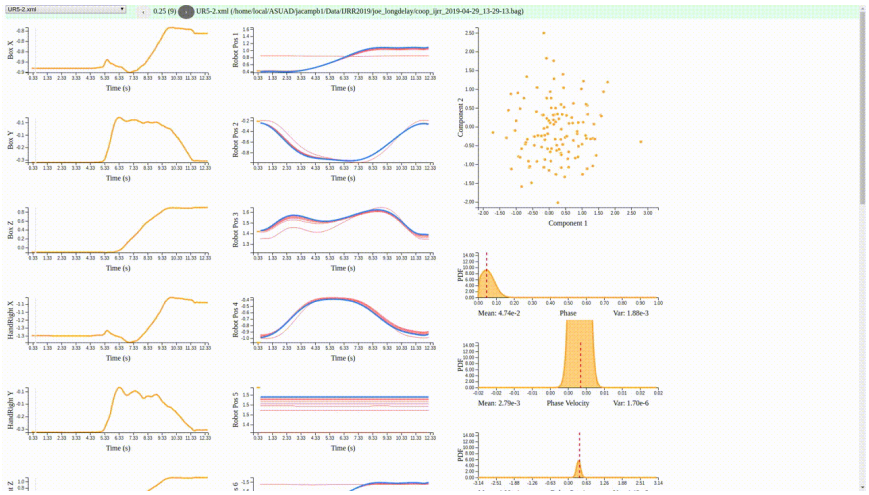
script to gain access to the analysis files, a simple Python server must be run to serve the XML files. Servers for Python 2 and Python 3 have been provided in the utils directory. Copy the appropriate file to the folder which contains the XML analysis files and run it from that directory. Then the visualization script will be able to access the analysis files from a drop down list.

Once an analysis file has been loaded, the script will prominently display three columns of data. The left column contains plots of the observed DoFs. Everytime `collect()` is called in the `StatCollector` it creates a timestep. These timesteps can be iterated through by pressing the the arrow buttons at the top of the page, using the arrow keys, or dragging the gray overlay box. A gray overlay box is displayed over the observed DoF plots to indicate which observations were given in that timestep. In the above code, observations are integrated one at a time so the corresponding box is very narrow.

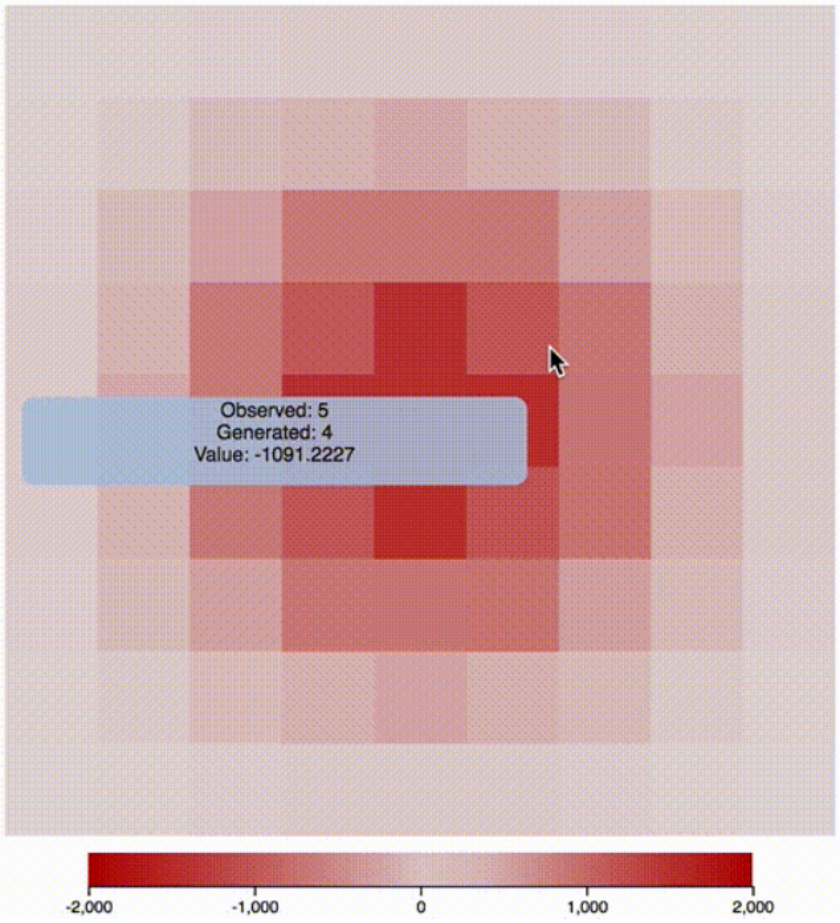
The middle column contains plots of the generated DoFs; these correspond to the DoFs that we directly want to control or infer (it is possible to generate all of the DoFs if that is desired). The orange line indicates what has been actually observed while the blue line indicates the inferred trajectory into the future. The red lines are previously inferred trajectories and are displayed for reference.

The right column contains information about the internal filter state. The top 2D plot is a projection of the ensemble to the two most significant principal components. Intuitively, this is a measure of the uncertainty in the ensemble. Ideally, this should projection should move around and collapse to a localized region, representing increased confidence in the state estimate. Below that are the PDFs for the phase system and the generated DoFs.

Lastly, at the very bottom is a visualization of the current covariance matrix. This



is useful for comparing the uncertainty between different DoFs, and it can be expanded to view the covariance of individual basis weights.


Observed: 5
Generated: 4
Value: -1091.2227
-2,000
-1,000
0
1,000
2,000

Chapter 5

CONCLUSION

In this dissertation, I have introduced a novel imitation learning method for human-robot interaction – Bayesian Interaction Primitives. While it seems natural and intuitive that human-robot interaction requires inferring correct actions in both a spatial and a temporal sense, prior works have focused primarily on only the spatial aspect while relegating temporal inference to a separate discrete step. Through Bayesian Interaction Primitives, I have shown that it is possible to jointly infer actions in a spatiotemporal sense by considering *what* the human partner is doing as well as *when* they are doing it.

The primary advantage of such an approach is that we are able to leverage the uncertainty in each sub-problem – spatial and temporal – to produce a more accurate overall inference. I have shown that this approach is conceptually related to simultaneous localization and mapping, under the conditions of known feature correspondence and a dynamic map. This is an important aspect since it enables us to leverage other techniques from a previously separate discipline in human-robot interaction.

The approach proposed in this dissertation thus provides a solution to each of the research problems identified in Chapter 1.1. In particular, I have shown that Bayesian Interaction Primitives is capable of:

1. performing joint spatiotemporal inference
2. operating without an explicit dynamics model
3. utilizing multiple sensor modalities

4. leveraging classical controllers in a hierarchical setting

That is not to say that the proposed approach is unassailable, there are definite trade-offs at play. For one, while each primitive is easy to train, the complexity is limited when compared to black box methods such as neural networks which are capable of a much higher model complexity. The fact that only errors in the first two statistical moments are modeled can exacerbate this problem, as this is again, a trade-off of computational complexity and accuracy. Additionally, while Bayesian Interaction Primitives operate without an explicit dynamics model and is a net benefit in many scenarios, it also prevents us from explicitly modeling environmental constraints such as joint limitations or other factors; there is nothing preventing us from generating physically impossible trajectories. A bigger detriment to the usage of this approach is the lack of capability for handling rich sensor input; for example, images nor natural language can be used directly, instead a form of feature extraction must first be performed to produce a latent sensor representation.

However, in this dissertation I have shown that when these trade-offs are deemed acceptable Bayesian Interaction Primitives are capable of impressive real-world interactions with human partners: cooperative object manipulation, handshaking with a musculoskeletal robot, catching a thrown object, and a whole-body haptic hug. Two of these experiments – catching and hugging – were also evaluated with novel partners and still produced successful interactions. As a result, this work has shown that Bayesian Interaction Primitives are a promising approach for complex physical human-robot interaction.

5.1 Future Research

Based on the work stemming from this dissertation, there are a few promising research directions for future work which are outlined here.

Function Approximation via Neural Networks: One of the most interesting areas for improvement is to approximate one or more of the nonlinear functions in this work with a neural network. There has been recent work in the approximation of particle filters [101] and Kalman filters with neural works, paving the way for efficient recursive Bayesian inference with better approximation capabilities. However, another potential benefit here is using the complexity advantage of neural networks to directly process and produce latent representations for high-dimensional sensor inputs, such as images or language. This obviates the necessity of employing feature extraction before passing sensor data into a primitive. Some progress has already been made in this regard [59], which demonstrates promising results.

Explicit Human Modeling: Another drawback in the simplicity of the approach proposed in this paper is that there is an explicit assumption that the human partner's mental model and the robot's mental model of the task are equivalent. Classical planning approaches [25] that can directly model the human are able to consider scenarios in which this is not the case, and also provide mechanisms for rectifying such a solution. Differing models may result in unexpected behavior, which is always a dangerous situation in physical human-robot interaction as this may lead to injury. As real-world interactions become more feasible, the capability to explicitly model humans will certainly become necessary, as assuming that the human and robot agree on mental models is a strong assumption that is likely to be violated.

REFERENCES

[1] Martin Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th USENIX symposium on operating systems design and implementation*. 2016, pp. 265–283.

[2] Baris Akgun et al. "Trajectories and Keyframes for Kinesthetic Teaching: A Human-robot Interaction Perspective". In: *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI '12. Boston, Massachusetts, USA: ACM, 2012, pp. 391–398. ISBN: 978-1-4503-1063-5. DOI: 10.1145/2157689.2157815. URL: http://doi.acm.org/10.1145/2157689.2157815.

[3] Heni Ben Amor et al. "Generalization of human grasping for multi-fingered robot hands". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 2043–2050.

[4] Heni Ben Amor et al. "Interaction primitives for human-robot cooperation tasks". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2831–2837.

[5] Heni Ben Amor et al. "Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction". In: *Annual conference on artificial intelligence*. Springer. 2009, pp. 492–499.

[6] Brenna D Argall and Aude G Billard. "A survey of tactile human–robot interactions". In: *Robotics and autonomous systems* 58.10 (2010), pp. 1159–1176.

[7] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[8] Kunal Bagewadi, Joseph Campbell, and Heni Ben Amor. "Multimodal dataset of human-robot hugging interaction". In: *arXiv preprint arXiv:1909.07471* (2019).

[9] Donald J Berndt and James Clifford. "Using dynamic time warping to find patterns in time series." In: *KDD workshop*. Vol. 10. 16. Seattle, WA. 1994, pp. 359–370.

[10] Aude Billard et al. "Robot programming by demonstration". In: *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[11] Christopher M Bishop. "Pattern Recognition and Machine Learning". In: (2006).

[12] Alexis E Block and Katherine J Kuchenbecker. "Softness, warmth, and responsiveness improve robot hugs". In: *International Journal of Social Robotics* 11.1 (2019), pp. 49–64.

[13] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. "Analysis scheme in the ensemble Kalman filter". In: *Monthly weather review* 126.6 (1998), pp. 1719–1724.

[14] Saylvain Calinon and Aude Billard. "A framework integrating statistical and social cues to teach a humanoid robot new skills". In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Workshop on Social Interaction with Intelligent Indoor Robots*. 2008.

[15] Sylvain Calinon and Aude Billard. "Incremental learning of gestures by imitation in a humanoid robot". In: *Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM. 2007, pp. 255–262.

[16] Sylvain Calinon, Florent Guenter, and Aude Billard. "On learning, representing, and generalizing a task in a humanoid robot". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.2 (2007), pp. 286–298.

[17] Sylvain Calinon et al. "Learning collaborative manipulation tasks by demonstration using a haptic interface". In: *2009 International Conference on Advanced Robotics*. IEEE. 2009, pp. 1–6.

[18] Joseph Campbell and Heni Ben Amor. "Bayesian Interaction Primitives: A SLAM Approach to Human-Robot Interaction". In: *Conference on Robot Learning*. 2017, pp. 379–387.

[19] Joseph Campbell, Michael Drolet, and Heni Ben Amor. *IntPrim Framework ROS: A ROS Framework for IntPrim*. https://github.com/ir-lab/intprim_framework_ros. 2020.

[20] Joseph Campbell, Simon Stepputtis, and Heni Ben Amor. "Probabilistic Multimodal Modeling for Human-Robot Interaction Tasks". In: *2019 Robotics: Science and Systems, RSS 2019*. 2019.

[21] Joseph Campbell, Simon Stepputtis, and Heni Ben Amor. *IntPrim: An Interaction Primitives Python Library*. https://github.com/ir-lab/intprim. 2017.

[22] Joseph Campbell and Katsu Yamane. "Learning Whole-Body Human-Robot Haptic Interaction in Social Contexts". In: *Robotics and Automation (ICRA), 2020 IEEE International Conference on*. IEEE. 2020.

[23] Joseph Campbell et al. "Learning Interactive Behaviors for Musculoskeletal Robots Using Bayesian Interaction Primitives". In: *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*. IEEE. 2019.

[24] Zhe Cao et al. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields". In: *arXiv preprint arXiv:1812.08008*. 2018.

[25] Tathagata Chakraborti et al. "Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior". In: *Proceedings of the international conference on automated planning and scheduling*. Vol. 29. 2019, pp. 86–96.

[26] L. Chen et al. "Learning human-robot collaboration insights through the integration of muscle activity in interaction motion models". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 491–496.

[27] Longxin Chen et al. "Learning human-robot collaboration insights through the integration of muscle activity in interaction motion models". In: *Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on*. IEEE. 2017, pp. 491–496.

[28] Sheng Chen, Colin FN Cowan, and Peter M Grant. "Orthogonal least squares learning algorithm for radial basis function networks". In: *IEEE Transactions on neural networks* 2.2 (1991), pp. 302–309.

[29] Ching-Ping Chou and Blake Hannaford. "Static and dynamic characteristics of McKibben pneumatic artificial muscles". In: *Proceedings of the 1994 IEEE international conference on robotics and automation*. IEEE. 1994, pp. 281–286.

[30] Geoffrey Clark, Joseph Campbell, and Heni Ben Amor. "Learning Predictive Models for Ergonomic Control of Prosthetic Devices". In: *arXiv preprint arXiv:2011.07005* (2020).

[31] Geoffrey Clark et al. "Predictive Modeling of Periodic Behavior for Human-Robot Symbiotic Walking". In: *Robotics and Automation (ICRA), 2020 IEEE International Conference on*. IEEE. 2020.

[32] Michael H. Coen. "Multimodal Integration: A Biological View". In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'01. Seattle, WA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 1417–1424.

[33] Yunduan Cui et al. "Environment-adaptive interaction primitives for human-robot motor skill learning". In: *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE. 2016, pp. 711–717.

[34] Yunduan Cui et al. "Environment-adaptive interaction primitives through visual context for human–robot motor skill learning". In: *Autonomous Robots* (2018).

[35] Oriane Dermy, François Charpillet, and Serena Ivaldi. "Multi-Modal Intention Prediction With Probabilistic Movement Primitives". In: *Human Friendly Robotics*. Springer, 2019, pp. 181–196.

[36] Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. "Legibility and predictability of robot motion". In: *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*. IEEE Press. 2013, pp. 301–308.

[37] Anca D Dragan et al. "Movement primitives via optimization". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2339–2346.

[38] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[39] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[40] Geir Evensen. "The ensemble Kalman filter: Theoretical formulation and practical implementation". In: *Ocean dynamics* 53.4 (2003), pp. 343–367.

[41] Marco Ewerton et al. "Learning multiple collaborative tasks with a mixture of Interaction Primitives". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1535–1542. DOI: 10.1109/ICRA.2015.7139393.

[42] Pedro F Felzenszwalb, Daniel P Huttenlocher, and Jon M Kleinberg. "Fast algorithms for large-state-space HMMs with applications to web usage analysis". In: *Advances in neural information processing systems*. 2004, pp. 409–416.

[43] J Randall Flanagan and David J Ostry. "Trajectories of human multi-joint arm movements: Evidence of joint level planning". In: *Experimental Robotics I*. Springer. 1990, pp. 594–613.

[44] Franka Emika GmbH. *Introducing the Franka Emika Robot*. https://www.franka.de/.

[45] J Rogelio Guadarrama-Olvera et al. "Enhancing Biped Locomotion on Unknown Terrain Using Tactile Feedback". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 1–9.

[46] Ji-Hyeong Han and Jong-Hwan Kim. "Consideration about the Application of Dynamic Time Warping to Human Hands Behavior Recognition for Human-Robot Interaction". In: *Robot Intelligence Technology and Applications 2: Results from the 2nd International Conference on Robot Intelligence Technology and Applications*. Ed. by Jong-Hwan Kim et al. Cham: Springer International Publishing, 2014, pp. 269–277.

[47] Eric L Haseltine and James B Rawlings. "Critical evaluation of extended Kalman filtering and moving-horizon estimation". In: *Industrial & engineering chemistry research* 44.8 (2005), pp. 2451–2460.

[48] Alexander Hildebrandt et al. "Cascaded control concept of a robot with two degrees of freedom driven by four artificial pneumatic muscle actuators". In: *Proceedings of the 2005, American Control Conference, 2005*. IEEE. 2005, pp. 680–685.

[49] Arne Hitzmann et al. "Anthropomorphic musculoskeletal 10 degrees-of-freedom robot arm driven by pneumatic artificial muscles". In: *Advanced Robotics* 32.15 (2018), pp. 865–878.

[50] Michael J. Hove and Jane L. Risen. "It's all in the timing: Interpersonal synchrony increases affiliation". In: *Social Cognition* 27.6 (2009), pp. 949–961. DOI: 10.1521/soco.2009.27.6.949.

[51] Geir E Hovland, Pavan Sikka, and Brenan J McCarragher. "Skill acquisition from human demonstration using a hidden markov model". In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 3. Ieee. 1996, pp. 2706–2711.

[52] Auke Jan Ijspeert et al. "Dynamical movement primitives: learning attractor models for motor behaviors". In: *Neural computation* 25.2 (2013), pp. 328–373.

[53] Shuhei Ikemoto, Yoichi Nishigori, and Koh Hosoda. "Direct teaching method for musculoskeletal robots driven by pneumatic artificial muscles". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3185–3191.

[54] InertialLabs. *Miniature and subminiature orientation sensors*. https://inertiallabs.com/3os.html.

[55] Tariq Iqbal and Laurel D Riek. "Human-robot teaming: Approaches from joint action and dynamical systems". In: *Humanoid Robotics: A Reference* (2019), pp. 2293–2312.

[56] Andrew Jansen et al. "Bio-inspired robot design considering load-bearing and kinematic ontogeny of chelonioidea sea turtles". In: *Conference on Biomimetic and Biohybrid Systems*. Springer. 2017, pp. 216–229.

[57] Mohsen Kaboli, Alex Long, and Gordon Cheng. "Humanoids learn touch modalities identification via multi-modal robotic skin and robust tactile descriptors". In: *Advanced Robotics* 29.21 (2015), pp. 1411–1425.

[58] A. Kaplish and K. Yamane. "Motion Regargeting and Control for Teleoperated Physical Human-Robot Interaction". In: *IEEE-RAS International Conference on Humanoid Robots*. 2019.

[59] Peter Karkus, David Hsu, and Wee Sun Lee. "Particle filter networks with application to visual localization". In: *Conference on Robot Learning*. PMLR. 2018, pp. 169–178.

[60] Joohyung Kim, Alexander Alspach, and Katsu Yamane. "3D printed soft skin for safe human-robot interaction". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 2419–2425.

[61] Glenn K Klute, Joseph M Czerniecki, and Blake Hannaford. "McKibben artificial muscles: pneumatic actuators with biomechanical intelligence". In: *Advanced Intelligent Mechatronics, 1999. Proceedings. 1999 IEEE/ASME International Conference on*. IEEE. 1999, pp. 221–226.

[62] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical review E* 69.6 (2004), p. 066138.

[63] Dana Kulic et al. "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 330–345.

[64] Nojun Kwak and Chong-Ho Choi. "Input feature selection by mutual information based on Parzen window". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (2002), pp. 1667–1671.

[65] W Gregory Lawson and James A Hansen. "Implications of stochastic and deterministic filters as ensemble-based data assimilation methods in varying regimes of error growth". In: *Monthly weather review* 132.8 (2004), pp. 1966–1981.

[66] Dongheui Lee and Yoshihiko Nakamura. "Mimesis Model from Partial Observations for a Humanoid Robot". In: *The International Journal of Robotics Research* 29.1 (2010), pp. 60–80.

[67] Jing Lei, Peter Bickel, and Chris Snyder. "Comparison of ensemble Kalman filters under non-Gaussianity". In: *Monthly Weather Review* 138.4 (2010), pp. 1293–1306.

[68] Kevin Sebastian Luck et al. "From the lab to the desert: Fast prototyping and learning of robot locomotion". In: *arXiv preprint arXiv:1706.01977* (2017).

[69] Christian Lundquist, Zoran Sjanic, and Fredrik Gustafsson. *Statistical Sensor Fusion: Exercises*. Sweden: Studentlitteratur AB, 2015.

[70] Guilherme Maeda et al. "Anticipative interaction primitives for human-robot collaboration". In: *2016 AAAI Fall Symposium Series*. 2016.

[71] Guilherme Maeda et al. "Learning interaction for collaborative tasks with probabilistic movement primitives". In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE. 2014, pp. 527–534.

[72] Jan Mandel. *Efficient implementation of the ensemble Kalman filter*. University of Colorado at Denver and Health Sciences Center, Center for Computational Mathematics, 2006.

[73] Robert N Miller, Michael Ghil, and Francois Gauthiez. "Advanced data assimilation in strongly nonlinear dynamical systems". In: *Journal of the atmospheric sciences* 51.8 (1994), pp. 1037–1056.

[74] Philipp Mittendorfer, Eiichi Yoshida, and Gordon Cheng. "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot". In: *Advanced Robotics* 29.1 (2015), pp. 51–67.

[75] Takahiro Miyashita et al. “Haptic communication between humans and robots”. In: *Robotics Research*. Springer, 2007, pp. 525–536.

[76] Michael Montemerlo and Sebastian Thrun. “FastSLAM 2.0”. In: *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics* (2007), pp. 63–90.

[77] Chrystopher L Nehaniv, Kerstin Dautenhahn, and Kerstin Dautenhahn. *Imitation in animals and artifacts*. MIT press, 2002.

[78] Kuniaki Noda et al. “Multimodal integration learning of robot behavior using deep neural networks”. In: *Robotics and Autonomous Systems* 62.6 (2014), pp. 721–736.

[79] Tomoyuki Noda et al. “Super-flexible skin sensors embedded on the whole body, self-organizing based on haptic interactions”. In: *Human-Robot Interaction in Social Robotics* (2012), p. 183.

[80] Toshiro Noritsugu and Toshihiro Tanaka. “Application of rubber artificial muscle manipulator as a rehabilitation robot”. In: *IEEE/ASME Transactions On Mechatronics* 2.4 (1997), pp. 259–267.

[81] Ozgur S. Oguz, Zhehua Zhou, and Dirk Wollherr. “A Hybrid Framework for Understanding and Predicting Human Reaching Motions”. In: *Frontiers in Robotics and AI* 5 (2018), p. 27.

[82] Alexandros Paraschos et al. “Probabilistic movement primitives”. In: *Advances in neural information processing systems*. 2013, pp. 2616–2624.

[83] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *arXiv preprint arXiv:1912.01703* (2019).

[84] Hanchuan Peng, Fuhui Long, and Chris Ding. “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 8 (2005), pp. 1226–1238.

[85] Luka Peternel and Jan Babič. “Learning of compliant human–robot interaction using full-body haptic interface”. In: *Advanced Robotics* 27.13 (2013), pp. 1003–1012.

[86] Jean Piaget. *The Child's Construction of Reality*. Routledge & Paul, 1955.

[87] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[88] L. R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[89] Dushyant Rao et al. “Multimodal learning and inference from visual and remotely sensed data”. In: *The International Journal of Robotics Research* 36.1 (2017), pp. 24–43.

[90] Patrick Rebeschini, Ramon Van Handel, et al. “Can local particle filters beat the curse of dimensionality?” In: *The Annals of Applied Probability* 25.5 (2015), pp. 2809–2866.

[91] Universal Robots. *UR5 Robot Arm*. https://www.universal-robots.com/products/ur5-robot/. [Online; accessed May 15, 2019].

[92] Brian C Ross. “Mutual information between discrete and continuous data sets”. In: *PloS one* 9.2 (2014), e87357.

[93] Sam Roweis and Zoubin Ghahramani. “A unifying review of linear Gaussian models”. In: *Neural computation* 11.2 (1999), pp. 305–345.

[94] Leonel Rozo et al. “Learning Controllers for Reactive and Proactive Behaviors in Human-Robot Collaboration”. In: *Frontiers in Robotics and AI* 3.30 (2016). Specialty Section Robotic Control Systems, pp. 1–11.

[95] Daniela Rus and Michael T Tolley. “Design, fabrication and control of soft robots”. In: *Nature* 521.7553 (2015), p. 467.

[96] S. Schaal. “Is imitation learning the route to humanoid robots?” In: *Trends Cogn Sci* 3.6 (1999), pp. 233–242. ISSN: 1364-6613.

[97] Stefan Schaal. “Dynamic movement primitives-a framework for motor control in humans and humanoid robotics”. In: *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[98] Masahiro Shiomi et al. “A robot that encourages self-disclosure by hug”. In: *International Conference on Social Robotics*. Springer. 2017, pp. 324–333.

[99] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. “On the non-trivial generalization of dynamic time warping to the multi-dimensional case”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 289–297.

[100] Shubham Sonawani et al. “Assistive Relative Pose Estimation for On-orbit Assembly using Convolutional Neural Networks”. In: *arXiv preprint arXiv:2001.10673* (2020).

[101] Simon Stepputtis et al. “Language-Conditioned Imitation Learning for Robot Manipulation Tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[102] Andrea Thomaz, Guy Hoffman, Maya Cakmak, et al. “Computational human-robot interaction”. In: *Foundations and Trends® in Robotics* 4.2-3 (2016), pp. 105–223.

[103] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[104] Bertrand Tondu et al. “A seven-degrees-of-freedom robot-arm driven by pneumatic artificial muscles for humanoid robots”. In: *The International Journal of Robotics Research* 24.4 (2005), pp. 257–274.

[105] Touchence Inc. *ShokacPot/ShokacCube Product Outline*. http://www.touchence.jp/en/cube/index.html.

[106] Nikolaos G Tsagarakis and Darwin G Caldwell. “Development and control of a ‘soft-actuated’exoskeleton for use in physiotherapy and training”. In: *Autonomous Robots* 15.1 (2003), pp. 21–33.

[107] Aleš Ude et al. “Orientation in cartesian space dynamic movement primitives”. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 2997–3004.

[108] Michäel Van Damme et al. “Proxy-based sliding mode control of a planar pneumatic manipulator”. In: *The International Journal of Robotics Research* 28.2 (2009), pp. 266–284.

[109] Jorge R Vergara and Pablo A Estévez. “A review of feature selection methods based on mutual information”. In: *Neural computing and applications* 24.1 (2014), pp. 175–186.

[110] D. Vogt et al. “A system for learning continuous human-robot interactions from human-human demonstrations”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2882–2889. DOI: 10.1109/ICRA.2017.7989334.

[111] Kazuyoshi Wada et al. “Effects of robot-assisted activity for elderly people and nurses at a day service center”. In: *Proceedings of the IEEE* 92.11 (2004), pp. 1780–1788.