OF PRIORS AND PARTICLES:

STRUCTURED AND DISTRIBUTED APPROACHES TO ROBOT PERCEPTION AND CONTROL

A Dissertation
Presented to
The Academic Faculty

By

Alexander Lambert

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Robotics

School of Electrical and Computer Engineering
Georgia Institute of Technology

August 2021

© Alexander Lambert 2021

OF PRIORS AND PARTICLES:

STRUCTURED AND DISTRIBUTED APPROACHES TO ROBOT PERCEPTION AND CONTROL

Dissertation committee:

Dr. Byron Boots
Paul G. Allen School of
Computer Science and Engineering
University of Washington

Dr. Matthew Gombolay
School of Interactive Computing
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Fabio Ramos
School of Computer Science
University of Sydney

Dr. Sonia Chernova
School of Interactive Computing
Georgia Institute of Technology

Date approved: May 28, 2021

Study hard what interests you the most

in the most undisciplined, irreverent and original manner possible.

Richard P. Feynman

To my parents.

ACKNOWLEDGMENTS

There are many people who have helped and supported me throughout my graduate studies. I would not have succeeded in completing this journey without them, and I must express my gratitude for the various roles they have played during my Ph.D experience.

I have to begin by thanking my advisor, Byron Boots, for his support and mentorship. I am grateful for his invaluable feedback, patience while listening to my often inchoate ideas, and help in focusing my efforts towards meaningful contributions. He has been a reliable source of technical expertise, professional guidance and inspiration. Being a part of his lab has been a privilege, and has led me to gain confidence and grow as a researcher.

I am honoured to have such exceptional committee members, and thank them for their help and encouragement during the making of this dissertation: Seth Hutchinson, Sonia Chernova, Matthew Gombolay, and Fabio Ramos. I would like to give special thanks to Fabio, in particular. Working with him has been a fun and exciting adventure, and has introduced a degree of playful creativity to my research process.

I am also grateful to a number of collaborators at NVIDIA, from whom I've learned a great deal. I'd like to thank Dieter Fox, who has been especially welcoming and generous during my internships with the SRL group, and has provided essential guidance and input. I would also like to thank Nathan Ratliff for his insights, stimulating discussions, and reminding me to check my intuitions and assumptions. I've also had the pleasure of working with Yi-Ting Chen and his team at HRI, which led to an enjoyable and educational research experience.

My transition to robotics would not have been possible without Henrik Christensen, who gave me the opportunity to start pursuing a career in the field. I am also indebted to Charles Isbell for allowing me to gain valuable teaching experience and educate others interested in machine learning. Heni Ben Amor also played a key role during his post-doc at Georgia Tech, being both a charitable friend and an encouraging mentor.

I am fortunate to have worked with a number of amazing researchers and colleagues. Thanks to Amir Shaban, Mustafa Mukadam, Bala Sundaralingam, Tucker Hermans, Lucas Barcelos, Adam Fishman, Ankur Handa, Stan Birchfield, Rafael Oliveira, Paulo Borges, Zhen Liu, and Amit Raj. Ching-An Cheng and Anqi Li have also kindly provided their feedback on a number of occasions.

I am lucky to have made a number of great friends throughout my time as a grad student. Thanks to Ashley Edwards, Himanshu Sahni, Shray Bansal, Yannick Shroecker, Steven Hickson, Kalesha Bullard and Meera Hahn for the awesome adventures outside of the lab, and making internship summers memorable. I'd also like to thank the "early-days" crew for being an indispensable source of laughs and good times when I first started Ph.D life. This includes Jon "Noodle" Scholz, Amber "Burrito" Numamoto, "Mis-tah" Martin Levihn, Michael "Misha" Novitzky, and Baris "Old Man" Akgun. There have also been many others that I've met and worked alongside of, who have been amazingly tolerable of my lame jokes, and with whom I've shared moments of levity and humor. Thank you all.

I'm grateful to all my family and friends back home. Thanks to my brothers Brandon and Bechir, to Alicia, and to my good friends Silvie, Devon, Ben, and Woody for welcoming me back with warmth and familiarity whenever I returned to visit.

Finally, I'd like to thank my parents for their love and support throughout this journey. My mother Joanna and step-dad Wally have been there for me the entire way, raising my spirits when the going got tough. My father Dave has largely been the reason I set out on this path. He laid the foundations early on, and has been an inspiration from the very beginning. This dissertation is dedicated to you three.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Applications of robotic systems have expanded significantly in their scope, moving beyond the caged predictability of industrial automation and towards more open, unstructured environments. These agents must learn to reliably perceive their surroundings, efficiently integrate new information and quickly adapt to dynamic perturbations. To accomplish this, we require solutions which can effectively incorporate prior knowledge while maintaining the generality of learned representations. These systems must also contend with uncertainty in both their perception of the world and in predicting possible future outcomes. Efficient methods for probabilistic inference are then key to realizing robust, adaptive behavior.

This thesis will first examine data-driven approaches for learning and combining perceptual models for both visual and tactile sensor modalities, common in robotics. Modern variational inference methods will then be examined in the context of online optimization and stochastic optimal control. Specifically, this thesis will contribute (1) data-driven visual and tactile perceptual models leveraging kinematic and dynamic priors, (2) a framework for joint inference with visuo-tactile sensing, (3) a family of particle-based, variational model predictive control and planning algorithms, and (4) a distributed inference scheme for online model adaptation.

CHAPTER 1
INTRODUCTION

Successful deployment of fully-autonomous, robotic systems in noisy and dynamic environments remains a challenge. These machines must contend with varying degrees of uncertainty characterized by multiple sources, including sensor measurements, modelling error, and stochastic dynamics. This can be accomplished by leveraging tools from machine learning and statistics to (1) learn generalize-able models for interpreting sensory signals, (2) efficiently integrate new information during execution, and (3) reason about current and future states in a principled and distributed way.

Although much progress has been made on developing sophisticated hardware for sensing and perception, deriving reliable analytical models to correctly extract useful information from high-dimensional measurements is often difficult and unreliable. Learning such models from experience offers a promising alternative for both improving accuracy and scalability. Recently, large-scale data-driven approaches have become increasingly popular for training large parametric models which can interpret raw, high-dimensional sensory measurements to actionable latent spaces. These have demonstrated improved generalization over purely model-based and engineered methods, minimizing the need for expert intervention and feature construction [1, 2, 3, 4]. A significant trend in this area has been to learn 'end-to-end' function approximators which map observations directly to system control commands [5, 6, 7, 8]. However, these methods use a large number of parameters, and consequently require vast amounts of training data. This can be expensive to obtain on real systems, making it prohibitive for many applications. Further still, naïve approaches often lack a sufficiently interpretable intermediate representation [9, 10]. Incorporating structure can be crucial for ensuring safety-critical intervention in the sense-to-act pipeline, and make engineering of complex autonomous systems far more tenable. This may even be

necessary to effectively scale learning and improve performance, in general [11, 12].

In order to make learning and inference for robotics more data-efficient and scalable, we should strive to integrate domain knowledge into our representations when possible, with careful consideration of inductive biases [10]. Luckily, robotics problems often exhibit rich *structure* and known dependencies which can be exploited in a variety of ways, particularly when learning predictive models for inference. Such examples include system appearance and geometry [13, 14], kinematics [15], and physics-based priors [16, 17, 18]. To learn effective sensory mappings, perceptual models should be designed to efficiently leverage such problem structure. Furthermore, as data can be arduous and expensive to acquire for real robot systems, data collection should be carefully targeted to capture essential modes of the target state distribution, with adequate coverage of critical corner cases [19, 20].

Bayesian inference is a fundamental statistical method that has generated many practical and theoretically-principled tools for probabilistic reasoning. It has been pivotal in the field of robotics, engineering, and related disciplines [21, 22, 23], providing a framework for parameter- and state-estimation which naturally incorporates uncertainty in modelling, measurement and prior information. Recently, developments in variational inference have led to a number of sample-efficient methods for approximating complex, multi-modal probability distributions. A subset of these techniques maintain an empirical representation as a system of interacting particles, with deterministic updates computed efficiently in batch. These representations are *distributed*, in that they consist of a collection of unique parameters which require local evaluation, but interact in a de-centralized way. This makes these approaches particularly amenable to parallel computation, yet their application has been largely limited to toy examples and simple offline datasets. With the growth in availability of on-board GPUs, effort should be driven towards developing online inference and control algorithms with parallelization in mind. Doing so will allow for the practical adoption of powerful techniques from the statistics community, and provide efficient means for handling high-dimensional state and control distributions common in robotics.

*Data-driven perceptual models for both predicting and inferring sensory observations can achieve robust performance from sparsely collected data by leveraging known priors. This can be accomplished by integrating system **structure** directly into representations used for learning and inference. Predicting distributions of current and future states for control and estimation can be scaled effectively by leveraging **distributed** probabilistic representations.*

In this thesis, I present novel methods for incorporating structure for learning and combining sensory models in robotics, followed by distributed inference methods for stochastic optimal control and parameter estimation.

Beginning with **Part I**, the problem of embedding structure in perceptual models for robot manipulation tasks is addressed for both visual and tactile sensing. **Chapter 2** describes a combined parameteric/non-parameteric learning representation which leverages key-frame data for improved image-based prediction, and demonstrates the use of generative models to infer system states for tracking tasks and detecting occlusions [14]. In **Chapter 3**, we propose a tactile perceptual model with integrated spatial structure for robust force prediction, and outline a data collection strategy to improve cross-domain learning [24].

Part II covers the role of incorporating structure via model-based priors for state estimation with multi-sensory observations. In **Chapter 4**, I describe a framework for efficient, online probabilistic inference for robot manipulation, combining both tactile and visual perceptual models [25]. The use of geometric and physics-based priors is shown to be a vital component in cross-modal compensation, particularly when dealing with heavy occlusion. Pose estimates from visual tracking systems can be enhanced by using contact force measurements and, conversely, visual information can be used to alleviate biases in tactile estimation.

In **Part III**, I examine how a modern particle variational inference method, known as Stein Variational Gradient Descent (SVGD) [26], can be adapted to solve control and esti-

mation problems for common robotics tasks, such as manipulation and navigation. **Chapter 5** presents a new class of control algorithms leveraging SVGD and parallel computation. This work is the first of its kind, providing the earliest known examples of Stein's Method [27, 28] being used for stochastic optimal control and motion planning [29]. By formulating MPC as an inference problem, and using appropriately factored kernels, favorable performance is shown for short and long planning horizons, mitigating the occurrence of poor local optima in the control solution. In **Chapter 6**, we extend this approach to include online estimation of system parameters [30]. Effective adaptation of these parameters is shown to improve performance of the MPC algorithm in a dynamically-changing environment, and tested on a real autonomous ground vehicle.

Part I

Structure in Learned Perceptual Models

CHAPTER 2
FLOW-BASED VISUAL PREDICTION AND TRACKING

2.1 Introduction

Several fundamental problems in robotics, including state estimation, prediction, and motion planning rely on accurate models that can map state to measurements (forward models) or measurements to state (inverse models). Classic examples include the measurement models for global positioning systems, inertial measurement units, or beam sensors that are frequently used in simultaneous localization and mapping [31], or the forward and inverse kinematic models that map joint configurations to workspace and *vice-versa*. Some of these models can be very difficult to derive analytically, and, in these cases, roboticists have often resorted to machine learning to infer accurate models directly from data. For example, complex nonlinear forward kinematics have been modeled with techniques as diverse as Bayesian networks [32] and Bezier Splines [33], and many researchers have tackled the problem of learning inverse kinematics with nonparametric methods like locally weighted projection regression (LWPR) [34, 35], mixtures of experts [36], and Gaussian Process Regression [37]. While these techniques are able to learn accurate models, they rely heavily on prior knowledge about the kinematic relationship between the robot state-space and work-space.

Despite the important role that forward and inverse models have played in robotics, there has been little progress in defining these models for very high-dimensional sensor data like images and video. This has been disappointing: cameras are a cheap, reliable source of information about the robot and its environment, but the precise relationship between a robot pose or configuration, the environment, and the generated image is extremely complex. A possible solution to this problem is to *learn* a forward model that directly maps

the robot pose or configuration to high-dimensional perceptual space or an inverse model that maps new images to the robot pose or configuration. Given these models, one can directly and accurately solve a wide range of common robotics problems including recursive state estimation, sequential prediction, and motion planning.

In the following chapter, we explore the idea of directly learning forward and inverse perceptual models that relate high-dimensional images and low-dimensional robot state. Specifically, we use deep neural networks to learn both forward and inverse perceptual models for a camera pointed at the manipulation space of a Barret WAM arm. While recent work on convolutional neural networks (CNNs) provides a fairly straightforward framework for learning inverse models that can map images to robot configurations, learning accurate generative (forward) models remains a challenge.

2.2 Related Work

Predictive models for visual data, such as image frames or video, define a mapping from a latent space to pixel-level observations. They have been used most recently for learning unsupervised visuo-motor policies [38, 13], visual task planning [39], and model-predictive control [4, 40, 41]. This has demonstrated the utility of defining desired *visual* states and trajectories directly in observation space, for both manipulation and navigation tasks.

Generative neural networks have recently shown much promise in addressing the problem of mapping low-dimensional encodings to a high-dimensional pixel-space [42, 43, 44, 45]. The generative capacity of these approaches is heavily dependent on learning a strictly parametric model to map input vectors to images. Using deconvolutional networks for learning controllable, kinematic transformations of objects has previously been demonstrated, as in [46, 47]. However, these models have difficulty reproducing clear images with matching textures, and have mainly been investigated on affine transformations of simulated objects.

Learning to predict frames has also been conducted on two-dimensional robot manipu-

lation tasks. Finn et al. [38] propose an LSTM-based network to predict next-frame images, given the current frame and state-action pair. In order to model pixel transformations, the authors make use of composited convolutions with either unconstrained or affine kernels. The generated image frames appear to reproduce linear motion in the scene, but also appear to have difficulty replicating multi-degree-of-freedom dynamics. Given that forward prediction is conducted by recursive input of predicted frames, error compounds during sequential prediction and prediction quality quickly degrades over future timesteps.

An alternative approach to generating images directly, after applying transformations in a low-dimensional encoding, is to learn a transformation in the high-dimensional output space. One can then re-use pixel information from observed images to reconstruct new views from the same scene. This has been proposed in previous studies [48, 49]. The authors in [49], for instance, learn a model to generate a flow-field transformation from an input image-pose pair derived from synthetic data. This is subsequently applied to a reference frame, effectively rotating the original image to a previously unseen viewpoint. Using confidence masks to combine multiple flow-fields generated from different reference frames is also proposed. However, these frames are selected randomly from the training data.

In this Chapter, we present a method for predicting photo-realistic observations in robot manipulation by leveraging a key fact: the geometry and kinematics of the system is effectively constant, and the configuration space is a well-defined closed set. By collecting key-frame data of different robot poses, a flow-based transformation can be learned to generate novel viewpoints from nearest-neighbour images. This can then be used for visual prediction of desired joint-space trajectories, with the added benefit of detecting occlusions in the task space.

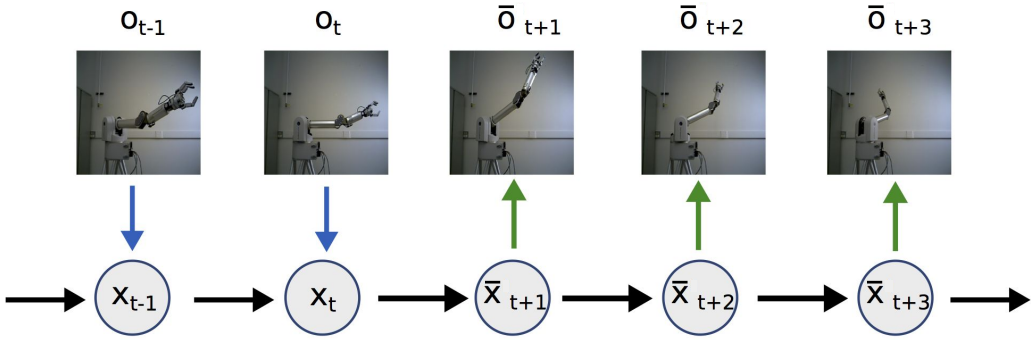


Figure 2.1: A conceptual diagram of the proposed framework, makes use of learned visual sensor models to infer states x_i from observations o_i, and predict observations from future states [14].

2.3 Model Design

Tracking and prediction are important tasks in robotics. Consider the problem of tracking state and predicting future images, illustrated in Fig. 2.1. Given a history of $O_t = (\mathbf{o}_i)_{i=0}^{t}$ where $\mathbf{o} \in \mathbb{R}^m$, we wish to track the state of the system $\mathbf{x} \in \mathbb{R}^n$ for the corresponding sequence $X_t = (\mathbf{x}_i)_{i=0}^{t}$. This can be accomplished using an *inverse model* that maps images to state. Conversely, given a sequence of expected states $\overline{X}_t = (\overline{\mathbf{x}}_i)_{i=t+1}^{T}$ (which may be obtained from planner, for instance), each state $\bar{\mathbf{x}}_i$ can be mapped to a corresponding observation prediction $\bar{\mathbf{o}}_i$ by a *forward model*.

The tasks of mapping the configuration-space joint values to (observed) robot poses, and *vice-versa*, is similar to the problem of forward/inverse kinematics for modeling robot manipulator mechanics. Traditionally, such mechanical models are integrated with visual information using separately-constructed image-projection models, for techniques such as Position-based Visual Servoing [50, 51]. In the present context, however, the kinematics and projection relations are represented jointly by the learned forward/inverse models.

If we assume that (1) the configuration space of the robot is fixed (kinematics do not change after training time) and (2) the camera intrinsic/extrinsic parameters are also constant (ex. a head-mounted camera on a humanoid robot), then the 3D projection of any un-occluded robot pose will lie within some fixed set. Given this assumption of constant

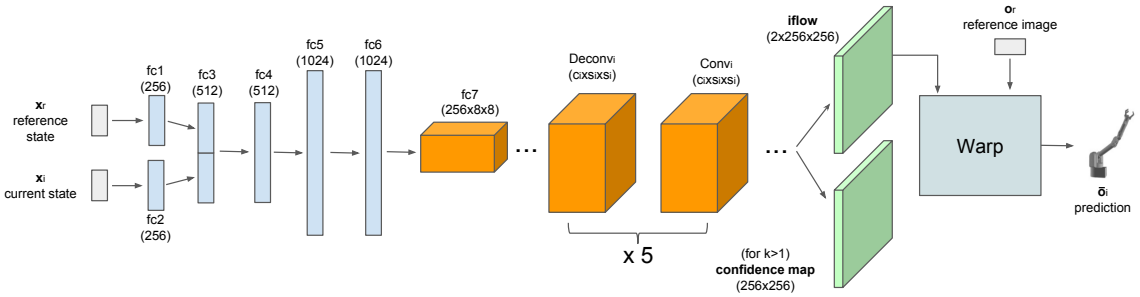


Figure 2.2: A depiction of a single parametric branch used in the forward model with flow-field output (i-Flow).

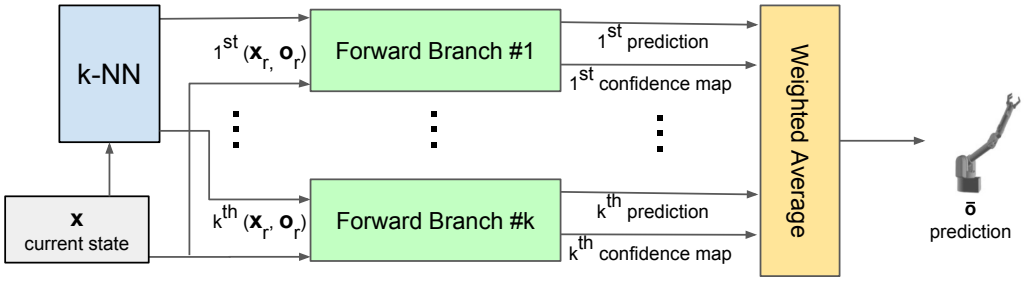


Figure 2.3: The complete k-NN-FLOW model architecture. k-Nearest neighbor (image, state)-pairs selected by the k-NN module, and passed to individual parametric branches (having shared weights). Resulting warped images are weighed by their corresponding confidence maps and summed together to produce final output image [14].

geometry in many robotics problems, we can leverage experienced key-point data to learn *nonparametric* models describing the system without explicitly defining the kinematic relationships *a priori*.

2.3.1 A Forward Sensor Model

We use a generative observation model g to perform a mapping from joint-space values $\mathbf{x}_i$ to pixel-space values $\mathbf{o}_i$ for a trajectory sequence of future states $\overline{X}_t$. Instead of training a neural network to predict the RGB values completely from scratch, our network is comprised of two parts: 1) a *non-parametric* component, where given a state value $\mathbf{x}_i$, a reference pair $(\mathbf{x}_r, \mathbf{o}_r)$ is found such that $\mathbf{o}_r$ has a similar appearance as the output image $\mathbf{o}_i$; and 2) a parametric component, where given the same state value $\mathbf{x}_i$ and the reference state-image pair $(\mathbf{x}_r, \mathbf{o}_r)$, we learn to predict the pixel flow field $\overrightarrow{h}(\mathbf{x}_i, \mathbf{x}_r)$ from reference

image $\mathbf{o}_r$ to the image $\mathbf{o}_i$ (using image-warping layers similar to [49]).

The final prediction is made by warping the reference image $\mathbf{o}_r$ with the predicted flow field $\overrightarrow{h}$: $g(\mathbf{x}_i) = \text{warp}(\mathbf{o}_r, \overrightarrow{h})$. As long as there is a high correlation between visual appearance of the reference image $\mathbf{o}_r$ and the output image $\mathbf{o}_i$, warping the reference image results in much higher-fidelity predictions when compared to models which map input directly to RGB values [47]. The overall architecture is shown in Figs. 2.2 and 2.3.

We use a nearest neighbor algorithm in the configuration space to find $\mathbf{o}_r$. More precisely, we take a subset of the training set $\mathcal{F}_p \subseteq \mathcal{F}$, consisting of configuration-image pairs and store the data in a KD-tree [52]. We can quickly find a reference pair by searching for the training data pair closest to the current joint configuration $\mathbf{x}_i$ in configuration space. Since the state-space is low-dimensional we can find this nearest reference efficiently. In practice, the output of Nearest Neighbor is limited to the elements inside the training set and cannot generalize to unseen data very well. To address this problem, we train a network to warp the predicted $\mathbf{o}_r$ to the target observation $\mathbf{o}_i$. A diagram of this parametric model is shown in Fig. 2.2. Given a current and reference state vector as input, the network is trained to produce a flow tensor to warp the reference image [49].

Intuitively, the Nearest Neighbor component (Fig. 2.3) provides the RGB values for a nearby joint configuration, and the neural network learns to move the pixels to produce an image which corresponds to the target joint vector. The input is first mapped to a high dimensional space using six fully-connected layers. The resulting vector is reshaped into a tensor with size $256 \times 8 \times 8$. The spatial resolution is increased from 8×8 to $2^8 \times 2^8$ by using 5 deconvolution layers. Following the deconvolutions with convolutional layers was found to qualitatively improve the output images. Finally, we use the predicted flow field of size $2 \times 256 \times 256$ to warp the reference image. To allow end-to-end training, we used a differentiable image sampling method with a bi-linear interpolation kernel in the warping layer [53]. To prevent over-fitting, we randomly sample one of 10-nearest neighbors in the training phase. Using L_2 prediction loss, the final optimization function is defined as

follows:

$$\min_{W} L(W) = \sum_{i} ||\mathbf{o}_i - \text{warp}(\mathbf{o}_r, \overrightarrow{h}_W(\mathbf{x}_i, \mathbf{x}_r))||^2 + \lambda||W||^2 \tag{2.1}$$

in which W contains the network parameters and λ is the regularization coefficient. Training on the forward model was performed end-to-end using the Caffe library [54], which we extended to support the warping layer. ADAM [55] was used to optimize the network.

2.3.2 Forward Model with k-Nearest Neighbors

Using a reference image works well if all visible portions of the arm in $\mathbf{o}_i$ are *also* visible in the reference image $\mathbf{o}_r$ [49]. However, portions of the arm may be visible in some reference images and not others. As such, we could improve the prediction fidelity by warping more than one reference image, and merging the results. The idea of warping a single nearest neighbor can be extended to warping an ensemble of k-nearest neighbor reference images, with each neighbor contributing separately to the prediction.

In addition to the flow field, each network in the ensemble also predicts a 256×256 confidence map (as shown in Fig. 2.2). We use these confidence maps to compute the weighted sum of different predictions in the ensemble and compute the final prediction. We refer to this general, multi-neighbor formulation of the forward model as k-NN-FLOW.

2.3.3 Prediction using the Forward Model

Given a trajectory or sequence of states $X_t = (\mathbf{x}_i)_{i=t+1}^{T}$, it may be desirable to obtain the corresponding observations $O_t = (\mathbf{o}_i)_{i=t+1}^{T}$. For instance, an expected trajectory $\overline{X}_t$ could be obtained by using a motion-planning algorithm in simulation. The resulting solution would then be translated into observation space using the forward sensor model, generating a predictive video containing photo-realistic image frames. Generating each image $\mathbf{o}_i$ is accomplished simply by performing a forward pass on each corresponding state value $\mathbf{x}_i$ using the architecture described in Section 2.3.1.

2.4 Tracking with Learned Visual Models

A common approach to state estimation is to use a Bayesian filter with a generative sensor model to provide a correction conditioned on measurement uncertainty. For instance, the forward model, described in the previous sections, could be used to provide such a state update, and potentially allow for a single model to be used for both tracking and prediction. To track a belief-state distribution, we derive an Extended Kalman Filter from this sensor model. For the parametric component, this is a straightforward process, as deep neural networks are inherently amenable to linearization. To perform the correction step, the Jacobian J is computed as the product of the layer-wise Jacobians: $J = J^{(L)} \times J^{(L-1)} \times \cdots \times J^{(0)}$, where L is the total number of layers in the network branch. The dimensionality of the observations makes it impractical to compute the Kalman Gain $\left(K = \Sigma J^T (J \Sigma J^T + R)^{-1}\right)$ directly. Instead, we use a low-rank approximation of the sensor-noise covariance matrix R, and perform the inversion in projected space:

$$(J^T \Sigma J + R)^{-1} \approx U (U^T J^T \Sigma J U + S_R)^{-1} U^T \tag{2.2}$$

where S_R contains the top-most singular values of R. Simple first-order linear dynamics is used for the transition model. We provide an initial prior over the state values with identical covariance, and an arbitrary mean-offset from ground-truth.

Although it is possible to define an EKF in this manner, this approach is only expected to perform accurately for sufficiently smooth functions. The non-parametric component described in Section 2.3.1 prevents the current forward model from having this property, and, the EKF derived for our forward model has limited accuracy in practice (as discussed in below). The stability of the tracking is also highly dependent on the quality of the generated images. Discontinuities arise from the non-parametric component of the k-NN-FLOW model, as selected nearest-neighbors may abruptly change during a tracked trajectory. Although this effect is mitigated by using a softmax mask for weighting of nearest-neighbor-

flow outputs, these sudden jumps in the value of the Jacobian and residual terms can lead to aberrations in the tracking performance. We find that using a de-convolutional network for this task is more stable, in general. Alternatively, we can learn an inverse sensor model, in analogy with inverse kinematics, to directly predict robot state from observations.

2.5 An Inverse Sensor Model

In order to infer the latent joint states $\mathbf{x}_i$ from observed images o_i, we define a discriminative model g_{fwd}. Given the capacity of convolutional neural network models to perform regression on high-dimensional input data, we used a modified version of the VGG CNN-S network [56] containing 5 convolutional layers and 4 fully-connected layers (shown in Fig. 2.4). The model was trained on $256 \times 256 \times 3$ input images and corresponding joint labels, optimizing an L_2 loss on joint values and regularized with dropout layers.

2.6 Tracking using the Inverse Sensor Model

The problem of tracking the state of a system can be generally defined as follows: given a history of observations $O_t = (\mathbf{o}_i)_{i=t-T}^{t}$, we would like to estimate the corresponding sequence of states $X_t = (\mathbf{x}_i)_{i=t-T}^{t}$. In the current application, we propose to use the inverse model described in Section 2.5 to infer each state value $\mathbf{x}_i$ directly from $\mathbf{o}_i$, and independently of other states ($\mathbf{x}_{j\neq i} \in X_t$) and observations ($\mathbf{o}_{j\neq i} \in O_t$). This assumes that the state of the system is fully-observable in a given image frame, and that the observations

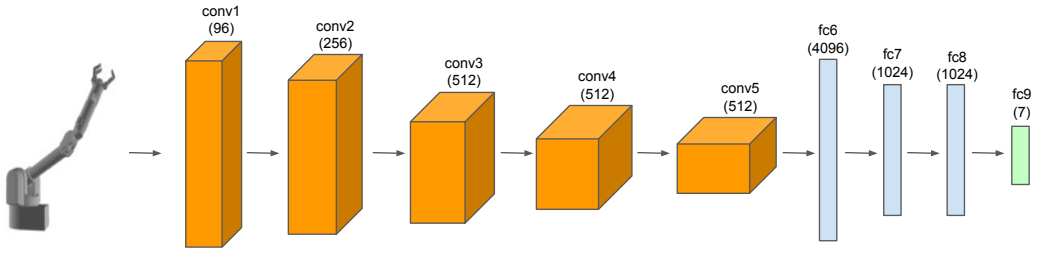


Figure 2.4: The inverse sensor model architecture, based on VGG-net.

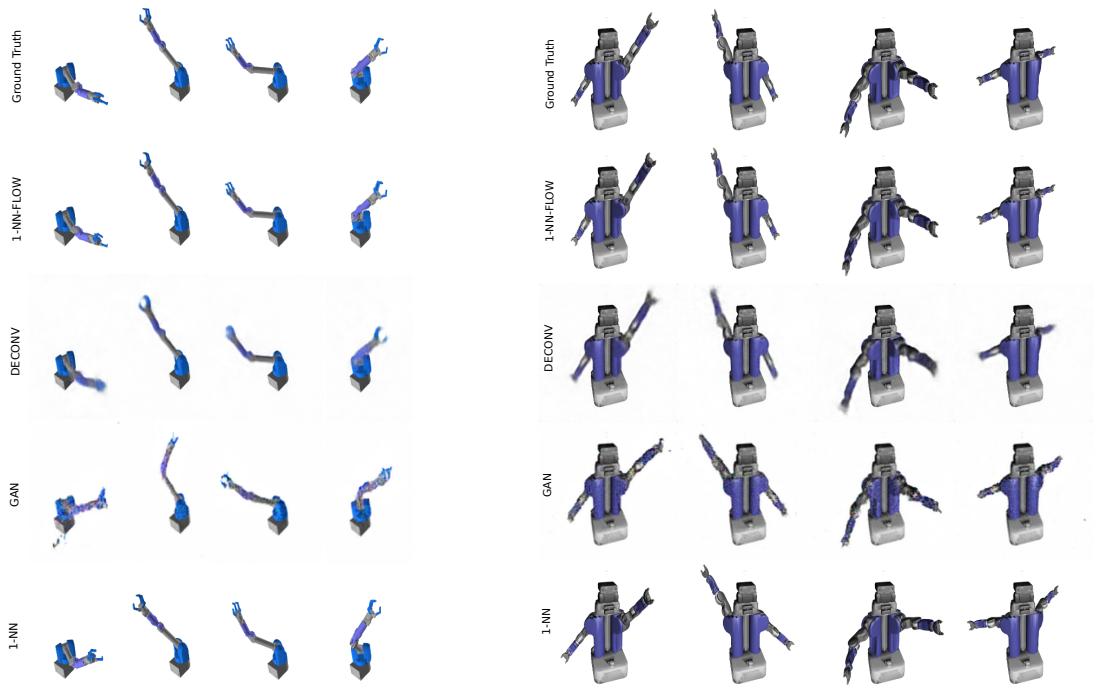
$\mathbf{o}_i$ are therefore free of occlusion.

2.7 Synthetic Dataset Generation

Images of robot-pose and corresponding joint values were captured for a Barrett WAM robot manipulator modeled in simulation using DART [57]. These were taken from a different viewpoint compared to those of the physical system, and the first four joints were sampled randomly. With a truly uniform sampling distribution, less data was required for training (20,000 for the training set and 10,000 for nearest-neighbor set). In addition, a simulation dataset of identical size was captured for a PR2 dual-manipulator robot, with random sampling of six degrees-of-freedom (first three joints in each arm).

2.8 Real-world Data Collection

Experiments were conducted using a Barrett WAM manipulator, a cable-actuated robotic arm. Data was captured from raw joint-encoder traces and a statically-positioned camera, collecting 640×480 RGB frames at 30 fps using a PrimeSense camera (the infra-red depth modality was not used for this study). Due to non-linear effects arising from joint flexibility and cable stretch, large joint velocities and accelerations induce discrepancies between recorded joint values and actual positions seen in the images [58]. In order to mitigate aliasing, the joint velocities were kept under $10^{\circ}/s$. This and other practical constraints imposed limitations on obtaining an adequate sampling density of the four-dimensional joint space. As such, the training data was collected while executing randomly generated trajectories using only the first four joints (trajectories were made linear for simplicity). A total of 500 trajectories were executed, resulting in 225,000 captured camera frames and corresponding joint values. 50,000 data points were reserved for the nearest-neighbor data-set, and the remaining for training data. Test data was collected for arbitrary joint trajectories with varying velocities and accelerations (without concern for joint flexibility and stretch).



(a) WAM 4-dof (b) PR2 6-dof

Figure 2.5: Examples of generated images for randomly-sampled input joint values, which were not encountered during training. Each row corresponds to the following (top-to-bottom): ground-truth images, 1-NN-FLOW predicted output, DECONV baseline, and GAN baseline. Differences in robot-pose arise for GAN predictions due to noise injection necessary for joint-conditioned training.

Dense sampling along sparsely-distributed trajectories results in non-uniformly sampled data. Simply constructing a nearest-neighbor dataset from randomly sampling collected data results in clusters of neighbors with high intra-cluster similarity. Picking k-nearest neighbors results in many reference images providing nearly identical viewpoints of the robot arm, which reduces the benefit of using multiple neighbors to begin with. To introduce more dissimilarity (and variation) into k-neighbor selection, it was ensured that no two nearest neighbors originated from the same trajectory executed during data collection.

Figure 2.6: Sequence of generated images for a test trajectory, where the top-to-bottom row correspondence is as follows: ground-truth images, 1-NN-FLOW predicted output, DECONV baseline, and GAN baseline. Times indicated for $t = 1, 20, 40, ..., 80$, from left to right, respectively

Model	WAM (4-dof)		PR2 (6-dof)	
	Mean L_1	RMS	Mean L_1	RMS
5-NN-FLOW (ours)	0.00255	0.02319	0.00878	0.04546
2-NN-FLOW (ours)	0.00222	0.02171	0.00867	0.04843
1-NN-FLOW (ours)	**0.00183**	**0.01905**	**0.00616**	**0.03868**
DECONV (baseline)	0.00726	0.02871	0.01567	0.04566
GAN (baseline)	0.03552	0.04262	0.12373	0.08151
1-NN (baseline)	0.00838	0.06471	0.02617	0.12283

Table 2.1: RGB prediction pixel-error results on simulated datasets for WAM (4-dof) and PR2 (6-dof) platforms. Shown are values for different k-NN-FLOW forward models ($k = 1, 2, 5$ nearest-neighbors), with comparison to 1-nearest-neighbor, DECONV and GAN baselines. Raw pixels values are within $[0, 1]$.

Model	Mean L_1	RMS
5-NN-FLOW (ours)	0.01246	0.02852
2-NN-FLOW (ours)	**0.01084**	**0.02269**
1-NN-FLOW (ours)	0.01223	0.02510
DECONV (baseline)	0.01303	0.02832
GAN (baseline)	0.01417	0.03017
1-NN (baseline)	0.01786	0.05210

Table 2.2: RGB prediction pixel-error results on the real-world 4-dof WAM dataset. Shown are values for different k-NN-FLOW forward models ($k = 1, 2, 5$ nearest-neighbors), with comparison to 1-nearest-neighbor, DECONV and GAN baselines. Raw pixels values are within $[0, 1]$.

2.9 Forward Sensor Model Evaluation

We first examine the predicted observations generated by the proposed forward sensor model (k-NN-FLOW). This includes three different variations of the proposed model, constructed for $k = 1, 2, 5$ nearest neighbors. A deconvolutional network (DECONV) similar to that used in [46] is selected as a baseline, with the absence of a branch for predicting segmentation masks (as these are not readily available from RGB data). Given recent successes of generative adversarial networks (GANs) for image generation, we also include a GAN model based on [59, 60]. GAN models can be conditioned for controlled output [61]. Instead of conditioning on discrete labels, we condition on a continuous signal provided by the joint angles. We stabilize the training by adding low variance noise to the input signal . Finally, we include results produced by a simple 1-Nearest Neighbor implementation.

Quantitative results for prediction accuracy are shown in Tables 2.1 and 2.2, where it is apparent that both 1-NN-FLOW and 2-NN-FLOW models outperform the DECONV, GAN and 1-NN baselines in both mean L_1 and RMS pixel error. The 5-NN-FLOW predictions are similar in appearance to the real-world predictions produced by both the DECONV and GAN networks, but outperform them on the simulated datasets. All models outperform simple 1-NN selection.

Qualitative comparisons between the ground-truth, forward-sensor predictions, DE-

CONV and GAN outputs are depicted in Fig. 2.5 for randomly selected joint input values in simulation. For the physical system, prediction results are shown in Fig. 2.6 for a sequence of state-input values at various times on a pair of real test trajectories.

Detailed texture and features have been preserved in the k-NN-FLOW predictions, and the generated robot poses closely match the ground truth images. Both DECONV and GAN outputs suffer from blurred reconstructions, as expected, and do not manage to render certain components of the robot arm (such as the end-effector).

2.10 Occlusion Prediction

Using generative models for frame-prediction results in another interesting benefit: the capacity to reason about constraints in image-space. We briefly demonstrate how the forward model, k-NN-Flow, can be used for predicting occlusions of articulated bodies. This may be desirable in instances where maintaining visibility of the robot is required for monitoring execution under partial observability, such as reaching or grasping in cluttered environments [62, 63, 64].

While collecting ground-truth optical flow values is infeasible in a real-world application, the core architecture shown in Fig. 2.2 learns to predict the inverse optical flow fields from reference observation $\mathbf{o}_r$ to the current observation $\mathbf{o}$ without any direct supervision. The forward optical flow can also be generated by changing the order of the inputs to the network. We use the assumption that forward and inverse optical flow should be symmetric for visible portions of the arm [65, 66]. Given two states $\mathbf{x}_1$ and $\mathbf{x}_2$, we compute the forward and backward optical flow. We then find the robot-pixels in the first image with symmetric forward and backward optical flow. Points that violate this property are assumed to be occluded in the second image. Fig. 2.7 illustrates a few qualitative examples that were generated using this method. Predicting occlusions by projecting from configuration to image space could allow such perceptual constraints to be used in defining desirable trajectories (which maintain end-effector visibility, for example), and could be used in conjunction

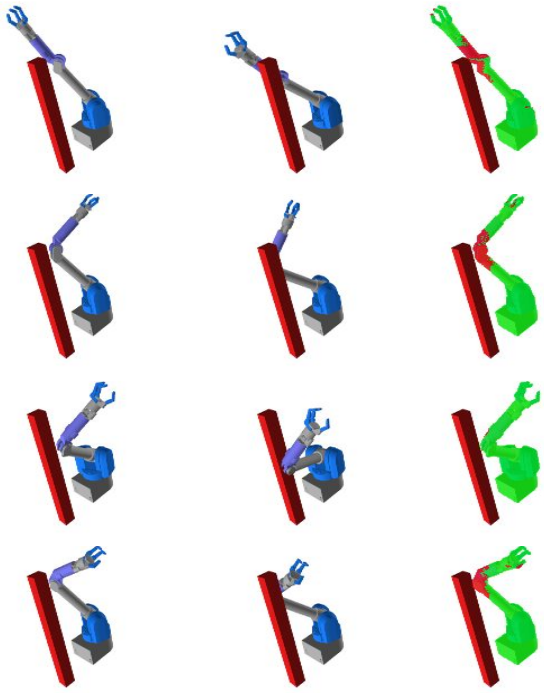


Figure 2.7: Each row shows an example of occlusion detection. The first column is the first observation $\mathbf{o}_1$ in which the arm is not occluded by the object. The goal is to predict which part of the arm would be occluded if it moves to the second state. Second columns show the second observation $\mathbf{o}_2$. Note that the observations are shown for demonstration purposes, and the network only uses the corresponding joint values $\mathbf{x}_1$ and $\mathbf{x}_2$. The third column depicts the where violation forward-backward symmetry is predicted for $\mathbf{o}_1$: red pixels indicate occluded regions, while green pixels indicate be observable surfaces.

with a planning framework.

2.11 Tracking Task Evaluation

We use the inverse model described in Section 2.5 to track the 4-dof robot joint positions from images. The performance is compared to tracking with an EKF formulated using the forward model defined in Section 2.3.1) and an EKF based on the DECONV model. The latter uses a Jacobian calculated in the same manner as described in Section 2.3.1.

Examples of tracking accuracy for a single test trajectory are shown in Fig. 2.8 and Fig. 2.9. Here, joint state estimates from the inverse model are compared to those provided by a 2-NN-FLOW EKF and a DECONV EKF. Both EKF models are initialized with different mean-offsets ($\mathbf{x}_0 = 10^\circ$ and $\mathbf{x}_0 = 20^\circ$), to demonstrate approximate knowledge of

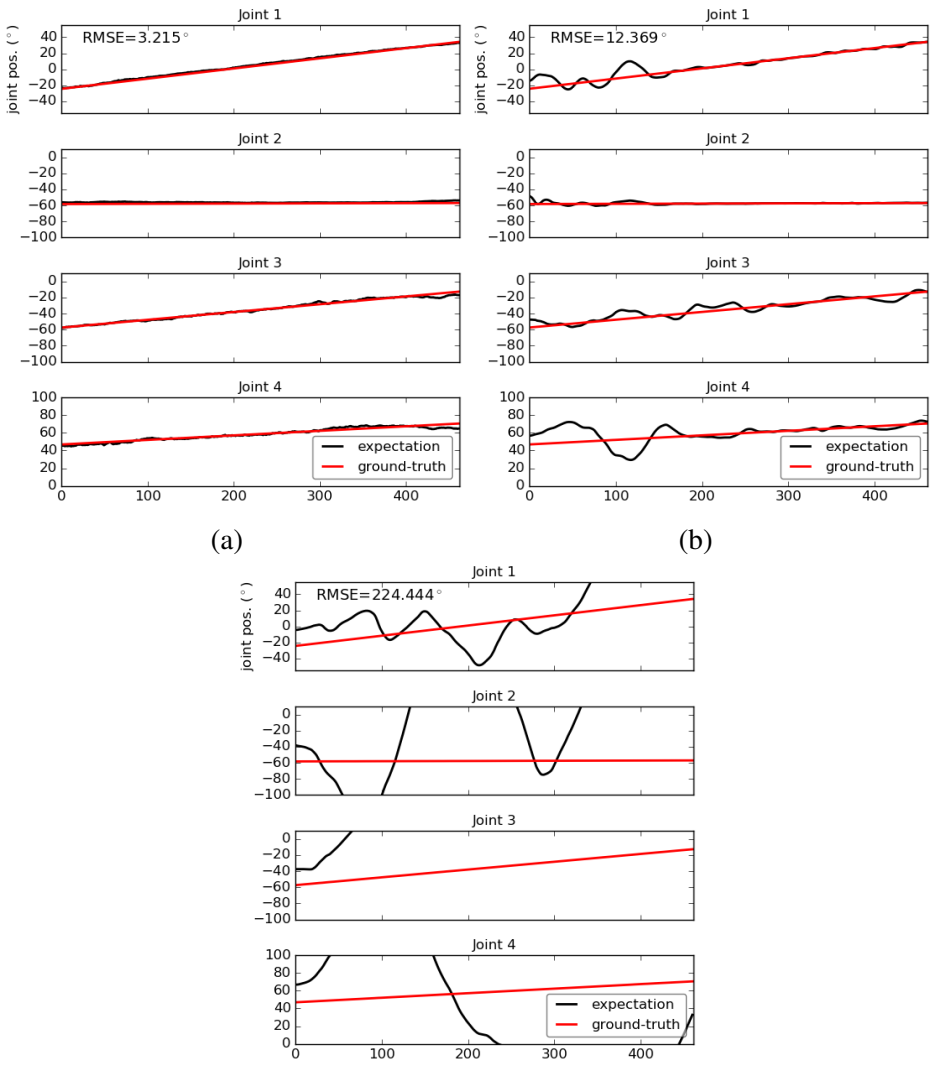


Figure 2.8: Comparison between the learned inverse sensor model (Fig. 2.4) and an EKF using the 2-NN-FLOW forward sensor model. Each row corresponds to a single joint evolving over 450 frames. The red line is the ground truth joint configuration, the black line is the estimated state. RMSE scores are shown. (a) The inverse model can robustly and accurately predict the state from an arbitrary image and unknown start state. (b) Tracking using an EKF and the learned DECONV model starting from a 10-degree offset and (c) 20-degree offset. The EKF works better when the state is already accurately tracked, but in general is much less robust and accurate than the learned inverse model.

the starting state at $t = 0$.

The results indicate that both EKF trackers are able to converge the state estimate to the true trajectory over time, given a favorable (10° offset) initial prior. RMSE tracking errors demonstrate similar tracking performance for this initialization. Failure cases for

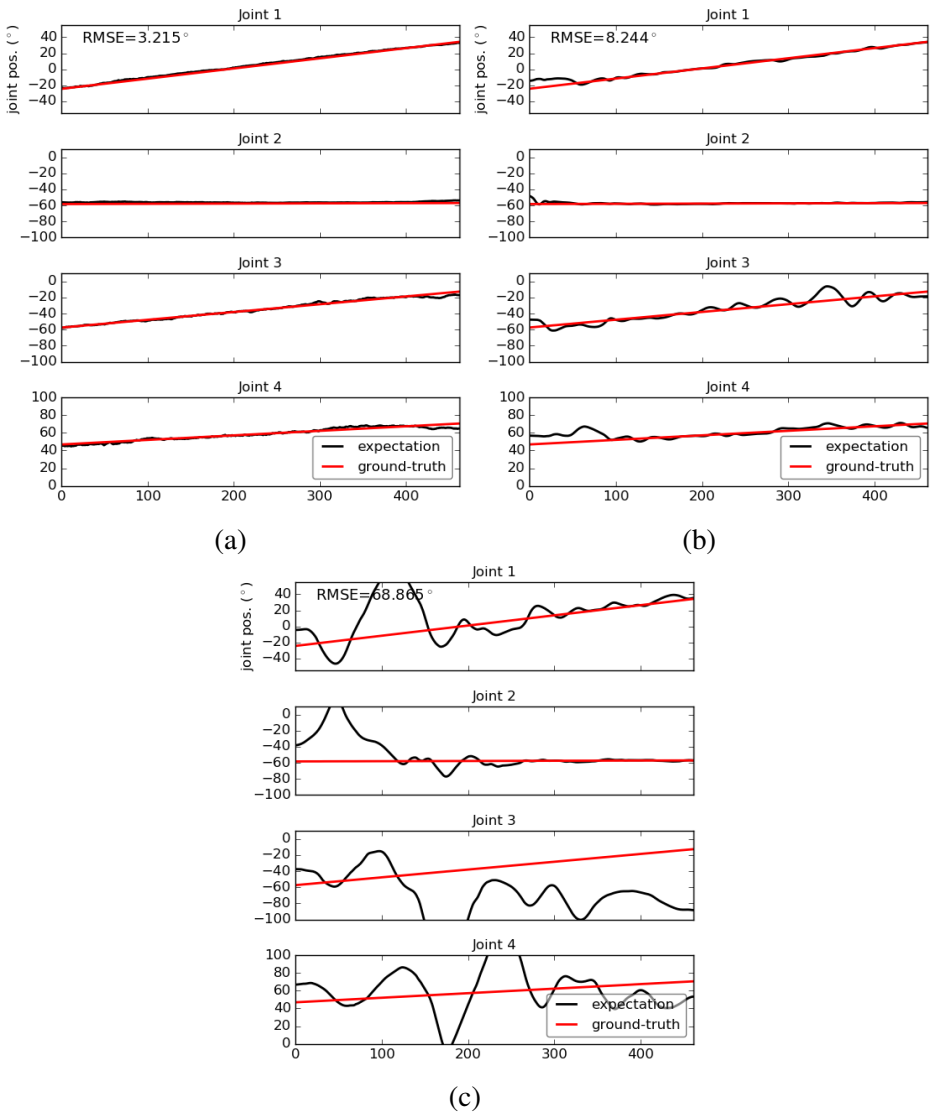


Figure 2.9: Tracking results and RMSE scores for DECONV EKF (same context as Fig. 2.8). The DECONV EKF works better when the state is already accurately tracked, but in general is much less robust and accurate than the learned inverse model.

2-NN-FLOW and DECONV EKF models are shown in Fig. 2.8c and Fig. 2.9c. Here, the state is initialized with a $20°$ offset, leading to instability in both trackers. Although performance has drastically deteriorated in both cases, it is worth noting that the DECONV EKF manages to recover reasonable joint state estimates for the first two joints. As mentioned in Section 2.4, EKF performance is dependent on the smoothness of the function, and may be a contributing factor to tracking robustness. The stability of the tracking is highly dependent on the quality of the generated images. Discontinuities arise from the non-parametric

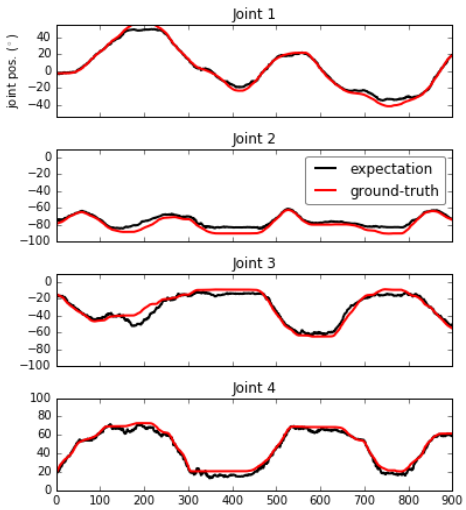


Figure 2.10: Inferred joint values from a sequence of images using inverse sensor model for an arbitrary trajectory.

component of the k-NN-FLOW model, as selected nearest-neighbors may abruptly change during a tracked trajectory. Although this effect is mitigated by using a softmax mask for weighting of nearest-neighbor-flow outputs, these sudden jumps in the value of the Jacobian and residual terms can lead to aberrations in the tracking performance. It is apparent that the inverse sensor model is considerably more suited to the tracking task in this domain. The tracker requires no initial prior, and finds an optimum in a single forward pass (as opposed to the iterative optimization performed by the EKF). The comparatively high robustness of the inverse model in tracking is further demonstrated by estimating the state of an arbitrary nonlinear test trajectory shown in Fig. 2.10. No latent dynamics model is assumed here, and state estimates are produced independently given a currently observed frame.

2.12 Discussion

This chapter presented a framework for tracking and prediction consisting of separate inverse and forward models, relating state to perceptual space, for purposes of state estimation and observation generation respectively. A novel approach was proposed, which combines the strengths of nearest neighbors and neural networks to generate high-quality

predictions of never-before-seen images. In both a quantitative and qualitative sense, this generative network produces improved results over the DECONV, GAN and 1-NN baselines. For state-estimation and tracking, the generative observation model can be used in an EKF-based framework to perform probabilistic inference on the underlying latent state, and track the manipulator state over a simple trajectory. However, we have shown that learning a convolutional neural network as a forward model results in better performance in practice. Several experiments were performed on a real robotic system, validating the approach and showing that our forward model is quantitatively and qualitatively state-of-the-art.

We have examined a technique for incorporating structure in *visual* perceptual models, where robot kinematic states and projected appearance served to provided non-parametric reference points for training a predictive module. Next, we will move on to the *tactile* sensing modality, and discuss how structure can be used to improve prediction in this context.

CHAPTER 3
LEARNED TACTILE SENSING AND FORCE ESTIMATION

3.1 Introduction

Tactile perception is an important sensing modality, enabling robots to gain critical information for safe interaction in the physical world [67, 68, 69]. The advent of sophisticated tactile sensors [70] have provided increased sensitivity to forces induced by contact dynamics, allowing for a diversity of applications in robotics ranging from object class and pose identification, surface texture reasoning, and slip detection [71, 72, 73, 74, 75, 76, 77, 78]. Yet, the output signals of these devices are typically noisy and difficult to interpret. This is largely due to the complexity and non-linearity of contact mechanics (such as visco-elastic deformation and adhesion). There has been lack of accurate, generalize-able models which can correctly map raw sensory signals to useful force information across different tasks. This has limited the use of these sensors to coarse, force-based control, acting as classifiers of salient events such as slip or contact [79, 75]. Providing reliable, continuous measurements on force direction and magnitude could allow for improved robustness in controller design used in contact-rich manipulation tasks.

These shortcomings were addressed in [24] using a large-scale data-driven approach, where training examples were collected across different contact domains in order to learn a tactile sensor model. To improve prediction of directional force measurements, the spatial structure and surface geometry of the sensor was encoded directly into the architecture of the network used for the model. The proposed approach was compared to the current state-of-the-art methods for force estimation using the same tactile-sensing device, including both analytic [80] and learned [81] baselines.

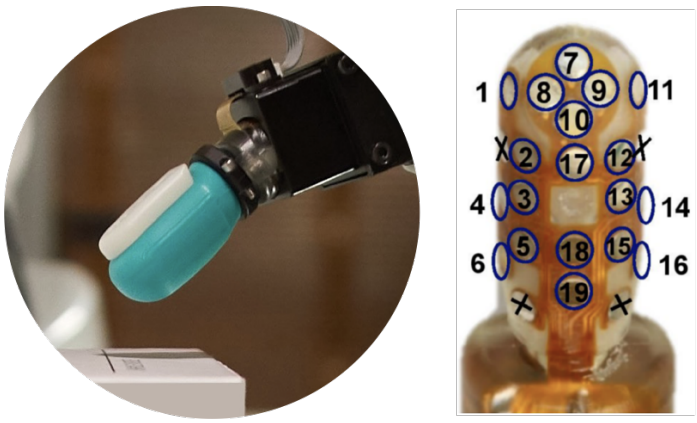


Figure 3.1: The BioTac sensor [82] consists of a rigid core, surrounded by a weakly conductive gel and a high friction elastomeric skin. Changes in impedance caused by fluid deformation during contact are captured by an array of 19 electrodes. However, these measurements must be converted into meaningful force values for many manipulation tasks.

3.2 Related Work

One limiting factor in existing approaches has been the lack of effective mappings from tactile signals to force measurements which scale robustly across different tasks. Current methods for force estimation on the SynTouch BioTac [83] fail to cover the entire range of forces applied during typical manipulation tasks. Analytic methods [80, 84] tend to produce very noisy estimates at small force values and their accuracy decreases as the imparted force angle relative to the sensor surface normal becomes large (*i.e.*, a large shear component relative to the compression force). On the other hand, learned force models [85, 81] tend to overfit to the dataset used in training and have not been sufficiently validated in predicting force across varied tasks.

More specifically, Wettel and Loeb [85] use machine learning techniques to estimate the force, contact location, and object curvature when a tactile sensor interacts with an object. Lin *et al.* [80] improve upon [85], formulating analytic functions for estimation of the contact point, force, and torque from the BioTac sensor readings. Navarro *et al.* [84] explore calibration of the force magnitude estimates by recording the DC pressure signal when the sensor is in contact with a force plate. They use these values in a linear least squares for-

mulation to estimate the gain. While they can estimate the magnitude of force, they cannot estimate force direction. Su *et al*. [81] explore using feed-forward neural networks to learn a model that maps BioTac signals to force estimates. The neural network more accurately estimates forces than the linear model from [80] and is used to perform grasp stabilization. Importantly, none of these methods validate their force estimates using a data source different from the method used to generate the training data. They also lack experimental comparison between different approaches in the context of robotic manipulation tasks.

In the following, we attempt to address these limitations by collecting a large scale ground truth dataset from different methods and by leveraging the sensor surface and spatial information in our proposed neural network architecture. For one of our collection methods, we infer force from the motion of an object on a planar surface, by formalizing the interaction as a system of particles, a deviation from the well-established velocity model for planar pushing [86] which does not reason about force magnitude. This scheme of force estimation allows us to obtain accurate small-scale forces (0.1-2N), enabling us to learn a precise force prediction model.

Motivated by [87], we compare our proposed method with the current state-of-the-art methods for force estimation for the BioTac sensor. We specifically compare the analytic model from [80] and the best performing feed-forward neural network model from [81]. We compare both in terms of force estimation accuracy on our dataset and also empirical experiments on a robot manipulation task. To summarize, we make the following contributions:

1. We provide a novel method to infer force from object motion on a planar surface by formalizing the mechanics as a system of particles and solving for the force in a least squares minimization problem, given the object motion and the point on the object where the force is imparted.

2. We introduce a novel 3D voxel grid, neural network encoding of tactile signals enabling the network to better leverage spatial relations in the signal. We further tailor

our learning to the tactile sensor through the introduction of a novel loss function used in training that scales the loss as a function of the angular distance between the imparted force and the surface normal.

3. We collected a large-scale dataset for the BioTac sensor, consisting of over 600 pushing episodes and 200 interactions between an arm-hand system equipped with the BioTac sensors and a force torque sensor.

We validate these contributions on our dataset and in an autonomous pick and place task. We show that our proposed method robustly learns a model to estimate forces from the BioTac tactile signals that generalize across multiple robot tasks. Our method improves upon the state of the art [81, 80] in tactile force estimation for the BioTac sensor achieving a median angular accuracy of 3.5 degrees in predicting force direction (66% improvement over the current state of the art) and a median magnitude accuracy of 0.06 N (93% improvement) on a test dataset.

3.3 Problem Definition & Proposed Approach

We describe the sensor's states in the following section, followed by a formal definition of the problem. We then describe the computation of ground truth force from planar pushing in Section 3.3.3 and our network architecture in Section 3.3.4.

3.3.1 BioTac Sensor

We use the BioTac sensor [82] from SynTouch. The sensor has a rigid core which is enveloped by a high friction elastomeric skin. A weakly conductive liquid is filled in the space between the core and the skin. There are 19 impedance sensing electrodes spread out on the core surface, giving measurements $e \in \mathbb{R}^{19}$. A thermistor coupled with heaters measures the fluid temperature $T_{dc} \in \mathbb{R}$ and temperature flow $T_{ac} \in \mathbb{R}$. A transducer measures the static pressure $p_{dc} \in \mathbb{R}$. High frequency changes to the pressure are mea-

sured by the transducer at 2.2 kHz and sent to the system in a buffer with the past 22 values $p_{ac} \in \mathbb{R}^{22}$ at 100 Hz along with the other signals. A single sensor sample is thus given by $z = [e, p_{dc}, p_{ac}, T_{dc}, T_{ac}]^\top \in \mathbb{R}^{44}$. Following [80], we use the tared signals from the sensor (*i.e.*, initial value subtracted). Using methods described by Lin et al. [80], we also compute the contact point $s_c \in \mathbb{R}^3$ on the BioTac sensor and the surface normal $s_n \in \mathbb{R}^3$ by approximating the BioTac surface geometry as a half-cylinder attached to a quarter-cylinder cap, both of the same radius r.

3.3.2 Problem Definition & Approach Overview

We define the problem as estimating the force $f \in \mathbb{R}^3$ with reference to the sensor frame B, given z, s_c, and s_n. We use feed-forward neural networks to learn the function $f = F(z, s_c, s_n)$ that maps from sensor readings z, the sensor surface contact point s_c, and the surface normal s_n, to the force f. In order to learn an accurate model, our training dataset needs to cover a wide range of forces (in magnitude and direction). Furthermore, to learn a robust model that transfers to new tasks, we generate ground truth data from three different sources. The first source is collected by rigidly attaching the BioTac to a wrist force/torque (FT) sensor similar to [81, 80] and pressing on the BioTac sensor using objects. We term this source *rigid-ft*. This requires a human to interact with the object and is biased by the human. This setup was used to cover very large forces. For the second source, we attach the same FT sensor to a ball, with which we interact using a robotic hand-arm system. We call this source *ball-ft*. The *ball-ft* source adds randomness to the orientation of the BioTac sensor frame with respect to the force torque sensor frame. The wrist force/torque sensor is noisy in ranges between 0.01 N to 0.1 N, making small force readings unreliable. To overcome this problem, we collect sensor readings from a planar pushing setup, where a robot pushes a box on a planar surface using the tactile sensor. We call this source *planar-pushing*. The ground truth force for planar pushing is computed by least squares optimization, described in the next section.

3.3.3 Mechanics of Planar Pushing as a System of Particles

Given an object with mass m in an SE(2) planar space, moving with a linear velocity v and an angular velocity ω, the net force causing this motion can be defined as

$$f_c = m\dot{v} \tag{3.1}$$

$$c \times f_c = I\dot{\omega} \tag{3.2}$$

where $f_c \in \mathbb{R}^2$ is the net force acting at a point $c \in \mathbb{R}^2$ with reference to the center of mass (CM) of the object. Given the linear acceleration, the net force can be obtained. However, if the measurement system is not able to observe small linear accelerations, solving Eq. (3.1) is intractable. There are two cases when the linear acceleration can be small: 1) when the force applied to the object is very small, causing very small linear and angular acceleration, 2) when the force applied is perpendicular to the radial line, in which case the object will have a large angular acceleration. In the latter case, Eq. (3.2) could give us the net force f_c. However, Eq. (3.2) is a degenerate system as we need $f_c \in \mathbb{R}^2$ from $\dot{\omega} \in \mathbb{R}$. We solve for f_c by formulating Eq. (3.1) and Eq. (3.2) as loss functions in a least squares minimization problem:

$$\underset{f_c}{\operatorname{argmin}} \quad k||f_c - m\dot{v}||^2 + |||[c]_\times f_c - I\dot{\omega}||^2 \tag{3.3}$$

where the weight k scales the linear acceleration loss and $[c]_\times$ is the skew symmetric matrix of vector c. Consider the object to be now resting on a planar surface with coefficient of friction μ_s between the object and the surface. The friction between the object and the planar surface will oppose the motion of the object with a frictional force f_f and moment n_f. If the contact region between the object and the surface is R, and r is any point on the

object in this region, the force and moment can be defined using Coulomb's law as

$$f_f = -\mu_s \int_R \frac{v(r)}{\|v(r)\|} p(r) dA \tag{3.4}$$

$$n_f = -\mu_s \int_R r \times \frac{v(r)}{\|v(r)\|} p(r) dA \tag{3.5}$$

where $v(\cdot)$ is a function that gives the velocity of the point. The pressure at r is given by $p(\cdot)$ and dA is a differential element of area at r. We derive the moment with reference to the object's center of mass. To make computation of the frictional force tractable for planar pushing, we make the following assumptions:

1. The pressure distribution in the contact region R is uniform.

2. The rigid body is made of n particles which are uniformly distributed.

The contact region is decomposed into n small regions, with center of mass for region i at r_i and the normal force applied by region i is $\int_i p(r_i) dA = \frac{mg}{N}$.

With the listed assumptions, we simplify the frictional force

$$f_f = -\mu_s \frac{mg}{N} \sum_{i=0}^{n} \frac{v(r_i)}{\|v(r_i)\|} \tag{3.6}$$

and the moment due to frictional force becomes

$$n_f = -\mu_s \frac{mg}{N} \sum_{i=0}^{n} r_i \times \frac{v(r_i)}{\|v(r_i)\|} \tag{3.7}$$

Including the frictional force f_f and moment n_f in our minimization problem of Eq. (3.3)),

$$\underset{f_c}{\operatorname{argmin}} \quad k||f_c + f_f - m\dot{v}||^2 + |||[c]_\times f_c + n_f - I\dot{\omega}||^2 \tag{3.8}$$

Optimizing Eq. (3.8) yields an estimate of the force f_c. The force f_c is 2D, parallel to the support surface. We obtain the ground truth force $f_{3d} = {}_B R_o f_c$ by transforming the force f_c

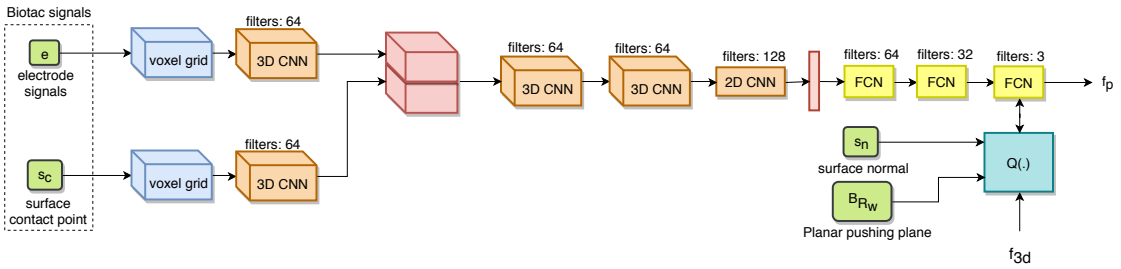


Figure 3.2: The force prediction neural network uses 3D voxelized inputs that preserve the spatial information. We use layer norm followed by ReLU after every convolutional and fully connected layer (FCN). Additionally, we use kernels and strides of 2 for every convolutional layer.

from the object's frame of reference o to the BioTac sensor frame B.

3.3.4 Network Architecture

Our proposed neural network architecture takes only the spatial signals[1] e, s_c from the BioTac to estimate the force as shown in Fig. 3.2. We create a 3D voxel grid and input the value of each electrode on the corresponding voxels based on the electrode's position with reference to the BioTac frame B. We create a second voxel grid for the contact point and input a value of 1 for the voxel at contact point s_c. These two voxel layers are concatenated and passed through two layers of 3D convolutions. The features are then flattened and passed through a layer of 2D convolutions, which is further flattened to a vector. This vector passes through fully connected layers to output the predicted force vector f_p of length 3.

3.3.5 Loss Functions

The predicted force vector f_p is compared to the 3D ground truth force f_{3d} via a scaled ℓ_2 norm.

$$Q_{3d}(f_{3d}, f_p) = \frac{1}{||f_{3d}||}||f_{3d} - f_p|| \tag{3.9}$$

[1]We found empirically that the other signals did not improve the force estimation accuracy.

For the planar pushing dataset, we use a projected ℓ_2 norm, as there could be forces acting perpendicular to the planar surface which the physics model does not take into account.

$$Q_{proj}(f_{3d}, f_p, {}_wR_B) = \frac{1}{||f_{3d}||}||({}_wR_B f_{3d} - {}_wR_B f_p)^\top \psi||^2 \tag{3.10}$$

where ψ is the orientation of the support surface plane.

The high friction of the BioTac surface allows for imparting force from directions other than the surface normal at a contact point. So the force could be applied from any contact point on the surface and is not only limited to contact points whose surface normal matches with the force direction. We hypothesize that as the angle between the force and the surface normal increases, the sensor's signals might be less meaningful. We scale the loss function with an adaptive weight function $\alpha(\cdot)$ to reflect this hypothesis.

$$\alpha(s_n, f_{3d}) = 2^{\beta(1-D(s_n, f_{3d}))} \tag{3.11}$$

$$D(s_n, f_{3d}) = \frac{\cos^{-1}(s_n^\top \hat{f}_{3d})}{\pi} \tag{3.12}$$

where β is a scalar weight and $\hat{f}_{3d}$ is the unit vector of the ground truth force vector f_{3d} and $D(\cdot)$ is the normalized cosine distance function. The loss function used in our network is defined as

$$Q(\cdot) = \begin{cases} \alpha(\cdot)Q_{proj}(f_{3d}, f_p, {}_wR_B) & \text{if planar pushing} \\ \alpha(\cdot)Q_{3d}(f_{3d}, f_p) & \text{otherwise} \end{cases}$$

3.4 Dataset Collection, Implementation Details, & Experimental Protocol

This section provides a concise description of our dataset collection procedure. We also provide implementation details of our neural network and describe the error metrics and comparison methods used to analyze our force model.

3.4.1 Dataset Collection

The setups for our three different data sources *rigid-ft*, *ball-ft*, and *planar-pushing* are shown in Fig. 1. For learning the force model, we only used samples from the dataset that have non-zero force readings; we term these samples "force samples". We use the OptoForce HEX-E 6-DOF force torque sensor to collect the *rigid-ft* and *ball-ft* data. For *rigid-ft*, we mounted the BioTac to the force torque (FT) sensor and pressed down on the finger using flat rigid plastic objects to collect data. This closely resembles the data collection performed by [80, 81] for BioTac force estimation. We collected a total of 20k force samples. For the *ball-ft* method the Allegro hand pushed the BioTac against a hard plastic ball mounted on a vertical bar attached to the same FT sensor. We generated a total of 200 random trajectories for the middle fingertip to make contact with the ball across 10 different wrist poses generating a total of 20k force samples. For the *planar-pushing* method, we mounted a BioTac on an ABB YuMi robot which pushed a known box weighing 0.65 kg. We generated a single straight-line, task-space position trajectory for the YuMi fingertip to follow using trajectory optimization. We use Riemannian motion policies [88] to execute the task space trajectory. We chose a random initial orientation for the BioTac and box for every execution of the task space position trajectory. The orientation of the box was sampled from a small range to keep the contact on the same face of the box for each push. We collected a total of 600 trials on the robot generating 100k force samples in total.

Our final dataset collected across all three data sources contains a total of 140k force samples. For *ball-ft* and *planar-pushing* setups, we track the robot with an ASUS Xtion RGB-D camera using Dense Articulated Real-Time Tracking (DART) [89]. We enable DART's contact prior when the FT sensor measures a force greater than 2 N for *ball-ft* and when the BioTac absolute pressure signal (p_{dc}) rises greater than 10 units for the *planar-pushing* setup.

We chose the parameters for the optimization described in Eq. (3.8) as $n = 80$ and $k = 10$. We found the number of particles n did not affect the force by much above this

size and any value of k greater than 2 gave similar performance. We chose the coefficient of friction $\mu_s = 0.1$ between the box and the planar surface by interpolating data from [90]. We solve the optimization using Sequential Least Squares Programming (SLSQP) [91] available through PAGMO [92].

3.4.2 Neural Network Implementation Details

We built our neural network in TensorFlow [93]. For each data source, we used 80% for training and approximately 10% for validation and from the remaining data, we picked 1.5k samples for testing; the data was split by trials (leaving whole trials). We run the training for a maximum of 200 epochs with a batch size of 512 and store the model only when the loss on the validation set improves. We optimize using the Adam optimizer [55]. We use an adaptive learning rate that starts at 10^{-4} and increases for the first 2 epochs by $2^{(i/50)}$ and later decreases by 0.95 each iteration i for the remaining epochs, motivated by [94].

We only send the BioTac signals to the network when the robot detects the fingertip is in contact. We use the absolute pressure (p_{dc}) signal from the BioTac to determine contact and classify the sensor as in contact if this p_{dc} signal maintains a value above 10 for the past 10 timesteps. This reduces false positives when the sensor is moving in free space. We set the voxel grid to have dimensions $15 \times 15 \times 7$, which allowed a unique voxel for each electrode of the BioTac sensor. We plan on studying the effect of voxel dimensions on the efficiency of learning in the future.

3.4.3 Error Metrics, Protocol & Comparison Methods

Given the predicted force f_p and the ground truth force f_{3d}, we compute the error in direction as the cosine similarity between the vectors. We scale this cosine similarity to give a percentage direction accuracy.

$$\text{Direction error}\% = 100 \times \frac{1}{\pi} \cos^{-1}(\hat{f}_{3d}^{\top} \hat{f}_p) \tag{3.13}$$

For computing the error in magnitude, we report the symmetric mean absolute percentage accuracy between the ground truth and predicted force magnitudes:

$$\text{Magnitude error\%} = 100 \times \frac{\text{abs}(||f_{3d}|| - ||f_p||)}{||f_{3d}|| + ||f_p||} \tag{3.14}$$

We report the absolute ℓ_1 norm between the ground truth and predicted force magnitude as "magnitude error (N)".

We compare our proposed network architecture against the NN-3 model from Su et al. [81] and also the linear model given by [80]:

$$f_p = [S_x \sum_{i=1}^{19} e_i n_{x,i}, S_y \sum_{i=1}^{19} e_i n_{y,i}, S_z \sum_{i=1}^{19} e_i n_{z,i}]^\top \tag{3.15}$$

We compute the scalar weights S_x, S_y, S_z using linear regression and $[n_{x,i}, n_{y,i}, n_{z,i}]^\top$ for the orientation of electrode i.

We report error on the test dataset which contains 1500 samples from each data source (4500 in total). To study how each dataset source affects the prediction model, we train each prediction model with the following source combinations:

1. *rigid-ft* to directly compare to Su et al. [81] and [80].
2. *planar-pushing* to study how 2D ground truth force obtained by optimizing over planar physics performs on predicting 3D forces.
3. *ball-ft* to study the effect of randomization of the sensor's frame with respect to a 6-DOF force sensor.
4. *mixed*, which includes all sources *rigid-ft*, *planar-pushing* and *ball-ft*, to study the ability of the prediction model to learn from different sources.

We also evaluate the effect of the two primary contributions of our network structure: spatial signal encoding by 3D voxelization and the proposed α weight. We train our pro-

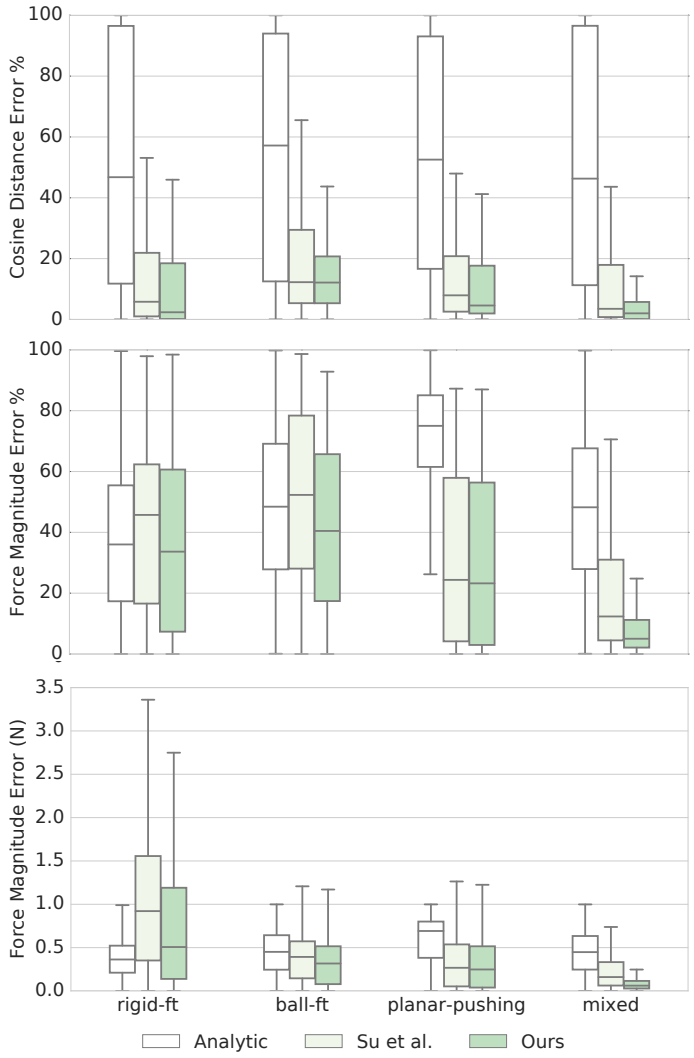


Figure 3.3: Predicted force error for different models and training sets. Analytic refers to the linear model from [80], Su *et al.* refers to the best performing model from [81].

posed network architecture with and without each of these contributions (making a total of four models) on the same source combinations as described above. We compare voxelization to four fully connected layers of the signals e and s_c, input to the proposed network's first fully connected layer.

3.5 Results

We now report the results on our test dataset. In all plots the middle line in the box plot defines the median error. The bottom and top borders indicate the first and third quartiles.

The whiskers indicate the extrema of the inliers within 1.5 times the interquartile range.

3.5.1 Prediction accuracy:

Our method trained on the *mixed* dataset achieves the best accuracy, as shown in Fig. 3.3. We achieve a median angular error of 0.06 radians (3.5 degrees) and a median magnitude error of 0.06 N compared to Su *et al.*'s median angular error of 0.18 radians (10 degrees) and median magnitude error of 0.91 N. We also compare our model to Su *et al.* (trained on *rigid-ft*) in a time series of force estimates in Fig. 3.4. We see that our model sufficiently captures the ground truth while Su *et al.* only covers the magnitude along z-axis. In Fig. 3.4, we suspect the FT ground truth to be noisy, causing the oscillating behaviour in the force along x-axis.

3.5.2 Effect of spatial encoding and α regularization:

We investigate the effect of α in the loss function. As seen in Fig. 3.5, without α, the interquartile is larger, specifically in *ball-ft* and *mixed* trained models, indicating that α indeed helps regularize the force predictions. Voxelization helps in lowering the prediction error further when combined with α highlighting that spatial information is important. Without α, voxelization performs worse only on the magnitude of *ball-ft* trained model.

3.6 Force Feedback for Object Manipulation

We analyze the generalization ability of the learned force model in an autonomous object lifting and placement task. We use this task to illustrate the utility of the learned force model and do not directly compare to other approaches to grasp stabilization and placement using tactile sensing [95, 73].

To perform the task the robot reaches its hand to a fixed pose with respect to the object. The robot closes its fingers and makes contact in the desired configuration. The robot then attempts to increase the force on all the fingertips to a desired threshold τ by increasing

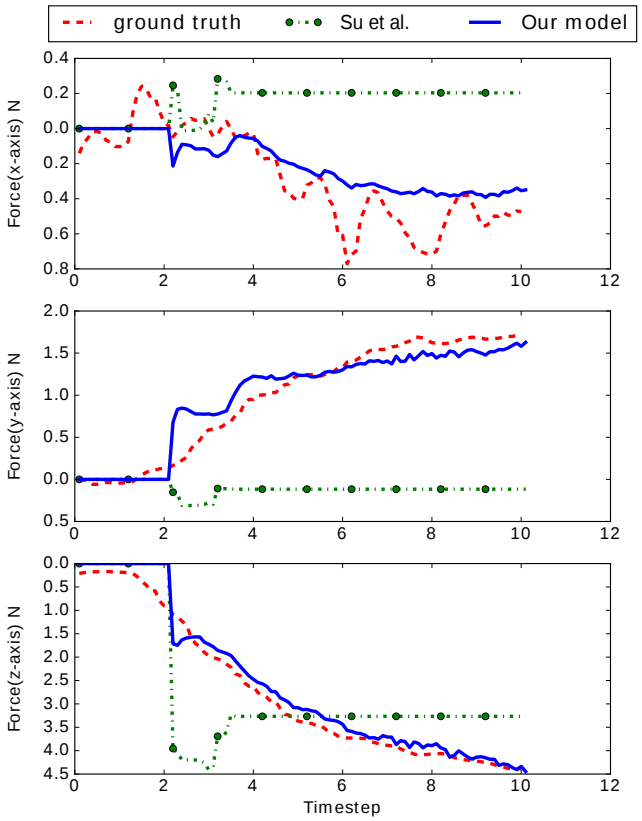


Figure 3.4: Estimated force from our model and Su *et al.* compared with the FT sensor, to which the BioTac is rigidly mounted. Our model sufficiently tracks the ground truth along all three axes.

the finger position along a predefined task-space vector. We experimentally selected τ to be 2 N as the minimal force needed for the method from [81] to lift the *soft-scrub* object. The robot then raises its arm to lift the object along a straight-vertical trajectory. After reaching the desired height, the robot lowers its arm down along the same straight line and the fingers release the object when a negative force in the normal direction to the support surface is detected on the index fingertip. All grasps we studied were three fingered grasps. We chose the index finger as it had the largest change in force direction caused by the support surface.

The lifting and placement tasks directly depend on the accuracy of the forces estimated from the tactile sensor. If the estimated force values are larger than the actual force, the

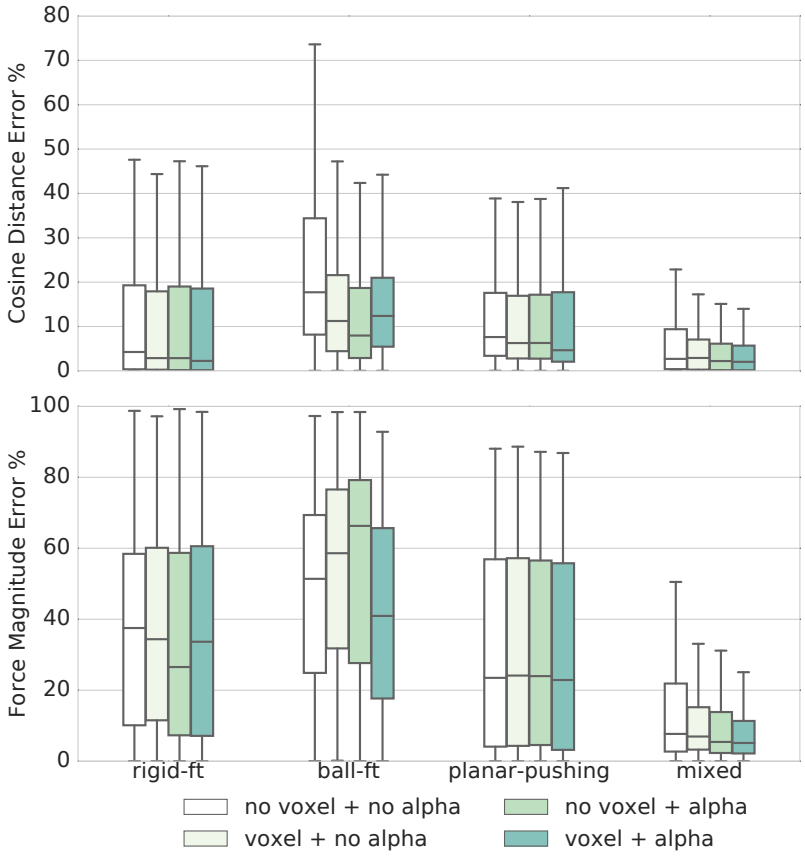


Figure 3.5: Effect of the spatial encoding ("voxel") and α on the prediction accuracy.

object will not be in a stable grasp and the object will not rise with the arm. If the values are smaller than the actual force, the grasp may deform the softer objects used in the experiments. If the force estimates are incorrect at placement, then the robot will either drop the object too soon or push down onto the table with excessive force, possibly knocking the object over.

We chose four objects from the YCB dataset [96] and two deformable objects—a paper cup and a plastic bottle. The objects are shown in Fig. 3.6. We show results for our *mixed* dataset trained prediction model and compare to Su et al.'s prediction model [81]. We train the Su et al. prediction model using the *rigid-ft* data to closely replicate the experiments in [81] and show the benefit of training on diverse sources of data. We ran 5 manipulation trials per object for each method. The initial and desired pose of the object was kept consistent across the two methods.

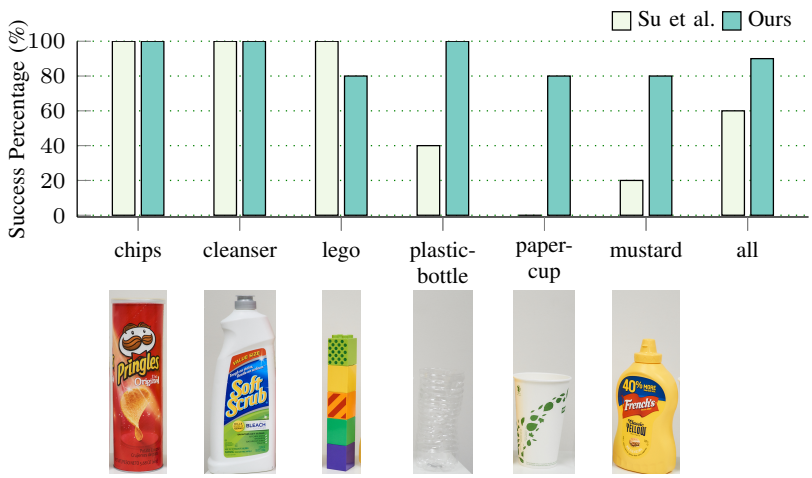


Figure 3.6: Success rates on the manipulation task of object lifting and placement. Our method performs significantly better on the deformable objects *plastic-bottle* and *paper-cup*.

On rigid objects, both methods performed similarly well, as shown in Fig. 3.6. However, on deformable objects *paper-cup* and *plastic-bottle*, our model performs significantly better. This shows our proposed approach has the ability to estimate accurate forces on "never seen" data, as we only collected our dataset on rigid objects. For the *mustard* object, the index fingertip was on the cap, as shown in Fig. 1 for four of the trials to check the predictions on non-flat contact surfaces. We see that only our method was able to detect placement and successfully release the object.

3.7 Discussion

In this chapter, we examined a method for embedding structure within *tactile* perceptual models. By incorporating spatial information of the input-signals directly into the neural architecture, improvement of contact-force predictions across a variety of manipulation tasks was demonstrated. This lends credence to the general idea of introducing practically-motivated inductive bias into our learned perceptual models for robotics applications. We additionally examined the use of task-specific loss functions, which are informed by knowledge of directional friction forces present during data collection and execution. When

trained across different tasks in combination, the approach exhibits significant improvement over baseline models.

The question of integrating structure for perception has been examined for both visual and tactile sensing modalities. In the following chapter, we will address the issue of leveraging structure in the form of *priors* in combining these two sensory input streams within an efficient framework for probabilistic inference and state estimation.

Part II

Priors for Multi-Sensory Integration

CHAPTER 4

JOINT INFERENCE FOR VISUO-TACTILE SENSING

4.1 Introduction & Related Work

Manipulation is a difficult problem, complicated by the challenge of robustly estimating the state of the robot's interaction with the environment. Parameters such as the contact point and the force vector applied at that point, can be very hard to robustly estimate. These parameters are generally partially observable and must be inferred from noisy information obtained via coarse visual or depth sensors and highly sensitive but difficult to interpret tactile sensors.

For instance, in the case of "in-hand" manipulation problems, where a held object is often partially occluded by an end-effector, tactile sensing offers an additional modality that can be exploited to estimate the pose of the object [89].

Vision and tactile sensors have been used to localize an object within a grasp using a gradient-based optimization approach [97]. This has been extended to incorporate constraints like signed-distance field penalties and kinematic priors [89]. However, the former is deterministic and the latter handles uncertainty only per time-step, which is insufficient since sensors can be highly noisy and sensitive. Particle filtering-based approaches have been proposed that can infer the latent belief state from bi-modal and noisy sensory data, to estimate the object pose for two-dimensional grasps [98] and online localization of a grasped object [99]. These approaches are often limited in scope. For example, [99] uses vision to only initialize the object pose and later relies purely on contact information and dynamics models. In general, particle filtering based methods also suffer from practical limitations like computational complexity, mode collapse, and particle depletions in tightly constrained state spaces.

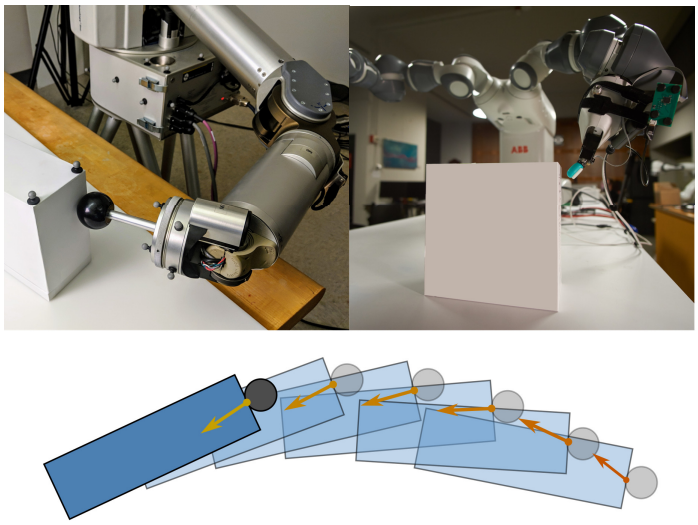


Figure 4.1: Tracking contact dynamics: (Top-left) Pushing probe with Force-Torque sensor on the WAM arm. (Top-right) Yumi robot with mounted biomimetic tactile sensor. (Bottom) Optimized kinematic and force trajectories on a pushed object.

Beyond manipulation, sate estimation is a classic problem in robotics. For example, Simultaneous Localization and Mapping (SLAM) has been studied for many decades, and many efficient tools have been developed to address noisy multi-modal sensor fusion in these domains [100, 101, 102]. One of the more successful tools, the smoothing and mapping (SAM) framework [102], uses factor graphs to perform inference and exploits the underlying sparsity of the estimation problem to efficiently find locally optimal distributions of latent state variables over temporal sequences. This technique offers the desired combination of being computationally fast while accounting for uncertainty over time, and has been recently incorporated into motion planning [103, 104].

This framework has also been explored for estimation during manipulation [105, 106, 107]. In particular, Yu et al. [106] formulate a factor graph of planar pushing interaction (for non-prehensile and underactuated object manipulation) using a simplified dynamics model, with both visual object-pose and force-torque measurements and show improved pose recovery over trajectory histories compared to single-step filtering techniques. However, the scope of [106] is limited to the use of a purpose-built system, equipped with small-diameter

pushing-rods kept at a vertical orientation, allowing for high-fidelity contact-point estimation. A fiducial-based tracking system is also used. Such high precision measurements are impractical in a realistic setting.

In this work, we extend the capabilities of such factor graph inference frameworks in several ways to perform planar pushing tasks in real world settings. We extend the representation to incorporate various geometric and physics-based constraints alongside multi-modal information from vision and tactile sensors. We perform ablation benchmarks to show the benefits of including such constraints, and benchmarks where the vision is occluded or the tactile sensors are very noisy, using data from on our own generalized systems. We conduct our tests on two systems, a dual-arm ABB Yumi manipulator equipped a gel-based Syntouch Biotac tactile sensor [108] and a Barrett WAM arm equipped with a pushing probe end effector mounted with a force torque sensor (Fig. 4.1). Both of these systems are also set up with a vision-based articulated tracking system that leverages a depth camera, joint encoders, and contact-point estimates [89].

Through inference, we jointly estimate the history of not only object poses, and end-effector poses, but also, contact points, and applied force vectors. Estimating contact points and applied force vectors can be very useful in tractable dynamics models to predict future states and can be beneficial to contact-rich planning and control for manipulation [109].

With our experiments, we show that we can contend with a range of multi-modal noisy sensor data and perform efficient inference in batch and incremental settings to provide high-fidelity and consistent state estimates.

4.2 Dynamics of Planar Pushing

In this section, we review the dynamics model for pushing on planar surfaces. The quasi-static approximation of this model is used in the next section to describe the motion model of the pushed object within the factor graph for estimation.

Given an object of mass m being pushed with an applied force f, we can describe the

planar dynamics of the rigid body through the primary equations of motion

$$f + f_\mu = m\ddot{x}_{CM}, \quad \tau + \tau_\mu = I_{CM}\omega \tag{4.1}$$

where x_{CM} is the object position measured at the center-of-mass (CM), ω the angular velocity of the object frame, I_{CM} the moment of inertia, and f_μ the linear frictional force. The applied and frictional moments are defined as $\tau = x_{CM} \times f$ and $\tau_\mu = x_{CM} \times f_\mu$ respectively.

We can estimate the frictional loads on the object by considering the contribution of each point on the support area A of the object [105]. The friction force f_μ and corresponding moment τ_μ is found by integrating Coulomb's law across the contact region of the object with the surface

$$f_\mu = -\mu_s \int_A \frac{v(r)}{|v(r)|} P(r) dA, \quad \tau_\mu = -\mu_s \int_A r \times \frac{v(r)}{|v(r)|} P(r) dA \tag{4.2}$$

where $v(r)$ denotes the linear velocity at a point r in area A, and $P(r)$ the pressure distribution. The coefficient of friction is assumed to be uniform across the support area.

For pusher trajectories that are executed at near-constant speeds, inertial forces can be considered negligible. The push is then said to be quasi-static, where the applied force is large enough to overcome friction and maintain a velocity, but is insufficient to impart an acceleration [110]. Then, the applied force f must lie on the limit surface. This surface is defined in (f_x, f_y, τ) space and encloses all loads under which the object would remain stationary [111]. It can be approximated as an ellipsoid with principal semi-axes f_{max} and τ_{max} [112]

$$\left(\frac{f_x}{f_{max}}\right)^2 + \left(\frac{f_y}{f_{max}}\right)^2 + \left(\frac{\tau}{\tau_{max}}\right)^2 = 1 \tag{4.3}$$

where $f_{max} = \mu_s f_n$, and f_n is the normal force. In order to calculate τ_{max}, we assume a uniform pressure distribution and define r with respect to the center of mass ($r = r_{CM}$):

$\tau_{max} = -\mu_s \frac{mg}{A} \int_A |r_{CM}| dA$. For quasi-static pushing, the velocity is aligned with the frictional load, and therefore must be parallel to the normal of the limit surface. This results in the following constraints on the object motion

$$\frac{v_x}{\omega} = c^2 \frac{f_x}{\tau}, \quad \frac{v_y}{\omega} = c^2 \frac{f_y}{\tau}, \quad \text{and} \quad c = \frac{\tau_{max}}{f_{max}} \tag{4.4}$$

used within our estimation factor graph in the next section.

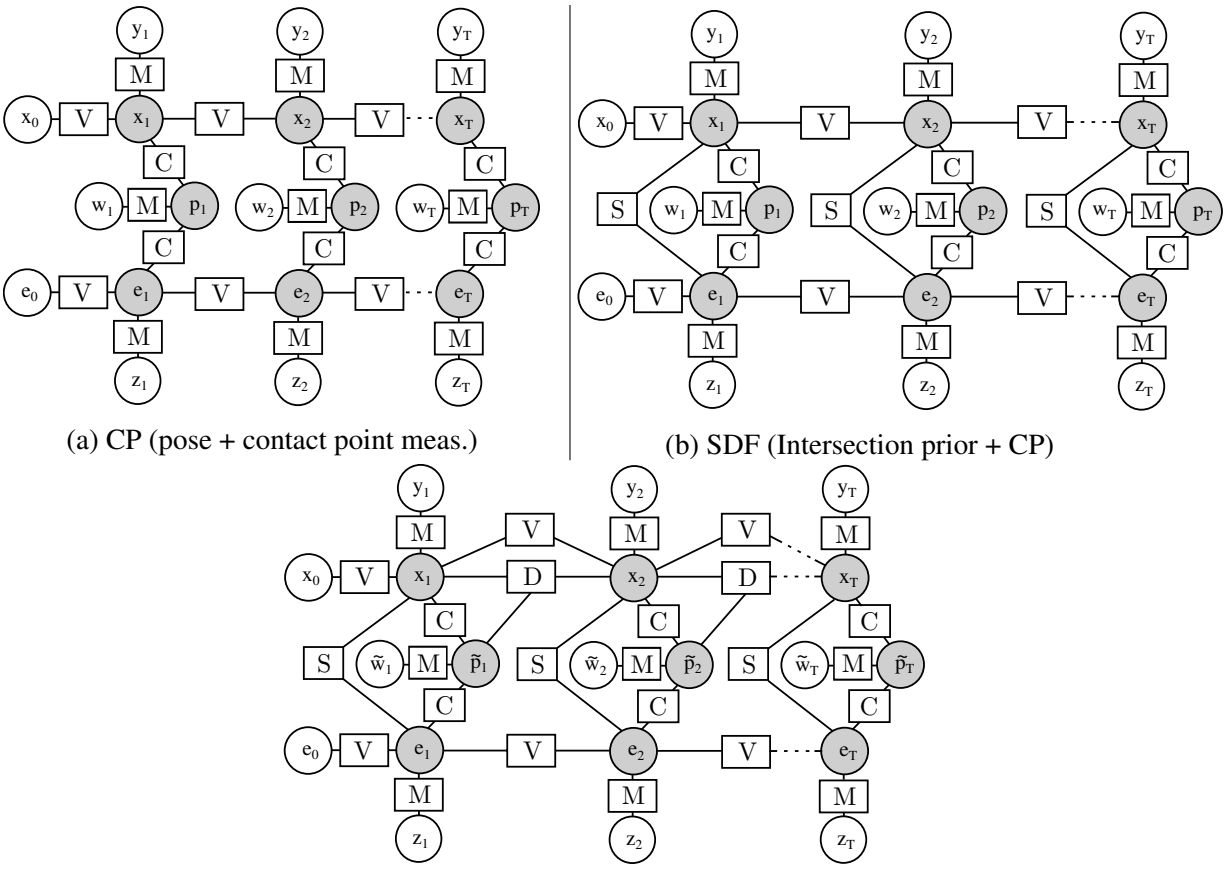
4.3 State Estimation with Factor Graphs

To solve state estimation during manipulation we formulate a factor graph of belief distributions over any state and force vector trajectory and perform inference over the trajectory given noisy sensor measurements. The graph construction and inference is performed with GTSAM [102, 113] via sparsity exploiting nonlinear least squares optimization to find the maximum a posteriori (MAP) trajectory that satisfies all the constraints and measurements. In the batch setting we use a Gauss-Newton optimizer and in an incremental setting we use iSAM2 that performs incremental inference via Bayes trees [114]. All random variables and measurements are assumed to have a Gaussian distribution. In the remainder of this section, we describe the construction of the relevant factor graphs depicted in Fig. 4.2.

We construct three different factor graphs for state estimation in our pushing task: CP, SDF, and QS (Fig. 4.2). All three models include the latent state variables for a given time t: the planar object pose $\mathbf{x}_t \in SE(2)$, the projected end-effector pose $\mathbf{e}_t \in SE(2)$, and the contact point $\mathbf{p}_t \in \mathbb{R}^2$.

4.3.1 Measurements

Each of the latent state variable is accompanied by an associated measurement factor M which projects corresponding measurements from $SE(3)$ into the pushing plane. The object poses are estimated by the visual tracking system with measurements $\mathbf{y}_t \in SE(3)$.



(a) CP (pose + contact point meas.)

(b) SDF (Intersection prior + CP)

(c) QS (Quasi-static dynamics + SDF)

Figure 4.2: Estimation graphs. Filled circles are unknown state variables, unfilled circles are measured values, and squares indicate factors.

Likewise, the end-effector pose measurements $\mathbf{z}_t \in SE(3)$ may be provided from robot forward kinematics, or from the tracking system (DART includes a prior on joint measurements). The contact-point measurements $\mathbf{w}_t \in SE(3)$ are provided by a tactile sensor model. In the QS graph (Fig. 4.2c), we include a new state variable for the applied planar contact force $\mathbf{f}_t \in \mathbb{R}^2$ with corresponding measurements $\boldsymbol{\alpha}_t \in \mathbb{R}^3$. For simplicity of graphical representation, we combine the contact point and force variables:

$$\tilde{\mathbf{p}}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{f}_t \end{bmatrix}, \quad \tilde{\mathbf{w}}_t = \begin{bmatrix} \mathbf{w}_t \\ \boldsymbol{\alpha}_t \end{bmatrix} \tag{4.5}$$

4.3.2 Geometric Constraints

We assume constant point-contact between the end-effector and the object. We include the factor C which incurs a cost on the difference between the contact point $\mathbf{p}_t$ and the closest point to a surface (ξ):

$$\mathrm{C}(\xi,\ \mathbf{p}_t) = G\left(\xi,\ \mathbf{p}_t\right) - \mathbf{p}_t \tag{4.6}$$

where $G\left(\xi,\ \mathbf{p}_t\right)$ is the projection of $\mathbf{p}_t$ onto ξ, and $\xi = \xi\left(\cdot\right)$ returns the surface geometry of a body with a given pose: $\xi = \xi\left(\mathbf{x}_t\right)$ for the object, and $\xi = \xi\left(\mathbf{e}_t\right)$ for the end-effector. Additionally, the object and the end-effector must be prevented from occupying the same region in space. Such a constraint is necessary in practice where contact-point estimation is often noisy. Therefore, we introduce a factor S to penalize intersecting geometries with a signed distance field. Let the point on the end-effector furthest into the object be denoted by $\boldsymbol{\delta} \in \mathbb{R}^2$, where $\boldsymbol{\delta} = \boldsymbol{\delta}\left(\mathbf{x}, \xi\left(\mathbf{e}\right)\right)$. The projection of δ onto $\xi\left(\mathbf{x}\right)$ (the surface of the object) is then defined by $G_\delta = G\left(\boldsymbol{\xi}\left(\mathbf{x}\right), \delta\right)$, and we can apply a penalty

$$\mathrm{S}(\mathbf{x}, \mathbf{e}) = \begin{cases} G_\delta - \boldsymbol{\delta}, & \text{if intersecting} \\ 0, & \text{otherwise} \end{cases}$$

4.3.3 Dynamics

We add a constant velocity prior V to impose smoothness on state transitions. For example, for finite-difference velocities of object poses we have

$$V(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}) = \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\Delta_t} - \frac{\mathbf{x}_{t+1} - \mathbf{x}_t}{\Delta_{t+1}} \tag{4.7}$$

where Δ_t and Δ_{t+1} denote the timestep sizes at t and $t+1$. Similar to [106], we introduce an additional factor D to condition object state transitions on quasi-static pushing. The

corresponding graphical model is denoted by QS and is shown in Fig. 4.2c. From Eq. (4.4) we get

$$D(\mathbf{x}_{t-1}, \mathbf{x}_t, \tilde{\mathbf{p}}_t) = \frac{\mathbf{v}_t}{\omega_t} - c^2 \frac{\mathbf{f}_t}{\tau_t} \tag{4.8}$$

where $\mathbf{v}_t = (\mathbf{x}_{\text{trans},t} - \mathbf{x}_{\text{trans},t-1})/\Delta_t$ and $\omega_t = (\mathbf{x}_{\text{rot},t} - \mathbf{x}_{\text{rot},t-1})/\Delta_{t-1}$ are the finite-difference linear and angular velocity, respectively. The final cost function is optimized with respect to the set of variables $\Phi = \{(\mathbf{x}, \mathbf{e}, \tilde{\mathbf{p}})\}_{t=1}^{t=T}$ over a trajectory of length T :

$$\begin{aligned}
\Phi^* = \arg\min_{\Phi} \sum_{t=1}^{T} \Big\{ & \|\mathrm{D}(\mathbf{x}_{t-1}, \mathbf{x}_t, \tilde{\mathbf{p}}_t))\|^2_{\Sigma_\mathrm{D}} + \|\mathrm{V}(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1})\|^2_{\Sigma_\mathrm{V}} \\
& + \|\mathrm{V}(\mathbf{e}_{t-1}, \mathbf{e}_t, \mathbf{e}_{t+1})\|^2_{\Sigma_\mathrm{V}} + \|\mathrm{C}(\mathbf{x}_t, \mathbf{e}_t)\|^2_{\Sigma_C} + \|C(\tilde{\mathbf{p}}_t, \mathbf{x}_t)\|^2_{\Sigma_C} \\
& + \|C(\tilde{\mathbf{p}}_t, \mathbf{e}_t)\|^2_{\Sigma_C} + \|S(\mathbf{x}_t, \mathbf{e}_t)\|^2_{\Sigma_S} + \|M(\mathbf{x}_t, \mathbf{y}_t)\|^2_{\Sigma_M} \\
& + \|M(\mathbf{e}_t, \mathbf{z}_t)\|^2_{\Sigma_M} + \|M(\tilde{\mathbf{p}}_t, \tilde{\mathbf{w}}_t)\|^2_{\Sigma_M} \Big\}
\end{aligned}$$

The above equation provides the locally optimal *i.e.*MAP solution of the estimation problem.

4.4 Baseline Comparison

In order to first ascertain the general performance of our approach, we evaluate the QS-graph on the MIT planar pushing dataset [115] using batch optimization. This data contains a variety of pushing trajectories for a single-point robotic pushing system. The object poses were tracked with a motion capture system, and contact forces were measured with a pushing probe mounted on a force-torque sensor. We use this data as ground truth, since it is considered to be sufficiently reliable. We restrict our experiments to a subset of this data, using trajectories with zero pushing acceleration and velocities under 10 cm/s in order to maintain approximately quasi-static conditions. Additionally, we only consider trajectories on the ABS surface, but examine different object types (ellip1, rect1, rect3) with approxi-

mately 100 trajectories per object and measurements provided at 100Hz. Gaussian noise is artificially added to the measurements prior to inference, with the following sigma values: $\sigma_{\mathbf{x}_{\text{trans}}} = 0.5\text{cm},\ \sigma_{\mathbf{x}_{\text{rot}}} = 0.5\text{rad},\ \sigma_{\mathbf{e}_{\text{trans}}} = 0.5\text{cm},\ \sigma_{\mathbf{e}_{\text{rot}}} = 0.5\text{rad},\ \sigma_{\mathbf{p}} = 0.5\text{cm},\ \sigma_{\mathbf{f}} = 0.5\text{N}$.

The resulting RMS and covariance values post-optimization are shown in Table 4.1. The optimized values exhibit marked reductions in error compared to the sigma values of the initial measurements. Note that, for object poses we only include values in which the object is in motion, in order to exclude trivial stationary estimates. All position-related values are in cm, with angular values in radians, and forces in Newtons. An example of an optimized trajectory is shown in Fig. 4.3. Although the observation noise is artificial, these results indicate that latent state estimates may still be successfully recovered with the addition of geometric and physics-based priors, and without over-constraining the optimization.

Table 4.1: RMS and Covariance values on the MIT Dataset.

Object	RMS $(\mathbf{x}_{\text{trans}})$	RMS $(\mathbf{x}_{\text{rot}})$	$\Sigma\,(\mathbf{x}_{\text{trans}})$	$\Sigma\,(\mathbf{x}_{\text{rot}})$
ellip1	0.0262	0.283	2.723e-4	4.171e-10
rect1	0.0253	3.471-5	2.931e-4	4.19e-10
rect3	0.0182	1.672e-5	2.563e-4	4.18e-10
Object	RMS $(\mathbf{e}_{\text{trans}})$	RMS $(\mathbf{e}_{\text{rot}})$	$\Sigma\,(\mathbf{e}_{\text{trans}})$	$\Sigma\,(\mathbf{e}_{\text{rot}})$
ellip1	7.73e-2	9.47e-2	4.74e-3	7.11e-3
rect1	8.59e-2	9.18e-2	5.89e-3	6.01e-3
rect3	0.372	0.376	0.148	0.154
Object	RMS $(\|\mathbf{f}\|)$	RMS $(\mathbf{f}_{\text{rot}})$	$\Sigma\,(\|\mathbf{f}\|)$	$\Sigma\,(\mathbf{f}_{\text{rot}})$
ellip1	0.118	9.543e-2	9.827e-3	1.635e-4
rect1	0.145	9.683e-2	9.862e-3	1.823e-4
rect3	0.113	9.754e-2	9.145e-3	1.856e-4
Object	RMS $(\mathbf{p}_{\text{trans}})$	—	$\Sigma\,(\mathbf{p}_{\text{trans}})$	—
ellip1	3.42e-2	—	2.54e-3	—
rect1	4.52e-2	—	6.21e-3	—
rect3	3.26e-2	—	3.41e-3	—

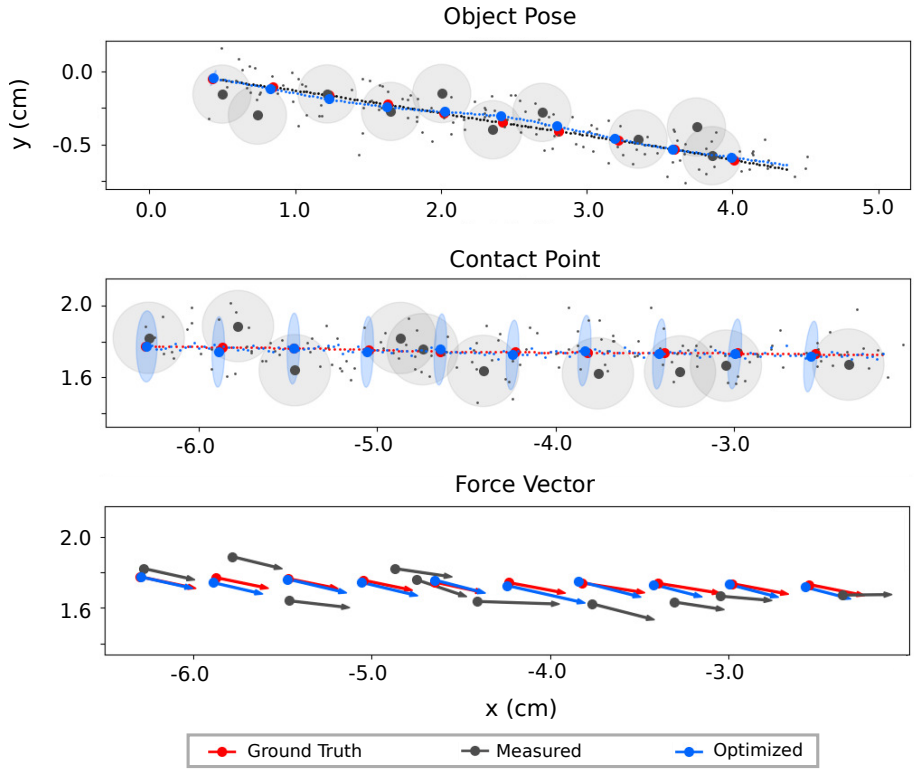


Figure 4.3: Example of performing the inference on a trajectory from the MIT pushing dataset, using the QS graph. Noise is artificially added to measurements prior to smoothing. Two-sigma contours and force vectors are displayed at every 15th time-step for visual clarity.

4.5 State Estimation in Open and Cluttered scenes

We first perform pushing experiments with the Barrett WAM manipulator acting on a laminated box as shown in Fig. 4.4. The system is observed by a stationary PrimeSense depth camera located 2.0m away from the starting push position of the end-effector. Vision-based tracking measurements of the object pose are provided by DART, configured with contact-based priors and joint estimates [89]. The robot is equipped with a Force-Torque sensor and a rigid end-effector mounted with a spherical hard-plastic pushing probe. The contact forces are measured by the F/T sensor, with contact point measurements provided through optimization in DART. Ground-truth poses are provided via a motion-capture system. The table is mounted with a smooth delrin sheet to provide approximately uniform friction across the pushing area.

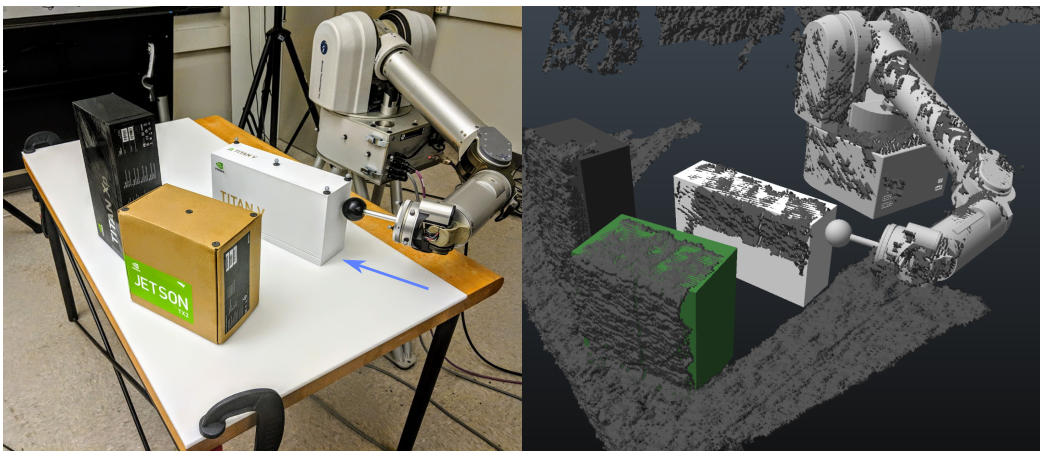
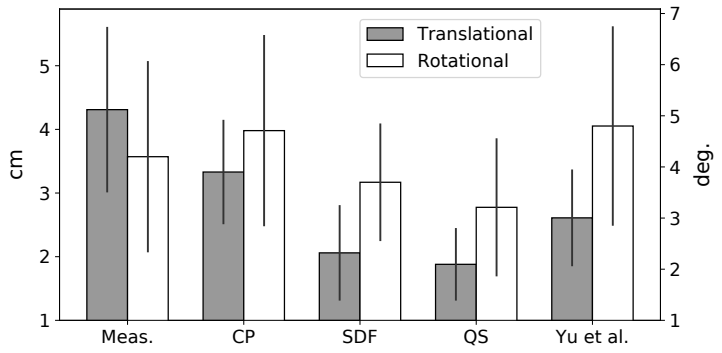


Figure 4.4: Left: Setup for pushing experiments with occlusion using Barrett-WAM manipulator. The white box is the pushed object, with general pushing direction indicated by the blue arrow. The system is observed by a depth camera to the left (out of frame). Right: visualization of the tracked system in DART [116], with the observed pointcloud marked in dark grey.

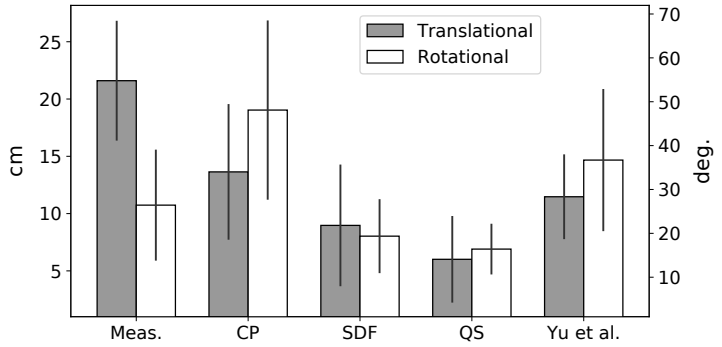
We performed 100 pushing trials with varying initial end-effector and object poses. The end-effector trajectories were varied in curvature and maintained a translational speed close to 6 cm/s to approximate quasi-static conditions. Object pose-tracking measurements were provided at roughly 25Hz, with end-effector poses and force/contact measurements published at 250Hz. Incremental inference of the factor graph is performed after 5 object pose measurements.

In order to provide real-time performance, DART maintains a belief distribution over state at a single timestep. However, this can make tracking susceptible to unreliable measurements that may arise from state-dependent uncertainty or partial observability. As such, we purposely include trajectories in which the object orientation changes significantly with respect to the camera orientation, causing large variations in pointcloud association. In addition, the pushing trajectories were also performed in cluttered scenes, as depicted in Fig. 4.4, with 85% occlusion of the pushing object occurring in the middle of the trajectory.

Examples of measured and estimated state trajectories are shown in Fig. 4.6. In the fully-observable (unocccluded) setting, distinct improvement of the object pose can be seen with both SDF and QS models. Under heavy occlusion, the visual tracking system



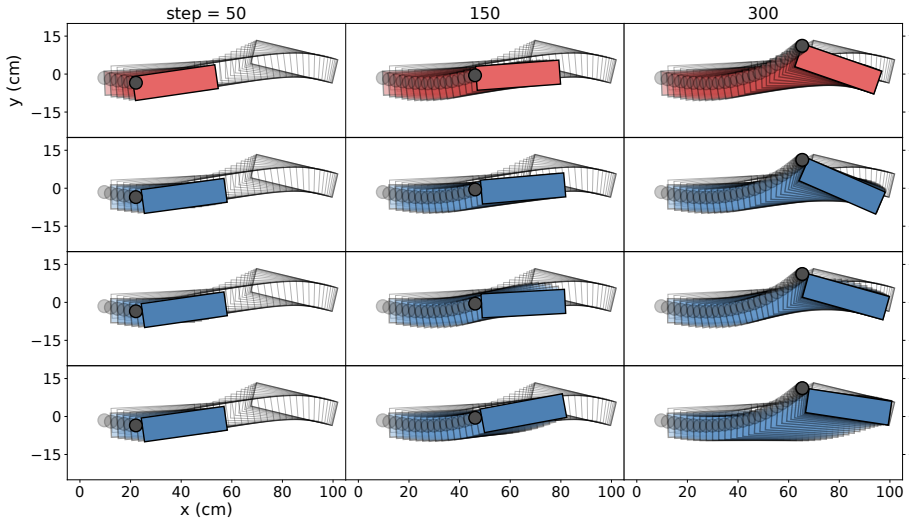
(a) Fully observable



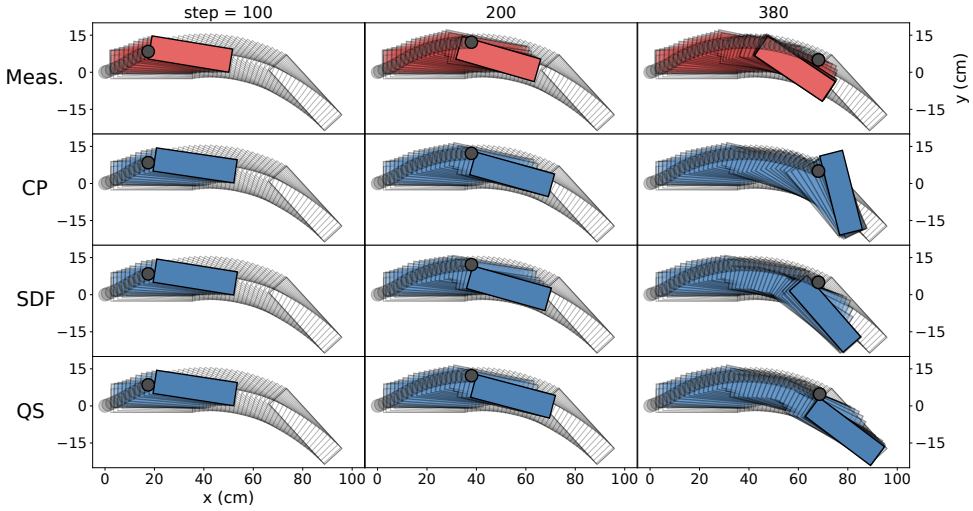
(b) Occluded

Figure 4.5: Mean error and standard deviations of object pose estimates (after the last iSAM2 step has been performed). CP, SDF, and QS model results are compared raw measured values, and to those produced by the graph described in Yu et al. [106]. Tracking performance is greatly improved with the inclusion of geometric and physics-based priors. The comparison with [106], which does not use SDF priors, indicates the importance of enforcing these constraints in practice.

loses track of the object and is unable to regain the trajectory state. Contact-point factors are insufficient for reliable tracking, and can cause object orientation to deviate wildly under occlusion. Incorporating SDF constraints helps to prevent many infeasible poses. The QS graph enforces pose changes which adhere to pushing mechanics. These physics-based priors inform the pose estimates, and stabilize the trajectory even under occlusion. The addition of both geometric and physics based priors to the factor graph result in realignment of the tracked object. Fig. 4.5 shows the tracking performance for fully observable trajectories using the CP, SDF, and QS factor graphs. The results are compared to the model proposed by Yu et al. [106], which includes quasi-static dynamics factors with contact and



(a) Trajectory 1 (fully observable)



(b) Trajectory 2 (with occlusion)

Figure 4.6: Examples of estimated object trajectories for both un-occluded and occluded scenarios. Measured object pose histories (pink) are shown in the top rows, and compared below to the incrementally-optimized trajectories (blue) using the CP, SDF, and QS factor graphs illustrated in Fig. 4.2. Each column depicts the state estimates at a particular timestep (with respect to object pose measurements). The trajectories are overlayed onto the full ground-truth trajectories derived from motion-capture, with every 10 timestep intervals shown. Trajectories of the end-effector (grey circle) are also represented. The measurements show how the tracking system performance degrades under certain orientations, since less of the object is seen as it turns away from the camera.

zero-velocity priors.

In addition to improving inference on kinematic trajectories, the QS graph can be used

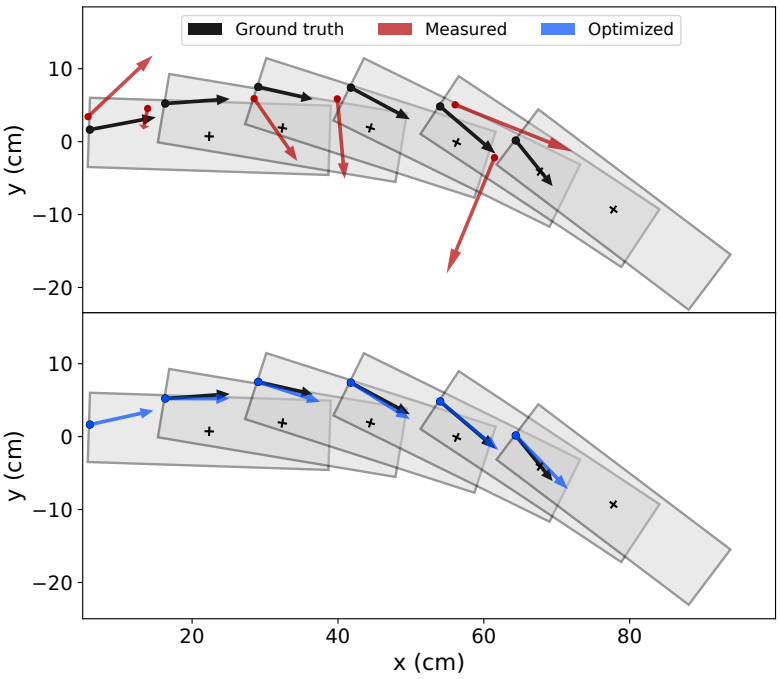
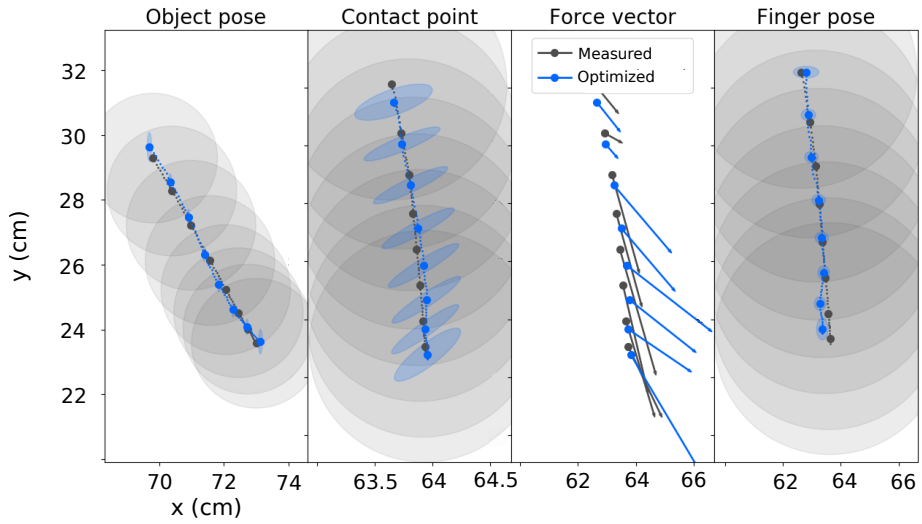


Figure 4.7: Example of force-estimation using the QS model with ground-truth poses and non-Gaussian noise added to force measurements and contact points. Force vectors and contact points are recovered by the optimization process.

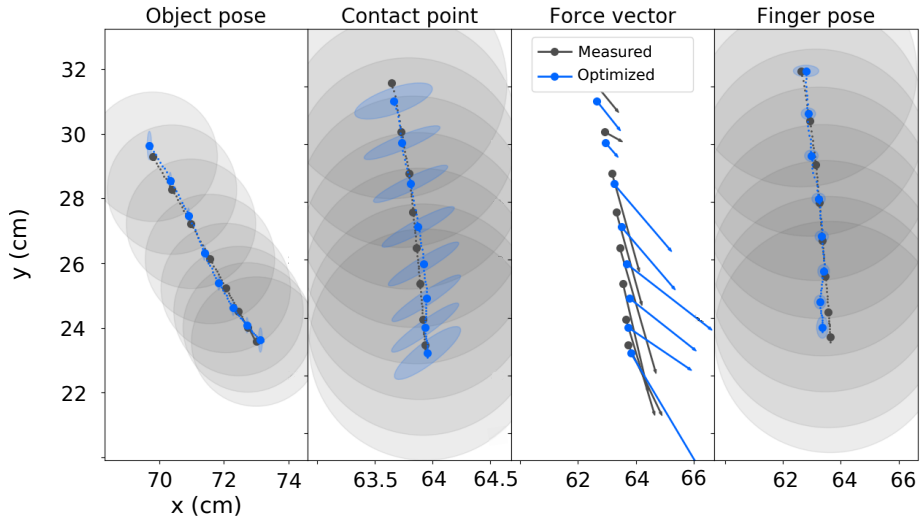
to improve contact point and force estimates. To demonstrate this, we artificially add non-Gaussian noise (bi-modal mixture of two triangular distributions) to contact points and force measurements on the ground-truth data. The resulting estimation errors after optimization are shown in Table 4.2, and indicate that our approach manages to recover true contact points and pushing forces. An an example of force-trajectory optimization is illustrated in Fig. 4.7.

Table 4.2: Error Results for force and contact Recovery

Component	RMSE	MAE	σ
Force magnitude (N)	0.352	0.195	0.043
Force direction (deg.)	3.15	2.54	0.78
Contact location (cm)	0.32	0.14	0.18



(a) Trajectory 1



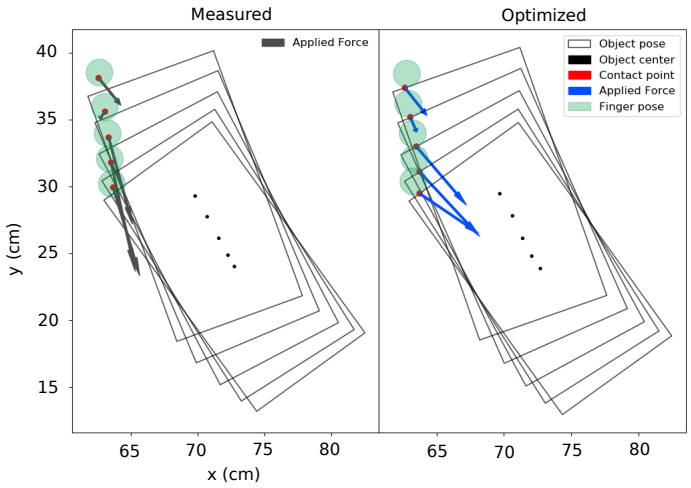
(b) Trajectory 2

Figure 4.8: Examples of pushing trajectories performed on the YUMI system. Initial object and finger pose estimates are provided by the DART tracking system. Contact points and force measurements are estimated by the analytic tactile sensor model [108]. Each trajectory is optimized using the QS graph depicted in Fig. 4.2c. Two-sigma values and force vectors shown at every 10th timestep for visual clarity. Joint inference over kinematic and force trajectories decreases uncertainty in poses as well as contact points and forces, and smoothens noisy tactile data to agree with physics-based constraints.

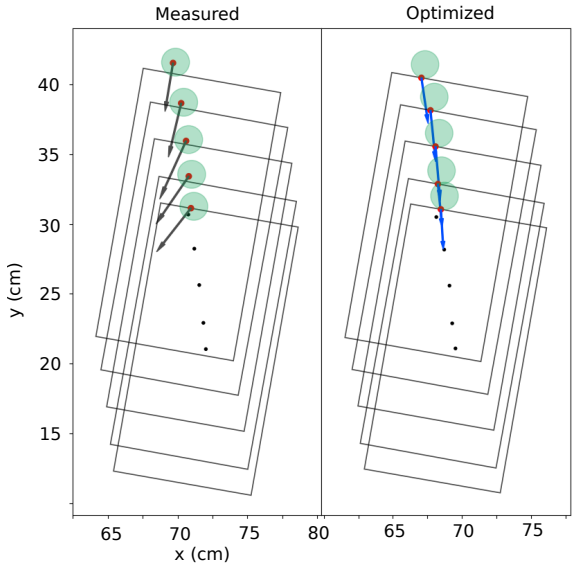
4.6 Force Estimation for Tactile Sensing

We further demonstrate inference on force trajectories using realistic (noisy) tactile data.

The Biotac sensor comprises of a solid core encased in an elastomeric skin and is filled with



(a) Trajectory 1



(b) Trajectory 2

Figure 4.9: Visualizations of measurements for corresponding trajectories in Fig. 4.8. Measured positions, contact points and force-vector outputs from the learned sensor model are shown on the left-hand side. Optimized values are shown on the right, indicating consistency of finger-object surface contact. Our approach produces force trajectories which more closely adhere to quasi-static mechanics. Joint inference allows kinematic trajectories to inform the force estimates, aligning forces to the object center of mass during linear motion, and correcting applied moments when motion is non-linear.

weakly-conductive gel [108]. The core surface is populated by an array of 19 electrodes, each measuring impedance as the thickness of the fluid between the electrode and the skin changes. A transducer provides static pressure readings which consist of a single scalar

value per time-step. This sensor is also equipped with a thermistor for measuring fluid temperature. Although the device does not directly provide a force distribution or contact point measurements, an analytical method for estimating these values is described in [108].

Using an ABB YUMI robot with a mounted Biotac sensor, we generated randomized linear trajectories of the end effector pushing a 0.65 kg box across a laminated surface (Fig. 4.1) starting from a number of different poses. We used the DART tracking system [89] to obtain object and end-effector pose measurements, along with approximate contact points. The analytical force sensor model [108], was used to provide initial force measurements.

Examples of initial and optimized trajectories are shown in Figs. 4.8 and 4.9. The presence of the contact surface factor shrinks the contact point covariance in the direction of push, as is expected. The covariances for finger and object pose estimates are drastically reduced, exhibiting the benefits of joint-inference across trajectory histories. Also, the dynamics factor aligns the force vector in the direction of motion of the object. This is further clarified in Fig. 4.9, where force vectors are correctly aligned with the object center-of-mass for linear trajectories, and provide a moment arm during angular displacement. This demonstrates the importance of contact and geometric factors in aligning the surface tangents of the finger and the object at the point of contact.

4.7 Discussion

We proposed a factor graph-based inference framework to solve estimation problems for robotic manipulation in batch and incremental settings. Our approach can leverage geometric and physics-based constraints in the form of probabilistic priors, along with vision and tactile based multi-modal sensor information to jointly estimate the history of robot and objects poses along with contact locations and force vectors. We perform several benchmarks on various datasets with multiple manipulators in real environments and show that our framework can contend with sensitive, noisy sensor data and occlusions in vision to

efficiently solve for locally optimal state estimates that closely match ground truth. Future work will include incorporating the approach within a motion planning context [104], combining vision and tactile modalities in learning predictive sensor models [14, 24], and the possibility of integration into a hierarchical task-planning framework.

Although the inference tools used in above framework are convenient for fast and online performance, the least-squares optimization objective is equivalent to a maximum-likelihood inference procedure. As such, this general class of probabilistic inference makes strong assumptions on the convexity / uni-modality of the inference problem, and may be susceptible to poor local minima for highly non-convex / multi-modal target functions. To handle this added complexity, we may wish to resolve multiple local solutions by recovering the *full* posterior distribution over inference parameters. This will require inquiry into methods which allow us to add flexibility into how we model target distributions, while considering the issue of scalability for robotics-based applications. Furthermore, we have so far dealt with the problem of inferring and predicting state for *perceptual models*. Next, we will change our focus to predicting distributions over future outcomes for *control and planning* problems common to robotics.

Part III

Variational Inference for Control and Dynamics Estimation

CHAPTER 5

STEIN VARIATIONAL MODEL PREDICTIVE CONTROL

5.1 Introduction

Model predictive control (MPC) is a powerful framework for sequential decision making in robotics [117, 118]. This success can be largely attributed to its simplicity and scalability in dealing with stochastic, non-stationary optimization problems encountered on real systems. MPC has been applied effectively in different areas, including autonomous driving [119, 117, 118], humanoid locomotion [120], and dextrous manipulation [121]. However, common approaches to MPC often fall short in their ability to adequately contend with complex, multi-modal distributions over possible actions. For instance, such distributions may arise from non-convexity of constraints, such as obstacles [122, 123] or from multiple goal locations [124]. Although common sampling-based SOC algorithms have been shown to exhibit symmetry-breaking in the presence of sudden disturbances [125] and multiple optima [126], the sampling scheme may inadequately resolve the true posterior [122]. We require a new class of MPC algorithms that can effectively contend with complex, non-Gaussian distributions.

In the following chapter, we formulate MPC as a Bayesian inference problem, where the target posterior is defined directly over control policy parameters or control inputs, as opposed to joint probabilities over states and actions [127, 128]. By taking this perspective, we can construct a relative-entropy minimization problem to approximate the posterior, and leverage recent advances in variational inference [129, 130, 131] to derive the optimal distribution over parameters. Specifically, we use Stein variational gradient descent (SVGD) [132], to infer a set of solutions which constitute a nonparametric approximation to the posterior distribution over decision parameters, given state observations and a defined cost

function. The generality of this approach offers flexibility in designing appropriate MPC algorithms, and is closely related to common MPC approaches. Additionally, we show how this framework can be extended to general trajectory optimization problems.

5.2 Related Work

The duality between probabilistic inference and optimization for stochastic optimal control (SOC) has been examined extensively in previous work [133, 134, 135, 136]. A wide range of approximate inference methods have been proposed, including Expectation-Propagation (EP) and moment-matching approaches to control and trajectory optimization [119, 135]. These methods typically assume a restricted form of the target posterior distribution over controls (usually in the exponential family), which, under simplifying assumptions, return the optimal control input. However, these distributions are often insufficiently expressive for many SOC problems, where non-convexity may arise from nonlinear dynamics or non-convex cost functions, for instance. Our interpretation of MPC as an approximate inference problem is perhaps most closely related to the formulation presented in [134, 127]. Here, an iterative KL-minimization problem for finite-horizon problems is presented, where the prior is defined to be the control distribution obtained from the previous iteration. However, this does not include a strategy for contending with non-stationary distributions, where the posterior may change between iterations, and the analysis is restricted to exact inference of the log-partition function.

Expectation-Maximization (EM) approaches to SOC have also been considered [137, 138, 139], which iteratively optimize a variational lower-bound. However, these approaches share the limitations inherent with common EM strategies, in that the form of the posterior distribution is assumed to be known and tractable. Additionally, representations based on mixture models must contend with mode-collapse and poor local minima, especially prevalent in higher dimensions [140, 141]. These issues can be mitigated, for example, by introducing entropy regularization heuristics [138].

The Path Integral (PI) formulation of stochastic optimal control [142, 143, 144] bears close resemblance to the open-loop-controls characterization of our proposed approach, where a particular choice of the marginal log-likelihood is assumed to contain an exponentiated cost. In fact, PI attempts to minimize a variational lower-bound with respect to the controlled stochastic dynamics. This follows from the perspective presented here. This comparison applies equally to KL-control [136], the discrete-time counterpart to PI-control.

The application of SVGD for approximate inference in decision making problems has been previously explored in the context of maximum entropy reinforcement learning, where policies are trained using collected experience and updated offline. In [145], the network-based policy is represented using a set of Stein particles to generate a single-step action. However, it is unclear how to appropriately design kernels and evaluate them efficiently in order to counter the effects of increasing dimensionality (*i.e.* network size) using this approach. By contrast, the soft Q-learning algorithm [146] evaluates the SVGD gradient over single-step actions. This is then subsequently backpropagated through the policy to update the parameters. To our knowledge, the non-parametric variational inference approach of [132] has yet to be applied to MPC, or general SOC and planning problems. The dependence on length of the planning horizon poses interesting challenges for scaling and sample-efficient computation. Furthermore, the proposed approach is developed with the online setting in mind, and addresses the problem of rapidly-changing, non-stationary target distributions. Lastly, we make explicit connections to existing SOC methods from a theoretical standpoint.

5.3 Model Predictive Control

We consider the discrete-time stochastic dynamical system: $\mathbf{x}_{t+1} \sim f(\mathbf{x}_t, \mathbf{u}_t)$, where at time t, the system state is denoted by $\mathbf{x}_t \in \mathbb{R}^n$ and the control input as $\mathbf{u}_t \in \mathbb{R}^d$. The stochastic transition map $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ randomly produces the subsequent state $\mathbf{x}_{t+1}$ and this state is accompanied by an instantaneous cost $c(\mathbf{x}_t, \mathbf{u}_t)$.

Over a time horizon H, we define a control trajectory as a sequence of control inputs beginning at time t : $U_t \triangleq (\mathbf{u}_t, \mathbf{u}_{t+1}, ..., \mathbf{u}_{t+H-1})$. Similarly, we define the state trajectory: $X_t \triangleq (\mathbf{x}_t, \mathbf{x}_{t+1}, ..., \mathbf{x}_{t+H-1}, \mathbf{x}_{t+H})$. The total cost incurred over H timesteps can then be specified as

$$C(X_t, U_t) = c_{\text{term}}(\mathbf{x}_{t+H}) + \sum_{h=0}^{H-1} c(\mathbf{x}_{t+h}, \mathbf{u}_{t+h}), \tag{5.1}$$

where $c_{\text{term}}(\cdot)$ is the terminal cost. As in [118], we define an instantaneous feedback policy, $\pi_{\theta_t}(\mathbf{x}_t)$, as a parameterized probability distribution $p(\mathbf{u}_t|\mathbf{x}_t;\ \theta_t)$ used to generate a control input at time t given the $\mathbf{x}_t$, *i.e.*, $\mathbf{u}_t \sim \pi_{\theta_t}(\mathbf{x}_t)$, where $\theta_t \in \Theta$, is the set of feasible parameter values. MPC describes the process of finding the optimal, time-indexed sequence of policy parameters $\boldsymbol{\theta}_t \triangleq (\theta_t, \theta_{t+1}, ..., \theta_{t+H-1})$, which determine the sequence of instantaneous feedback policies $\boldsymbol{\pi}_{\boldsymbol{\theta}_t} \triangleq \left(\pi_{\theta_t}, \pi_{\theta_{t+1}}, ..., \pi_{\theta_{t+H-1}}\right)$. At each time step, we must find $\boldsymbol{\theta}_t$, the parameters that define the optimal policy. We can do this by defining a statistic $J(\cdot)$ on cost $C(X_t, U_t)$ where the minimal $J(\cdot)$ occurs at the optimal $\boldsymbol{\theta}_t$. In real-world situations, the true dynamics function f is often unavailable, and is commonly estimated using a parameterized function $\hat{f}_\xi$ with parameters ξ. As such, we define the surrogate loss function $\hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; \mathbf{x}_t) = \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[C(X_t, U_t)\Big]$. For each MPC-step, the optimal decision is defined as $\boldsymbol{\theta}_t = \text{argmin}_{\boldsymbol{\theta}}\ \hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; \mathbf{x}_t)$ which parameterizes the optimal policy π_{θ_t}, from which we can sample a new control value for the first timestep: $\mathbf{u}_t \sim \pi_{\theta_t}(\mathbf{x}_t) = p(\mathbf{u}_t|\mathbf{x}_t; \theta_t)$. This is then executed on the physical system to generate the next state value: $\mathbf{x}_{t+1} \sim f(\mathbf{x}_t, \mathbf{u}_t)$.

5.4 MPC as Bayesian Inference

Optimal control can be framed as Bayesian inference by considering the distribution over parameters $\boldsymbol{\theta}$. Similarly to [134, 128], we introduce an auxiliary binary random variable $\mathcal{O}_\tau \in \{0, 1\}$ to indicate optimality of the state-action trajectory $\tau = (X_t, U_t)$ with respect to the cost function $C(\cdot)$. Using Bayes' rule, the distribution of parameters $\boldsymbol{\theta}$ conditioned on

the requirement for optimal trajectories ($\mathcal{O}_\tau = 1$) and the current state $\mathbf{x}_t$ can be expressed as

$$p_t(\boldsymbol{\theta}|\mathcal{O}_\tau = 1; \xi, \mathbf{x}_t) = \frac{p_t(\mathcal{O}_\tau = 1|\boldsymbol{\theta}; \xi, \mathbf{x}_t)\; p_t(\boldsymbol{\theta}; \mathbf{x}_t)}{\int p_t(\mathcal{O}_\tau = 1|\boldsymbol{\theta}; \xi, \mathbf{x}_t)\; p_t(\boldsymbol{\theta}; \mathbf{x}_t)\; d\boldsymbol{\theta}}, \tag{5.2}$$

where explicit dependence on state $\mathbf{x}_t$ is included for generality. In the remaining discussion, we will denote $\mathcal{O}_\tau = 1$ simply as $\mathcal{O}_\tau$ without ambiguity. The likelihood $p_t(\mathcal{O}_\tau|\boldsymbol{\theta}; \xi, \mathbf{x}_t)$ is defined as the marginal probability over all possible control and state trajectories:

$$p_t(\mathcal{O}_\tau|\boldsymbol{\theta}; \xi, \mathbf{x}_t) = \int\int p(\mathcal{O}_\tau|X_t, U_t)\; p(X_t, U_t\; |\boldsymbol{\theta}; \xi, \mathbf{x}_t)\; dU_t\; dX_t \tag{5.3}$$

where $p(\mathcal{O}_\tau|X_t, U_t)$ is the probability of optimality given the observed trajectory $\tau = (X_t, U_t)$, and $p(X_t, U_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t)$ is the joint probability of state-control trajectories, conditioned on parameters $\boldsymbol{\theta}$ and assumed dynamics model $\hat{f}_\xi = p_\xi(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. In the discrete-time case, the joint probability can be factorized as

$$p(X_t, U_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t) = \prod_{h=0}^{H-1} p_\xi(\mathbf{x}_{t+h+1}|\mathbf{x}_{t+h}, \mathbf{u}_{t+h})\pi_{\theta_h}(\mathbf{x}_{t+h}), \tag{5.4}$$

where the current state $\mathbf{x}_t$ is observed. If dynamics parameters must also be inferred, the equations can be extended to include ξ as an inference variable, where the posterior is defined over both parameters: $p_t(\boldsymbol{\theta}, \xi|\mathcal{O}_\tau; \mathbf{x}_t) \propto p_t(\mathcal{O}_\tau|\boldsymbol{\theta}, \xi; \mathbf{x}_t)\, p_t(\boldsymbol{\theta}|\xi; \mathbf{x}_t)\, p(\xi)$, suggesting the definition of an alternative latent random variable, such as $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \xi)$.

We can further model $p(\mathcal{O}_\tau|X_t, U_t)$ using a non-negative function $\mathcal{L}(\tau) \propto p(\mathcal{O}_\tau|X_t, U_t)$, which we refer to as the "cost-likelihood". This is defined to be the composition $\mathcal{L} = g \circ C$ of the cost function $C(\cdot)$ with a monotonically decreasing function $g(\cdot)$. The likelihood in Eq. (5.3) then takes the form

$$p_t(\mathcal{O}_\tau|\boldsymbol{\theta}; \xi, \mathbf{x}_t) \propto \int \mathcal{L}(\tau)\; p(\tau\; |\boldsymbol{\theta}; \xi, \mathbf{x}_t)\; d\tau = \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\; \mathcal{L}(\tau)\Big]\; . \tag{5.5}$$

5.5 Nonparametric Bayesian MPC

Instead of performing inference over policy parameters, the Bayesian MPC formulation can be used for inference over control input sequences. In this case, the inference variable $\boldsymbol{\theta}$ is defined as the sequence of open-loop controls: $\boldsymbol{\theta} \triangleq (\mathbf{u}_t, \mathbf{u}_{t+1}, ..., \mathbf{u}_{t+H-1})$. This can be interpreted as a nonparametric version of Bayesian-MPC, as no assumption is made on the existence of a parametrized policy $\boldsymbol{\pi}$. Although the relationship in Eq. (5.2) still holds, the likelihood function $p_t(\mathcal{O}_\tau|\boldsymbol{\theta}; \xi, \mathbf{x}_t)$ must then be re-defined as

$$p_t(\mathcal{O}_\tau|\boldsymbol{\theta}; \xi, \mathbf{x}_t) = \int p(\mathcal{O}_\tau|X_t, \boldsymbol{\theta})\; p(X_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t)\; dX_t \;\propto\; \mathbb{E}_{\hat{f}_\xi}\Big[\; \mathcal{L}(X_t, \boldsymbol{\theta})\Big]\;, \tag{5.6}$$

where $p(X_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t)$ is the probability of state trajectories, conditioned on decisions $\boldsymbol{\theta}$ and assumed dynamics model $\hat{f}_\xi = p_\xi(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. In the discrete-time case, this can be written as the product of state transition probabilities along a trajectory:

$$p(X_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t) = \prod_{h=0}^{H-1} p_\xi(\mathbf{x}_{t+h+1}|\mathbf{x}_{t+h}, \theta_{t+h})\;. \tag{5.7}$$

Furthermore, if we assume a fixed current state ($\mathbf{x}_t = \text{const.}$), we can apply this model to trajectory optimization problems by performing approximate inference on the posterior $p(\boldsymbol{\theta}|\mathcal{O}_\tau; \xi)$, and taking the *maximum a posteriori* estimate, $\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}}\; p(\boldsymbol{\theta}|\mathcal{O}_\tau; \xi)$.

5.6 Variational Inference

Variational inference poses posterior estimation as an optimization task where a candidate distribution $q^*(\boldsymbol{\theta})$ within a distribution family $\mathcal{Q} = \{q(\boldsymbol{\theta})\}$ is chosen to best approximate the target distribution $p(\boldsymbol{\theta}|\mathcal{O}_\tau)$. This is typically obtained by minimizing the Kullback-

Leibler (KL) divergence:

$$q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \; D_{KL}\left(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta}|\mathcal{O}_\tau)\right). \tag{5.8}$$

The solution also maximizes the Evidence Lower Bound (ELBO), as expressed by the following objective:

$$q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmax}} \; \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] - D_{KL}\left(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right) \tag{5.9}$$

This optimization seeks to maximize the log-likelihood of the observations with the first term while penalizing for differences between the target and the prior with the second term. For a high-capacity model space $\mathcal{Q}$ that includes the target distribution, the second term becomes increasingly small. Selecting a model space $\mathcal{Q}$ with both high capacity and computational efficiency is critical to variational inference, but remains a challenging problem.

Proof 1:

$$q^* = \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \; D_{KL}\left(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta}|\mathcal{O}_\tau)\right) \tag{5.10}$$

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \int \log q(\boldsymbol{\theta}) dq(\boldsymbol{\theta}) - \int \log p(\boldsymbol{\theta}|\mathcal{O}_\tau) dq(\boldsymbol{\theta}) \tag{5.11}$$

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \int \log q(\boldsymbol{\theta}) dq(\boldsymbol{\theta}) - \int \log p(\mathcal{O}_\tau|\boldsymbol{\theta}) dq(\boldsymbol{\theta}) - \int \log p(\boldsymbol{\theta}) dq(\boldsymbol{\theta}) + \log Z \tag{5.12}$$

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \int \log q(\boldsymbol{\theta}) dq(\boldsymbol{\theta}) - \int \log p(\mathcal{O}_\tau|\boldsymbol{\theta}) dq(\boldsymbol{\theta}) - \int \log p(\boldsymbol{\theta}) dq(\boldsymbol{\theta}) \tag{5.13}$$

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmin}} \; \mathbb{E}_q\Big[\log q(\boldsymbol{\theta})\Big] - \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\Big] \tag{5.14}$$

$$= \underset{q \in \mathcal{Q}}{\operatorname{argmax}} \; \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] - D_{KL}\left(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right) \tag{5.15}$$

where Z is the normalizing constant. The functional in the objective is also known as the

variational free energy $F_q(\boldsymbol{\theta}) := \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] - D_{KL}\left(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right)$. An alternative derivation can be found by deriving the lower-bound to the log marginal-likelihood:

Proof 2:

$$\log \mathbb{E}_p\Big[p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] = \log \int p(\mathcal{O}_\tau|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{5.16}$$

$$= \log \int p(\mathcal{O}_\tau|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{5.17}$$

$$= \log \int p(\mathcal{O}_\tau|\boldsymbol{\theta})\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}q(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{5.18}$$

$$= \log \mathbb{E}_q\Big[p(\mathcal{O}_\tau|\boldsymbol{\theta})\frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\Big] \tag{5.19}$$

$$\geq \mathbb{E}_q\Big[\log \frac{p(\mathcal{O}_\tau|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\Big] \tag{5.20}$$

$$= \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] - \mathbb{E}_q\Big[\log \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\Big] \tag{5.21}$$

$$= \mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] - D_{KL}\left(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right) \tag{5.22}$$

where Eq. (5.20) results from log-concavity and application of Jensen's inequality. Equality is obtained when q matches the posterior probability. In other words, provided that $p(\boldsymbol{\theta}|\mathcal{O}_\tau) \in \mathcal{Q}$, we have:

$$q^*(\boldsymbol{\theta}) = \frac{p(\mathcal{O}_\tau|\boldsymbol{\theta})\; p(\boldsymbol{\theta})}{\int p(\mathcal{O}_\tau|\boldsymbol{\theta})\; p(\boldsymbol{\theta})d\boldsymbol{\theta}} = p(\boldsymbol{\theta}|\mathcal{O}_\tau) \tag{5.23}$$

As such, we can define the minimization:

$$-\log \mathbb{E}_p\Big[p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] = \min_{q\in\mathcal{Q}} -\mathbb{E}_q\Big[\log p(\mathcal{O}_\tau|\boldsymbol{\theta})\Big] + D_{KL}\left(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta})\right) \tag{5.24}$$

5.7 Stein Variational Gradient Descent

In order to circumvent the challenge of determining an appropriate $\mathcal{Q}$, while also addressing Eq. (5.8), we develop an algorithm based on Stein variational gradient descent (SVGD) for

Bayesian inference. The nonparametric nature of SVGD is advantageous as it removes the need for assumptions on restricted parametric families for q. This approach approximates a posterior $p(\boldsymbol{\theta}|x)$ with a set of particles $\{\boldsymbol{\theta}^i\}_{i=1}^m$, $\boldsymbol{\theta}^i \in \mathbb{R}^p$. The particles are iteratively updated according to

$$\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i + \epsilon \boldsymbol{\phi}^*(\boldsymbol{\theta}^i) \tag{5.25}$$

given a step-size ϵ. The function $\boldsymbol{\phi}^*(\cdot)$ lies in the unit-ball of an $\mathbb{R}^p$-valued reproducing kernel Hilbert space (RKHS) of the form $\mathcal{H} = \mathcal{H}_0 \times ...\mathcal{H}_0$, where $\mathcal{H}_0$ is a scalar-valued RKHS with kernel $k(\boldsymbol{\theta}', \boldsymbol{\theta})$. It characterizes the optimal perturbation or velocity field (i.e. gradient direction) which maximally decreases the KL-divergence:

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi} \in \mathcal{H}}{\operatorname{argmax}} \Big\{ -\nabla_\epsilon D_{KL}\left(q_{[\epsilon\boldsymbol{\phi}]} || p(\boldsymbol{\theta}|x)\right) \text{ s.t. } \|\boldsymbol{\phi}\|_{\mathcal{H}} \leq 1 \Big\}, \tag{5.26}$$

where $q_{[\epsilon\boldsymbol{\phi}]}$ indicates the particle distribution resulting from taking an update step $\boldsymbol{\theta} = \boldsymbol{\theta} + \epsilon\boldsymbol{\phi}(\boldsymbol{\theta})$. This has been shown to yield a closed-form solution [132] which can be interpreted as a functional gradient in RKHS and approximated with the set of particles:

$$\hat{\boldsymbol{\phi}}^*(\boldsymbol{\theta}) = \frac{1}{m}\sum_{j=1}^{m} \Big[k(\boldsymbol{\theta}^j, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}^j} \log p(\boldsymbol{\theta}^j || x) + \nabla_{\boldsymbol{\theta}^j} k(\boldsymbol{\theta}^j, \boldsymbol{\theta}) \Big]. \tag{5.27}$$

Eq. (5.27) has two terms that control different aspects of the algorithm. The first term is essentially a scaled gradient of the log-likelihood over the posterior's particle approximation. The second term is known as the *repulsive force*. Intuitively, it pushes particles apart when they get too close to each other and prevents them from collapsing into a single mode. This allows the method to approximate complex, possibly multi-modal posteriors in MPC. When there is only a single particle, the method reduces to a standard optimization of the log-likelihood or a MAP estimate of the posterior as the repulsive force term vanishes, *i.e.* $\nabla_{\boldsymbol{\theta}} k(\boldsymbol{\theta}, \boldsymbol{\theta}) = 0$. SVGD's optimization structure empirically provides better

particle efficiency than other popular sampling procedures, such as Markov Chain Monte Carlo [147].

5.8 Stein Variational MPC

In this section we present our novel method for Stein inference, specifically designed around MPC requirements. The full algorithm can is outlined in Algorithm 1.

5.8.1 Posterior Sequential Updates

The Bayesian interpretation of MPC seeks to find the posterior distribution over decision parameters at time t. Recalling Eq. (5.2):

$$p_t(\boldsymbol{\theta}|\mathcal{O}_\tau;\xi,\mathbf{x}_t) = \frac{p_t(\mathcal{O}_\tau|\boldsymbol{\theta};\xi,\mathbf{x}_t)\ \tilde{q}_t(\boldsymbol{\theta};\mathbf{x}_t)}{\int p_t(\mathcal{O}_\tau|\boldsymbol{\theta};\xi,\mathbf{x}_t)\ \tilde{q}_t(\boldsymbol{\theta};\mathbf{x}_t)\ d\boldsymbol{\theta}} \quad . \tag{5.28}$$

with a prior $\tilde{q}_t(\boldsymbol{\theta};\mathbf{x}_t)$. Our approach approximates the posterior over decision parameters using a *weighted* set of particles $\{\boldsymbol{\theta}^i\}_{i=1}^m$, where the proposal q is defined as the empirical distribution $q(\boldsymbol{\theta}) = \sum_{i=1}^m w^i \delta(\boldsymbol{\theta}^i)$ with weights evaluated according to:

$$w^i = \frac{p_t(\mathcal{O}_\tau|\boldsymbol{\theta}^i;\xi,\mathbf{x}_t)\ \tilde{q}_t(\boldsymbol{\theta};\mathbf{x}_t)}{\sum_{j=1}^m p_t(\mathcal{O}_\tau|\boldsymbol{\theta}^j;\xi,\mathbf{x}_t)\ \tilde{q}_t(\boldsymbol{\theta};\mathbf{x}_t)} \tag{5.29}$$

such that $\sum_{i=1}^m w^i = 1$. Following the procedure outlined in Eq. (5.25)-Eq. (5.27), an SVGD update can be computed for the individual particles by computing the functional gradient

$$\hat{\boldsymbol{\phi}}^*(\boldsymbol{\theta}^i) = \frac{1}{m}\sum_{j=1}^m \Big[k(\boldsymbol{\theta}^j,\boldsymbol{\theta}^i)\nabla_{\boldsymbol{\theta}^j}\log p_t(\boldsymbol{\theta}^j|\mathcal{O}_\tau;\xi,\mathbf{x}_t) + \nabla_{\boldsymbol{\theta}^j}k(\boldsymbol{\theta}^j,\boldsymbol{\theta}^i)\Big] \tag{5.30}$$

and performing the gradient step: $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i + \epsilon\hat{\boldsymbol{\phi}}^*(\boldsymbol{\theta}^i)$. The evaluation of Eq. (5.30) requires computation of the log-posterior gradient, which can be written as the sum of the gradients

of both the log-prior and log-likelihood:

$$\nabla_{\boldsymbol{\theta}^i} \log p_t(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi,\mathbf{x}_t) = \nabla_{\boldsymbol{\theta}^i} \log p_t(\mathcal{O}_\tau|\boldsymbol{\theta}^i;\xi,\mathbf{x}_t) + \nabla_{\boldsymbol{\theta}^i} \log \tilde{q}_t(\boldsymbol{\theta}^i;\mathbf{x}_t) \tag{5.31}$$

$$= \nabla_{\boldsymbol{\theta}^i} \log \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}^i},\hat{f}_\xi}\Big[\ \mathcal{L}(\tau)\Big] + \nabla_{\boldsymbol{\theta}^i} \log \tilde{q}_t(\boldsymbol{\theta}^i;\mathbf{x}_t) \tag{5.32}$$

The first RHS term requires that we define the cost-likelihood function $\mathcal{L}$. The SV-MPC framework allows for different possible definitions for $\mathcal{L}$. We will consider two in particular:

- Exponentiated Utility (EU): $\mathcal{L}(\tau) = \exp\left(-\alpha C(X_t,U_t)\right), \quad \text{where } \alpha > 0$ (5.33)
- Probability of Low Cost (PLC): $\mathcal{L}(\tau) = \mathbb{1}_{C\leq C_{\max}}(C(X_t,U_t))$ (5.34)

It is generally assumed that the cost function $C(X_t,U_t)$ is non-differentiable with respect to the decision parameter $\boldsymbol{\theta}$, and that resulting expectations are difficult to evaluate analytically. As such, the gradients can be estimated via Monte-Carlo sampling, where a set of N control and state trajectory samples are drawn from the policy and the modeled dynamics: $\{\tau^s\}_{s=1}^N \sim p(X_t,U_t|\boldsymbol{\theta};\xi,\mathbf{x}_t)$, where $\tau^s = (X_t^s, U_t^s)$. This leads to the following approximation:

$$\nabla_{\boldsymbol{\theta}^i} \log \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}^i},\hat{f}_\xi}\Big[\ \mathcal{L}(\tau)\Big] = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}^i},\hat{f}_\xi}\Big[\mathcal{L}(\tau)\nabla_{\boldsymbol{\theta}^i}\log\boldsymbol{\pi}_{\boldsymbol{\theta}^i}\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}^i},\hat{f}_\xi}\Big[\mathcal{L}(\tau)\Big]} \tag{5.35}$$

$$\approx \frac{\sum_{s=1}^N \mathcal{L}(\tau^s)\nabla_{\boldsymbol{\theta}^i}\log\boldsymbol{\pi}_{\boldsymbol{\theta}^i}(U_t^s)}{\sum_{s=1}^N \mathcal{L}(\tau^s)} \tag{5.36}$$

For a single particle ($m = 1$), the full gradient in Eq. (5.30) reduces to : $\boldsymbol{\phi}^*(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log p_t(\boldsymbol{\theta}|\mathcal{O}_\tau;\xi,\mathbf{x}_t)$. SVGD then produces a local MAP estimate of the posterior distribution over $\boldsymbol{\theta}$. As a result, the SV-MPC update step exhibits strong similarity to common MPC algorithms (such as MPPI [117] and CEM [148]) depending on the chosen likelihood function, exhibiting equivalence under parameter values and choice of prior.

5.8.2 Exponentiated Utility (EU)

By picking the cost-likelihood to be: $\mathcal{L}(\tau) = \exp\left(-\alpha C(X_t, U_t)\right)$, we obtain the likelihood function

$$\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\ \mathcal{L}(\tau)\Big] = \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]. \tag{5.37}$$

This is otherwise known as the Free-Energy of the cost function $C(\mathbf{x}_t, \mathbf{u}_t)$ [143], as well as the "soft maximum" or "risk-aware" loss. This yields the likelihood-gradient:

$$\nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\ \mathcal{L}(\tau)\Big] = \nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big] \tag{5.38}$$

$$= \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi}_{\boldsymbol{\theta}}\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} \tag{5.39}$$

As $\alpha \to \infty$, regions of high-cost are assigned lower probability, making the distribution of resulting policies risk-averse. Conversely, as $\alpha \to 0$, high-cost regions have higher likelihood, making the policy distributions more risk-seeking.

With this form of the gradient, we can choose a control policy as a sequence of Gaussian distributions over open-loop controls with fixed covariance: given that $\boldsymbol{\pi}_{\boldsymbol{\theta}_t} = \left(\pi_{\theta_t}, \pi_{\theta_{t+1}}, ..., \pi_{\theta_{t+H-1}}\right)$, we have the instantaneous policy $\pi_{\theta_t} = \mathcal{N}(\mu_t, \Sigma_t)$, our decision parameter is thus $\boldsymbol{\theta}_t = \boldsymbol{\mu}_t = (\mu_t, \mu_{t+1}, ...\mu_{t+H-1})$, where $\mu_t \in \mathbb{R}^d$, $\Sigma_t \in \mathbb{R}^{d\times d}$. Considering the SV-MPC step in Eq. (5.30) for a single particle, we can derive the update for a parameter element $h \in (t, t+1, ..., t+H-1)$:

$$\theta'_h = \theta_h + \epsilon\Big(\nabla_{\theta_h} \log p_t(\mathcal{O}_\tau \mid \boldsymbol{\theta}; \xi, \mathbf{x}_t) + \nabla_{\theta_h} \log \tilde{q}_t(\boldsymbol{\theta})\Big) \tag{5.40}$$

$$= \theta_h + \epsilon \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) \nabla_{\theta_h} \log \pi_{\theta_h}\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} + \epsilon \nabla_{\theta_h} \log \tilde{q}_t(\boldsymbol{\theta}) \tag{5.41}$$

$$= \theta_h + \epsilon\, \Sigma_t^{-1} \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) (\mathbf{u}_h - \theta_h)\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} + \epsilon \nabla_{\theta_h} \log \tilde{q}_t(\boldsymbol{\theta}) \tag{5.42}$$

$$= (I - \epsilon\, \Sigma_t^{-1})\, \theta_h + \epsilon\, \Sigma_t^{-1} \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) \mathbf{u}_h\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} + \epsilon \nabla_{\theta_h} \log \tilde{q}_t(\boldsymbol{\theta}) \tag{5.43}$$

where the gradient of the log-prior will depend on the particular choice of transition probability, $p_t(\boldsymbol{\theta}|\tilde{\boldsymbol{\theta}}; \mathbf{x}_{t+1})$, used in the shift operation (Section 5.8.6). The update reduces to that found in MPPI [117] if we consider the control distribution to be uncorrelated across dimensions (as is often done in practice): $\Sigma_t = \sigma_t^2 I$. Setting the step-size to $\epsilon = \sigma^2$, and assuming a uniform prior on controls, $p_t(\theta|\tilde{\theta}; \mathbf{x}_{t+1}) = \mathcal{U}(\theta_{min}, \theta_{max})$:

$$\theta'_h = (I - \sigma_t^2\, \sigma_t^{-2} I)\, \theta_h + \sigma_t^2\, \sigma_t^{-2} I \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) \mathbf{u}_h\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} + 0 \tag{5.44}$$

$$= \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right) \mathbf{u}_h\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{f}_\xi}\Big[\exp\left(-\alpha C(X_t, U_t)\right)\Big]} \quad , \tag{5.45}$$

recovering the MPPI update rule.

5.8.3 Probability of Low Cost (PLC)

We can incorporate a threshold-utility to indicate preference for costs below a given threshold, using the indicator function: $\mathcal{L}(\tau) = \mathbb{1}_{C \leq C_{t,\max}}(C(X_t, U_t))$. The likelihood then takes

the form

$$\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\ \mathcal{L}(\tau)\Big] = \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\mathbb{1}_{C\leq C_{t,\max}}(C(X_t,U_t))\Big]. \tag{5.46}$$

with the resulting gradient:

$$\nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\ \mathcal{L}(\tau)\Big] = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\mathbb{1}_{C\leq C_{t,\max}}(C(X_t,U_t))\nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi}_{\boldsymbol{\theta}}\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\mathbb{1}_{C\leq C_{t,\max}}(C(X_t,U_t))\Big]} \tag{5.47}$$

The threshold parameter $C_{t,\max}$ is set adaptively as the largest cost of the top member in the elite fraction of sampled trajectories. Using the same derivation for the case of Exponentiated Utility likelihood, the choice of a Gaussian policy with a threshold-utilty reduces to the update rule for the Cross Entropy Method [148]:

$$\theta'_h = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\mathbb{1}_{C\leq C_{t,\max}}(C(X_t,U_t))\mathbf{u}_h\Big]}{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\theta}},\hat{f}_{\xi}}\Big[\mathbb{1}_{C\leq C_{t,\max}}(C(X_t,U_t))\Big]} \quad . \tag{5.48}$$

A significant advantage of the SV-MPC formulation is the robustness to highly-peaked posterior distributions. If particles are initialized poorly, or if the target posterior changes significantly between time-steps, many particles may find themselves in regions of low-probability. Indeed, this may occur frequently for likelihoods $p(\mathcal{O}_\tau|\boldsymbol{\theta}^i;\xi,\mathbf{x}_t)$ with exponentiated cost (see Section 5.8.2, for example). However, the shared gradient terms in Eq. (5.30) allow these particles to overcome this degeneracy quickly, while avoiding collapse due to the repulsive term (the reader may refer to Figure 1. in [149] for an intuitive illustration of this phenomenon.) As a consequence, SV-MPC avoids the problem of particle depletion often encountered in Sequential Monte Carlo methods [150].

5.8.4 Kernels for trajectories

High-dimensional inference problems pose significant challenges for SVGD as the repulsive force given by the derivative of the kernel with respect to the inputs diminishes as the dimensionality increases [151]. Inspired by probabilistic graphical models and the conditional independence assumptions encoded in Markov random fields, we tackle this issue by devising a kernel that factorizes a high-dimensional input into a sum of kernels defined over cliques of dimensions. This allows the exploitation of the Markov structure of the trajectories to address the curse of dimensionality. For example, assume that the posterior over the parameters $\boldsymbol{\theta}$ satisfies the conditional independence relations encoded in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with vertices $\mathcal{V}$, and edges $\mathcal{E}$ such that $p(\boldsymbol{\theta}) \propto \prod_{d \in \mathcal{E}} \psi_d(\theta_d) \prod_{(dt) \in \mathcal{E}} \psi_{dt}(\theta_d, \theta_t)$, where $\psi_d(\theta_d)$ and $\psi_{dt}(\theta_d, \theta_t)$ are unary and pairwise potential functions respectively. We define the kernel over parameters as,

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{d \in \mathcal{V}} k(\theta_d, \theta_d) + \sum_{(d,t) \in \mathcal{E}} k(\theta_{(d,t)}, \theta'_{(d,t)}) \tag{5.49}$$

The kernel is a sum of positive semi-definite kernels, so the result is a valid reproducing kernel Hilbert space [152] but less sensitive to the curse of dimensionality. In this paper we adopt the smooth RBF kernel; $k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp\{-\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2 / h\}$, where h is evaluated using the median heuristic on the set of particles: $h = \operatorname{med}\left(\{\boldsymbol{\theta}^i\}\right)^2 / \log m$.

5.8.5 Action Selection

The variational inference procedure results in an approximation of the posterior distribution over $\boldsymbol{\theta}$. Following this step, a decision must be made on $\boldsymbol{\theta}_t$, such that a control action can be generated from the resulting policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_t}$ and executed on the real system. Here we outline two possible methods for choosing an appropriate $\boldsymbol{\theta}_t$.

We can first consider the relative probabilistic weight of the particles as an approxima-

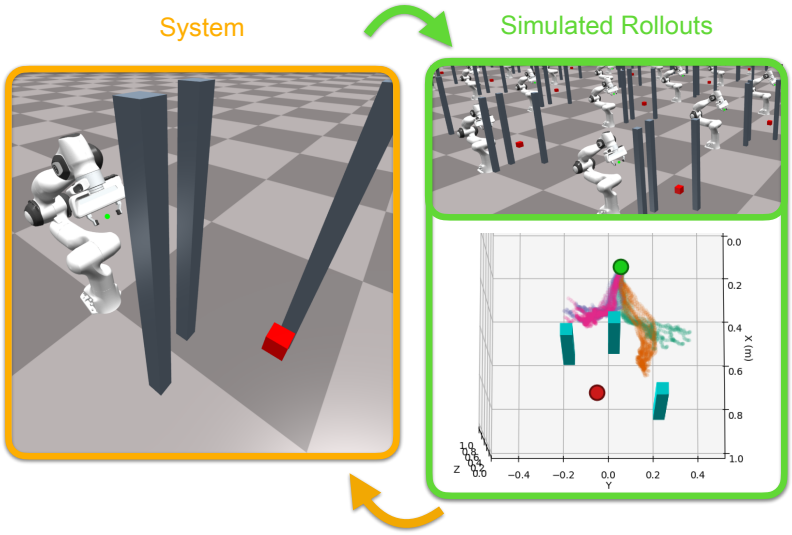


Figure 5.1: A 7-dof reaching task. The SV-MPC framework is capable of reasoning over multi-modal distributions of trajectories in high-dimensional spaces. Here, the controller iteratively explores the posterior over joint-velocities by simulating trajectories in parallel (green frame) in order to guide the system (orange frame). Each particle-generated distribution is shown by a unique coloring over the generated state trajectories, as seen from a top-down view of the workspace. The robot arm manages to reach the goal (red), while avoiding poor local minima.

tion to their posterior probabilities:

$$w_i = \frac{p(\mathcal{O}_\tau|\boldsymbol{\theta}^i;\xi,\mathbf{x}_t)\;\tilde{q}_{t-1}(\boldsymbol{\theta}^i)}{\sum_{j=1}^{m} p(\mathcal{O}_\tau|\boldsymbol{\theta}^j;\xi,\mathbf{x}_t)\;\tilde{q}_{t-1}(\boldsymbol{\theta}^j)} \tag{5.50}$$

$$\approx p(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi,\mathbf{x}_t). \tag{5.51}$$

One strategy to selecting $\boldsymbol{\theta}_t$ is to pick the highest-weighted particle $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{i^*}$, which corresponds to the approximate MAP solution:

$$i^* = \underset{i}{\text{argmax}}[w_i] \tag{5.52}$$

$$\approx \underset{i}{\text{argmax}}\, p(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi,\mathbf{x}_t). \tag{5.53}$$

An alternative approach is to randomly sample from the posterior distribution, which can be approximated by sampling from the set of particles according to their weight w_i. This

Algorithm 1: Stein Variational MPC

* Components which apply only in the presence of a parameterized policy π are marked in red.

Input: Initial state $\mathbf{x}_0$, dynamics $\hat{f}_\xi$, cost-likelihood $\mathcal{L}$, prior $p_0(\boldsymbol{\theta}; \cdot)$, kernel k, policy π

Initialize $\tilde{q}_0(\boldsymbol{\theta}) = p_0(\boldsymbol{\theta}; \mathbf{x}_0)$
Sample $\{\boldsymbol{\theta}^i\}_{i=1}^m \sim \tilde{q}_0(\boldsymbol{\theta})$
for $t = 0, 1, ..., T-1$ **do**
  for $i = 1, 2, ..., m$ **do in parallel**
    $\nabla_{\boldsymbol{\theta}^i} \log p_t(\boldsymbol{\theta}^i | \mathcal{O}_\tau; \xi, \mathbf{x}_t) = \nabla_{\boldsymbol{\theta}^i} \log \mathbb{E}_{\pi_{\boldsymbol{\theta}^i}, \hat{f}_\xi}\Big[\mathcal{L}(\tau)\Big] + \nabla_{\boldsymbol{\theta}^i} \log \tilde{q}_t(\boldsymbol{\theta}^i)$
  end
  for $i = 1, 2, ..., m$ **do in parallel**
    $\Delta\boldsymbol{\theta}^i \leftarrow \frac{1}{m} \sum_{j=1}^m k(\boldsymbol{\theta}^j, \boldsymbol{\theta}^i) \nabla_{\boldsymbol{\theta}^j} \log p_t(\boldsymbol{\theta}^j | \mathcal{O}_\tau; \xi, \mathbf{x}_t) + \nabla_{\boldsymbol{\theta}^j} k(\boldsymbol{\theta}^j, \boldsymbol{\theta}^i)$
    $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i + \epsilon \Delta\boldsymbol{\theta}^i$
  end
  $w^i \leftarrow p(\mathcal{O}_\tau | \boldsymbol{\theta}^i; \mathbf{x}_t) \tilde{q}_t(\boldsymbol{\theta}^i)$
  $w^i \leftarrow \frac{w^i}{\sum_{j=1}^m w^j}$
  Pick $\boldsymbol{\theta}^*$ (using Eq. (5.50) or Eq. (5.52), for example)
  Get control input: $\mathbf{u}_t \leftarrow \boldsymbol{\theta}^*$ or sample $\mathbf{u}_t \sim \pi_{\boldsymbol{\theta}^*_t}(\mathbf{x}_t)$
  Sample true dynamics: $\mathbf{x}_{t+1} \sim f(\mathbf{x}_t, \mathbf{u}_t)$
  Shift particles: $\tilde{\boldsymbol{\theta}}^i = \Phi(\boldsymbol{\theta}^i)$
  Update prior: $\tilde{q}_{t+1}(\boldsymbol{\theta}; \mathbf{x}_{t+1}) = \sum_{i=1}^m w^i p_t(\boldsymbol{\theta} | \tilde{\boldsymbol{\theta}}^i; \mathbf{x}_{t+1})$
end

can be performed by drawing from the categorical distribution over particle weights:

$$i^* \sim \text{Cat}(w_i). \tag{5.54}$$

5.8.6 Shifting the distribution

The prior $\tilde{q}_t(\boldsymbol{\theta}; \mathbf{x}_t)$ is obtained by an operation akin to the prediction step commonly found in Bayes filtering and sequential Monte-Carlo methods, and can be interpreted as a probabilistic version of the shift operator defined in [118] which serves to bootstrap the previous

MPC solution to initialize the current iteration. Specifically, the proposal distribution q is propagated after each round of MPC by marginalizing the transition over the approximate posterior distribution obtained from the previous iteration:

$$\tilde{q}_{t+1}(\boldsymbol{\theta}; \mathbf{x}_{t+1}) = \int p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t; \mathbf{x}_{t+1}) q_t(\boldsymbol{\theta}_t) d\boldsymbol{\theta}_t \tag{5.55}$$

given a transition probability, $p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}; \mathbf{x}_t)$ which includes a dependence on the currently observed state $\mathbf{x}_t$ for generality. This operation serves to approximate the prior at the new iteration. Given the empirical distribution q_t, we can simplify the expression above:

$$\tilde{q}_{t+1}(\boldsymbol{\theta}; \mathbf{x}_{t+1}) = \sum_{i=1}^{m} \int p_t(\boldsymbol{\theta}|\boldsymbol{\theta}_t; \mathbf{x}_{t+1})\ w^i \delta_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_t) d\boldsymbol{\theta}_t = \sum_{i=1}^{m} w^i p_t(\boldsymbol{\theta}|\boldsymbol{\theta}^i; \mathbf{x}_{t+1}) \tag{5.56}$$

resulting in a mixture of conditional probabilities.

In many open-loop MPC implementations [117, 118], it is assumed that the solution does not change significantly between each round, given an accurate dynamics model and sufficiently high controller frequency to resolve the dynamics and possible perturbations. This motivates a common heuristic used to reduce the computational burden between timesteps, which is to shift the control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}_t}$, forward-in-time by one step. That is, given an initial parameter sequence $\boldsymbol{\theta}_t = (\theta_t, \theta_{t+1}, ..., \theta_{t+H-1})$:

$$\tilde{\boldsymbol{\theta}}_{t+1} = \Phi(\boldsymbol{\theta}_t) \tag{5.57}$$

$$= (\theta_{t+1}, \theta_{t+2}, ..., \theta_{t+H-1}, \tilde{\theta}_{t+H-1}) \tag{5.58}$$

where the new parameter $\tilde{\theta}_{t+H-1}$ is chosen to reflect the expected final action. In the implementation, we adopt a similar heuristic, where the empirical distribution is first shifted deterministically according to:

$$\tilde{\boldsymbol{\theta}}^i = \Phi(\boldsymbol{\theta}^i) \quad \forall \quad i \in 1 : m \tag{5.59}$$

and setting the resulting distribution as $\tilde{q}_t(\boldsymbol{\theta}) = q_t(\boldsymbol{\theta})$. The shifted particles, $\tilde{\boldsymbol{\theta}}^i$, are then used in the following iteration to approximate the posterior, and the prior is updated according to Eq. (5.56). The shift operation can then be summarized by the following two sub-steps:

1. Shift particles: $\tilde{\boldsymbol{\theta}}^i = \Phi(\boldsymbol{\theta}^i)$

2. Update prior: $\tilde{q}_{t+1}(\boldsymbol{\theta}; \mathbf{x}_{t+1}) = \sum_{i=1}^{m} w^i p_t(\boldsymbol{\theta}|\tilde{\boldsymbol{\theta}}^i; \mathbf{x}_{t+1})$

5.9 Non-parametric SV-MPC

Inference can be performed directly over the posterior of open-loop control input sequences $\boldsymbol{\theta} \triangleq (\mathbf{u}_t, \mathbf{u}_{t+1}, ..., \mathbf{u}_{t+H-1})$. To accommodate this, we can define the cost-likelihood as a function of state: $\mathcal{L} = \mathcal{L}(X_t)$, and introduce a cost on control by defining an additional prior factor $p(\boldsymbol{\theta})$, and combining it with the transition probability defined in Section 5.8.6. If the dynamics are :

$$\nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\hat{f}_\xi}\Big[\, \mathcal{L}(X_t)\Big] = \frac{\mathbb{E}_{\hat{f}_\xi}\Big[\mathcal{L}(X_t)\, \nabla_{\boldsymbol{\theta}} \log p(X_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t)\Big]}{\mathbb{E}_{\hat{f}_\xi}\Big[\mathcal{L}(X_t)\Big]} \tag{5.60}$$

Similarly to the parametric formulation, the form of the cost-likelihood $\mathcal{L}$ will result in a particular update rule. The gradient can generally be evaluated by approximating the expectations with Monte Carlo sampling of state trajectories given the controlled stochastic dynamics, and evaluating the gradients on sampled trajectories. Under certain conditions, however, it may be evaluated in closed form. Such is the case for a Linear-Quadratic-Gaussian (LQG) system, for example [143]. An analytic expression to the log-likelihood can be derived by following steps similar to deriving the LEQR loss in [118], and taking the gradient of the resulting quadratic equation.

Algorithm 2: SV-TrajOpt

Input: Initial state $\mathbf{x}_0$, dynamics f, cost function C, prior $p_0(\boldsymbol{\theta})$, kernel k, termination condition $\mathrm{Done}(\cdot)$

Sample $\{\boldsymbol{\theta}^i\}_{i=1}^m \sim p_0(\boldsymbol{\theta})$

Set $\Delta\boldsymbol{\theta}^i = \mathrm{Inf}\ \forall\ i \in 1:m$

while $\mathrm{Done}(\{\Delta\boldsymbol{\theta}^i\})$ *is False* **do**

  for $i = 1, 2, ..., m$ **do in parallel**

    Forward dynamics : $X^i = X(f, \boldsymbol{\theta}^i, \mathbf{x}_0)$

    $\nabla_{\boldsymbol{\theta}^i} \log p(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi) = -\alpha\nabla_{\boldsymbol{\theta}^i} C(X^i) + \nabla_{\boldsymbol{\theta}^i} \log p_0(\boldsymbol{\theta}^i)$

  end

  for $i = 1, 2, ..., m$ **do in parallel**

    $\Delta\boldsymbol{\theta}^i \leftarrow \frac{1}{m}\sum_{j=1}^m k(\boldsymbol{\theta}^j, \boldsymbol{\theta}^i)\nabla_{\boldsymbol{\theta}^j} \log p(\boldsymbol{\theta}^j|\mathcal{O}_\tau;\xi) + \nabla_{\boldsymbol{\theta}^j} k(\boldsymbol{\theta}^j, \boldsymbol{\theta}^i)$

    $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}^i + \epsilon\Delta\boldsymbol{\theta}^i$

  end

end

(Optional: SGD refinement)

for $iter = 1, ..., N$ **do**

  $\Delta\boldsymbol{\theta}^i \leftarrow \nabla_{\boldsymbol{\theta}^i} \log p(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi) \quad \forall\ i \in 1:m$

  $\boldsymbol{\theta}^i \leftarrow \epsilon_r\Delta\boldsymbol{\theta}^i \quad \forall\ i \in 1:m$

end

$\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}_i} \log p(\boldsymbol{\theta}^i|\mathcal{O}_\tau;\xi)$

5.10 Trajectory optimization

The variational inference framework defined by SV-MPC can be modified to accommodate general motion planning problems common to many robotics applications. This special case can be considered by (1) using the non-parametric formulation, (2) assuming a stationary posterior distribution (no shifting), and (3) defining a deterministic dynamics model $\bar{f}$ such that state trajectory probabilities can be represented as $p(X|\boldsymbol{\theta}) = \delta(X - \bar{f}(\boldsymbol{\theta}, x_0))$. A prior over sequences $p(\boldsymbol{\theta})$ can be defined to encourage desired behavior such as smoothness [103]. Because individual particles are not guaranteed to produce a MAP estimate once the SVGD optimization has converged, they can be subsequently refined by applying a deterministic gradient descent update using posterior gradients without kernelization. Se-

lection of a feasible and optimal plan can then be generated by simply picking the best particle. In order to derive the likelihood gradient, we can consider the special case for deterministic dynamics by defining the trajectory distributions using a dirac measure on the space of trajectories: $p(X_t|\boldsymbol{\theta}; \xi, \mathbf{x}_t) = \delta(X_t - \overline{X}_t)$, where $\overline{X}_t = F(\boldsymbol{\theta}, \xi, \mathbf{x}_t)$, and F performs consecutive application of the deterministic dynamics f. Setting $\mathcal{L}(X_t, \boldsymbol{\theta}_t) = \exp(-\alpha C(X_t, \boldsymbol{\theta}_t))$, the gradient then reduces to :

$$\nabla_{\boldsymbol{\theta}} \log \mathbb{E}_{\hat{f}_\xi}\Big[\, \mathcal{L}(X_t, \boldsymbol{\theta})\Big] = \frac{\nabla_{\boldsymbol{\theta}} \int \exp\left(-\alpha C(X_t, \boldsymbol{\theta})\right) \delta(X_t - \overline{X}_t) dX_t}{\int \exp\left(-\alpha C(X_t, \boldsymbol{\theta})\right) \delta(X_t - \overline{X}_t) dX_t} \tag{5.61}$$

$$= \frac{\nabla_{\boldsymbol{\theta}} \exp\left(-\alpha C(\overline{X}_t, \boldsymbol{\theta})\right)}{\exp\left(-\alpha C(\overline{X}_t, \boldsymbol{\theta})\right)} \tag{5.62}$$

$$= \frac{-\alpha \exp\left(-\alpha C(\overline{X}_t, \boldsymbol{\theta})\right) \nabla_{\boldsymbol{\theta}} C(\overline{X}_t, \boldsymbol{\theta})}{\exp\left(-\alpha C(\overline{X}_t, \boldsymbol{\theta})\right)} \tag{5.63}$$

$$= -\alpha \nabla_{\boldsymbol{\theta}} C(\overline{X}_t, \boldsymbol{\theta}) \tag{5.64}$$

The gradient can then be evaluated in a straightforward manner via back-propagation on the cost function, through the dynamics. The final motion planning algorithm, SV-TrajOpt, is fully outline in Algorithm 2.

In the following section, we apply SV-MPC to common robotics problems: navigation, manipulation and locomotion. We end with an example of the SV-TrajOpt algorithm applied to a planar motion-planning scenario. All control algorithms were implemented in PyTorch, with batched gradient computation across particles and parallel generation of rollouts using either simulated or analytic dynamical models. Additional experimental details and results can be found in the appendix.

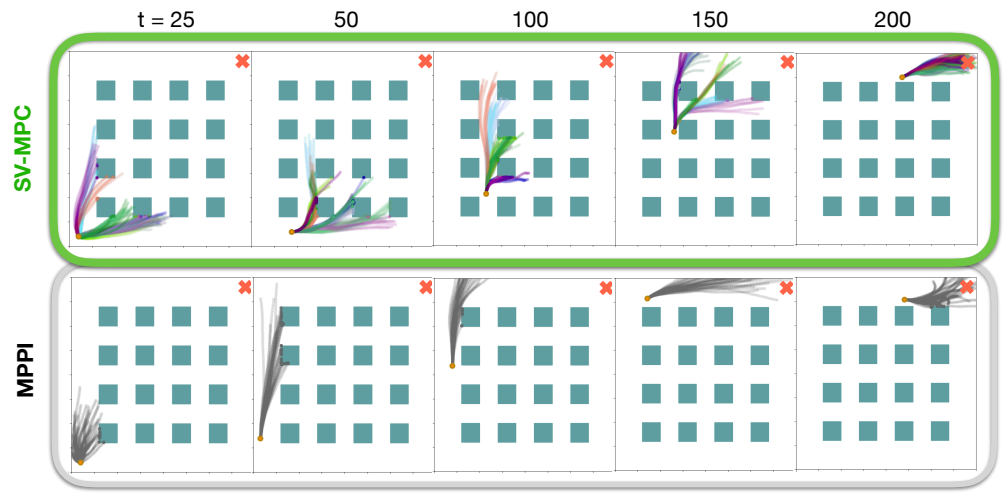


Figure 5.2: Depiction of the planar navigation task. The robot (orange dot) attempts to reach the goal location (red cross) while avoiding obstacles. Each frame depicts the environment state at a particular time-step, along with the distributions of sampled state-trajectory rollouts generated by the MPC controllers using the modeled dynamics. Each trajectory color is associated with a single particle from SV-MPC. The multi-modal distribution of SV-MPC is able to explore passages between obstacles and find shorter paths to the goal.

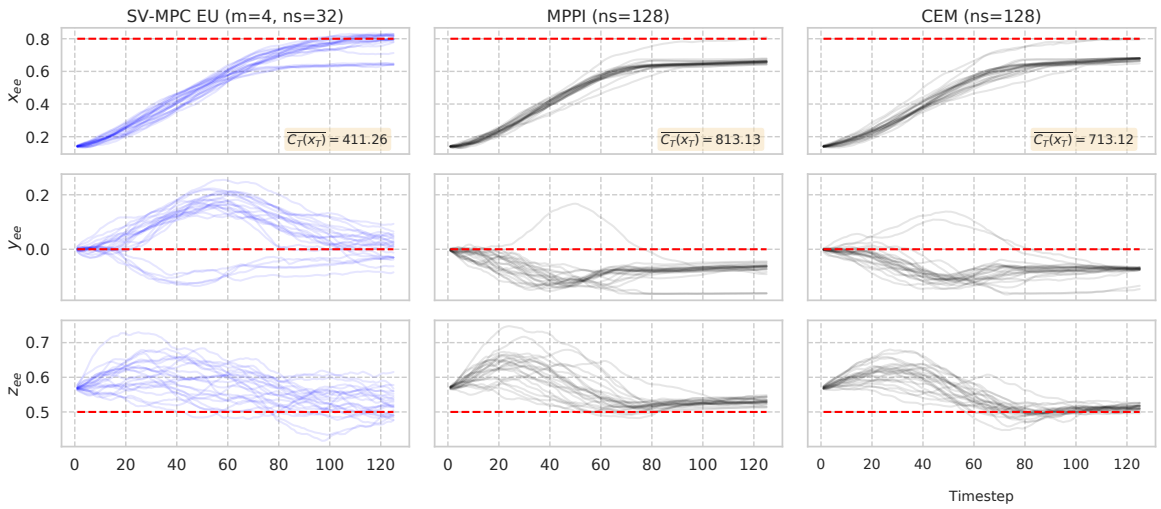


Figure 5.3: Examples of end-effector Cartesian trajectories resulting from application of different MPC algorithms on the Franka reaching experiment. The relative distance to the fixed target location is plotted over the length of each episode. The dashed red line indicates the coordinates of the target. The sample-averaged terminal cost for the final state $\overline{C_T(\mathbf{x}_T)}$ is evaluated over the 24 independent trials. With four particles ($m = 4$), SV-MPC with Exponentiated utility likelihood manages to avoid bad local minima, despite higher-variance gradients due to fewer samples used to evaluate gradients ($\mathrm{ns} = 32$ vs. $\mathrm{ns} = 128$).

Table 5.1: Statistics for planar navigation task over 25 trials (4x4 obstacle grid)

Controller	Num. of particles	Avg. cost of success ($\times 10^3$)	Success rate (%)
SV-MPC	32	20.7	96
	12	21.8	96
	6	25.6	84
MPPI	—	26.5	64
CEM	—	25.4	64

5.11 Experiments

5.11.1 Planar Navigation

We construct a 2D robot navigation task, where a holonomic point-robot must reach a target location while avoiding obstacles (Fig. 5.2). Hitting an obstacle will cause the agent to "crash" and prevent any further movement. This added non-differentiability makes the problem particularly challenging, but also suitable for sampling-based control schemes. The system exhibits stochastic dynamics, and is defined as a double-integrator model with additive Gaussian noise. The SV-MPC controller with exponentiated-utility (EU) is constructed with a set of 6 to 32 particles, where the gradient of each particle is estimated via Monte-Carlo sampling of control and state trajectories. At each round, the best-performing particle is chosen to generate the action using Eq. (5.52).

For state $\mathbf{x}_t = (x_t, \dot{x}_t)$ and control $\mathbf{u}_t = \ddot{x}_t$, where x_t, $\dot{x}_t$, $\ddot{x}_t$ are the 2D position, velocity and acceleration, respectively, and the x_{goal} the target 2D goal position, we define the instantaneous and terminal costs (with respect to Eq. (5.1)):

$$c(\mathbf{x}_t, \mathbf{u}_t) = 0.5(x_t - x_{goal})^\top (x_t - x_{goal}) + 0.25\dot{x}_t^\top \dot{x}_t + 0.2\ddot{x}_t^\top \ddot{x}_t \tag{5.65}$$

$$c_{\text{term}}(\mathbf{x}_t) = 1000(x_t - x_{goal})^\top (x_t - x_{goal}) + 0.1\dot{x}_t^\top \dot{x}_t \tag{5.66}$$

The SV-MPC controller is compared against MPPI (Fig. 5.2) and CEM. We include quantitative summaries of performance across control types in Table 5.1. The performance of

the SV-MPC controller improves with increasing particle number.

5.11.2 Manipulation

We demonstrate SV-MPC on a 7-DOF reaching task (Fig. 5.1). Velocity-based control commands are generated in the configuration space of a simulated robot manipulator, which must reach a stationary goal in its work-space. We leverage the GPU-accelerated Isaac-Gym library [153] for parallel computation of trajectory rollouts in simulation during MPC iterations. The use of simulation provides the ability to efficiently compute highly-resolved geometric constraints between the robot and obstacles, eliminating the need for coarsely-defined heuristics for collision detection (such as signed-distance fields [103]). Although the dynamics are deterministic, the problem remains challenging since the posterior probability distribution implies a nonlinear mapping (via inverse kinematics) from the work-space to the sampling space. A sampling-based SV-MPC controller is compared against MPPI and CEM for an open-loop, constant-covariance Gaussian control distribution: $\boldsymbol{\pi}_{\boldsymbol{\theta}} = \mathcal{N}(U_t; \boldsymbol{\theta}, \Sigma)$. The obstacles and the target are placed in order to demonstrate a local-minima trap: given a finite H, the optimal control solution is to move left. However, the opening between the obstacles in this direction is too narrow for the robot to move through. To avoid getting stuck, a sampling-based control scheme must generate a sequence which will move the robot in the other direction. For a uni-modal distribution, this may have a very low-probability, and recovery will not occur. Increasing the sampling covariance would mitigate this, but would require a larger amount of samples to reduce variance. The cost-function consists of a cost on cartesian distance-to-goal from the end-effector, as well as a penalty on control. For state $\mathbf{x}_t = (e_t, \dot{e}_t)$ and control $\mathbf{u}_t = \dot{q}_t$, where e_t, $\dot{e}_t$, are the cartesian end-effector positions and velocities, respectively, $\dot{q}$ the joint velocities, and the e_{goal} the target 3D goal position, we define the instantaneous and terminal

costs (with respect to Eq. (5.1)):

$$c(\mathbf{x}_t, \mathbf{u}_t) = 1(e_t - e_{goal})^\top (e_t - e_{goal}) + 0.25\dot{e}_t^\top \dot{e}_t + 0.1\dot{q}_t^\top \dot{q}_t \tag{5.67}$$

$$c_{\text{term}}(\mathbf{x}_t) = 5000(e_t - e_{goal})^\top (e_t - e_{goal}) + 0.1\dot{e}_t^\top \dot{e}_t \tag{5.68}$$

Examples of trajectory executions are shown in Fig. 5.3. The total number of generated control samples is held constant across algorithms. Both MPPI and CEM tend towards the sub-optimal local minimum, leading the robot to get stuck between two obstacles. Using a particle-based representation of the posterior, SV-MPC can resolve multiple optima simultaneously, switching to lower-cost modes around obstacles and successfully reaching the goal.

5.11.3 Stochastic Half-Cheetah

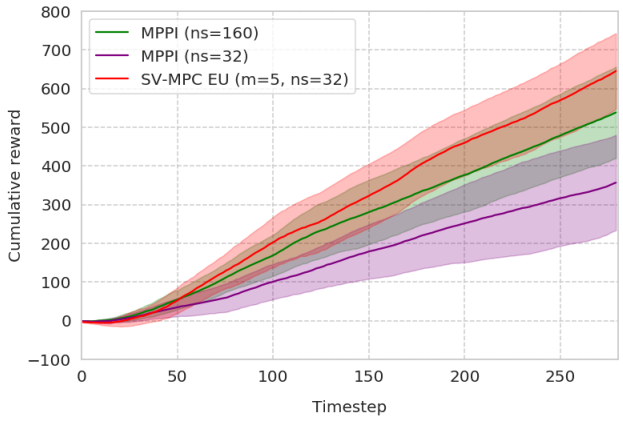
To test our approach on a complex nonlinear system with discontinuous dynamics, we consider an environment common in many Reinforcement Learning benchmarks: the Half-Cheetah [154]. We use a stochastic version of the dynamics with additive noise in the control space. We modify the cost function to reward forward velocity only if the agent is forward-facing, as done in [138], along with a control penalty. Without this alteration, progress can be made fairly easily by applying torque commands in a single direction, and 'cart-wheeling' the system. For instantaneous forward velocity v_t and body angle β,

$$c(\mathbf{x}_t, \mathbf{u}_t) = 0.1\mathbf{u}_t^\top \mathbf{u}_t - \frac{v_t}{2}(1 + \text{sgn}(\cos\beta)) \tag{5.69}$$

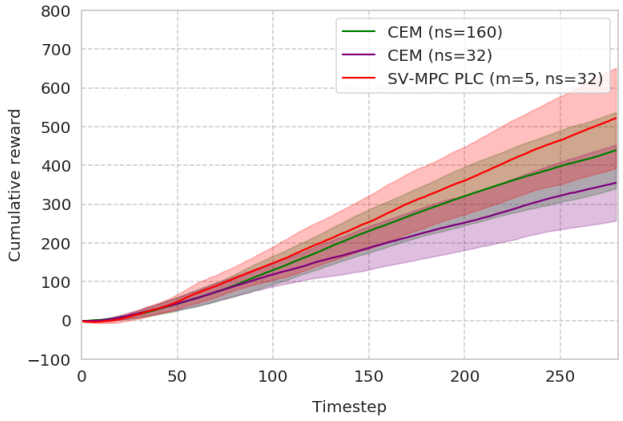
The cumulative rewards over multiple trials are plotted in Fig. 5.4.

5.11.4 Motion Planning

We use the SV-TrajOpt algorithm Algorithm 2 to infer a distribution over optimal control sequences for a planar motion planning problem on a point robot (Fig. 5.5). The robot must



(a) Exponentiated Utility (EU)



(b) Probability of Low-Cost (PLC)

Figure 5.4: Comparisons of cumulative-reward distributions for the Stochastic HalfCheetah task. Results are collected over 16 independent trials, with mean and standard deviations shown. SV-MPC is capable of finding high-reward trajectories, using the same total amount of samples as MPPI and CEM.

find a velocity-based control sequence which results in a low-cost, feasible path around a set of obstacles to reach the goal. The occupancy map is fully-differentiable, allowing a gradient on obstacle cost to be computed numerically. A 'smoothness' prior is defined over velocities as a multi-variate Gaussian with a tri-diagonal precision matrix, which favors low-acceleration trajectories. The dynamics consist of a deterministic, velocity controlled single-integrator model on the 2D position: $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_t \Delta t$, where $\mathbf{x}_t, \mathbf{u}_t \in \mathbb{R}^{d \times 1}$ ($d = 2$). In the example, we use a constant-control (*i.e.* zero-acceleration) prior on velocities:

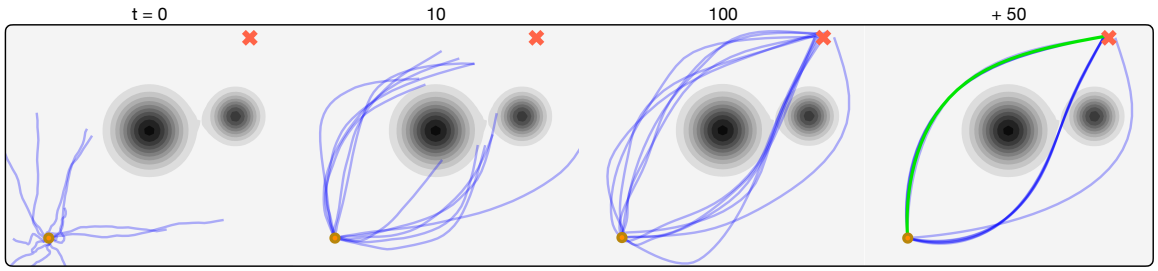


Figure 5.5: The SV-TrajOpt algorithm is applied to a motion-planning problem, where a velocity-controlled holonomic robot must reach the goal (red cross). Each blue state-trajectory results from a single particle control-sequence. Particles are randomly initialized from the prior ($t = 0$), and are optimized until convergence ($t = 10, 100$). Independent local MAP approximations are generated after 50 iterations of gradient-descent refinement, and the lowest-cost particle shown in green.

$\mathbf{u}_{t+1} = \mathbf{u}_t + \mathbf{w}_t,\ \mathbf{w}_t \sim \mathcal{N}(0, \Sigma)$. We can construct the prior over sequences by first considering the convolution of control inputs over the planning horizon T:

$$\begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{T-1} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ I & I & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & I & I & \cdots & I \end{bmatrix} \begin{bmatrix} \mathbf{w}_{-1} \\ \mathbf{w}_0 \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{T-2} \end{bmatrix}, \tag{5.70}$$

where we assume the initial control is drawn from the same zero-mean distribution: $\mathbf{u}_0 = \mathbf{w}_{-1} \sim \mathcal{N}(0, \Sigma)$. For the sequences $U \triangleq [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{T-1}]^\top$ and $W \triangleq [\mathbf{w}_{-1}, \mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_{T-2}]^\top$, we can write the above as:

$$U = LW \tag{5.71}$$

with L defined as the lower-triangular matrix. We can then define the prior over control sequences:

$$U \sim p(U) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{5.72}$$

with covariance $\mathbf{\Sigma} = LDL^T$, where $D = \text{diag}(\Sigma, \Sigma, ..., \Sigma) \in \mathbb{R}^{Td \times Td}$. Note that the precision matrix $\mathbf{\Sigma}^{-1}$ is tri-diagonal, implying a graphical structure with Markovian dependencies. In the SV-MPC optimization, the prior can also be interpreted as a penalty which encourages smooth state trajectories (similarly to the GP-prior described in [103]). In the motion planning problem, we use this multi-variate Gaussian as the prior over particles: $p(\boldsymbol{\theta}) = \mathcal{N}(0, \mathbf{\Sigma})$.

Furthermore, the cost function defined for this problem includes a smooth obstacle cost-map. This is generated using a bi-modal mixture of Gaussians, with the probability of collision given by $p_{\text{obs}}(x_t)$. The cost function is then:

$$c(\mathbf{x}_t) = 1 \times 10^5 \, p_{\text{obs}}(x_t) \tag{5.73}$$

$$c_{\text{term}}(\mathbf{x}_t) = 1000(x_t - x_{goal})^\top (x_t - x_{goal}) \tag{5.74}$$

5.12 Complexity

The implementation of SV-MPC requires the computation of the kernel Gram-matrix $K(\theta^i, \theta^j) \; \forall \; i, j \; \in \; (1, ..., m)$ for the SVGD update. This requires an inner-product operation for all particle pairs, resulting in a computational complexity of $\mathcal{O}(m^2 h d)$ (where m : number of particles, h : horizon, d : control dimension). However, by exploiting structured kernels for trajectories, the scaling with respect to the horizon can be removed by parallel computation of kernel factors. Evaluation of a kernel factor is then a constant-time operation ($\mathcal{O}(1)$), and the overall SV-MPC complexity reduces to $\mathcal{O}(m^2)$. In practice a relatively low order of particles is required: $m << 1 \times 10^3$. The core bottleneck is typically the generation of rollout trajectories from control samples during each iteration of MPC, which can be performed in parallel but is linear in time with respect to the horizon h. As with most MPC applications, the horizon length, number of samples, etc. can be varied to balance accuracy with runtime complexity, depending on the constraints of the system.

5.13 Connection to Path Integral Control

The equation derived in Eq. (5.24) is well-known in statistical thermodynamics, where the random variable under consideration is the energy $C(x) : \Omega_x \mapsto \mathbb{R}$, a non-negative real-valued measurable property, and $x \in \mathbb{R}^n$ is the state of the system. The Helmholtz free energy of C, with respect to probability density p, is defined as the function:

$$-\frac{1}{\alpha} \log \mathbb{E}_p\Big[\exp\left(-\alpha C(x)\right) \Big], \tag{5.75}$$

where $\alpha > 0$ is the (inverse) temperature. The corresponding variational inequality, known as the Donsker-Varadahn principle [155, 156], relates the free energy as the Legendre transform of the entropy:

$$-\frac{1}{\alpha} \log \mathbb{E}_p\Big[\exp\left(-\alpha C(x)\right) \Big] = \min_q \ -\frac{1}{\alpha}\mathbb{E}_q\Big[\log \exp\left(-\alpha C(x)\right) \Big] + \frac{1}{\alpha} D_{KL}\left(q(x) \,||\, p(x)\right) \tag{5.76}$$

$$= \min_q \ \mathbb{E}_q\Big[C(x)\Big] + \frac{1}{\alpha} D_{KL}\left(q(x) \,||\, p(x)\right) \tag{5.77}$$

Following the same derivation presented in Eqs. (5.16) and (5.24), the solution is then found to be the Gibbs distribution:

$$q^* = \frac{\exp\left(-\alpha C(x)\right) p(x)}{\int \exp\left(-\alpha C(x)\right) p(x) dx}. \tag{5.78}$$

In the context of stochastic dynamics, the above is also applicable to random paths generated by a Markov diffusion process [156]. This can then be extended to optimal control, by addressing the KL-minimization problem between controlled and uncontrolled stochastic systems. The connection was developed and explored for continuous-time dynamics in previous work, such as [143, 144]. Here, the nonlinear-affine dynamics under consideration

are subject to Brownian motion:

$$\mathbf{x}_{t+1} = \bar{f}(\mathbf{x}_t) + G(\mathbf{x}_t)\mathbf{u}_t + B(\mathbf{x}_t)\mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, \Sigma) \tag{5.79}$$

$$= f(\mathbf{x}_t, \mathbf{u}_t) \tag{5.80}$$

for the discrete-time case. For a SOC expected-cost objective:

$$\min_{U_t} J(X_t) = \min_{U_t=(\mathbf{u}_0,\ldots,\mathbf{u}_T)} \mathbb{E}_f \left[c_{\text{term}}(\mathbf{x}_T) + \sum_{t=0}^{T} c_t(\mathbf{x}_t) + \frac{1}{2}\mathbf{u}_t^\top R \mathbf{u}_t \right], \tag{5.81}$$

the optimal control distribution could be derived by a change of measure using Girsanov's theorem, which was shown to satisfy the Hamilton-Jacobi-Bellman (HJB) equations for optimality. To solve the HJB equations, an exponentiated form of the value function, the desirability-function, was defined: $\Psi(\mathbf{x}_t) = \exp\left(-\alpha \mathrm{V}(\mathbf{x}_t)\right)$. By application of the Feynman-Kac lemma, the transformed HJB partial differential equation can be solved as:

$$\Psi(\mathbf{x}_t) = \int \exp\left(-\alpha C(X_t)\right) p_f(X_t; \mathbf{x}_t) dX_t \tag{5.82}$$

$$= \mathbb{E}_{p_f} \left[\exp\left(-\alpha C(X_t)\right) \right] \tag{5.83}$$

where p_f denotes the passive dynamics *i.e.* the probability density of trajectories X_t, resulting from the stochastic dynamics $f(\mathbf{x}_t, \mathbf{u}_t)$ with $\mathbf{u}_t = 0$. Under the assumption that $R = \frac{1}{\alpha}\Sigma^{-1}$, this can then be used to evaluate the optimal control law:

$$\begin{aligned} \mathbf{u}_t^* &= -\alpha \Sigma\, G(\mathbf{x}_t)^\top \nabla_{\mathbf{x}} \mathbf{V}(\mathbf{x}_t) \\ &= \Sigma\, G(\mathbf{x}_t)^\top \nabla_{\mathbf{x}} \log \Psi(\mathbf{x}_t) \\ &= \Sigma\, G(\mathbf{x}_t)^\top \frac{\nabla_{\mathbf{x}} \Psi}{\Psi} \,. \end{aligned} \tag{5.84}$$

For receding-horizon control, we can borrow from [117] and consider a discrete-time case of Path-Integral control for a nonlinear stochastic dynamical system:

$$\mathbf{x}_{t+1} = \bar{f}(\mathbf{x}_t, \mathbf{v}_t),\ \mathbf{v}_t \sim p(\mathbf{v}_t|\mathbf{u}_t) \tag{5.85}$$

with nominal deterministic dynamics $\bar{f}$, commanded control input $\mathbf{u}_t$, and stochastic perturbations $\mathbf{v}_t$ which are exhibited in the control input channel. The uncontrolled system is then realized when $\mathbf{u}_t = 0$, and is controlled otherwise. Given a sequence of perturbations: $V_t = (\mathbf{v}_t, \mathbf{v}_{t+1}, ...\mathbf{v}_{t+H-1})$, we can write the resulting state trajectory as $X_t = F(\mathbf{x}_t, V_t)$, where F performs consecutive application of the dynamics $\bar{f}$ given $\mathbf{x}_t$ and a sequence V_t. We can then consider probability distributions directly over V_t, with $p(V_t)$ for uncontrolled dynamics and $q(V_t|U_t)$ for the controlled system, with the sequence of control inputs given by $U_t = (\mathbf{u}_t, \mathbf{u}_{t+1}, ...\mathbf{u}_{t+H-1})$. A cost on state $C(X_t)$ is mapped to the random variable V_t by the convolution $S = C \circ F$. Similarly to equations Eqs. (5.24) and (5.77) above, we then seek a solution to minimize the variational objective:

$$\begin{aligned} q^* &= \underset{q}{\operatorname{argmin}}\ D_{KL}\left(q \,||\, q^*(V_t)\right) && (5.86)\\ &= \underset{q}{\operatorname{argmin}}\ -\frac{1}{\alpha}\mathbb{E}_q\Big[\log\exp\left(-\alpha S(V_t)\right)\Big] + \frac{1}{\alpha}D_{KL}\left(q \,||\, p\right) && (5.87)\\ &= \underset{q}{\operatorname{argmin}}\ \mathbb{E}_q\Big[S(V_t)\Big] + \frac{1}{\alpha}D_{KL}\left(q \,||\, p\right) && (5.88) \end{aligned}$$

where the optimal distribution is then known to be

$$q^*(V_t) = \frac{\exp\left(-\alpha S(V_t)\right)p(V_t)}{\int \exp\left(-\alpha S(V_t)\right)p(V_t)dV_t}\ . \tag{5.89}$$

Although we have the form of the optimal control distribution, an analytic solution is generally intractable due to the partition function. The algorithm for Model Predictive Path Integral Control (MPPI) [119, 117] proceeds by defining a surrogate cross-entropy mini-

mization problem:

$$\min_{q \in \mathcal{Q}} \; D_{KL}\left(q^*(V_t) \,||\, q\right) \tag{5.90}$$

for a tractable family of distributions $\mathcal{Q}$, typically fixed-covariance Gaussians with mean parameters $\mu_t = U_t$ and an equivalent assumption on the controlled dynamics distribution.

In the SV-MPC framework, the original variational objective of Path Integral control can be addressed, where

$$\min_{q \in \mathcal{Q}} \; D_{KL}\left(q \,||\, q^*(V_t)\right). \tag{5.91}$$

We can apply the non-parametric Bayesian-MPC formulation (Setion Section 5.5) to solve for $q^*(V_t)$ directly over V_t, setting the inference parameter to be $\boldsymbol{\theta} = V_t$ with a prior given by the passive dynamics $p(\boldsymbol{\theta}; \mathbf{x}_t) = p(V_t)$. The likelihood term $p(\mathcal{O}_\tau | \boldsymbol{\theta}; \xi, \mathbf{x}_t)$ can be derived by using the exponentiated-utility $\mathcal{L}(X_t) = \exp\left(-\alpha C(X_t)\right)$ and defining a dirac measure over state trajectories to denote the probability density of X_t given $\boldsymbol{\theta}$:

$$p(\mathcal{O}_\tau | \boldsymbol{\theta}; \xi, \mathbf{x}_t) \propto \int \exp\left(-\alpha C(X_t)\right) \delta(X_t - F(\mathbf{x}_t, \boldsymbol{\theta})) dX_t \tag{5.92}$$

$$= \exp\left(-\alpha C(F(\mathbf{x}_t, \boldsymbol{\theta}))\right) \tag{5.93}$$

$$= \exp\left(-\alpha S(\boldsymbol{\theta})\right) \; . \tag{5.94}$$

where we drop ξ-notation for the dynamics parameters for simplicity. We then recover the Bayesian formulation:

$$q^*(\boldsymbol{\theta}) = \frac{p(\mathcal{O}_\tau | \boldsymbol{\theta}; \xi, \mathbf{x}_t) \; p(\boldsymbol{\theta}; \mathbf{x}_t)}{\int p(\mathcal{O}_\tau | \boldsymbol{\theta}; \xi, \mathbf{x}_t) \; p(\boldsymbol{\theta}; \mathbf{x}_t) d\boldsymbol{\theta}} \tag{5.95}$$

$$= \frac{\exp\left(-\alpha S(\boldsymbol{\theta})\right) \; p(\boldsymbol{\theta}; \mathbf{x}_t)}{\int \exp\left(-\alpha S(\boldsymbol{\theta})\right) \; p(\boldsymbol{\theta}; \mathbf{x}_t) d\boldsymbol{\theta}} \tag{5.96}$$

$$= p(\boldsymbol{\theta} | \mathcal{O}_\tau; \xi, \mathbf{x}_t) \tag{5.97}$$

Approximate inference on the posterior $p(\boldsymbol{\theta}|\mathcal{O}_\tau; \xi, \mathbf{x}_t)$ can then be performed using Algorithm 1. We can further consider the special case of control-affine dynamics of the form:

$$\mathbf{x}_{t+1} = \bar{f}(\mathbf{x}_t) + G(\mathbf{x}_t)(\mathbf{u}_t + \mathbf{v}_t), \quad \mathbf{v}_t \sim \mathcal{N}(0, \Sigma). \tag{5.98}$$

with passive dynamics:

$$\mathbf{x}_{t+1} = \bar{f}(\mathbf{x}_t) + G(\mathbf{x}_t)\mathbf{v}_t, \quad \mathbf{u}_t = 0 \; \forall \; t \; . \tag{5.99}$$

We refer to this in trajectory-wise form as $X_t = F(\mathbf{x}_t, V_t)$. By application of Eq. (5.84), the optimal control action can then be determined:

$$\mathbf{u}_t^* = \Sigma \, G(\mathbf{x}_t)^\top \frac{\nabla_{\mathbf{x}} \Psi}{\Psi} = \Sigma \, G(\mathbf{x}_t)^\top \frac{\nabla_{\mathbf{x}} \mathbb{E}_{V_t \sim p(V_t)} \Big[\exp(-\alpha S(V_t)) \Big]}{\mathbb{E}_{V_t \sim p(V_t)} \Big[\exp(-\alpha S(V_t)) \Big]} \tag{5.100}$$

$$= -\alpha \Sigma \, G(\mathbf{x}_t)^\top \frac{\mathbb{E}_{V_t \sim p(V_t)} \Big[\exp(-\alpha S(V_t)) \nabla_{\mathbf{x}} S(V_t)|_{\mathbf{x}=\mathbf{x}_t} \Big]}{\mathbb{E}_{V_t \sim p(V_t)} \Big[\exp(-\alpha S(V_t)) \Big]} \tag{5.101}$$

$$\approx -\alpha \Sigma \, G(\mathbf{x}_t)^\top \frac{\sum_{i=1}^m \exp(-\alpha S(\boldsymbol{\theta}_i)) \nabla_{\mathbf{x}} S(\boldsymbol{\theta}_i)|_{\mathbf{x}=\mathbf{x}_t}}{\sum_{i=1}^m \exp(-\alpha S(\boldsymbol{\theta}_i))} \tag{5.102}$$

where $\nabla_{\mathbf{x}} S(\boldsymbol{\theta}_i)|_{\mathbf{x}=\mathbf{x}_t}$ is the gradient of the path cost with respect to the current state $\mathbf{x}_t$. This can be expressed in terms of the nominal dynamics by using the chain rule:

$$\nabla_{\mathbf{x}} S(\boldsymbol{\theta}_i)^\top|_{\mathbf{x}=\mathbf{x}_t} = \frac{dC(X_t)}{dX_t}\bigg|_{X_t = F(\mathbf{x}_t, \boldsymbol{\theta}_i)} \cdot \frac{dF(\mathbf{x}, \boldsymbol{\theta}_i)}{d\mathbf{x}}\bigg|_{\mathbf{x}=\mathbf{x}_t} \tag{5.103}$$

Numerical evaluation of this gradient can be performed via backpropagation for each i-th particle.

5.14 Discussion

A novel formulation for Bayesian model predictive control is presented, where inference is performed directly over the control parameters and inputs. An algorithm for approximate inference is then proposed, where the posterior is represented as a set of particles and is optimized via SVGD. In contrast to pure Monte-Carlo sampling methods, gradient-based information can be exploited to improve particle efficiency, where many computationally-intensive operations can be run in parallel using effective GPU implementation. The flexibility of the approach can accommodate different cost transformations to modulate risk-seeking behavior, and can naturally be extended to trajectory optimization problems. We compare against common MPC baselines, demonstrating improved performance on a variety of control tasks.

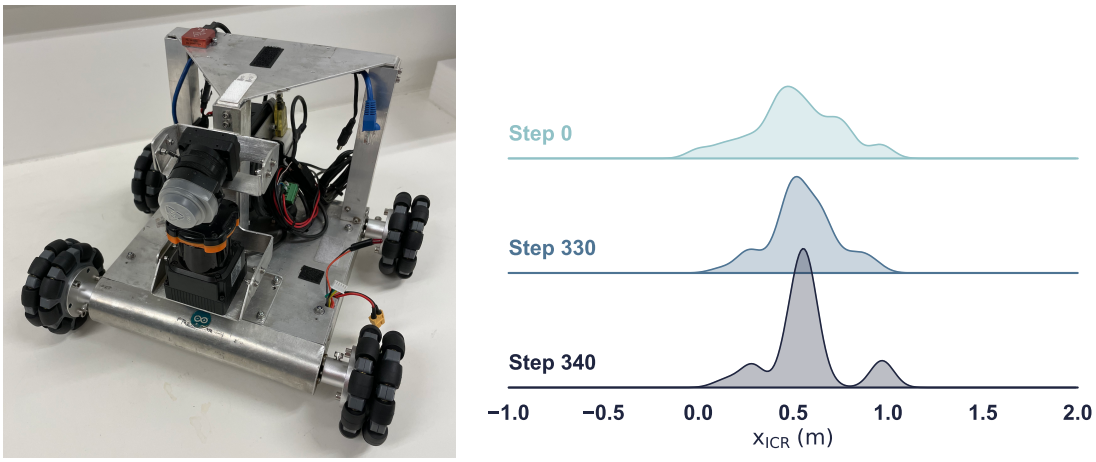
In the next chapter, we propose an extension of this distributed control algorithm to include inference over dynamics parameters, such as mass and inertial values. This is examined in the online setting, where incoming streams of measurement are used to update belief over the physical properties of the system model.

CHAPTER 6

PARTICLE-BASED INFERENCE FOR ONLINE PARAMETER ESTIMATION

6.1 Introduction

In Chapter 5, model predictive control was formulated as a Bayesian inference problem, where the posterior distribution is estimated over control parameters given the state and "cost-based" conditioning for optimal trajectories. This approach was shown to be better suited at handling the multi-modality of the distribution over actions, but did not attempt to dynamically adapt to changes in environment parameters. Previous work has demonstrated that incorporating uncertainty of system parameters in the evaluation of SOC estimates can improve performance [157], particularly when this uncertainty is periodically re-estimated [158]. In the following chapter, we extend the application of Stein Variational Gradient Descent in control problems to include the estimation of uncertainty over model parameters on-the-fly. We represent the structural uncertainty over parameters as a collection of particles, representing an implicit variational distribution. These particles are then updated sequentially in an online fashion, and can capture complex multi-modal distributions. We demonstrate the approach on both simulated and real-world navigation tasks requiring real-time execution in the face of dynamically changing environments. The proposed algorithm is implemented on a real autonomous ground vehicle (AGV) (Fig. 6.1a), illustrating the applicability of the method in real time. Experiments show how the control and parameter inference are leveraged to adapt the behaviour of the robot under varying conditions, such as changes in mass. Simulation results are presented for an inverted pendulum and a 2D obstacle grid, demonstrating an effective adaptation to dynamic changes in model parameters.



(a) Wombot AGV

(b) Posterior distribution over x_{ICR}

Figure 6.1: Online parameter estimation for autonomous ground vehicles. Distributions over system parameters such as the inertial center of rotation (ICR), are adapted in real-time. (a) The custom built skid-steer robot platform used in experiments. (b) Distribution over x_{ICR} at different time steps. The mass load on the robot is suddenly increased during system execution. The parameter distribution estimate quickly changes to include a second mode that better explains the new dynamics. Our particle-based control scheme can accommodate such multi-modal uncertainty and adapt to dynamically changing environments.

6.2 Related Work

Model predictive control is a reactive control scheme, and can accommodate modelling errors to a limited degree. However, its performance is largely affected by the accuracy of long-range predictions. Modelling errors can compound over the planning horizon, affecting the expected outcome of a given control action. This can be mitigated by accounting for model uncertainty, leading to better estimates of expected cost. This has been demonstrated to improve performance in stochastic optimal control methods and model-based reinforcement learning [159, 160]. Integrating uncertainty has typically been achieved by learning probabilistic dynamics models from collected state-transition data in an episodic setting, where the model is updated in between trajectory-length system executions [161, 162, 163, 164]. A variety of modelling representations have been explored, including Gaussian processes [160], neural network ensembles [161], Bayesian regression, and meta-learning [165]. Alternatively, the authors in [164, 158] estimate posterior distributions of

physical parameters for black-box simulators, given real-world observations.

Recent efforts have examined the online setting, where a learned probabilistic model is updated based on observations made *during execution* [166, 167, 168, 169]. The benefits of this paradigm are clear: incorporating new observations and adapting the dynamics *in situ* will allow for better predictions, improved control, and recovery from sudden changes to the environment. However, real-time requirements dictate that model adaptation must be done quickly and efficiently, and accommodate the operational timescale of the controller. This typically comes at the cost of modelling accuracy, and limits the application of computationally-burdensome representations, such as neural networks and vanilla GPs. Previous work has included the use of sparse-spectrum GPs and efficient factorization to incrementally update the dynamics model [166, 170]. In [165], the authors use a meta-learning approach to train a network model offline, which is adapted to new observations using Bayesian linear regression operating on the last layer. However, these approaches are restricted to Gaussian predictive distributions, and may lack sufficient modelling power for predicting complex, multi-modal distributions.

Perhaps most closely related to our approach is the work presented in [167]. The authors propose to track a distribution over simulation parameters using a sequential Monte Carlo method akin to a particle filter. The set of possible environments resulting from the parameter distribution is used by an MPPI controller to generate control samples. Each simulated trajectory rollout is then weighted according to the weight of the corresponding environment parameter. Although such an approach can model multi-modal posterior distributions, we should expect similar drawbacks to particle filters, which require clever re-sampling schemes to avoid mode collapse and particle depletion. Our method also leverages a particle-based representation of parameter distributions, but performs deterministic updates based on new information and is more sample efficient than MC sampling techniques.

6.3 Joint Inference for Control and Dynamics

We can generalise the framework presented in Chapter 5 to simultaneously refine our knowledge of the dynamical system, while estimating optimal policy parameters $\boldsymbol{\theta}_t$. We can assume to have access to a model or simulation of the dynamics, which is a function of a set of physical parameters ξ (such as mass, friction, geometry, etc). To perform inference over ξ, we can store observations from the environment into a dataset $\mathcal{D}_{1:t} := \{(\mathbf{x}_t^r, \mathbf{u}_{t-1}^r, \mathbf{x}_{t-1}^r)\}_{t=1}^{N_\mathcal{D}}$ at each timestep t. The current posterior probability in Eq. (5.2) can then be expressed as:

$$p(\boldsymbol{\theta}_t, \xi|\mathcal{O}, \mathcal{D}_{1:t}) = p(\boldsymbol{\theta}_t|\mathcal{O}, \xi)p(\xi|\mathcal{D}_{1:t})\,, \tag{6.1}$$

Note that the policy parameters $\boldsymbol{\theta}$ are conditionally independent from the system observations, given ξ. We can then perform an iterative inference procedure, where at each time-step the posterior over dynamics parameters $p(\xi|\mathcal{D}_{1:t})$ is first updated based on recent measurements. Following this, the posterior $p(\boldsymbol{\theta}_t|\mathcal{O}, \xi)$ over control parameters can then be inferred. The latter can be achieved using a the SV-MPC approach described in Chapter 5.

6.3.1 Real-time Dynamics Inference

We now focus on the problem of updating the posterior over the simulator parameters ξ_t. Note that, due to the independence of each inference problem, the frequency in which we update $p(\xi|\mathcal{D}_{1:t})$ can be different from the control policy update. Ideally, it is preferable to update the parameter distribution in *real-time*, allowing a robotic system to quickly adapt to sudden changes in the environment. To that end, we require an efficient way of incorporating recent measurements.

The inference problem at a given time-step can then be written as:

$$p(\xi|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_t|\xi, \mathcal{D}_{1:t-1})p(\xi|\mathcal{D}_{1:t-1})\,. \tag{6.2}$$

Note that in this formulation, ξ is considered time-invariant. This is based on the implicit assumption that the frequency with which we gather new observations is significantly larger than the covariate shift to which $p(\xi|\mathcal{D}_{1:t})$ is subject to as we traverse the environment.

In general, we do not have access to direct measurements of ξ, only to the system state. Therefore, in order to perform inference over the dynamics parameters, we rely on a generative model, *i.e.*the simulator $\hat{f}_\xi$, to generate samples in the state space $\mathcal{X}$ for different values of ξ. However, unlike in the policy inference step, for the dynamics parameter estimation we are not computing deterministic simulated rollouts, but rather trying to find the explanatory parameter for each observed transition in the environment. Namely, we have:

$$\mathbf{x}_t^r = f(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}) = \hat{f}_\xi(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}) + \eta_t \,, \tag{6.3}$$

where $\mathbf{x}_t^r$ denotes the true system state and η_t is a time-dependent random variable closely related to the *reality gap* in the sim-to-real literature [171] and incorporates all the complexities of the real system not captured in simulation, such as model mismatch, unmodelled dynamics, etc. As result, the distribution of η_t is unknown, correlated over time and hard to estimate.

In practice, for the feasibility of the inference problem, we make the standard assumption that the noise is distributed according to a time-invariant normal distribution $\eta_t \sim \mathcal{N}(0, \Sigma_{\text{obs}})$, with an empirically chosen covariance matrix. More concretely, this allows us to define the likelihood term in Eq. (6.2) as:

$$\begin{aligned} \ell(\xi|\mathcal{D}_{1:t}) &:= p(\mathcal{D}_t|\xi, \mathcal{D}_{1:t-1}) \\ &= p(\mathbf{x}_t^r|\xi, \mathcal{D}_{1:t-1}) \\ &= \mathcal{N}(\mathbf{x}_t^r; \hat{f}_\xi(\mathbf{x}_{t-1}^r, \mathbf{u}_{t-1}), \Sigma_{\text{obs}}) \,, \end{aligned} \tag{6.4}$$

where we leverage the symmetry of the Gaussian distribution to center the uncertainty around $\mathbf{x}_t^r$. The likelihood of $\ell(\xi|\mathcal{D}_{1:t})$ depends only on the current observation tuple given

by $\mathcal{D}_t$, and we can drop the conditioning on previously observed data. In other words, we now have a way to quantify how likely is a given realisation of ξ based on the data we have collected from the environment. Furthermore, let us define a single observation $\mathcal{D}_t = (\mathbf{x}_t^r, \mathbf{u}_{t-1}^r, \mathbf{x}_{t-1}^r)$ as the tuple of last applied control action and observed state transition. Inferring exclusively over the current state is useful whenever frequent observations are received and prevents us from having to store information over the entire observation dataset.

Equipped with Eq. (6.4) and assuming that an initial prior $p(\xi)$ is available, we can proceed by approximating each prior at time t with a set of Stein particles $\{\xi^i\}_{i=1}^{N_\xi}$ following $q(\xi|\mathcal{D}_{1:t-1})$, so that our posterior over ξ at a given time t can then be rewritten as:

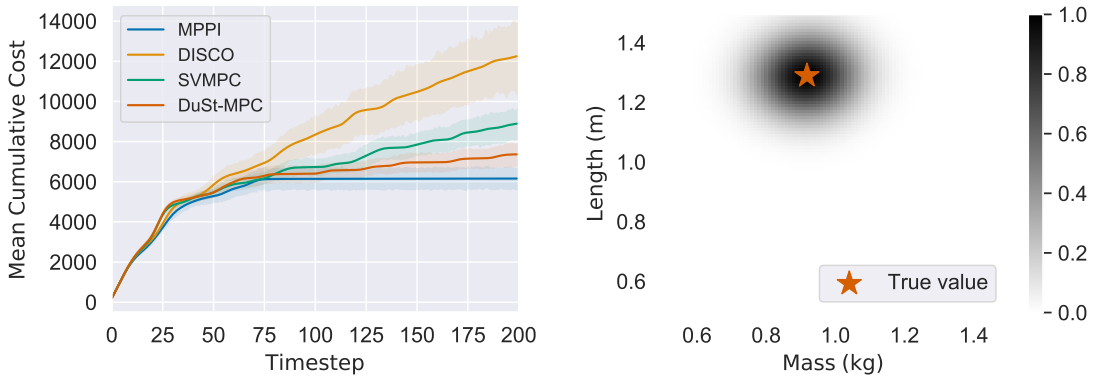
$$p(\xi|\mathcal{D}_{1:t}) \approx q(\xi|\mathcal{D}_{1:t}) \propto \ell(\xi|\mathcal{D}_t) q(\xi|\mathcal{D}_{1:t-1}) \,, \tag{6.5}$$

and we can make recursive updates to $q(\xi|\mathcal{D}_{1:t})$ by employing it as the prior distribution for the following step. Namely, we can iteratively update $q(\xi|\mathcal{D}_{1:t})$ a number of steps L by applying the SVGD functional gradient update (Eq. (5.25), where the score function can approximated as

$$\nabla_\xi \log p_t(\xi|\mathcal{D}_{1:t}) \approx \nabla_\xi \log p(\mathcal{D}_t|\xi) + \nabla_\xi \log q_t(\xi|\mathcal{D}_{1:t}) \,. \tag{6.6}$$

An element needed to evaluate Eq. (6.6) is an expression for the gradient of the posterior density. An issue in sequential Bayesian inference is that there is no exact expression for the posterior density [172]. Namely, we know the likelihood function, but the prior density is only represented by a set of particles, not the density itself.

One could forge an empirical distribution $q(\xi|\mathcal{D}_{1:t}) = \frac{1}{N_\xi} \sum_{i=1}^{N_\xi} \delta(\xi^i)$ by assigning Dirac functions at each particle location, but we would still be unable to differentiate the posterior. In practice, we need to apply an efficient density estimation method, as we need to compute the density at each optimisation step. We choose to approximate the posterior density with



(a) Pendulum costs

(b) Pendulum posterior

Figure 6.2: Inverted pendulum results. (a) The image shows the mean cumulative cost over 10 episodes. The shaded region represents the 50% confidence interval. The high variance is expected since each scenario has parameters sampled from a uniform distribution. (b) Plot of the posterior distribution over the pendulum pole-mass at the final step of one of the episodes. The true latent value is shown by the red star marker.

an equal-weight Gaussian Mixture Model (GMM) with a fixed diagonal covariance matrix:

$$q(\xi|\mathcal{D}_{1:t}) = \frac{1}{N_\xi}\sum_{i=1}^{N_\xi}\mathcal{N}(\xi;\xi^i,\Sigma_\mathrm{s})\,, \tag{6.7}$$

where the covariance matrix Σ_s can be predetermined or computed from data. One option, for example, is to use a Kernel Density bandwidth estimation heuristic, such as Improved Sheather Jones [173], Silverman's [174] or Scott's [175] rule, to determine the standard deviation σ and set $\Sigma_\mathrm{s} = \sigma^2\mathrm{I}$.

6.4 Experiments

In the following section we present experiments, both in simulation and with a physical autonomous ground vehicle (AGV), to demonstrate the correctness and applicability of our method.

	Point-mass		Pendulum	
	Cost ($\mu \pm \sigma$)	Succ.[†]	Cost ($\mu \pm \sigma$)	Succ.[‡]
MPPI[§]	—	—	30.8 ± 12.6	100%
DISCO	250.8 ± 29.9	20%	61.3 ± 40.0	70%
SVMPC	191.7 ± 56.5	25%	44.5 ± 17.9	70%
DuSt-MPC	$\mathbf{118.3 \pm 07.9}$	100%	$\mathbf{36.8 \pm 14.0}$	80%

Table 6.1: Simulation results. Summary of results for simulation experiments. The mean episode cost is given by the sum of the instant costs over the episode length. Values shown do not include the crash penalty for a more comparable baseline. [§]Not used in the navigation task; has perfect knowledge in the pendulum task. [†]Successes are episodes with no crashes. [‡]Successes are episodes whose last five steps have a instant cost below 4 ($\approx$10° from the upright position).

6.4.1 Inverted pendulum with uncertain parameters

We first investigate the performance of DuSt-MPC in the classic inverted pendulum control problem. As usual, the pendulum is composed of a rigid pole-mass system controlled at one end by a 1-degree-of-freedom torque actuator. The task is to balance the point-mass upright, which, as the controller is typically under-actuated, requires a controlled swing motion to overcome gravity. Contrary to the typical case, however, in our experiments the mass and length of the pole-mass are unknown and equally likely within a range of 0.5 kg to 1.5 kg and 0.5 m to 1.5 m, respectively.

At each episode, a set of latent model parameters is sampled and used in the simulated environment. Each method is then deployed utilising this same parameter set. MPPI is used as a baseline and has *perfect knowledge* of the latent parameters. This provides a measure of the task difficulty and achievable results. As discussed in Section 6.2, we compare against DISCO and SVMPC as additional baselines. We argue that, although these methods perform no online update of their knowledge of the world, they offer a good underpinning for comparison since the former tries to leverage the model uncertainty to create more robust policies, whereas the latter shares the same variational inference principles as our method. DISCO is implemented in its unscented transform variant applied

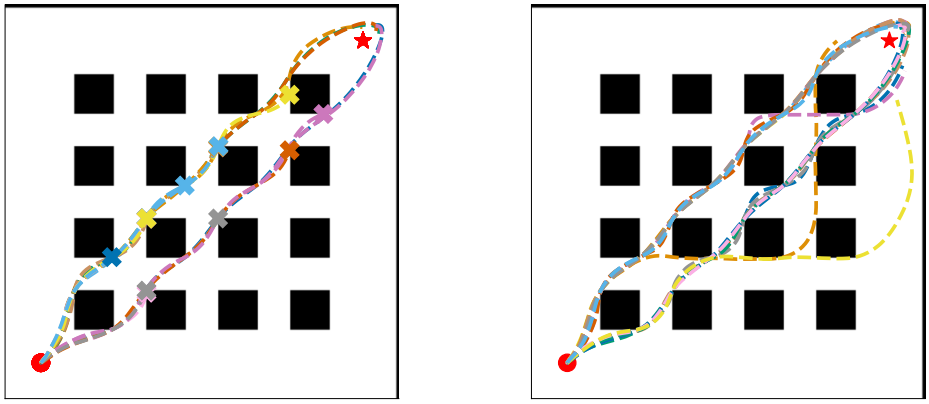
to the uninformative prior used to generate the random environments. SVMPC uses the mean values for mass and length as point-estimates for its fixed parametric model. For more details on the hyper-parameters used, refer to Appendix A.

Figure 6.2a presents the average cumulative costs over 10 episodes. Although the results show great variance, due to the randomised environment, it is clear that DuSt-MPC outperforms both baselines. Careful consideration will show that the improvement is more noticeable as the episode progresses, which is expected as the posterior distribution over the model parameters being used by DuSt-MPC gets more informative. The final distribution over mass and length for one of the episodes is shown in Fig. 6.2b. Finally, a summary of the experimental results is presented in Table 6.1.

6.4.2 Point-mass navigation on an obstacle grid

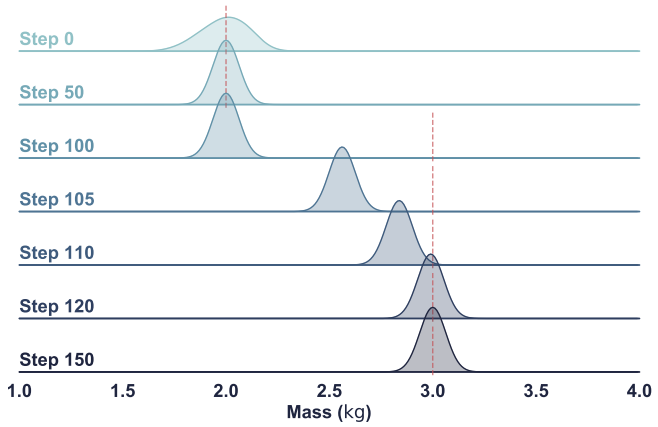
Here, we reproduce and extend the planar navigation task presented in [29]. We construct a scenario in which an holonomic point-mass robot must reach a target location while avoiding obstacles. As in [29], colliding with obstacles not only incurs a high cost penalty to the controller, but prevents all future movement, simulating a crash. The non-differentiable cost function makes this a challenging problem, well-suited for sampling-based approaches. Obstacles lie in an equally spaced 4-by-4 grid, yielding several multi-modal solutions. This is depicted in Fig. 6.3. Additionally, we include barriers at the boundaries of the simulated space to prevent the robot from easily circumventing obstacles.

The system dynamics is represented as a double integrator model with non-unitary mass m, he particle acceleration is given by $\ddot{\mathbf{x}} = m^{-1}\mathbf{u}$ and the control signal is the force applied to the point-mass. In order to demonstrate the inference over dynamics, we forcibly change the mass of the robot at a fixed step of the experiment, adding extra weight. This has a direct parallel to several tasks in reality, such as collecting a payload or passengers while executing a task. Assuming the goal position is denoted by $\mathbf{x}_g$, the cost function that defines



(a) SVMPC trajectories

(b) DuSt-MPC trajectories



(c) Ridge plot of mass distribution (in kg)

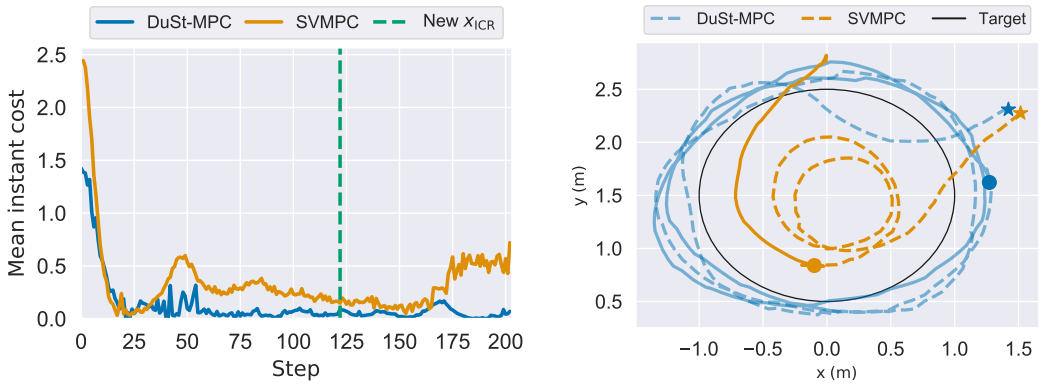
Figure 6.3: Point-mass navigation task. The plots shows trajectories from the start position (red dot) towards the goal (red star). (a) Trajectories executed by SVMPC. Note that, as the mass of the robot changes, the model mismatch causes many of the episodes to crash (**x** markers). (b) Trajectories executed by DuSt-MPC. Depending on the state of the system when the mass change occurs, a few trajectories deviate from the centre path to avoid collisions. A few trajectories are truncated due to the fixed episode length. (c) Ridge plot of the distribution over mass along several steps of the simulation. The vertical dashed line denotes the true mass. Mass is initially set at 2 kg, and changed to 3 kg at step 100.

the task is given by:

$$c(\mathbf{x}_t, \mathbf{u}_t) = 0.5\mathbf{e}_t^\mathsf{T}\mathbf{e}_t + 0.25\dot{\mathbf{x}}_t^\mathsf{T}\dot{\mathbf{x}}_t + 0.2\mathbf{u}_t^\mathsf{T}\mathbf{u}_t + p \cdot \mathbb{1}\{\text{col.}\}$$

$$c_{term}(\mathbf{x}_t, \mathbf{u}_t) = 1000\mathbf{e}_t^\mathsf{T}\mathbf{e}_t + 0.1\dot{\mathbf{x}}_t^\mathsf{T}\dot{\mathbf{x}}_t\,,$$

where $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_g$ is the instantaneous position error and $p = 10^6$ is the penalty when



(a) Raw costs

(b) Trajectories

Figure 6.4: AGV trajectory tracking results. (a) Raw cost over time. Amount of steps before and after the change of mass are normalised for proper comparison. (b) Trajectories executed by each method. Line style changes when mass changes. Markers denote initial and change of mass position.

a collision happens. A detailed account of the hyper-parameters used in the experiment is presented in Chapter A.

As a baseline, we once more compare against DISCO and SVMPC. In Fig. 6.3 we present an overlay of the trajectories for SVMPC and DuSt-MPC over 20 independent episodes and we choose to omit trajectories of DISCO for conciseness. Collisions to obstacles are denoted by a **x** marker. Note that in a third of the episodes SVMPC is unable to avoid obstacles due to the high model mismatch while DuSt-MPC is able to avoid collisions by quickly adjusting to the new model configuration online. A typical sequential plot of the posterior distribution induced by fitting a GMM as in Eq. (6.7) is shown in Fig. 6.3c for one episode. There is little variation between episodes and the distribution remains stable in the intermediate steps not depicted.

6.4.3 Trajectory tracking with autonomous ground vehicle

We now present experimental results with a physical autonomous ground robot equipped with a skid-steering drive mechanism. The kinematics of the robot are based on a modified unicycle model, which accounts for skidding via an additional parameter [176]. The

parameters of interest in this model are the robot's wheel radius r_{w}, axial distance a_{w}, i.e. the distance between the wheels, and the displacement of the robot's ICR from the robot's centre x_{ICR}. A non-zero value on the latter affects turning by sliding the robot sideways. The robot is velocity controlled and, although it possess four-wheel drive, the controls is restricted to two-degrees of freedom, left and right wheel speed. Individual wheel speeds are regulated by a low-level proportional-integral controller.

The robot is equipped with a 2D Hokuyo LIDAR and operates in an indoor environment in our experiments. Prior to the tests, the area is pre-mapped using the gmapping package [177] and the robot is localised against this pre-built map. Similar to the experiment in Section 6.4.2, we simulate a change in the environment that could be captured by our parametric model of the robot to explain the real trajectories. However, we are only applying a relatively simple kinematic model in which the effects of the dynamics and ground-wheel interactions are not accounted for. Therefore, friction and mass are not feasible inference choices. Hence, out of the available parameters, we opted for inferring x_{ICR}, the robot's centre of rotation. Since measuring x_{ICR} involves a laborious process, requiring different weight measurements or many trajectories from the physical hardware [178], this also makes the experiment more realistic. To circumvent the difficulties of ascertaining x_{ICR}, we use the posterior distribution estimated in [157], and bootstrap our experiment with $x_{\mathrm{ICR}} \sim \mathcal{N}(0.5, 0.2^2)$.

To reduce the influence of external effects, such as localisation, we defined a simple control task of following a circular path at a constant tangential speed. Costs were set to make the robot follow a circle of 1 m radius with $c(\mathbf{x}_t) = \sqrt{d_t^2 + 10(s_t - s_0)^2}$, where d_t represents the robot's distance to the edge of the circle and $s_0 = 0.2\,\mathrm{m/s}$ is a reference linear speed.

The initial particles needed by DuSt-MPC in Eq. (6.6) for the estimation of x_{ICR} are sampled from the bootstrapping distribution, whereas for SVMPC we set $x_{\mathrm{ICR}} = 0.5\,\mathrm{m}$, the distribution mean. Again, we want to capture whether our method is capable of adjusting

to environmental changes. To this end, approximately halfway through the experiment, we add an extra load of approximately 5.3 kg at the rear of the robot in order to alter its centre of mass. These moments are indicated on the trajectories shown in Fig. 6.4b. In Fig. 6.4a we plot the instant costs for a fixed number of steps before and after the change of mass. For the complete experiment parameters refer to Appendix A.

We observe that considering the uncertainty over x_{ICR} and, crucially, refining our knowledge over it allows DuSt-MPC to significantly outperform the SVMPC baseline. Focusing on the trajectories from SVMPC we note that our estimation of x_{ICR} is probably not accurate. As the cost function emphasises the tangential speed over the cross-track error, this results in circles correctly centred, but of smaller radius. Crucially though, the algorithm cannot overcome this poor initial estimation. DuSt-MPC initially appears to find the same solution, but quickly adapts, overshooting the target trajectory and eventually converging to a better result. This behaviour can be observed both prior to and after the change in the robot's mass. Conversely, with the addition of mass, the trajectory of SVMPC diverged and eventually led the robot to a halt.

6.5 Discussion

We present a method capable of simultaneously estimating model parameters and controls. The method expands previous results in control as implicit variational inference and provides the theoretical framework to formally incorporate uncertainty over simulator parameters. By encapsulating the uncertainty on dynamic systems as distributions over a parametric model, we are able to incorporate domain knowledge and physics principles while still allowing for a highly representative model. Crucially, we perform an *online* refinement step where the agent leverages system feedback in a sequential way to efficiently update its beliefs regardless of the size of observation dataset.

Simulated experiments are presented for a randomised inverted pendulum environment and obstacle grid with step changes in the dynamical parameters. Additionally, a trajec-

tory tracking experiment utilising a custom built AGV demonstrates the feasibility of the method for online control. The results illustrate how the simplifications on dynamic inference are effective and allow for a quick adjustment of the posterior belief resulting in a *de facto* adaptive controller. Consider, for instance, the inverted pendulum task. In such periodic and non-linear system, untangling mass and length can pose a difficult challenge as there are many plausible solutions [164]. Nonetheless, the results obtained are quite encouraging, even with very few mapping steps per control loop.

Finally, we demonstrated how, by incorporating uncertainty over the model parameters, DuSt-MPC produces more robust policies to dynamically changing environments, even when the posterior estimation of the model parameters is rather uncertain. This can be seen, for instance, in the adjustments made to trajectories in the obstacle grid.

CHAPTER 7
CONCLUSION

Interpretable sensory integration is an important component to building robust autonomous systems. Training perceptual components from collected data can benefit from informed design during data collection, model construction, and for merging perceptual modalities. Incorporating structure and prior knowledge directly into parameteric models is a useful strategy for improving scalability and performance.

Reasoning about uncertainty and probability distributions in an online fashion can be expensive, particulary for high-dimensional problems encountered in robotics. Fortunately, the use of GPU-accelerated computation for robotic platforms is increasing. This is timely, as new developments in distributed Bayesian inference are well positioned to take advantage of this parallelization.

This thesis addressed these two central themes, providing methods and formal insights to further the adoption of modern learning and inference techniques in robotics. In Chapter 2 and Chapter 3, we examined the role of both *non-parametric* and *parametric* structure in designing learnable perception modules for both visual and tactile modalities. Forward models for visual prediction were demonstrated to benefit from key-frame robot pose data, learning ensembles of residual transformations to accurately predict projected appearance from arbitrary robot configurations. This was shown to be additionally capable of predicting occluded regions in cluttered scenes. We then described a novel method for interpreting raw signals from a sophisticated bio-mimetic tactile sensor, where electrode array structure and sensor geometry were encoded into the network architecture. A strategy for automating data collection with minimal human intervention was outlined to scale learning across manipulation tasks. Chapter 4 demonstrated the use of geometric and physics-based priors for fusing visual and tactile models in a probabilistic framework. This was shown to

improve estimation of object poses and contact forces during partially-observable manipulation tasks.

In Chapter 5 and Chapter 6, we examined distributed approaches to variational inference for control, planning and parameter estimation. A family of sampling-based, model-predictive controllers was derived, based on equivalent formulations between optimal control and Bayesian posterior approximation. Optimization was performed using Stein Variational Gradient Descent (SVGD), using appropriate Markovian kernels for trajectory-wise inference and leveraging parallel evaluation of control policies. This also led to a formulation for trajectory optimization problems exhibiting deterministic dynamics, where structured priors can be incorporated to plan for smooth motions. The SVGD approach was then extended to combine particle-based MPC with online parameter estimation, addressing scenarios involving uncertainty in physical values such as mass and inertial center of rotation. Augmenting the inference procedure to include model adaptation demonstrated improved robustness on simulated and real systems.

7.1 Future Directions

The SV-MPC perspective outlined in Chapter 5 is one example of using particle-based representations for variational inference in control and planning. A variety of other possible methods exist, including Hamiltonian Monte Carlo (HMC) [179], Particle Mirror Descent (PMD) [180], and Particle Optimization (PO) [181], among others. Although SVGD has been shown numerically to outperform many of these candidates in low-dimensional problems [26], the specific trade-offs between these choices with regards to convergence, computational overhead, and accuracy for inference over trajectory spaces remain to be fully elaborated and compared.

Theoretical analysis of SVGD and related algorithms is also still in its infancy, but is being actively developed [28]. To-date, convergence analysis of SVGD has largely been constrained to the large-sample asymptotic limit (*e.g.* infinite particles), as in [182]. The

KL divergence is guaranteed to decrease monotonically, with a rate bounded from above by the kernelized Stein discrepancy. A recent study has derived bounds for finite-sample approximations [183]. How these rates translate to performance in the MPC and planning context is currently unclear, but offers an exciting direction for future research.

A promising avenue for distributed posterior representations would be to include the application of particle-based control schemes to KL-regularized reinforcement learning [184]. The approximate inference problem described in Eq. (5.8) is directly related to entropy-regularized policy evaluation and value approximation. As such, the problem of finding low-variance, sample-efficient approximation of expected-costs may be particularly amenable to modern variational inference methods.

Additionally, the incorporation of parameter uncertainty into model-based reinforcement learning has been generally limited to the episodic setting [185, 138]. To realize safe and robust learning, we should consider how the agent manages uncertainty and adapts its belief during execution. This would mean including online parameter adaptation, active learning and adaptive control for minimizing Bayesian regret at both the continual and episodic level [186, 163]. Efficient, online methods for resolving complex posterior distributions should be very helpful in this regard.

Further, by considering a Bayesian formulation of model-predictive control, we can incorporate priors over action spaces in a principled way. Obtaining meaningful priors is non-trivial. However these can be derived from expert or human demonstrations [187, 188], learned from experience [189, 190] or take the form of a trajectory or skill library [191]. Ideally, such informed priors may be conditioned on the context, such as the task and environment setting [192].

With the availability of fast, GPU-accelerated simulators [153] and increasingly sophisticated methods for bridging the sim-to-real gap [193, 194, 164, 195], it is becoming conceivable to employ simulators within sampling-based control and state-estimation loops during real-time execution [167, 196]. However, to effectively utilize such parallelized

computation, we need principled methods for resolving high-dimensional uncertainty over actions and model parameters. This could be achieved by considering the natural, non-Euclidean geometry induced by system kinematics and constraints of the system [197]. By ensuring that our sampling space lies on a known Riemannian manifold, for example, we can improve sample efficiency by implicitly accounting for system geometry [198].

Appendices

APPENDIX A

CHAPTER 6: EXPERIMENT PARAMETERS

Parameter	Inverted Pendulum	Point-mass Navigation	AGV Traj. Tracking
Initial state, $\mathbf{x}_0$	[3rad, 0 m/sec]	—	—
Environment maximum speed	5m/sec	—	—
Environment maximum acceleration	10m/sec^2	—	—
Policy samples, N_a	32	64	50
Dynamics samples, N_s	8	4	4
Cost Likelihood inverse temperature, α	1.0	1.0	1.0
Control authority, Σ	2.0^2	5.0^2	0.1^2
Control horizon, H	20	40	20
Number of policies, N_π	3	6	2
Policy Kernel, $k_\pi(\cdot,\cdot)$		Radial Basis Function	
Policy Kernel bandwidth selection		Silverman's rule	
Policy prior covariance, Σ_a	2.0^2	5.0^2	1.0^2
Policy step size, ϵ	2.0	100.0	0.02
Dynamics prior distribution			
Dynamics number of particles, N_ξ	50	50	50
Dynamics Kernel, $k_\xi(\cdot,\cdot)$		Radial Basis Function	
Dynamics GMM covariance, Σ_s	Improved Sheather Jones	0.25^2	0.0625^2
Dynamics likelihood covariance, Σ_obs	0.1^2	0.1^2	0.1^2
Dynamics update steps, L	20	20	5
Dynamics step size, ϵ	0.001	0.01	0.05
Dynamics in log space	No	Yes	No
Unscented Transform spread [157], α	0.5	—	—

Table A.1: Hyperparameters used in the experiments.

REFERENCES

[1] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9426–9432.

[2] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects", in *Conference on Robot Learning*, PMLR, 2018, pp. 306–316.

[3] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning", in *Robotics: science and systems*, 2018.

[4] C. Finn and S. Levine, "Deep visual foresight for planning robot motion", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2786–2793.

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies", *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[6] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov, "Learning deep visuomotor policies for dexterous hand manipulation", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 3636–3643.

[7] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8958–8964.

[8] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control", in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 8299–8310.

[9] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned", *The International Journal of Robotics Research*, p. 0 278 364 920 987 859, 2021.

[10] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, *et al.*, "The limits and potentials of deep learning for robotics", *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[11] A. Webb, C. Reynolds, W. Chen, H. Reeve, D. Iliescu, M. Lujan, and G. Brown, "To ensemble or not ensemble: When does end-to-end training fail?", *stat*, vol. 1050, p. 6, 2020.

[12] T. Glasmachers, "Limits of end-to-end learning", in *Asian Conference on Machine Learning*, PMLR, 2017, pp. 17–32.

[13] A. Byravan, F. Leeb, F. Meier, and D. Fox, "Se3-pose-nets: Structured deep dynamics models for visuomotor control", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3339–3346.

[14] A. Lambert, A. Shaban, A. Raj, Z. Liu, and B. Boots, "Deep forward and inverse perceptual models for tracking and prediction", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 675–682.

[15] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks", in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 173–180.

[16] R. Jonschkowski, R. Hafner, J. Scholz, and M. Riedmiller, "Pves: Position-velocity encoders for unsupervised learning of structured state representations", *arXiv preprint arXiv:1705.09805*, 2017.

[17] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning", in *International Conference on Learning Representations*, 2018.

[18] M. Lutter, J. Silberbauer, J. Watson, and J. Peters, "Differentiable physics models for real-world offline model-based reinforcement learning", *arXiv preprint arXiv:2011.01734*, 2020.

[19] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving", *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[20] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars", in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[21] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press Cambridge, 2000, vol. 1.

[22] S. Thrun, "Probabilistic robotics", *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[23] Z. Chen *et al.*, "Bayesian filtering: From kalman filters to particle filters, and beyond", *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[24] B. Sundaralingam, A. S. Lambert, A. Handa, B. Boots, T. Hermans, S. Birchfield, N. Ratliff, and D. Fox, "Robust learning of tactile force estimation through robot interaction", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9035–9042.

[25] A. S. Lambert, M. Mukadam, B. Sundaralingam, N. Ratliff, B. Boots, and D. Fox, "Joint inference of kinematic and force trajectories with visuo-tactile sensing", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 3165–3171.

[26] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm", in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 2378–2386.

[27] C. Stein *et al.*, "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables", in *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, The Regents of the University of California, 1972.

[28] A. Anastasiou, A. Barp, F.-X. Briol, B. Ebner, R. E. Gaunt, F. Ghaderinezhad, J. Gorham, A. Gretton, C. Ley, Q. Liu, L. W. Mackey, C. Oates, G. Reinert, and Y. Swan, "Stein's method meets statistics: A review of some recent developments", 2021.

[29] A. Lambert, A. Fishman, D. Fox, B. Boots, and F. Ramos, "Stein variational model predictive control", in *Proceedings of the 4th Annual Conference on Robot Learning*, 2020.

[30] L. Barcelos, A. Lambert, R. Oliveira, P. Borges, B. Boots, and F. Ramos, "Dual online stein variational inference for control and dynamics", in *Robotics: Science and Systems*, 2021.

[31] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[32] A. Dearden and Y. Demiris, "Learning forward models for robots", in *IJCAI*, vol. 5, 2005, p. 1440.

[33] S. Ulbrich, M. Bechtel, T. Asfour, and R. Dillmann, "Learning robot dynamics with kinematic bezier maps", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 3598–3604.

[34] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: Incremental real time learning in high dimensional space", in *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 2000, pp. 1079–1086.

[35] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics", in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, IEEE, vol. 1, 2001, pp. 298–303.

[36] B. Damas and J. Santos-Victor, "An online algorithm for simultaneously learning forward and inverse kinematics", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1499–1506.

[37] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning", in *Advances in Neural Information Processing Systems*, 2009, pp. 1193–1200.

[38] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction", in *Advances In Neural Information Processing Systems*, 2016, pp. 64–72.

[39] C. Paxton, Y. Barnoy, K. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning", in *2019 international conference on robotics and automation (ICRA)*, IEEE, 2019, pp. 8832–8838.

[40] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections.", in *CoRL*, 2017, pp. 344–356.

[41] N. Hirose, F. Xia, R. Martin-Martin, A. Sadeghian, and S. Savarese, "Deep visual mpc-policy learning for navigation", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3184–3191, 2019.

[42] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks", in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2758–2766.

[43] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network", in *Advances in Neural Information Processing Systems*, 2015, pp. 2539–2547.

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[45] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", *arXiv preprint arXiv:1511.06434*, 2015.

[46] A. Dosovitskiy, J. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks", *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[47] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Single-view to multi-view: Reconstructing unseen views with a convolutional network", *CoRR abs/1511.06702*, 2015.

[48] Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky, "Deepwarp: Photorealistic image resynthesis for gaze manipulation", in *European Conference on Computer Vision*, Springer, 2016, pp. 311–326.

[49] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow", in *European Conference on Computer Vision*, Springer, 2016, pp. 286–301.

[50] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches", *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[51] P. I. Corke, "Visual control of robot manipulators-a review", *Visual servoing*, vol. 7, pp. 1–31, 1993.

[52] S. Maneewongvatana and D. M. Mount, "It's okay to be skinny, if your friends are fat", in *Center for Geometric Computing 4th Annual Workshop on Computational Geometry*, vol. 2, 1999, pp. 1–8.

[53] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks", in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[54] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", *arXiv preprint arXiv:1408.5093*, 2014.

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

[56] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets", *arXiv preprint arXiv:1405.3531*, 2014.

[57] C. K. Liu and S. Jain, "A quick tutorial on multibody dynamics", *Online tutorial, June*, p. 7, 2012.

[58] B. Boots, A. Byravan, and D. Fox, "Learning predictive models of a depth camera and manipulator from raw execution traces", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 4021–4028.

[59] M. Mirza and S. Osindero, "Conditional generative adversarial nets", *arXiv preprint arXiv:1411.1784*, 2014.

[60] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", *arXiv preprint arXiv:1511.06434*, 2015.

[61] J. Gauthier, "Conditional generative adversarial nets for convolutional face generation", *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, vol. 2014, no. 5, p. 2, 2014.

[62] G. Kahn, P. Sujan, S. Patil, S. Bopardikar, J. Ryde, K. Goldberg, and P. Abbeel, "Active exploration using trajectory optimization for robotic grasping in the presence of occlusions", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 4783–4790.

[63] S.-K. Kim and M. Likhachev, "Planning for grasp selection of partially occluded objects", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 3971–3978.

[64] J. Pajarinen and V. Kyrki, "Robotic manipulation of multiple objects as a pomdp", *Artificial Intelligence*, vol. 247, pp. 213–228, 2017.

[65] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez, "Symmetrical dense optical flow estimation with occlusions detection", *International Journal of Computer Vision*, vol. 75, no. 3, pp. 371–385, 2007.

[66] S. Ince and J. Konrad, "Occlusion-aware optical flow estimation", *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1443–1451, 2008.

[67] A. A. Shenoi, T. Bhattacharjee, and C. C. Kemp, "A CRF that combines touch and vision for haptic mapping", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2255–2262.

[68] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, "Deep learning for tactile understanding from visual and haptic data", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 536–543.

[69] R. Calandra, J. Lin, A. Owens, J. Malik, U. Berkeley, D. Jayaraman, and E. H. Adelson, "More than a feeling: Learning to grasp and regrasp using vision and touch", in *NIPS*, 2017, pp. 1–10.

[70] P. S. Girão, P. M. P. Ramos, O. Postolache, and J. M. D. Pereira, "Tactile sensors for robotic applications", *Measurement*, vol. 46, no. 3, pp. 1257–1271, 2013.

[71] D. Xu, G. E. Loeb, and J. A. Fishel, "Tactile identification of objects using bayesian exploration", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3056–3061.

[72] R. Li, R. Platt, W. Yuan, A. ten Pas, N. Roscup, M. A. Srinivasan, and E. Adelson, "Localization and manipulation of small parts using GelSight tactile sensing", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2014, pp. 3988–3993.

[73] F. Veiga, H. van Hoof, J. Peters, and T. Hermans, "Stabilizing Novel Objects by Learning to Predict Tactile Slip", in *iros*, Hamburg, Germany, 2015.

[74] N. Wettels, A. R. Parnandi, J.-H. Moon, G. E. Loeb, and G. S. Sukhatme, "Grip control using biomimetic tactile sensing systems", *IEEE/ASME Transactions On Mechatronics*, vol. 14, no. 6, pp. 718–723, 2009.

[75] M. Meier, G. Walck, R. Haschke, and H. J. Ritter, "Distinguishing sliding from slipping during object pushing", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5579–5584.

[76] I. Agriomallos, S. Doltsinis, I. Mitsioni, and Z. Doulgeri, "Slippage detection generalizing to grasping of unknown objects using machine learning with novel features", *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 942–948, 2018.

[77] B. Heyneman and M. R. Cutkosky, "Slip classification for dynamic tactile array sensors", *The International Journal of Robotics Research*, vol. 35, no. 4, pp. 404–421, 2016.

[78] J. Hoelscher, J. Peters, and T. Hermans, "Evaluation of Tactile Feature Extraction for Interactive Object Recognition", in *IEEE-RAS International Conference on Humanoid Robotics*, Seoul, Korea, 2015.

[79] F. Veiga, J. Peters, and T. Hermans, "Grip stabilization of novel objects using slip prediction", *IEEE Transactions on Haptics*, 2018.

[80] L. Chia-Hsien, J. A. Fishel, and G. E. Loeb, "Estimating point of contact, force and torque in a biomimetic tactile sensor with deformable skin", *Tech. Rep.*, 2013.

[81] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor", in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 297–303.

[82] N. Wettels, J. A. Fishel, and G. E. Loeb, "Multimodal tactile sensor", in *The Human Hand as an Inspiration for Robot Hand Development*, Springer, 2014, pp. 405–429.

[83] J. Fishel, G. Lin, and G. Loeb, "SynTouch LLC BioTac product manual, v. 16", *Tech. Rep.*, 2013.

[84] B. Navarro, P. Kumar, A. Fonte, P. Fraisse, G. Poisson, and A. Cherubini, "Active calibration of tactile sensors mounted on a robotic hand", in *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[85] N. Wettels and G. E. Loeb, "Haptic feature extraction from a biomimetic tactile sensor: Force, contact location and curvature", in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2471–2478.

[86] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning", *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996.

[87] J. Reinecke, A. Dietrich, F. Schmidt, and M. Chalon, "Experimental comparison of slip detection strategies by tactile sensing with the BioTac® on the DLR hand arm system", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2742–2748.

[88] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A computational graph for automatic motion policy generation", in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2018, pp. 441–457.

[89] T. Schmidt, K. Hertkorn, R. Newcombe, Z. Marton, M. Suppa, and D. Fox, "Depth-based tracking with physical constraints for robot manipulation", in *ICRA*, IEEE, 2015, pp. 119–126.

[90] *Coefficients of friction between materials*, https://www.engineersedge.com/coeffients_of_friction.htm, Accessed: 2018-09-13.

[91] D. Kraft, "A software package for sequential quadratic programming", DLR German Aerospace Center - Institute for Flight Mechanics, Koln, Germany, Tech. Rep. DFVLR-FB 88-28, 1988.

[92] F. Biscani, D. Izzo, and C. H. Yam, "A global optimisation toolbox for massively parallel engineering optimisation", *arXiv preprint arXiv:1004.3824*, 2010.

[93] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[94] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates", *arXiv preprint arXiv:1708.07120*, 2017.

[95] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing", *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.

[96] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research", *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/15, p. 36, 2015.

[97] J. Bimbo, L. D. Seneviratne, K. Althoefer, and H. Liu, "Combining touch and vision for the estimation of an object's pose during manipulation", in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 4021–4026.

[98] L. Zhang and J. C. Trinkle, "The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing", in *Robotics and automation (ICRA), 2012 IEEE international conference on*, IEEE, 2012, pp. 3805–3812.

[99] M. Chalon, J. Reinecke, and M. Pfanne, "Online in-hand object localization", in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 2977–2984.

[100] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem", *Aaai/iaai*, vol. 593598, 2002.

[101] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures", *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[102] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing", *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[103] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time Gaussian process motion planning via probabilistic inference", *The International Journal of Robotics Research (IJRR)*, vol. 37, no. 11, pp. 1319–1340, 2018.

[104] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Simultaneous trajectory estimation and planning via probabilistic inference", in *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[105] K.-T. Yu, J. Leonard, and A. Rodriguez, "Shape and pose recovery from planar pushing", in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 1208–1215.

[106] K.-.-T. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing. application to planar manipulation", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7778–7785.

[107] K.-T. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing for inserting a suction-held object", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1628–1635.

[108] G. E. Loeb, "Estimating point of contact, force and torque in a biomimetic tactile sensor with deformable skin", 2013.

[109] F. R. Hogan and A. Rodriguez, "Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics", *arXiv preprint arXiv:1611.08268*, 2016.

[110] K. M. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback.", in *IROS*, 1992, pp. 416–421.

[111] M. T. Mason, "Mechanics and planning of manipulator pushing operations", *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[112] S. H. Lee and M. Cutkosky, "Fixture planning with friction", *Journal of Engineering for Industry*, vol. 113, no. 3, pp. 320–327, 1991.

[113] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction", Georgia Institute of Technology, Tech. Rep., 2012.

[114] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "Isam2: Incremental smoothing and mapping using the bayes tree", *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[115] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing", in *Intelligent*

Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, IEEE, 2016, pp. 30–37.

[116] T. Schmidt, R. A. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking.", in *Robotics: Science and Systems*, vol. 2, 2014.

[117] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving", *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[118] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots, "An online learning approach to model predictive control", *arXiv preprint arXiv:1902.08967*, 2019.

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

[120] M. Kasaei, N. Lau, and A. Pereira, "A robust closed-loop biped locomotion planner based on time varying model predictive control", *arXiv preprint arXiv:1909.06873*, 2019.

[121] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 378–383.

[122] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory", *Journal of statistical mechanics: theory and experiment*, vol. 2005, no. 11, P11011, 2005.

[123] V. Gómez, S. Thijssen, A. Symington, S. Hailes, and H. J. Kappen, "Real-time stochastic optimal control for multi-agent quadrotor systems", in *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[124] W. Wiegerinck, B. v. d. Broek, and H. Kappen, "Stochastic optimal control in continuous space-time multi-agent systems", *arXiv preprint arXiv:1206.6866*, 2012.

[125] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information.", in *Robotics: Science and Systems*, 2018.

[126] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems", *Physical review letters*, vol. 95, no. 20, p. 200 201, 2005.

[127] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference", in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[128] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review", *arXiv preprint arXiv:1805.00909*, 2018.

[129] M. I. Jordan, Z. Ghahramani, and et al., "An introduction to variational methods for graphical models", in *Machine Learning*, MIT Press, 1999, pp. 183–233.

[130] M. J. Beal, "Variational algorithms for approximate bayesian inference", University College London, Tech. Rep., 2003.

[131] M. Wainwright and M. Jordan, "Graphical Models, Exponential Families, and Variational Inference", *Foundations and Trends in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[132] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm", in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 2378–2386.

[133] E. Todorov, "General duality between optimal control and estimation", in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 4286–4292.

[134] K. Rawlik, M. Toussaint, and S. Vijayakumar, "Approximate inference and stochastic optimal control", *arXiv preprint arXiv:1009.3958*, 2010.

[135] M. Toussaint, "Robot trajectory optimization using approximate inference", in *Proceedings of the 26th annual international conference on machine learning*, ACM, 2009, pp. 1049–1056.

[136] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem", *Machine learning*, vol. 87, no. 2, pp. 159–182, 2012.

[137] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes", in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 945–952.

[138] M. Okada and T. Taniguchi, "Variational inference mpc for bayesian model-based reinforcement learning", in *Conference on Robotics Learning (CoRL)*, 2019.

[139] J. Watson, H. Abdulsamad, and J. Peters, "Stochastic optimal control as approximate input inference", in *Conference on Robot Learning*, 2020, pp. 697–716.

[140] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[141] D. Ormoneit and V. Tresp, "Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates", *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 639–650, 1998.

[142] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning", *The Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[143] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and kl control", in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, pp. 1466–1473.

[144] E. A. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations", *Entropy*, vol. 17, no. 5, pp. 3352–3375, 2015.

[145] Y. Liu, P. Ramachandran, Q. Liu, and J. Peng, "Stein variational policy gradient", in *Conference on Uncertainty on Artificial Intelligence (UAI)*, 2017.

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

[147] W. Y. Chen, A. Barp, F.-X. Briol, J. Gorham, M. Girolami, L. Mackey, and C. J. Oates, "Stein point markov chain monte carlo", in *ICML*, 2019.

[148] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization", in *Handbook of statistics*, vol. 31, Elsevier, 2013, pp. 35–59.

[149] Q. Liu, "Stein variational gradient descent: Theory and applications",

[150] N. Kantas, J. Maciejowski, and A. Lecchini-Visintini, "Sequential monte carlo for model predictive control", in *Nonlinear model predictive control*, Springer, 2009, pp. 263–273.

[151] J. Zhuo, C. Liu, J. Shi, J. Zhu, N. Chen, and B. Zhang, "Message passing Stein variational gradient descent", in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholm, Sweden: PMLR, Oct. 2018, pp. 6018–6027.

[152] B. Schölkopf and A. Smola, *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2001.

[153] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk, "Non-smooth newton methods for deformable multi-body dynamics", *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–20, 2019.

[154] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym", *arXiv preprint arXiv:1606.01540*, 2016.

[155] S. S. Varadhan, *Large deviations and applications*. SIAM, 1984.

[156] C. Hartmann, L. Richter, C. Schütte, and W. Zhang, "Variational characterization of free energy: Theory and algorithms", *Entropy*, vol. 19, no. 11, p. 626, 2017.

[157] L. Barcelos, R. Oliveira, R. Possas, L. Ott, and F. Ramos, "DISCO: Double likelihood-free inference stochastic control", in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation*, Paris, France: IEEE Robotics and Automation Society, May 31, 2020, p. 7.

[158] R. Possas, L. Barcelos, R. Oliveira, D. Fox, and F. Ramos, "Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement", presented at the International Conference on Intelligent Robots and Systems (IROS), 2020, p. 8.

[159] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming", *Advances in Neural Information Processing Systems*, vol. 27, pp. 1907–1915, 2014.

[160] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search", p. 8,

[161] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models", *arXiv:1805.12114 [cs, stat]*, Nov. 2, 2018. arXiv: 1805.12114.

[162] M. Okada and T. Taniguchi, "Variational inference MPC for bayesian model-based reinforcement learning", *arXiv:1907.04202 [cs, eess, stat]*, Oct. 6, 2019. arXiv: 1907.04202.

[163] K. P. Wabersich and M. Zeilinger, "Bayesian model predictive control: Efficient model exploration and regret bounds using posterior sampling", in *Learning for Dynamics and Control*, PMLR, 2020, pp. 455–464.

[164] F. Ramos, R. Possas, and D. Fox, "BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators", in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, Jun. 2019.

[165] J. Harrison, A. Sharma, and M. Pavone, "Meta-learning priors for efficient online bayesian regression", in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2018, pp. 318–337.

[166] Y. Pan, X. Yan, E. Theodorou, and B. Boots, "Adaptive probabilistic trajectory optimization via efficient approximate inference", *29th Conference on Neural Information Processing System*, 2016.

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

[168] D. Fan, A. Agha, and E. Theodorou, "Deep learning tubes for tube MPC", in *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation, Jul. 12, 2020, ISBN: 978-0-9923747-6-1.

[169] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems", *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019.

[170] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, "Prediction under uncertainty in sparse spectrum gaussian processes with applications to filtering and control", in *Proceedings of the 34th international conference on machine learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of machine learning research, tex.pdf: http://proceedings.mlr.press/v70/pan17a/pan17a.pdf, vol. 70, International Convention Centre, Sydney, Australia: PMLR, Aug. 6, 2017, pp. 2760–2768.

[171] E. Valassakis, Z. Ding, and E. Johns, "Crossing The Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics", presented at the International Conference on Intelligent Robots and Systems (IROS), 2020. arXiv: 2008.06686 [cs].

[172] M. Pulido and P. J. van Leeuwen, "Sequential monte carlo with kernel embedded mappings: The mapping particle filter", *Journal of Computational Physics*, vol. 396, pp. 400–415, Nov. 2019.

[173] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion", *The Annals of Statistics*, vol. 38, no. 5, pp. 2916–2957, Oct. 2010.

[174] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Boston, MA: Springer US, 1986, ISBN: 978-1-4899-3324-9.

[175] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*, 1st ed., ser. Wiley Series in Probability and Statistics. Wiley, Aug. 17, 1992, ISBN: 978-0-470-31684-9.

[176] K. Kozlowski and D. Pazderski, "Modeling and control of a 4-wheel skid-steering mobile robot", *Int. J. Appl. Math. Comput. Sci.*, vol. 14, no. 4, pp. 477–496, 2004.

[177] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters", *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[178] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation", *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1087–1097, Oct. 2009, Conference Name: IEEE Transactions on Robotics.

[179] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo", *arXiv preprint arXiv:1701.02434*, 2017.

[180] B. Dai, N. He, H. Dai, and L. Song, "Provable bayesian inference via particle mirror descent", in *AISTATS*, 2015.

[181] C. Chen and R. Zhang, "Particle optimization in stochastic gradient mcmc", *arXiv preprint arXiv:1711.10927*, 2017.

[182] Q. Liu, "Stein variational gradient descent as gradient flow", in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 3115–3123.

[183] A. Korba, A. Salim, M. Arbel, G. Luise, and A. Gretton, "A non-asymptotic analysis for stein variational gradient descent", *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[184] J. T. Springenberg, N. Heess, D. Mankowitz, J. Merel, A. Byravan, A. Abdolmaleki, J. Kay, J. Degrave, J. Schrittwieser, Y. Tassa, *et al.*, "Local search for policy iteration in continuous control", *arXiv preprint arXiv:2010.05545*, 2020.

[185] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models", in *NeurIPS*, 2018.

[186] E. Arcari, L. Hewing, M. Schlichting, and M. Zeilinger, "Dual stochastic mpc for systems with parametric and structural uncertainty", in *Learning for Dynamics and Control*, PMLR, 2020, pp. 894–903.

[187] R. Jeong, J. T. Springenberg, J. Kay, D. Zheng, Y. Zhou, A. Galashov, N. Heess, and F. Nori, "Learning dexterous manipulation from suboptimal experts", in *Conference on Robot Learning*, 2020.

[188] N. Rhinehart, R. McAllister, and S. Levine, "Deep imitative models for flexible inference, planning, and control", in *International Conference on Learning Representations*, 2019.

[189] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function", in *International Conference on Learning Representations*, 2018.

[190] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning", *arXiv preprint arXiv:2011.10024*, 2020.

[191] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai, "Learning generalizable locomotion skills with hierarchical reinforcement learning", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 413–419.

[192] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors", in *Conference on Robot Learning*, 2020.

[193] R. Antonova, A. Rai, T. Li, and D. Kragic, "Bayesian optimization in variational latent spaces with dynamic compression", in *Conference on Robot Learning*, PMLR, 2020, pp. 456–465.

[194] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers", in *Conference on Robot Learning*, PMLR, 2017, pp. 47–56.

[195] H. Bharadhwaj, Z. Wang, Y. Bengio, and L. Paull, "A data-efficient framework for training and sim-to-real transfer of navigation policies", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 782–788.

[196] J. Liang, A. Handa, K. Van Wyk, V. Makoviychuk, O. Kroemer, and D. Fox, "In-hand object pose tracking via contact feedback and gpu-accelerated robotic simulation", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 6203–6209.

[197] S. Calinon, "Gaussians on riemannian manifolds: Applications for robot learning and adaptive control", *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 33–45, 2020.

[198] C. Liu and J. Zhu, "Riemannian stein variational gradient descent for bayesian inference", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.