Probabilistic modeling of planar pushing

by

Maria Bauza Villalonga

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Signature redacted

Author ....
Department of Mechanical Engineering
May 2, 2018

Signature redacted

Certified by ....
Alberto Rodriguez
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by ....
Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses

Probabilistic modeling of planar pushing

by

Maria Bauza Villalonga

Submitted to the Department of Mechanical Engineering
on May 2, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This work studies the problem of data-driven modeling and stochastic filtering of complex dynamical systems. The main contributions are GP-SUM, a filtering algorithm tailored to systems expressed as Gaussian processes (GP), and the probabilistic modeling of planar pushing by combining input-dependent GPs and GP-SUM.

The main advantages of GP-SUM for filtering are that it does not rely on linearizations or unimodal Gaussian approximations of the belief. Moreover, it can be seen as a combination of a sampling-based filter and a probabilistic Bayes filter as GP-SUM operates by sampling the state distribution and propagating each sample through the dynamic system and observation models. Effective sampling and accurate probabilistic propagation are possible by relying on the GP form of the system, and a Gaussian mixture form of the belief. In this thesis we show that GP-SUM outperforms several GP-Bayes and Particle Filters on a standard benchmark.

To characterize the dynamics of pushing, we use input-dependent GPs to learn the motion of the pushed object after a short time step. With this approach we show that we can learn accurate data-driven models that outperform analytical models after less than 100 samples and saturate in performance with less than 1000 samples. We validate the results against a collected dataset of repeated trajectories, and use the learned models to study questions such as the nature of the variability in pushing, and the validity of the quasi-static assumption.

Finally, we illustrate how our learned model for pushing can be combined with GP-SUM, and demonstrate that we can predict heteroscedasticity, i.e., different amounts of uncertainty, and multi-modality when naturally occurring in pushing.

Thesis Supervisor: Alberto Rodriguez
Title: Associate Professor

Acknowledgments

First of all, I would like to thank my advisor Prof. Alberto Rodriguez for his support and advice during these last 3 years. Through our meetings and discussions I have learned how research must be properly conducted and how time and hard work are the right ways to empower our ideas and goals. From Prof. Rodriguez I have also learned the importance of being nice, cheerful and a careful listener. Personally, his way of interacting with students has made my graduate experience much more pleasant and easy.

I also want to specially thank MIT and its Mechanical Engineering Department, CFIS and LaCaixa for supporting my career and showing me that high aspirations and objectives build a path worth pursuing. I want to stress and thank how important the people that constitutes these organizations have become in this process. They have been there when needed, making any step in my studies easier by providing advice and unconditional help.

Since I arrived to Boston, I have been lucky enough to always found myself surrounded by very special people. From MIT, I want to thank all my labmates for how much I have learned from them, not only about research, but about life itself. Similarly, I must appreciate the tranquility and joy that I have gotten from my roommates Dani Rodan, Alex Armengol, Xavi Guell, Zrinka Anic, Marta Pita and Alvaro Gomez. Returning home and finding their smiling faces is priceless. I also thank all my new friends since I arrived to MIT for making this transition a lot more easy and fun, but also all my friends from Spain who have supported me despite distance and remained in contact.

None of this would be possible without the initial guidance and help from my family. They have always believed in my capabilities and supported me even when far from home.

Finally, my most special thank is for my dear boyfriend Ferran Alet. He is probably the main reason why all this makes sense. I have learned from his passion and creativity that it is worth pursuing our own ideas no matter how much time or effort it takes or how hard they seem. Moreover, time with him is probably what makes the biggest difference in my daily routine. He fills it with joy, smiles, fun, and tenderness in a way that is hard to explain but makes me feel deeply grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics and uncertainty come hand in hand. One of the defining challenges of robotics research is to design uncertainty-resilient behavior to overcome noise in sensing, actuation and/or dynamics. So far however most models of physical interaction in robotics are driven by experimental laws of friction and impact. These laws, such as Coulomb friction, describe the macroscopic behavior of contact by compounding variations at the microscopic level. As a result, one expects them to be accurate at most in a *statistical* sense.

This work studies the problems of data-driven modeling, simulation and filtering in systems with stochastic dynamics, with a particular interest in cases where the uncertainty in the system is complex either because it is action-dependent (heteroscedastic) or because the distribution over the state space cannot be realistically approximated by a single Gaussian distribution.

For instance, complex beliefs can naturally arise in manipulation tasks where state or action noise can make the difference between contact/separation or between sticking/sliding. The ordinary task of push-grasping a cup of coffee into your hand in Figure 1-1 illustrates the naturally occurring multimodality. Dogar and Srinivasa [8] used the observation that a push-grasped object tends to cluster into two clearly distinct outcomes—inside and outside the hand—to plan robust grasp strategies. Multimodality and complex belief distributions have been experimentally observed in a variety of manipulation actions such as planar pushing [41, 4], ground impacts [9], and bin-picking [32].

In this work, we study and model the dynamics of pushing as it is a simple manipulation

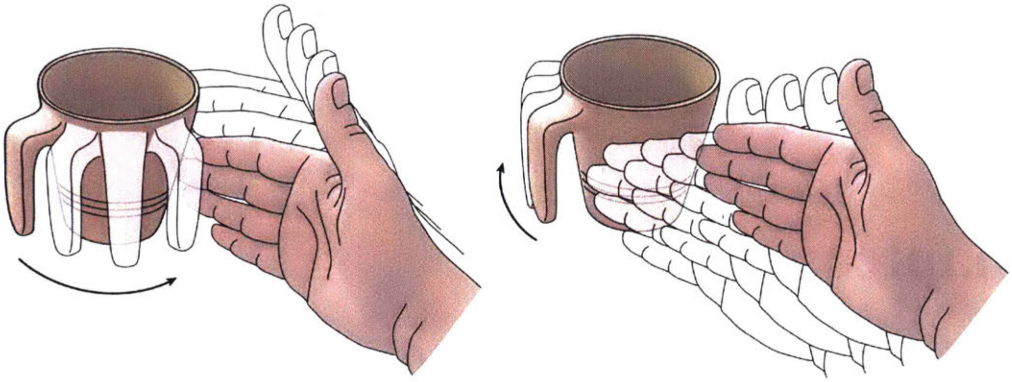

Figure 1-1: Push-grasp of a coffee cup. A small change in the initial contact between the hand and the cup, or a small change in the hand's motion can produce a very distinct—multimodal—outcome either exposing (top) or hiding (bottom) the cup's handle from the hand's palm.

task which already shows interesting statistical behavior. In previous work [41] we provide empirical evidence of the variability in the outcome of a planar push and the complexity of the state distributions formed over time when repeating the same action. Figure 1-2 shows that a series of pushes (center), indistinguishable to sensor and actuator resolution, yields divergent outcomes, while a different set of pushes (left) yields a more convergent set of outcomes. Some pushes are more precise than others, and some yield multi-modal behavior (right).

The main contributions of this work are:

- Propose a new algorithm GP-SUM to track complex state beliefs through a dynamic system expressed as a Gaussian process (GP) without the need to either linearize the transition or observation models or relying on unimodal Gaussian approximations of the belief.

- Learn a compact data-driven model that captures the first two moments, i.e., mean and variance, of the expected behavior of a pushed object after a short period of time.

- Combine the learned model for the dynamics of pushing with the algorithm GP-SUM to reason about the propagation of uncertainty over long pushes. We expect that this technique can be the basis for more realistic simulation, can aid in the design of robust plans or control policies, and can yield more statistically sound inference.

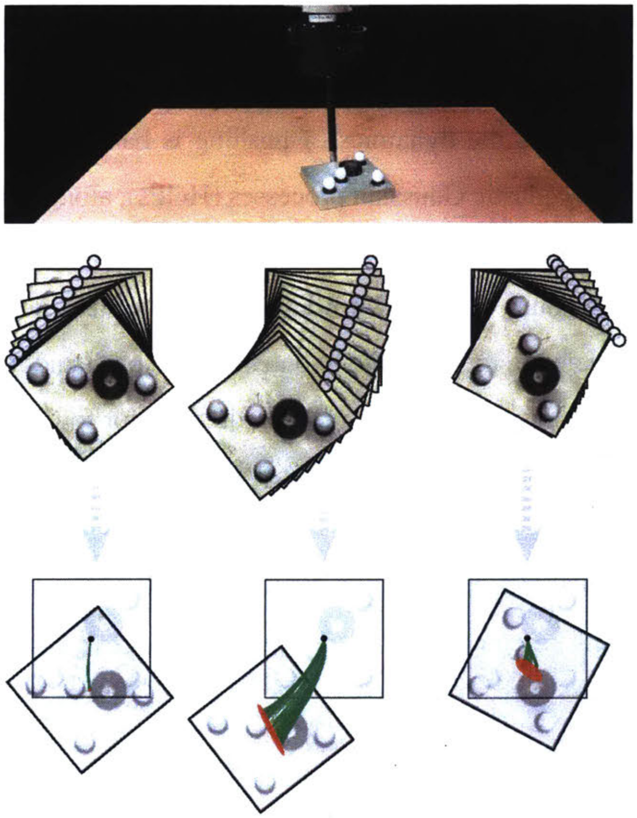

Figure 1-2: This work learns a data-driven model of the most likely outcome and the expected variability involved in pushing an object. The image shows three different pushes whose outcome (repeated 100 times) yields very different distributions: (left) convergent, (center) divergent, and (right) multi-modal. We show in green the trajectories of the center of mass of the block, and in red an ellipse approximating the distribution of final poses.

1.1 Model learning: heteroscedasticity

We are interested in learning the statistical mechanics of the planar pusher-slider system [26, 10, 24, 12] where a frictional point contact pushes on a planar object sliding on a frictional surface. To this aim we rely on the MIT pushing dataset [41] to learn and test the model and contribute in Section 5.2 with an addendum to the dataset with repeated pushes in a grid of pushing locations and directions designed to validate the variances predicted by the model (the new dataset is available online [1]).

The model used to learn the dynamics of pushing is based on a family of Gaussian processes called Heteroscedastic Gaussian processes (HGPs), along with their state-of-the-art variational implementation [22]. This model targets phenomena with input-dependent noise, i.e., when the amount of noise introduced by the system depends on the action. Section 6.1 uses it to estimate the most likely outcome of a push and its variance. To prove the accuracy of the model we evaluate the push predictions for four objects sliding on four materials. The accuracy is measured by the mean square prediction error (NMSE) and the normalized log probability density (NLPD), and compared to normal Gaussian processes and a common analytical model [25]. Finally, in Section 6.3 we validate the predicted probability distributions based on the KL-divergence distance to ground truth estimates of the distribution from repeated pushes.

1.2 Uncertainty propagation: GP-SUM

Given a robotic system where its dynamics are described by Gaussian processes, we propose the algorithm GP-SUM to propagate over time the evolution of the system state in a probabilistic manner. GP-SUM operates by sampling the state distribution, so it can be viewed as a sampling-based filter. It also maintains the basic structure of a Bayes filter by exploiting the GP form of the dynamic and observation models to provide a probabilistic sound interpretation of each sample, so it can also be viewed as a GP-Bayes filter.

We compare GP-SUM's performance to other existing GP-filtering algorithms like GP-UKF, GP-ADF and GP-Particle Filter in a standard benchmark [6, 31]. GP-SUM shows better filtering results both after a single and multiple filtering steps with a variety of metrics, and requires significantly less samples than standard particle filtering techniques.

We demonstrate the use of GP-SUM to predict the expected distribution of outcomes when pushing an object. From modeling the dynamics of pushing we show in Section 6.1 that planar pushing produces heteroscedastic and multimodal behavior, i.e., some actions are more deterministic than others and distributions can break down into components. GP-SUM successfully recovers both when applied to a GP learned model of planar pushing. We compare the results against trajectories simulated with less efficient Monte Carlo-based simulations.

Both actions and sensed information determine the shape of the belief distribution. This thesis provides an efficient algorithm for tracking distributions tailored to the case where the observation and transition models are expressed as GPs.

Chapter 2

Related work

In this chapter we describe in detail both the previous work done to model pushing behaviour, and the existing literature on Bayes filtering algorithms that resemble GP-SUM.

2.1 Data-driven modeling of pushing

There is significant excitement surrounding empirical data-driven techniques for robotic manipulation [33, 2]. Recently Huang et al. [13] surveyed efforts to create datasets of object manipulation. The high-fidelity dataset on planar pushing interaction by Yu et al. [41] is specially relevant to this work. It contains recordings of pushing motions and forces for different dimensions of shape, material, pushing direction, location, velocity, and acceleration. It also provides empirical evidence of the variability of the pushing process, which is the basis of the learned models in this thesis.

Over the years, several works have applied data-driven techniques to the problem of planar pushing [36, 40, 21, 27, 42]. Most recently, Zhou et al. [42] presented a data-driven but physics-inspired model for planar friction. The algorithm approximates the limit surface representation of the relationship between frictional loads and motion twists at a planar contact, and is the state-of-the-art in data-efficient friction modeling in robotics. All these algorithms study the problem of controlling a pushed object in a data-driven fashion, but to our knowledge, no previous work has attempted to model both the expected behavior and the experimental variability.

In this work we use a probabilistic model of the Gaussian processes family to predict the outcome of a push. Gaussian processes are used often to capture both mean and variance of a dynamic system. For example Paolini et al. [32] use Gaussian processes to learn both the transition dynamics and the observation model of prehensile manipulation tasks. We explicitly consider the dependence of the noise in the transition dynamics with the input action by considering the heteroscedastic Gaussian processes introduced by Le and Smola [23]. Kersting et al. [14] proposed a simplified learning algorithm based on maximum likelihood approach, which tends to underestimate noise levels. Lazaro-Gredilla and Titsias [22] solve this problem by introducing a variational heteroscedastic Gaussian process algorithm which we use in this work.

2.2 GP-Bayes filtering: GP-SUM

Gaussian processes (GPs) have proved to be a powerful tool to model the dynamics of complex systems [17, 30, 32], and have been applied to different contexts of robotics including planning and control [29, 7, 28], system identification [17, 11, 3], or filtering [16, 6, 31]. In this work, we study the problem of propagating and filtering the state of a system by providing accurate distributions of the state. When the models for the dynamics and measurements are learned through GP regression, the filtering algorithms are referred as GP-Bayes filters. Among these algorithms, the most frequently considered are GP-EKF [16], GP-UKF [16] and GP-ADF [6], with GP-ADF regarded as the state-of-the-art.

All these GP-filters rely on the assumption that the state distribution is well captured by a single Gaussian and depend on several approximations to maintain that Gaussianity. GP-EKF is based on the extended Kalman filter (EKF) and linearizes the GP models to guarantee that the final distributions are Gaussian. GP-UKF is based on the unscented Kalman filter (UKF) and provides a Gaussian distribution for the state using an appropriate set of sigma points that captures the moments of the state. Finally, GP-ADF computes the first two moments of the state distribution by exploiting the structure of GPs and thus returns a Gaussian distribution for the state.

GP-SUM instead is based on sampling from the state distributions and using Gaussian mixtures to represent these probabilities. This links our algorithm to the classical problem of particle filtering where each element of the mixture can be seen as a sample with an associated weight and a Gaussian. As a result, GP-SUM can be understood as a sampling algorithm that benefits from the parametric structure of Gaussians to simplify its computations and represent the state through weighted Gaussians. Another sampling algorithm for GP-filtering is provided in [16] where they propose the GP-PF algorithm based on the classical particle filter (PF). However, when compared to GP-UKF or GP-EKF, GP-PF is less reliable and more prone to give inconsistent results.

In the broader context of Bayes filtering where the dynamics and observation models are known, multiple algorithms have been proposed to recover non-Gaussian state distributions. For instance, we can find some resemblances between GP-SUM and the algorithms Gaussian Mixture Filter (GMF) [38], Gaussian Sum Filter (GSF) [19], and Gaussian Sum Particle Filtering (GSPM) [19]; all using different techniques to propagate the state distributions as sum of Gaussians. GPM considers a Gaussian mixture model to represent the state distribution, but the covariance of each Gaussian is equal and comes from sampling the previous state distribution and computing the covariance of the resulting samples; GP-SUM instead recovers the covariance of the mixture from the dynamics of the system. GSF is as a set of weighted EKF running in parallel. As a consequence it requires to linearize the system models while GP-SUM does not. Finally GSPM, which has proven to be superior to GSF, is based on the sequential importance sampling filter (SIS) [19]. GSPM samples from the importance function which is defined as the likelihood of a state x given an observation z, $p(x|z)$. GP-SUM instead does not need to learn this extra mapping, $p(x|z)$, to effectively propagate the state distributions.

Other algorithms relevant to GP-SUM are the multi-hypothesis tracking filter (MHT) [5] and the manifold particle filter (MPF) [20]. MHT is designed to solve a data association problem for multiple target tracking by representing the joint distribution of the targets as a Gaussian mixture. Instead, MPF is a sample-based algorithm that has been recently applied to several manipulation tasks with complex dynamics. MPF exploits the contact manifolds of the system by collapsing the distribution defined by the samples into that manifold.

An advantage of GP-SUM is that it can be viewed as both a sampling technique and a parametric filter. Therefore most of the techniques employed for particle filtering are applicable. Similarly, GP-SUM can also be adapted to special types of GPs such as heteroscedastic or sparse GPs. For instance, GP-SUM can be easily extended to the case where sparse spectrum Gaussian processes (SSGPs) are considered by using the work from Pan et al. [31]. This implies that GP-SUM can be made significantly faster by using sparse GPs for the dynamics and observation models of the system.

Chapter 3

Background: Gaussian processes and Bayes filtering

This work focuses on the classical problem of modeling and filtering when the dynamics and observation models of the system are learned through Gaussian process regression. In this section, we introduce the reader to the concepts of Bayes filtering and Gaussian processes.

3.1 Bayes filters

The goal of a Bayes filter is to track the state of the system, x_t, in a probabilistic setting. At time t, we consider that an action u_{t-1} is applied to the system making its state evolve from x_{t-1} to x_t, and an observation of the new state, z_t, is obtained. As a result, a Bayes filter computes the state distribution, $p(x_t)$, conditioned on the history of actions and observations obtained: $p(x_t|u_{1:t-1}, z_{1:t})$. This probability is often referred as the belief of the state at time t.

In general, a Bayes filter is composed of two steps: the prediction update and the measurement or filter update following the terminology from [39].

Prediction update. Given a model of the system dynamics, $p(x_t|x_{t-1}, u_{t-1})$, the prediction update computes the *prediction belief*, $p(x_t|u_{1:t-1}, z_{1:t-1})$, as:

$$\begin{gathered} p(x_t|u_{1:t-1}, z_{1:t-1}) = \\ \int p(x_t|x_{t-1}, u_{t-1}) p(x_{t-1}|u_{1:t-2}, z_{1:t-1}) dx_{t-1} \end{gathered} \tag{3.1}$$

where $p(x_{t-1}|u_{1:t-2}, z_{1:t-1})$ is the belief of the system before applying the action u_{t-1}. Thus the prediction belief can be understood as the *pre-observation* distribution of the state at time t while the belief would be the *post-observation* distribution. In general, the integral in (3.1) can not be solved analytically and different approximations must be used to simplify its computation. Among these simplifications, it is common to linearize the dynamics of the system as it is classically done in the EKF or to directly assume that the prediction belief is Gaussian distributed [39].

Measurement update. Given a new measurement of the state, z_t, the belief at time t can be obtained by filtering the prediction belief. The belief is recovered using the observation model of the system $p(z_t|x_t)$ and applying the Bayes' rule:

$$p(x_t|u_{1:t-1}, z_{1:t}) = \frac{p(z_t|x_t) p(x_t|u_{1:t-1}, z_{1:t-1})}{p(z_t|u_{1:t-1}, z_{1:t-1})} \tag{3.2}$$

This expression usually can not be solved in a closed form and several approximations are required to estimate the new belief. Linearizing the observation model or assuming Gaussianity are again common approaches [39].

Note that the belief at time t can be directly expressed in a recursive manner using the previous belief, and the transition and observation models:

$$\begin{gathered} p(x_t|u_{1:t-1}, z_{1:t}) \propto \\ p(z_t|x_t) \int p(x_t|x_{t-1}, u_{t-1}) p(x_{t-1}|u_{1:t-2}, z_{1:t-1}) dx_{t-1} \end{gathered} \tag{3.3}$$

We will show in Section 4.1 that the same idea of recursion can be applied to the prediction belief, which is a key element for our algorithm GP-SUM.

In general, the dynamics and observation models are considered known and given by parametric descriptions. However, in real systems it is often the case that these models are unknown and it is convenient to learn them using non-parametric approaches such as Gaussian processes. This proves specially beneficial when the actual models are complex and parametric approaches do not provide a fair representation of the system behavior [7, 4].

3.2 Gaussian processes

Gaussian processes (GPs) provide a flexible and non-parametric framework for function approximation and regression [35]. In this work, GPs are considered when modeling the dynamics of the system as well as its observation model. There are several advantages in using GPs over traditional parametric models. First, GPs can learn high fidelity models from noisy data as well as estimate the intrinsic noise in the system. Moreover, GPs can also quantify how certain are their predictions given the available data hence measuring the quality of the regression. For each point in the space, GPs provide the value of the expected output together with its variance. In practice, for each input considered, a GP returns a Gaussian distribution over the output space.

In classical GPs [35], the noise in the output is assumed to be Gaussian and constant over the input:

$$y(x) = f(x) + \varepsilon \tag{3.4}$$

where $f(x)$ is the latent or unobserved function that we want to regress, $y(x)$ is a noisy observation of this function at the input x, and $\varepsilon \sim N(0, \sigma^2)$ represents zero-mean Gaussian noise with variance σ^2.

The assumption of constant Gaussian noise together with a GP prior on the latent function $f(x)$ makes analytically inference possible for GPs (3.5). In practice, to learn a GP model over $f(x)$ you only need a set of training points, $D = \{(x_i, y_i)\}_{i=1}^n$, and a kernel function, $k(x, x')$. Given a new input x_*, a trained GP assigns a Gaussian distribution to the output $y_* = y(x^*)$ that can be expressed as:

$$\begin{aligned} p(y_*|x_*, D, \alpha) &= N(y_*|a_*, c_*^2 + \sigma^2) \\ a_* &= k_*^T(K + \sigma^2 I)^{-1} y \\ c_*^2 &= k_{**} - k_*^T(K + \sigma^2 I)^{-1} k_* \end{aligned} \tag{3.5}$$

where K is a matrix that evaluates the kernel in the training points, $[K]_{ij} = k(x_i, x_j)$, k_* is a vector with $[k_*]_i = k(x_i, x_*)$ and k_{**} is the value of the kernel at x_*, $k_{**} = k(x_*, x_*)$. Finally, y represents the vector of observations from the training set, and α is the set of hyperparameters including σ^2 and the kernel parameters that are optimized during the training process.

A notable property of GPs is that the expected variance of the output y_* comes from the addition of two variances: σ^2 and c_*^2 (3.5). The first one, σ^2, is constant and represents the overall noise of the data. The second one, c_*^2, depends on the input x_* and is only related to the regression error.

In this work we consider the ARD-SE kernel [35] which provides smooth representations of $f(x)$ during GP regression and is the most common kernel employed in the literature of GPs. However, it is possible to extend our algorithm to other kernel functions as it is done in [31].

3.3 Heteroscedastic Gaussian processes

Assuming that the noise of the process σ^2 is constant over the input space is sometimes too restricting. Allowing some regions of the input to be more noisy than others is specially beneficial for those systems with converging and diverging dynamics. Algorithms where GPs incorporate input-dependent noise have proven useful in different context such as mobile robot perception [14], volatility forecasting [22] and robotic manipulation [4]. In Section 4.3, we explore the benefits of combining GP-SUM with input-dependent GPs to characterize the long term dynamics of planar pushing.

The extensions of GPs that incorporate input-dependent noise are often referred as Heteroscedastic Gaussian processes (HGPs). This implies that they can regress both the mean and the variance of the process for any element of the input space. Then, the main conceptual difference between GP and HGP regression is that for HGPs observations are assumed to be drawn from:

$$y(x) = f(x) + \varepsilon(x) \tag{3.6}$$

where $\varepsilon(x) \sim N(0, \sigma^2(x))$ explicitly depends on x compared to (3.4) where ε is a random variable independent of x.

To model the dynamics of planar pushing in this work we consider the state of art algorithm for HGPs called variational HGP (VHGP) proposed by Lazaro-Gredilla and Titsias [22]. Using Bayesian variational theory they derive closed-form solutions for the mean and variance of the process considering input dependent noise. As a result, the probability of an observation y_* is given by:

$$\begin{aligned} p(y_*|x_*, D, \alpha) &= N(y_*|a_*, c_*^2 + e^{b_* + d_*^2/2}) \\ a_* &= k_{f*}^T (K_f + R)^{-1} y \\ c_*^2 &= k_{f**} - k_{f*}^T (K_f + R)^{-1} k_{f*} \\ b_* &= k_{g*}^T (\Lambda - \frac{1}{2} I) 1 + \mu_0 \\ d_*^2 &= k_{g**} - k_{g*}^T (K_g - \Lambda^{-1})^{-1} k_{g*} \end{aligned} \tag{3.7}$$

where k_f and k_g are the ARD-SE kernels of the mean $f(x)$ and the logarithm of the variance $g(x) = \log \sigma^2(x)$. The matrix R is diagonal with $[R]_{ii} = \sigma^2(x_i)$, μ_o is the hyperparameter of the log-variance mean and Λ is a positive semidefinite diagonal matrix optimized together with the other hyperparameters using conjugate gradient descent.

The similarity between the inference equations for GPs and VHGPs is remarkable. The term c_*^2 makes equally reference to the mean error due to the regression process and the equation for a_* is essentially the same. The constant noise σ^2 in GPs however is substituted by the input-dependent expression $e^{b_* + d_*^2/2}$, the expected noise of the process at the input x_*. In terms of computational cost, GPs and VHGPs scale alike with the amount of training data.

Given its nature, GP-SUM can be extended to systems where the models for the transitions and measurements are given by heteroscedastic GPs. This is exemplified in Chapter 7 during the study of planar pushing where, depending on the type of push, the motion of the object is more or less predictable [4].

Chapter 4

GP-SUM Bayes filter

In this chapter we present GP-SUM, discuss its main assumptions, describe its computational complexity, and evaluate and compare its performance.

4.1 The algorithm

Given that GP-SUM is a GP-Bayes filter, our main assumption is that both the dynamics and the measurement models are represented by GPs. This implies that for any state and action on the system the probabilities $p(x_t|x_{t-1}, u_{t-1})$ and $p(z_t|x_t)$ are available and are Gaussian.

To keep track of complex beliefs, GP-SUM does not approximate them by single Gaussians, but considers the weaker assumption that they are well approximated by Gaussians mixtures. Given this assumption, in Section 4.1.1 we exploit that the transition and observation models are GPs to correctly propagate the prediction belief, i.e. the pre-observation state distribution. In Section 4.1.2 we obtain a close-form solution for the belief expressed as a Gaussian mixture.

4.1.1 Updating the prediction belief

The main idea behind GP-SUM is described in Algorithm 1 and derived below. Consider (3.1) and (3.3), then the belief at time t in terms of the prediction belief is:

$$p(x_t|u_{1:t-1}, z_{1:t}) \propto p(z_t|x_t) \cdot p(x_t|u_{1:t-1}, z_{1:t-1}) \tag{4.1}$$

If the prediction belief at time $t-1$ is well approximated by a finite Gaussian mixture, we can write:

$$\begin{gathered} p(x_{t-1}|u_{1:t-2}, z_{1:t-2}) = \\ \sum_{i=1}^{M_{t-1}} \omega_{t-1,i} \cdot \mathcal{N}(x_{t-1}|\mu_{t-1,i}, \Sigma_{t-1,i}) \end{gathered} \tag{4.2}$$

where M_{t-1} is the number of components of the Gaussian mixture and $\omega_{t-1,i}$ is the weight associated with the i-th Gaussian of the mixture $\mathcal{N}(x_{t-1}|\mu_{t-1,i}, \Sigma_{t-1,i})$.

Then we can compute the prediction belief at time t combining (3.1) and (4.1) as:

$$\begin{gathered} p(x_t|u_{1:t-1}, z_{1:t-1}) = \\ \int p(x_t|x_{t-1}, u_{t-1}) p(x_{t-1}|u_{1:t-2}, z_{1:t-1}) dx_{t-1} \propto \\ \int p(x_t|x_{t-1}, u_{t-1}) p(z_{t-1}|x_{t-1}) p(x_{t-1}|u_{1:t-2}, z_{1:t-2}) dx_{t-1} \end{gathered} \tag{4.3}$$

Given the previous observation z_{t-1} and the action u_{t-1}, the prediction belief at time t can be recursively computed using the prediction belief at time $t-1$ together with the transition and observation models. If $p(x_{t-1}|u_{1:t-2}, z_{1:t-1})$ has the form of a Gaussian mixture, then we can take M_t samples, $\{x_{t-1,j}\}_{j=1}^{M_t}$, and approximate (4.3) by:

$$\begin{gathered} p(x_t|u_{1:t-1}, z_{1:t-1}) \propto \\ \sum_{j=1}^{M_t} p(x_t|x_{t-1,j}, u_{t-1,j}) p(z_{t-1}|x_{t-1,j}) \end{gathered} \tag{4.4}$$

Because the dynamics model is a GP, $p(x_t|x_{t-1,j}, u_{t-1})$ is the Gaussian $\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$, and $p(z_{t-1}|x_{t-1,j})$ is a constant value. As a result, we can take:

$$\omega_{t,j} = \frac{p(z_{t-1}|x_{t-1,j})}{\sum_{k=1}^{M_t} p(z_{t-1}|x_{t-1,k})} \tag{4.5}$$

and express the updated prediction belief again as a Gaussian mixture:

$$p(x_t|u_{1:t-1}, z_{1:t-1}) = \sum_{j=1}^{M_t} \omega_{t,j} \cdot \mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) \tag{4.6}$$

In the ideal case where M_t tends to infinity for all t, the Gaussian mixture approximation for the prediction belief converges to the real distribution and thus the propagation over time of the prediction beliefs will remain correct. This property of GP-SUM contrasts with previous GP-Bayes filters where the prediction belief is approximated as a single Gaussian. In those cases, errors from previous approximations inevitably accumulate over time.

Note that the weights in (4.5) are directly related to the likelihood of the observations. As in most sample-based algorithms, if the weights are too small before normalization, it becomes a good strategy to re-sample or modify the number of samples considered. In Section 4.3 we address this issue by re-sampling again from the distributions while keeping the number of samples constant.

Algorithm 1 Prediction belief recursion

```
**SUM-GP**($\{\mu_{t-1,i}, \Sigma_{t-1,i}, \omega_{t-1,i}\}_{i=1}^{M_{t-1}}$, $u_{t-1}$, $z_{t-1}$, $M_t$):
$\{x_{t-1,j}\}_{j=1}^{M_t}$ = sample($\{\mu_{t-1,i}, \Sigma_{t-1,i}, \omega_{t-1,i}\}_{i=1}^{M_{t-1}}$, $M_t$)
**for** $j \in \{1, \ldots, M_t\}$ **do**
  $\mu_{t,j} = GP_\mu(x_{t-1,j}, u_{t-1})$
  $\Sigma_{t,j} = GP_\Sigma(x_{t-1,j}, u_{t-1})$
  $\omega_{t,j} = p(z_{t-1}|x_{t-1,j})$
**end for**
$\{\omega_{t,j}\}_{j=1}^{M_t}$ = normalize_weights($\{\omega_{t,j}\}_{j=1}^{M_t}$)
**return** $\{\mu_{t,j}, \Sigma_{t,j}, \omega_{t,j}\}_{j=1}^{M_t}$
```

4.1.2 Recovering the belief from the prediction belief

After computing the prediction belief, we can use the observation z_t to compute the belief as another Gaussian mixture using (4.1):

$$\begin{aligned} p(x_t|u_{1:t-1}, z_{1:t}) &\propto p(z_t|x_t) \sum_{j=1}^{M_t} \omega_{t,j} \cdot \mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) \\ &= \sum_{j=1}^{M_t} \omega_{t,j} \cdot p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) \end{aligned} \tag{4.7}$$

Note that if $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ could be normalized and expressed as a Gaussian distribution, then the belief at time t would directly be a Gaussian mixture. In most cases, however, $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ is not proportional to a Gaussian. For those case, we find different approximations in the literature (see Algorithm 2). For instance, the algorithm GP-EKF [16] linearizes the observation model to express the previous distribution as a Gaussian.

Algorithm 2 Belief recovery

```
belief_computation($\{\mu_{t,j}, \Sigma_{t,j}, \omega_{t,j}\}_{j=1}^{M_t}$, $z_t$, $M_t$):
for $j \in \{1, \dots, M_t\}$ do
    $\hat{\mu}_{t,j}, \hat{\Sigma}_{t,j}$ = Gaussian_approx( $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ )
end for
$\{\hat{\omega}_{t,j}\}_{j=1}^{M_t} = \{\omega_{t,j}\}_{j=1}^{M_t}$
return $\{\hat{\mu}_{t,j}, \hat{\Sigma}_{t,j}, \hat{\omega}_{t,j}\}_{j=1}^{M_t}$
```

In this work, we exploit the technique proposed by Deisenroth et al. [6] as it preserves the first two moments of $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ and has proven to outperform GP-EKF [6]. This approximation assumes that $p(x_t, z_t|u_{1:t-1}, z_{1:t-1}) = p(z_t|x_t)p(x_t|u_{1:t-1}, z_{1:t-1})$ and $p(z_t|u_{1:t-1}, z_{1:t-1}) = \int p(x_t, z_t|u_{1:t-1}, z_{1:t-1})dx_t$ are both Gaussians. Note that this is an approximation, and that is only true when there is a linear relation between x_t and z_t. Using this assumption and that $p(z_t|x_t)$ is a GP, $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ can be approximated as a Gaussian by analytically computing its first two moments [6]. As a result, we recover the belief as a Gaussian mixture.

It is important to note that this approximation is only necessary to recover the belief, but does not incur in iterative filtering error. GP-SUM directly keeps track of the prediction belief, for which the approximation is not required.

4.2 Computational complexity

The computational complexity of GP-SUM depends on the number of Gaussians considered at each step. For simplicity, we will now assume that at each time step the number of components is constant, M. Note that the number of samples taken from the prediction belief corresponds to the number of components of the next distribution. Propagating the prediction belief one step then requires taking M samples from the previous prediction belief and evaluate M times the dynamics and measurement models. The cost of sampling once a Gaussian mixture can be considered constant, $O(1)$, while evaluating each model implies computing the output of a GP and takes $O(n^2)$ computations where n is the size of data used to train the GPs [35]. Therefore the overall cost of propagating the prediction belief is $O(Mn^2 + M)$ where n is the largest size of the training sets considered. Approximating the belief does not represent an increase in $O-$complexity as it also implies $O(Mn^2)$ operations [6].

Consequently GP-SUM's time complexity increases linearly with the size of the Gaussian mixture while providing a more realistic approximation of the belief and better filtering results (see Section 4.3). To further reduce the computational cost of GP-SUM, sparse GPs can be considered. For instance, combining GP-SUM with SSGPs [31] makes the evaluation of the GP models decrease from $O(n^2)$ to $O(k^2)$ with $k \ll n$.

4.3 Evaluation and comparison

We evaluate the performance of our algorithm in a standard synthetic task for nonlinear state space models [6, 15] where our algorithm proves it can outperform previous GP-Bayes filters[1].

[1] The implementations of GP-ADF and GP-UKF are based on [6] and can be found at `https://github.com/ICL-SML/gp-adf`.

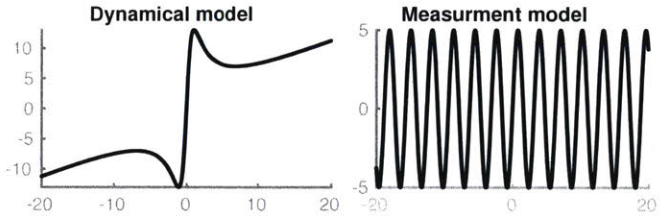

Figure 4-1: Non-linear dynamics and measurement model of the 1D system presented in Section 4.3. The dynamics are specially non-linear around zero which will potentially lead to unstable behavior and multi-modal state distributions.

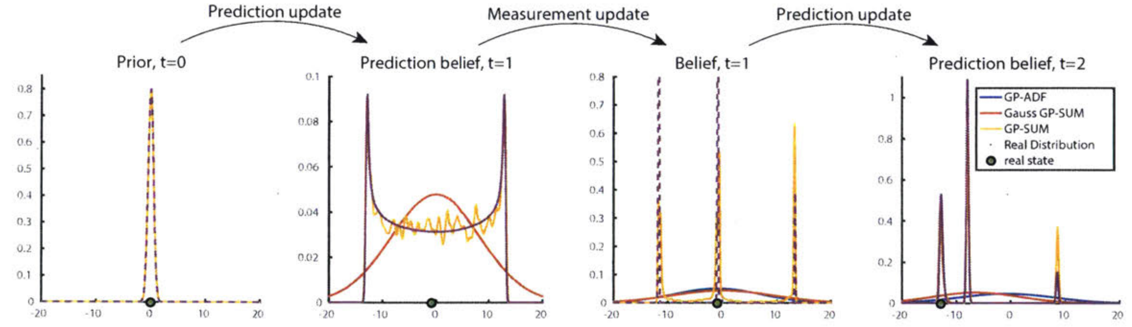

Figure 4-2: Graphical example of how GP-SUM, GP-ADF and Gauss GP-SUM propagate complex state distributions. In this case, the prior has $\mu_0 = 0$ making the dynamics extremely nonlinear around it. As a result, the prediction belief and the belief become multimodal in the first time steps. Compared to GP-ADF, GP-SUM can express properly these complex distributions and its predictions are more accurate. The belief at time $t = 1$ for GP-SUM clearly points out that there are three narrow regions for the location of the true state while GP-ADF only considers a Gaussian wide enough to enclose them all.

To compare GP-SUM to other existing GP-Bayes filters we consider the following 1D dynamical system:

$$x_{t+1} = \frac{1}{2}x_t + \frac{25x_t}{1+x_t^2} + w \qquad w \sim \mathcal{N}(0, 0.2^2) \tag{4.8}$$

and observation model:

$$z_{t+1} = 5\sin 2x_t + v \qquad v \sim \mathcal{N}(0, 0.01^2) \tag{4.9}$$

illustrated in Figure 4-1. The GP models for prediction and measurement are trained using 1000 samples uniformly distributed around the interval $[-20, 20]$. GP-SUM uses the same number of Gaussian components over all time steps, $M = M_t = 1000$. The prior distribution of x_0 is Gaussian with variance $\sigma_0^2 = 0.5^2$ and mean $\mu_0 \in [-10, 10]$. μ_0 is modified 200 times to assess the filters in multiple scenarios and becomes specially interesting around $x = 0$ where the dynamics are highly nonlinear. For each value of μ_0, the filters take 10 time steps. This procedure is repeated 300 times to average the performance of GP-SUM, GP-ADF, GP-UKF, and GP-PF, described in Section 2. For GP-PF, the number of particles is the same as GP-SUM components, $M = 1000$.

The error in the final state of the system is evaluated using 3 metrics. The most relevant is the negative log-likelihood, NLL, which measures the likelihood of the true state according to the predicted belief. We also report the root-mean-square error, RMSE, even though it only uses the mean of the belief instead of its whole distribution. Similarly, the Mahalanobis distance, Maha, only considers the first two moments of the belief so we have to approximate the belief from GP-SUM by a Gaussian. For the GP-PF we only compute the RMSE given that particle filters do not provide close-form distributions. In all the metrics proposed, low values imply better performance.

From Table 4.1 and Table 4.2, it is clear that GP-SUM outperforms the other algorithms in all the metrics proposed and can be considered more stable as it obtains the lowest variance in most of the metrics. In the first time step, GP-PF is already outperformed by GP-SUM and GP-ADF, and after a few more steps in time, particle starvation becomes a major issue for GP-PF as the likelihood of the observations becomes extremely low. For

Table 4.1: Comparison between GP-filters after 1 time step.

	GP-ADF	GP-UKF	GP-SUM	GP-PF
Error	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
NLL	**0.49** ± 0.17	**95.03** ± 97.02	**-0.55** ± 0.34	-
Maha	**0.69** ± 0.06	**2.80** ± 0.72	**0.67** ± 0.04	-
RMSE	**2.18** ± 0.39	**34.48** ± 23.14	**2.18** ± 0.38	**2.27** ± 0.35

Table 4.2: Comparison between GP-filters after 10 time steps

	GP-ADF	GP-UKF	GP-SUM	GP-PF
Error	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
NLL	**9.58** ± 15.68	**1517.17** ± 7600.82	**-0.24** ± 0.11	-
Maha	**0.99** ± 0.31	**8.25** ± 3.82	**0.77** ± 0.06	-
RMSE	**2.27** ± 0.16	**12.96** ± 16.68	**0.19** ± 0.02	**N/A**

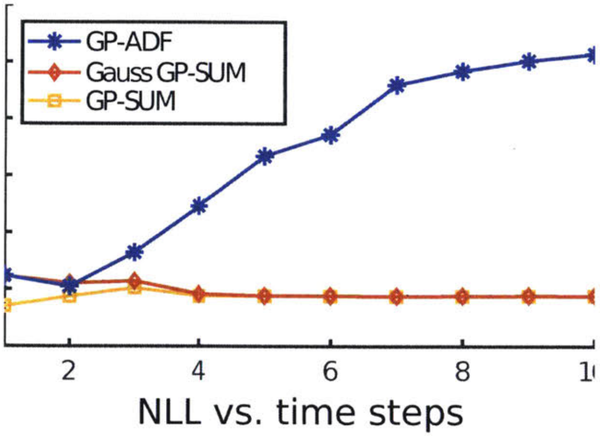

Figure 4-3: Evolution over time of the negative log-likelihood, NLL. In the first time steps the shape of the real belief is more likely to be non-Gaussian and this explains why GP-SUM performs better than both GP-ADF and the Gaussian approximation of GP-SUM, Gauss GP-SUM. As time evolves, GP-SUM and Gauss GP-SUM converge meaning that the shape of the real belief tends to become Gaussian. On the other hand, GP-ADF's performance worsens over time because for the cases where the dynamics are highly nonlinear its predicted variance tends to increase making the likelihood of the true state lower with time.

this reason, we did not report a RMSE value for the GP-PF after 10 time steps. GP-UKF performance is clearly surpassed by GP-SUM and GP-ADF after 1 and 10 time steps.

In Figure 4-2 we compare the true state distributions (computed numerically) to the distributions obtained by GP-ADF, GP-SUM, and a simplified version of GP-SUM, Gauss GP-SUM, that takes a Gaussian approximation of GP-SUM at each time step. It becomes clear that by allowing non-Gaussian beliefs GP-SUM can assign higher likelihood to the actual state while better approximating the true belief. Instead, GP-ADF can only assign a single Gaussian wide enough to cover all the high density regions.

In Figure 4-3 we study the temporal evolution of the Negative Log-Likelihood (NLL) metric for GP-ADF, GP-SUM and Gauss GP-SUM. As the number of steps increases, GP-SUM and Gauss GP-SUM tend to coincide because GP-SUM tends to become more confident on the location of the true state and its belief becomes more Gaussian. Figure 4-3 also shows that GP-ADF worsens its performance over time. This is due to those cases where the dynamics are highly non-linear, i.e. around zero, and the variance of GP-ADF increases over time. As a result, at the cost of larger computational expense, the expressiveness of its distributions and the lack of heavy assumptions makes GP-SUM a good fit for those systems where multimodality and complex behaviors can not be neglected.

Chapter 5

Pusher-Slider Data

Given that this thesis aims to model the probabilistic behavior of the pusher-slider system, illustrated in Figure 5-1, we include this chapter to describe the data used to learn and validate the models. We are interested in learning the behavior of the slider as an input-output relationship. As illustrated in Figure 5-2 the representation for the input space is:

v_p : Magnitude of the velocity of the pusher.

c : Contact point on the perimeter of the slider.

β : Pushing angle.

and the representation for the output space is:

Δx : COM x displacement in the pusher ref. frame.

Δy : COM y displacement in the pusher ref. frame.

$\Delta\theta$: Orientation change.

after a push for Δt seconds. These parameters are sufficient to characterize simple models of planar point pushing [25, 12].

In this work we use two sets of data: a large-scale general purpose dataset of planar pushing for learning, and a dataset with repeated pushes we collected for the purpose of validating the results. In particular we validate the model prediction of the variance for the outcome of a push.

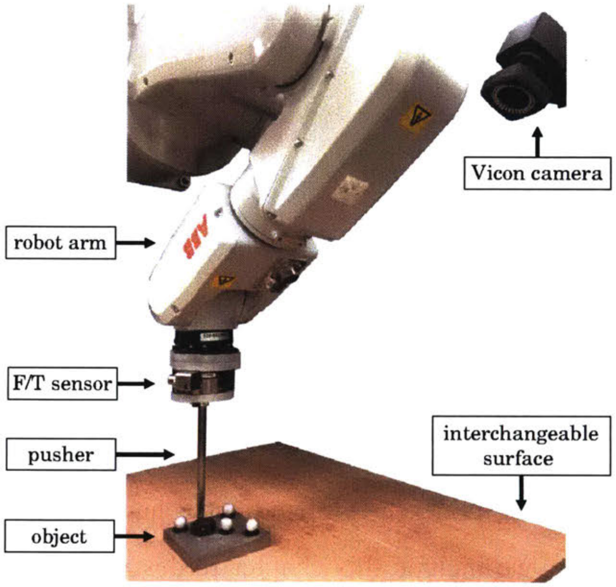

Figure 5-1: Experimental setup. The interaction between the pusher (vertical rod) and the object (square block) is recorded through a Vicon tracking system and a force-torque sensor. The system has been designed to provide a clean interface between the pusher and slider, and is described in detail in Yu et al. [41].

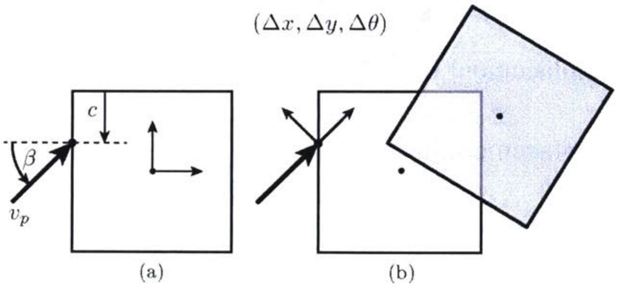

Figure 5-2: Input/output space for the dynamics of the pusher-slider system. (a) The input is parametrized with the distance c that captures the contact point along an edge of the square (for objects without symmetries, contact points are evenly spaced along their perimeter), the angle of push β, and its velocity v_p. (b) The output is parametrized with $(\Delta x, \Delta y, \Delta \theta)$, the displacement of the object in a reference frame aligned with the push direction.

5.1 Learning Data

In previous work [41] we captured a large set of planar pushes with the robotic setup in Figure 5-1, composed of a high-precision industrial robot fitted with a cylindrical pusher and stainless steel objects of different shapes sliding on surfaces of different materials. The system records the trajectories of the pusher and object and the force at the interaction. We will show in later sections that 100 samples are sufficient in average to outperform analytical models, and model accuracy saturates with less than 1000 samples.

5.2 Validation Data

Our goal is to predict reliably the object's expected motion and its variance:

$$\begin{aligned}
\Delta x &\sim \mu_x(u), \sigma_x^2(u) \\
\Delta y &\sim \mu_y(u), \sigma_y^2(u) \\
\Delta \theta &\sim \mu_\theta(u), \sigma_\theta^2(u)
\end{aligned} \tag{5.1}$$

where $u = (v_p, c, \beta)$ is the input/action and $\mu(u)$ and $\sigma^2(u)$ are the *input-dependent* expected outcome and variance. The dependence of the variance $\sigma^2(u)$ with the input u is the key complexity we address in this work, motivated by the example in Figure 1-2, and leads us to consider HGPs instead of standard GPs.

To validate the observation that the output noise depends on the input, we collected a new dataset in the same setup in Figure 5-1 containing 100 repetitions of each push considered, which gives us an approximate distribution of the object motion. In this new dataset, the pusher follows a straight trajectory of 1cm long at 20mm/s. The initial contact angles go from -1.5 to $+1.5$ radians spaced by 0.1 radians while we consider 11 different initial contact points evenly spaced on the side of the object. This produces a sufficiently dense grid of pushes allowing us to extract for each push the expected mean and variance of the object motion. This dataset is available online [1].

Besides validating the proposed model, the new dataset also provides empirical evidence to study the variability of the pushing process, including multi-modality effects.

Chapter 6

The Dynamics of Pushing

6.1 The learned model

In this section we describe in more detail the process to learn the pushing model for the four objects in Figure 6-1 sliding on a horizontal surface of four different materials.

To compute the pushing model of a certain pair object-surface, we train three independent VHGPs, one for each output Δx, Δy, and $\Delta\theta$. While not optimal from a data-efficiency perspective, since we neglect the existing correlations between the outputs variables, it proves sufficient in terms of performance as long as enough training data is used. For future work it would be interesting to combine multi-task GP prediction with heteroscedastic GPs.

Once each VHGP is trained, we visualize the effect of different pushes given a fixed velocity. Each push is defined by the contact point c and the pushing angle β. Figure 6-2 shows the regressed distribution for the case where the pusher's velocity is v_p = 20mm/s, the object shape is a square, and the surface is plywood. In accordance with intuition, we

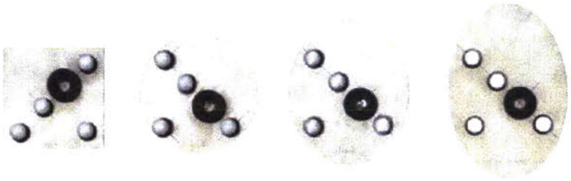

Figure 6-1: Objects used for learning, from Yu et al. [41].

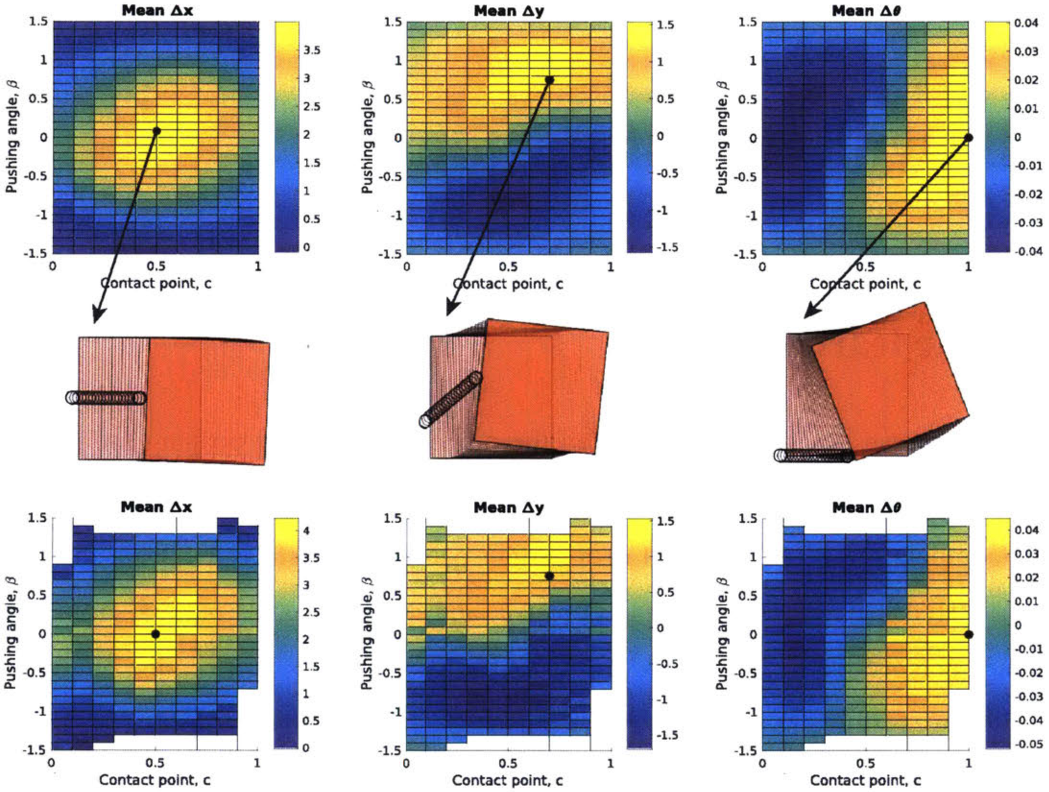

Figure 6-2: **Expected displacement after a push (units: mm and rad).** The first row shows the predicted outputs by the learned VHGP model for a fixed velocity $v_p = 20$mm/s after a push of $\Delta t = 0.2$ sec, for different contact point locations and push angles. The result is remarkably similar to the expected outputs from the validation data in the 3rd row. The symmetries in the plots are due to the symmetries of the square. The second row shows pushes that are predicted to yield high displacement in Δx ($c = 0.5$, $\beta = 0$), Δy ($c = 0.7$, $\beta = 0.7$), and $\Delta\theta$ ($c = 1.0$, $\beta = 0$) correspondingly.

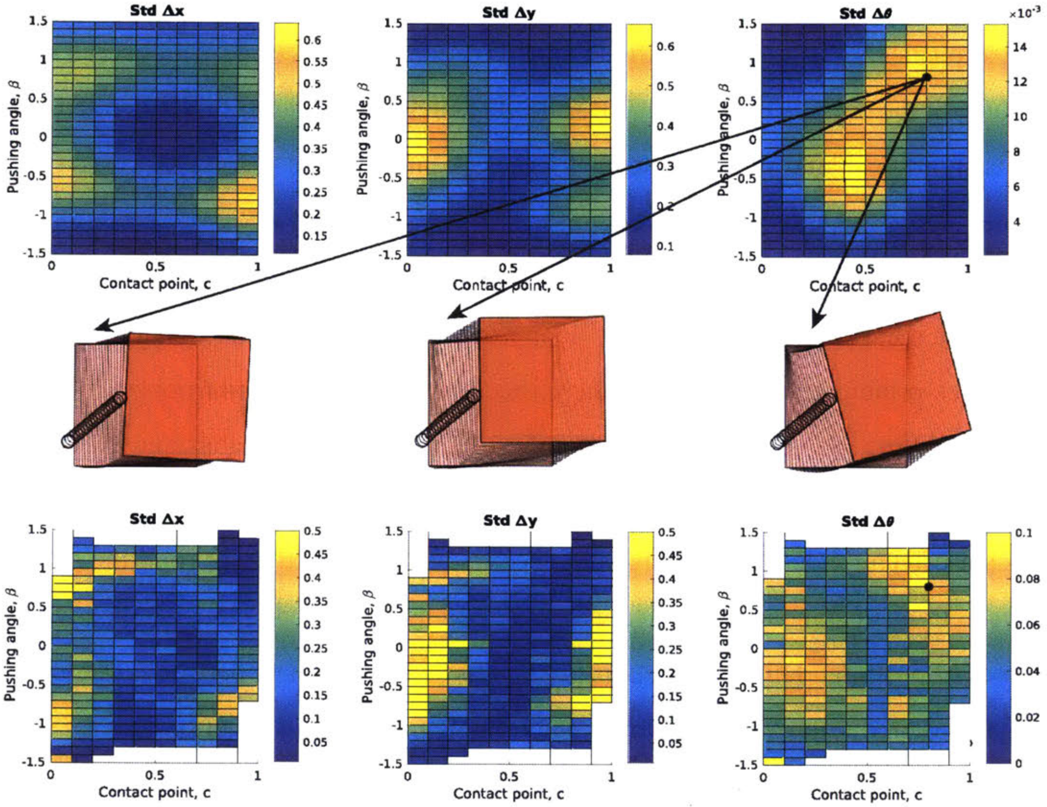

Figure 6-3: **Expected variability after a push (units: mm and rad).** The first row shows the predicted standard deviation (std) of each output obtained from the VHGP model, as compared to the observed from the validation dataset in the third row. Some regions are of high variance and from our experience these seem to be independent of the type of surface considered, suggesting that some pushes are more stable than others regardless of the surface. The middle row shows three examples from the learning data where the push considered has the maximum variance in the change of orientation. The outcome does seem to vary considerably.

see that the displacement in the pusher's direction Δx is maximum when pushes are done at the center of the edge, $c = 0.5$, and perpendicular to the edge, $\beta = 0$. The maximum change in orientation happens when pushing in between the edge center and the vertex with a pushing angle of $\beta = +30^o$ if $c = 0.75$ and $\beta = -30^o$ if $c = 0.25$.

Analogously, Figure 6-3 shows the modeled and experimental standard deviation of the predicted pushes, as a function of the contact location and direction. We observe that the magnitude of the noise is in between 10% and 40% of the magnitude of the expected output, which is significant, and further motivates the work in this thesis.

There are well defined regions that present more noise than others. To a certain degree the prediction matches well with the validation data, also shown in Figure 6-3. The precision of the measurement equipment (Vicon tracking system), suggests that the regressed noise comes principally from the pushing process and not from sensor noise. It is further indicative that the shape of the regressed noise remains more o less constant when considering the same object but a different surface.

A data-driven model that captures the uncertainty of interaction, beyond the deterministic predictions of standard analytical models gives us a more complete perspective of the dynamics of pushing. This information can be used to differentiate between more and less stable pushes, and improve multi-step prediction, by propagating uncertainty.

6.2 Evaluation of the model

We evaluate the performance of the learned model with the standard metrics normalized mean square error (NMSE) and normalized log probability density (NLPD):

$$\text{NMSE} = \frac{\sum_{j=1}^{m}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{m}(y_j - \bar{y})^2} \tag{6.1}$$

$$\text{NLPD} = -\frac{1}{m}\sum_{j=1}^{m}\log p(y_j|D) \tag{6.2}$$

where m is the number of elements in the test set and $\{y_j\}_{j=1}^{m}$ the observations. We note with $\hat{y}_j$ the predicted value from the VHGP model at x_j and with $\bar{y}$ the mean of the observations in the training set. We consider the total NMSE of the model as the sum of the individual NMSEs for each output. For the total NLPD, we consider $p(y_j|D)$ as the product of the individual probabilities of each output given their respective VHGP.

Note that NMSE, as defined in (6.1), does not take into account the variability of motion and only considers its expected value . It reflects the squared distance between the model outputs and the real process normalized by the variance of the observations. The NMSE is especially useful when comparing the results with deterministic models of pushing. The NLPD metric instead, computes how likely are the observations according to a given probabilistic model. It is more appropriated for evaluating our data-driven models as they also regress the uncertainty of the motion in (6.2). For the NLPD, lower results imply a higher likelihood of the data. Consequently NLPD penalizes both underconfident and overconfident models, so deterministic models can not be fairly evaluated using NLPD.

6.2.1 Evaluation on different objects and materials

Figure 6-4 and Figure 6-5 show the evaluation of the model when trained on four different surfaces (plywood, Delrin, polyurethane, and ABS) and for four different object shapes (square, circle and two ellipses, as shown in Figure 6-1). An advantage of VHGPs models is that they do not require a large amount of data. The plots show that less than 1000 samples are sufficient to saturate performance, which is equivalent to collecting about 5 minutes of pushing experiments in our setup if we use $\Delta t = 0.2$s. Therefore our VHGP models can provide good planar push models of new objects and surfaces without having to go through an extensive data collection. The fact that VHGPs performance remains more or less constant after a certain amount of data also suggests that the pushing motion is a sufficiently well defined process and simple enough to be learned in general from a reduced exploration of the environment.

6.2.2 Comparison with other models

In this n we compare the performance of VHGP against a standard GP model, and a commonly used analytical model [25]. GP and VHGP show a similar NMSE. This is expected

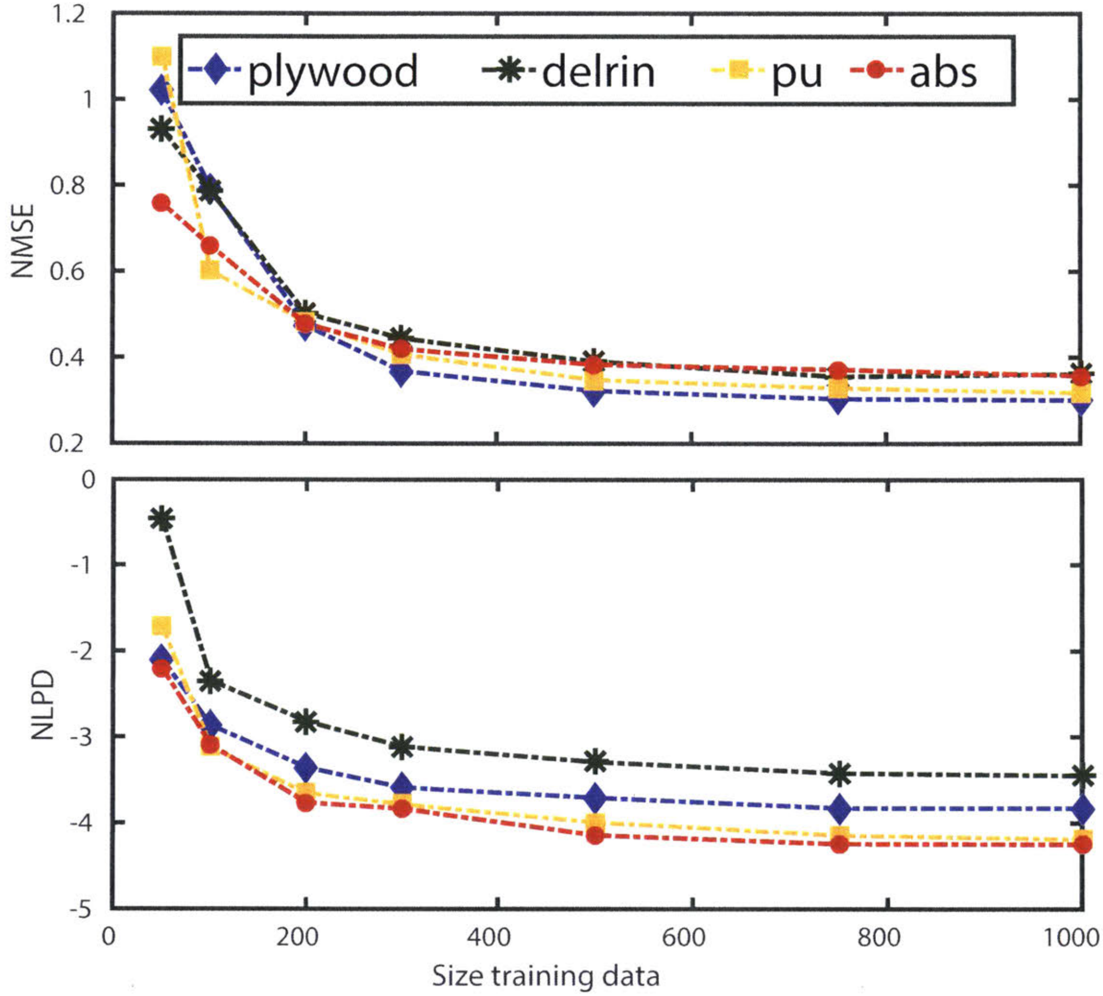

Figure 6-4: VHGP model evaluated on the four surfaces used in [41] depending on the size of the training set. We observe that the model is not very sensitive to the type of surface considered as the errors in NMSE and NLPD are reasonably similar. Higher NLPD in $delrin$ and $plywood$ could be due to higher degradation with time.

Table 6.1: NMSE and NLPD comparisons between models

Outputs	Analytical	GP	VHGP
NMSE	0.72	0.38	0.37
NLPD	–	-2.82	-3.73

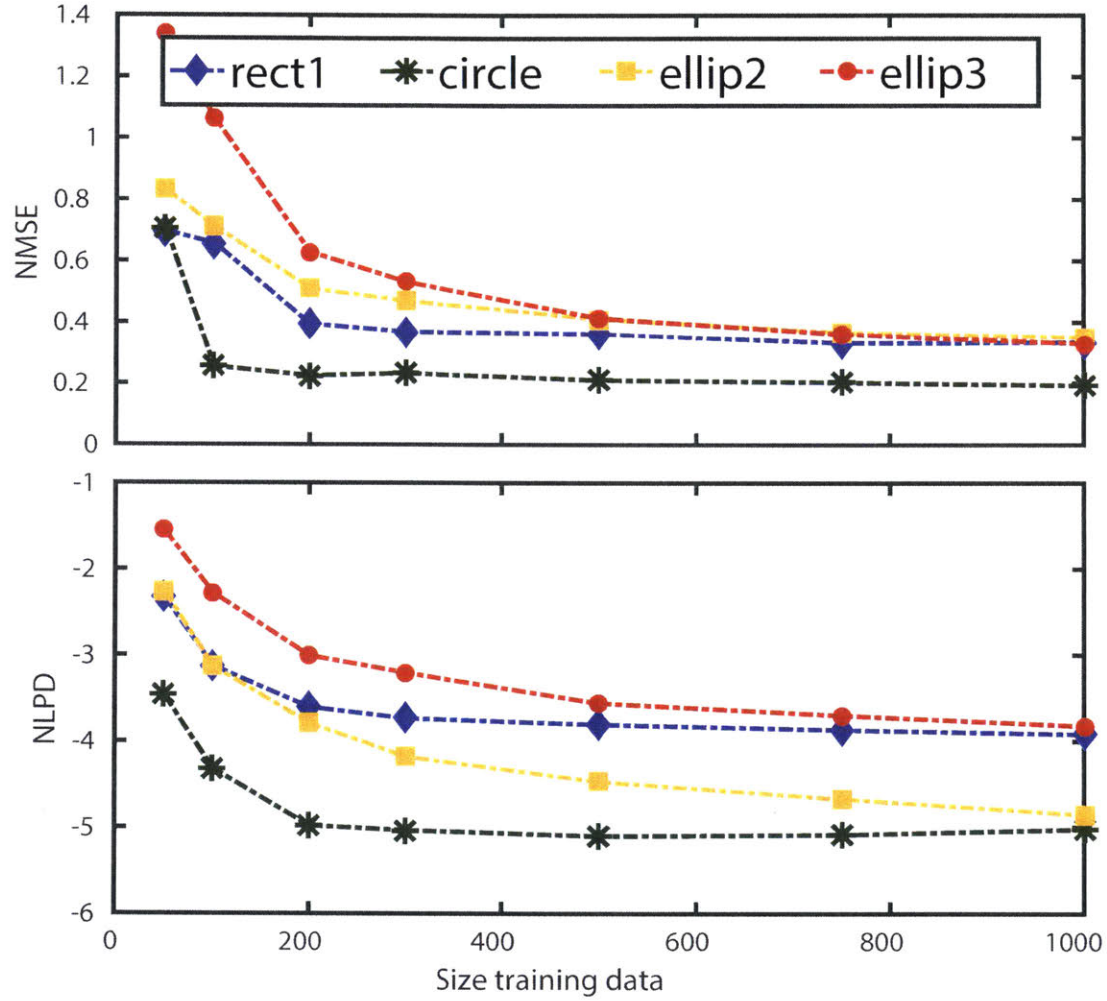

Figure 6-5: Evaluation of the VHGP model for the objects in Figure 6-1. We observe that the circle is easier to learn than the others shapes. This is probably due to the fact the all contacts points in a circle are indeed equivalent. Thus the contact point dimension can be considered redundant, simplifying the training of the model.

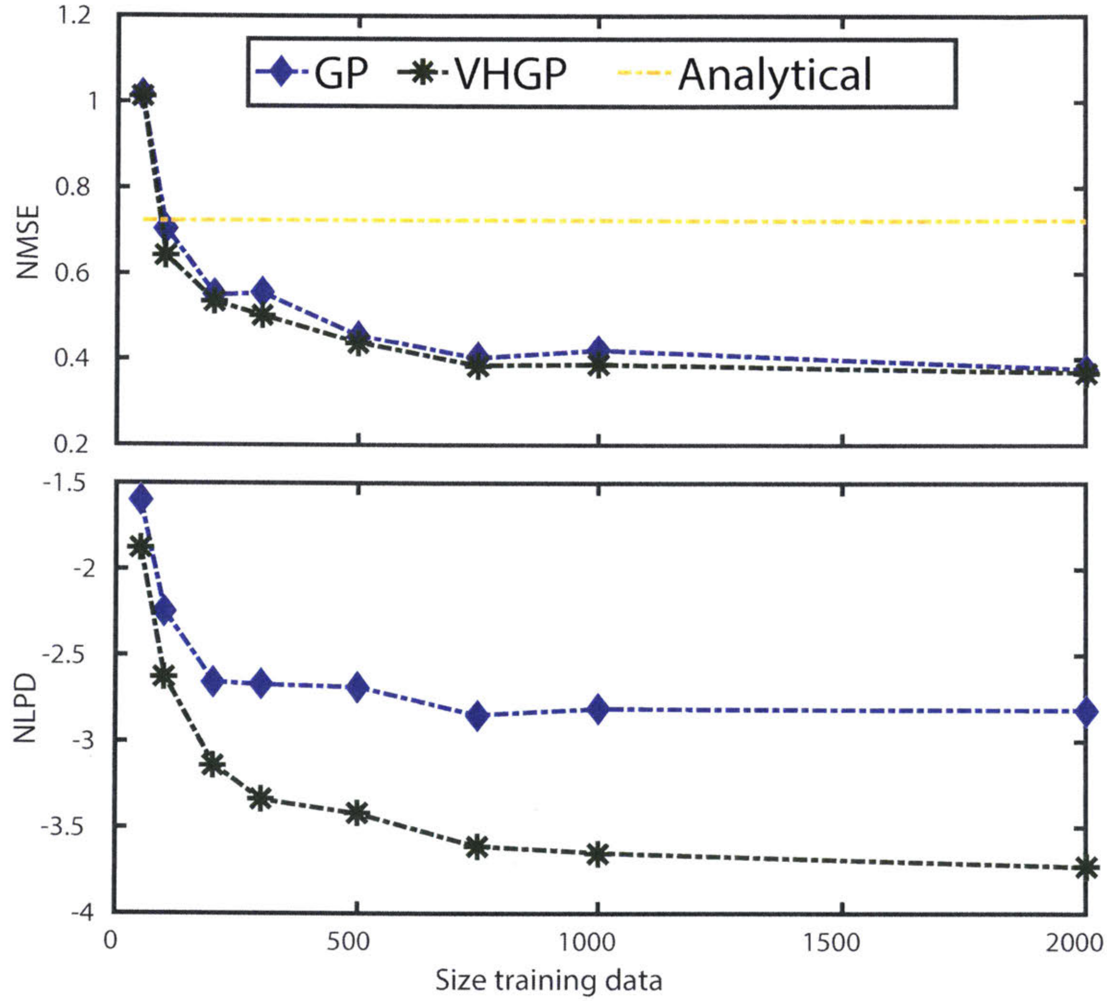

Figure 6-6: NMSE and NLPD comparison for different models as a function of the training data. Both GPs and VHGPs outperform the analytical model after 100 samples. As expected, VHGP outperforms GP measured with NLPD, since it has more flexibility to capture the stochasticity of the data.

since NMSE only evaluates the most likely outcome, and both GPs and VHGPs have a very similar formulation for the regressed mean. The difference in NLPD however, justifies the need of input-dependent noise to explain the stochastic behaviour of pushing. As all inputs in a classical GP are supposed to have the same noise, all the observations become equally weighted during training. Instead, in VHGPs those observations from lower noise regions have higher relevance in the regression process.

We also compare our data-driven model with the analytical model proposed by Lynch et al. [25], that relies on assumptions of quasi-static, uniform pressure distribution, uniform coulomb friction, and an ellipsoidal approximation to the limit surface.

The proposed data-driven model, instead of relying on a perfect knowledge of the object-surface interaction, outputs a distribution of the motion's outcome that encloses unexpected behaviours. Figure 6-6 shows that both GPs and VHGPs outperform the analytical model after 100 samples approximately.

We also observed that for high velocities the analytical model is more unreliable. This is reasonable as the analytical model assumes a quasi-static interaction and does not take into account inertia. In our model, those dynamic effects are captured by adding the pusher velocity as an input and through the uncertainty of the distribution.

6.3 Validation

The training and testing set used from [41] does not contain sufficient repeated pushes to estimate ground truth variance for a given push. To validate the distributions predicted by our model, we captured a benchmark dataset that incorporates repeated trajectories so that a reliable notion of uncertainty can be extracted from them (the dataset is available in [1]). The differences between repeated pushes should mainly include the uncertainty in the object-surface interaction, i.e., the stochastic side of the push itself. Given the samples of each repeated push, we compute the expected motion of the object and its variance from the benchmark. Section 5.2 contains a more detailed description of the dataset, and the bottom rows of Figure 6-2 and Figure 6-3 illustrate the obtained means and variances.

To evaluate the results of the model learned from the training dataset in [41], against the

Table 6.2: KL divergence between the validation data and the models

KL divergence	**GP**	**VHGP**
Average KL	22.63	15.32
Median KL	5.74	5.34

benchmark dataset, we use the average KL divergence over the pushes. If we assume that the real distribution of each push comes from three independent Gaussians, we can directly use the KL divergence between two Gaussian distributions:

$$KL(p,q) = \frac{1}{2}\left(log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1\right) \tag{6.3}$$

where $p = N(\mu_1, \sigma_1^2)$ and $q = N(\mu_2, \sigma_2^2)$.

The average and median KL divergences of each model show that our model not only regresses properly the expected motion, but also can predict reasonably well its distribution (Table 6.2). This is crucial to create robust probabilistic models that can be latter incorporated into multi-step planning and control.

Chapter 7

Uncertainty propagation in pushing

Planar pushing is an underdetermined and sometimes undecidable physical interaction [26], except when making sufficient assumptions and simplifications [10, 24]. It has also been shown experimentally that uncertainty in frictional interaction yields stochastic pushing behavior [4, 41]. Moreover, the type of push has a strong influence in the amount of expected uncertainty, i.e., the level of "noise" in the pushing dynamics is action dependent, a.k.a., heteroscedastic. This can be observed in Figure 1-2 where three different pushes repeated multiple times lead to very different distributions for the object position including multimodality.

A pushing controller could benefit from a model of the heteroscedasticity in pushing dynamics by preferring those pushes that lead to lower uncertainty. To this end, it is necessary an heteroscedastic pushing model for the dynamics together with an algorithm that reliably and efficiently propagates uncertainty over time. In this work, we use HGPs to model the dynamics of pushing as done by Bauza and Rodriguez [4], and GP-SUM to propagate the uncertainty of the system.

As illustrated in Figure 5-2, we model planar pushing as a HGP that takes as inputs the contact point between the pusher and the object, the pusher's velocity, and its direction. The output of the dynamics is the displacement of the object relative to the pusher's motion. We use the learned HGP model described in Section 6.1 with the real data from the MIT pushing dataset [41].

Because we are only concerned about simulating the propagation of uncertainty over

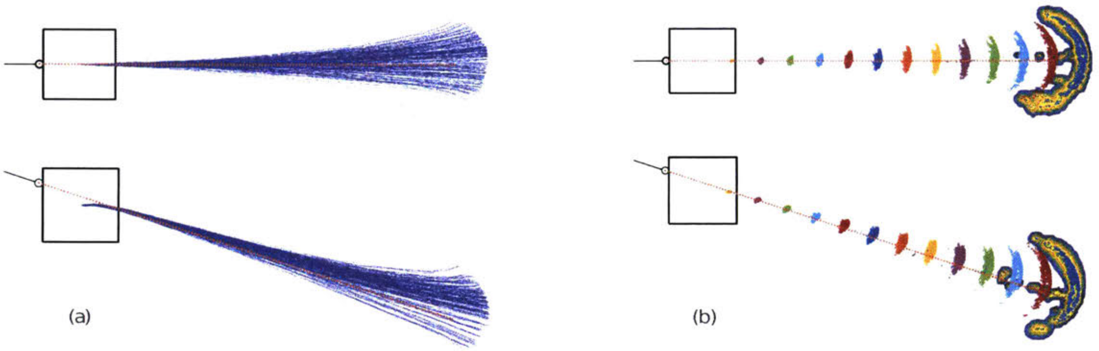

Figure 7-1: Uncertainty propagation during pushing. Figure (a) represents two different pushes and several trajectories obtained from those pushes by propagating the HGP dynamics in time. Figure (b) shows the evolution of the state distribution provided by GP-SUM over time. For simplicity, until the last time step we only show the mean of each Gaussian component where different colors represent different time steps. Both pushes present ring-shaped distributions and multimodality: the one on top presents more symmetric distributions while the one on the bottom is asymmetric because the object is more likely to remain to the left of the pusher's direction. The modes disconnected from the main distribution are due to the lose of contact with the pusher. This can be due to the object orientation which has been omitted in the plots to enable a clear visualization of the results.

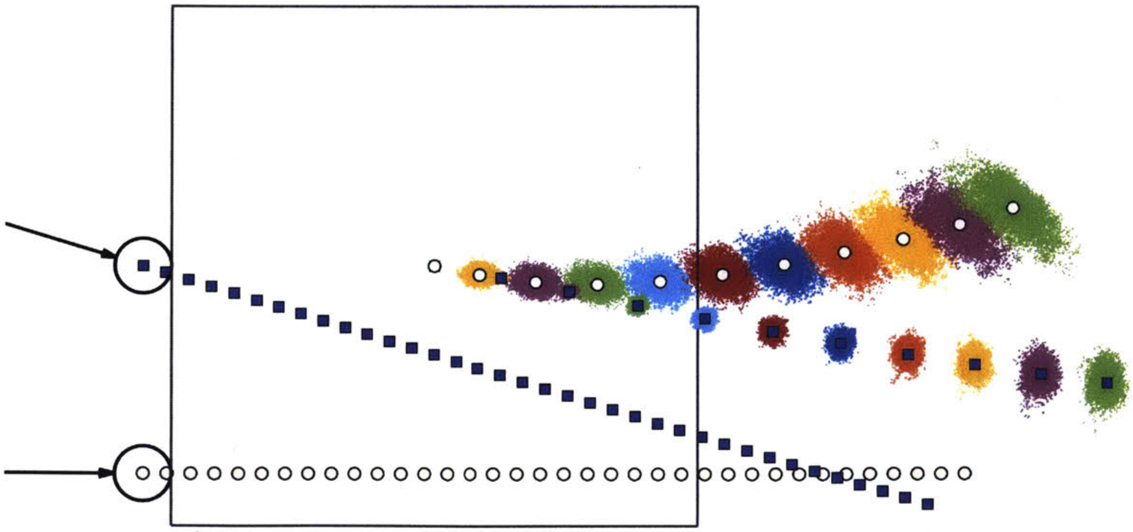

Figure 7-2: Comparison of two pushes with different levels of noise. One push is clearly noisier than the other as it generates a much wider distribution of the object position from the beginning. This is in part possible because the dynamics, given by an HGP, consider input-dependent noise.

time, we do not consider a measurement model. As a result, when using GP-SUM the prediction belief and the belief coincide and all the components in the Gaussian mixtures have the same weights.

Given that the distribution of the object position tends to become non-Gaussian overtime, GP-SUM can obtain more accurate results than other algorithms. Figure 7-1 shows the resemblance between sampled trajectories from propagating the dynamics in time in a Monte-Carlo fashion and the distributions obtained by GP-SUM. It also becomes clear that the shape of the distributions recovered by GP-SUM are ring-shaped and even multimodal which can not be captured by standard GP-Bayes filters that predict single Gaussian distributions.

Being able to propagate the uncertainty of the object position over time exposes interesting properties of the planar push system. For instance in Figure 7-2 we observe two different pushes and how the belief for the object position evolves. It becomes clear that one of the pushes leads to more noisy distributions. Being able to recover these behaviors is specially useful when deciding what are the best pushes to exert. If our goal is to push an object to a specific region of the space, then it would be advantageous to consider those pushes that lead to narrower (low-variance) distributions and avoid those that involve multimodal outcomes because they are often harder to control.

Chapter 8

Conclusion

In this chapter we discuss the main advantages and limitations of GP-SUM, review the implications of our probabilistic modeling of pushing and finish by describing future work directions.

8.1 Discussion on GP-SUM

GP-Bayes filters are a powerful tool to model and track systems with complex and noisy dynamics. Most approaches rely on the assumption that the belief is Gaussian or can be iteratively approximated by a Gaussian. The Gaussian assumption is an effective simplification. It enables filtering with high frequency updates or in high dimensional systems. It is most reasonable in systems where the local dynamics are simple, i.e., linearizable, and when accurate observation models are readily available to continuously correct for complex or un-modelled dynamics.

In this work we look at situations where the Gaussian belief is less reasonable. That is the case of contact behavior with non-smooth local dynamics due to sudden changes in stick/slip or contact/separation, and is the case in stochastic simulation where, without the benefit of sensor feedback, uncertainty distributions naturally grow over time. We propose the GP-SUM algorithm which considers the use of Gaussian mixtures to represent complex state distributions.

Our approach is sample-based in nature, but has the advantage of using a minimal number of assumptions compared to other GP-filters based on single Gaussian distributions or the linearization of the GP-models. Since GP-SUM preserves the probabilistic nature of a Bayes filter, it also makes a more effective use of sampling than particle filters.

When considering GP-SUM, several aspects must be taken into account:

Number of samples. Choosing the appropriate number of samples determines the number of Gaussians in the prediction belief and hence its expressiveness. Adjusting the number of Gaussian over time might be beneficial in order to properly cover the state space. Similarly, high-dimensional states might require higher values of M_t to ensure a proper sampling of the prediction belief. Because of the sample-based nature of GP-SUM, many techniques from sample-based algorithms can be effectively applied such as resampling or adding randomly generated components to avoid particle deprivation.

Likelihood of the observations. There is a direct relation between the weights of the beliefs and the likelihood of the observations. We can exploit this relationship to detect when the weight of the samples degenerates and correct it by re-sampling or modifying the number of samples.

Computational cost. Unlike non-sampling GP-filters, the cost of GP-SUM scales linearly with the number of samples. Nevertheless, for non-linear systems we showed that our algorithm can recover the true state distributions more accurately and thus obtain better results when compared to faster algorithms such as GP-ADF, GP-UKF or GP-PF.

GP extensions. The structure of GP-SUM is not restricted to classical GPs for the dynamics and observation models. Other types of GPs such as HGPs or sparse GPs can be considered. For instance, combining GP-SUM with SSGPs [31] would make the computations more efficient.

8.2 Discussion on push modeling

In recent work [41, 18] we provided empirical evidence that planar frictional interaction shows non-trivial statistical behavior, and suggested that a probabilistic model might yield a practical and less over-confident approach to model frictional contact. This work is a step in that direction. We focus on the problem of planar pushing, which has proven essential for many types of interaction, both simple and complex, and for which the robotics community has developed a good and long-standing analytical understanding.

Input-dependent noise. This work starts from the empirical observation that the magnitude of the observed variability under a constant push can vary up to an order of magnitude with the pushing action, i.e., pushing location, pushing direction or pushing velocity. A model that accounts for that variability could be used, for example, to avoid actions that yield unpredictable behavior.

Data-driven modeling. Building from the recent pushing dataset by Yu et al. [41], we propose to use a data-driven modeling approach based on a Variational Heteroscedastic Gaussian process model (VHGP) to capture the mean and variance of a planar frictional push. Unlike traditional Gaussian processes, which assume a level of output noise independent of their input, heteroscedastic Gaussian processes model the dependence of both the mean and variance of the outcome of a planar push as functions of the input pushing action.

The learned models are specific to the particular object and material. Generalization over materials and shapes is an interesting question that we plan to explore in future work, which will require a model less computationally expensive to train, such as those based on sparse Gaussian processes [37] or Random Features [34].

Evaluation. We show that an order of 100 samples is sufficient to overcome the performance of a standard analytical model, and evaluate the improvement of the VHGP framework over a traditional homoscedastic GP. The accuracy of the VHGP model tails off at about 10^3 training samples, data that can be captured in about 5 minutes in our training setup.

We validate the model against a new dataset collected in the same setup as [41] for the purpose of benchmarking. This new dataset is composed of 100 identical pushes for each of a set of more than 300 different pushing actions, which gives an empirical footprint of the stochasticity of planar pushing. The performance, evaluated by means of the KL divergence shows a clear improvement over a normal GP.

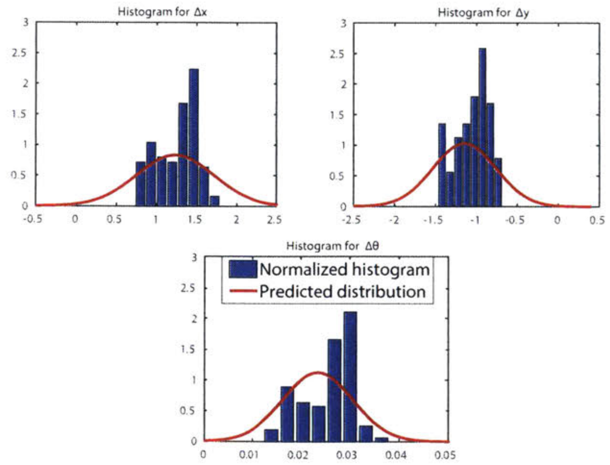

Figure 8-1: These histograms represent the different outputs obtained when repeating the same push. They have been normalized so that they can be compared with the distribution provided by our VHGPs. In this case, distributions do not adjust clearly to a Gaussian distribution but our model still obtains a good approximation for them.

Gaussianity vs. multi-modality. While VHGPs are the state-of-the-art for data efficient regression for input dependent processes, Gaussianity and unimodality of the underlying dynamics is still a key assumption. We know that this is not always the case (Figure 8-1). However, a model with input-dependent noise can express multi-modal behavior when integrated over time. We are interested in exploring further the idea of capturing finite multi-modal behavior with uni-modal Gaussian instantaneous models.

Dependence with velocity. A very common assumption in robotic manipulation is to neglect the inertial effects of interaction, i.e. the quasi-static assumption. In this work we avoid that assumption by making the velocity of the pusher an explicit input to the model.

To evaluate the importance of considering velocity as an input, we group data from different velocities by time-scaling the interaction (under the quasi-static assumption, an action executed twice as fast will have the same outcome, except in half the time) and train models without the velocity as input. If the system is indeed quasi-static, combining the data from different velocities should increase the amount of training data and yield better (or not worse) performance. Otherwise, we should see the performance worsen. Figure 8-2 shows the evolution of the performance of the learned model as we add data from larger velocities, all time-scaled to 10mm/s. We can see that the performance peaks at around

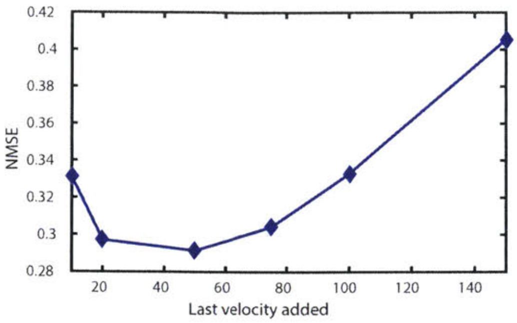

Figure 8-2: The x-axis of the plot shows the max velocity added to train a pushing model without velocity as an input. The plot shows the evolution of the performance (NMSE) of these velocity-independent models trained for a larger bracket of velocities in the training data. Initially, adding more velocities improves performance. However, after 50-80mm/s the performance of the model worsens. We conclude that the quasi-static assumption does not hold anymore after that.

50-80mm/s and degrades after that. We conclude that the quasi-static assumption does not hold after that, and hence there is value in adding velocity as an explicit input.

8.3 Future work

Future research will focus on combining GP-SUM with planning and control techniques. Being able to detect multimodality or noisy actions can help to better navigate the complex and noisy dynamics of the system and reduce the final uncertainty in the state distribution.

Bibliography

[1] Website for the data set. URL `https://mcube.mit.edu/push-dataset`.

[2] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, 2016.

[3] K Ažman and Jus Kocijan. Dynamical systems identification using gaussian process models with incorporated local models. *Engineering Applications of Artificial Intelligence*, 2011.

[4] Maria Bauza and Alberto Rodriguez. A probabilistic data-driven model for planar pushing. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015, 2017.

[5] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, Jan 2004. ISSN 0885-8985. doi: 10.1109/MAES.2004.1263228.

[6] Marc Deisenroth, Marco Huber, and Uwe Hanebeck. Analytic moment-based gaussian process filtering. In *ICML*, pages 225–232, 2009.

[7] MP Deisenroth, CE Rasmussen, and D Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems*, volume 7, pages 57–64, 2012.

[8] Mehmet R. Dogar and Siddhartha S. Srinivasa. Push-Grasping with Dexterous Hands: Mechanics and a Method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2123–2130, 2010.

[9] Nima Fazeli, Elliott Donlon, Evan Drumwright, and Alberto Rodriguez. Empirical Evaluation of Common Impact Models on a Planar Impact Task. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[10] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar Sliding with Dry Friction Part 1 . Limit Surface and Moment Function. *Wear*, 143, 1991.

[11] Gregor Gregorčič and Gordon Lightbody. Nonlinear system identification: From multiple-model networks to gaussian processes. *Engineering Applications of Artificial Intelligence*, 2008.

[12] François Hogan and Alberto Rodriguez. Feedback Control of the Pusher-Slider System: A Story of Hybrid and Underactuated Contact Dynamics. In *WAFR*, 2016.

[13] Y. Huang, M. Bianchi, M. Liarokapis, and Yu Sun. Recent data sets on object manipulation: A survey. In *Big Data*, volume 4, 2016.

[14] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *ICML*, 2007.

[15] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996. ISSN 10618600.

[16] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[17] Juš Kocijan, Agathe Girard, Blaž Banko, and Roderick Murray-Smith. Dynamic systems identification with gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 2005.

[18] Roman Kolbert, Nikhil Chavan Dafle, and Alberto Rodriguez. Experimental Validation of Contact Dynamics for In-Hand Manipulation. In *ISER*, 2016.

[19] Jayesh Kotecha and Petar Djuric. Gaussian sum particle filtering. *IEEE Transactions on signal processing*, 2003.

[20] M. C. Koval, M. Klingensmith, S. S. Srinivasa, N. S. Pollard, and M. Kaess. The manifold particle filter for state estimation on high-dimensional implicit manifolds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4673–4680, May 2017. doi: 10.1109/ICRA.2017.7989543.

[21] M. Lau, J. Mitani, and T. Igarashi. Automatic Learning of Pushing Strategy for Delivery of Irregular-Shaped Objects. In *ICRA*, 2011.

[22] M. Lazaro-Gredilla and M.K. Titsias. Variational Heteroscedastic Gaussian Process Regression. In *ICML*, 2011.

[23] Q. Le and A. Smola. Heteroscedastic gaussian process regression. In *ICML*, 2005.

[24] Kevin M. Lynch and Matthew T. Mason. Stable Pushing: Mechanics, Controllability, and Planning. *IJRR*, 15(6), 1996.

[25] Kevin M Lynch, Hitoshi Maekawa, and Kazuo Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IROS*, 1992.

[26] Matthew T. Mason. Mechanics and Planning of Manipulator Pushing Operations. *IJRR*, 5(3), 1986.

[27] T. Meric, M. Veloso, and H.L. Akin. Push-manipulation of complex passive mobile objects using experimentally acquired motion models. *Autonomous Robots*, 38(3), 2015.

[28] Mustafa Mukadam, Xinyan Yan, and Byron Boots. Gaussian process motion planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15. IEEE, 2016.

[29] Roderick Murray-Smith and Daniel Sbarbaro. Nonlinear adaptive control using nonparametric gaussian process prior models. *IFAC Proceedings Volumes*, 35(1):325–330, 2002.

[30] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[31] Yunpeng Pan, Xinyan Yan, Evangelos A Theodorou, and Byron Boots. Prediction under uncertainty in sparse spectrum gaussian processes with applications to filtering and control. In *ICML*, pages 2760–2768, 2017.

[32] Robert Paolini, Alberto Rodriguez, Siddhartha S. Srinivasa, and Matthew T. Mason. A Data-Driven Statistical Framework for Post-Grasp Manipulation. *IJRR*, 33(4), 2014.

[33] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours. In *ICRA*, 2016.

[34] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In *NIPS*, 2008.

[35] Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[36] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini. A vision-based learning method for pushing manipulation. Technical report, IRCS-93-47, U. of Pennsylvania, Department of Computer and Information Science, 1993.

[37] E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In *NIPS*, 2006.

[38] Andreas S Stordal, Hans A Karlsen, Geir Nævdal, Hans J Skaug, and Brice Vallès. Bridging the ensemble kalman filter and particle filters: the adaptive gaussian mixture filter. *Computational Geosciences*, 15(2):293–305, 2011.

[39] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[40] S. Walker and J. K. Salisbury. Pushing Using Learned Manipulation Maps. In *ICRA*, 2008.

[41] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a Million Ways to be Pushed. A High-Fidelity Experimental Data Set of Planar Pushing. In *IROS*, 2016.

[42] Jiaji Zhou, Robert Paolini, J. Andrew Bagnell, and Matthew T Mason. A Convex Polynomial Force-Motion Model for Planar Sliding: Identification and Application. In *ICRA*, 2016.