GP-SUM. Gaussian Processes Filtering of non-Gaussian Beliefs

Maria Bauza and Alberto Rodriguez
Mechanical Engineering Department — Massachusetts Institute of Technology
<bauza, albertor>@mit.edu

Abstract—This work centers on the problem of stochastic filtering for systems that yield complex beliefs. The main contribution is GP-SUM, a filtering algorithm for dynamic systems expressed as Gaussian Processes (GP), that does not rely on linearizations or Gaussian approximations of the belief. The algorithm can be seen as a combination of a sampling-based filter and a probabilistic Bayes filter. GP-SUM operates by sampling the state distribution and propagating each sample through the dynamic system and observation models. Both, the sampling of the state and its propagation, are made possible by relying on the GP form of the system. In practice, the belief has the form of a weighted sum of Gaussians.

We evaluate the performance of the algorithm with favorable comparisons against multiple versions of GP-Bayes filters on a standard synthetic problem. We also illustrate its practical use in a pushing task, and demonstrate that GP-SUM can predict heteroscedasticity, i.e., different amounts of uncertainty, and multi-modality when naturally occurring in pushing.

I. INTRODUCTION

Robotics and uncertainty come hand in hand. Certainly, one of the defining challenges of robotics research is to design uncertainty-resilient behavior that can deal with noise in sensing, actuation or dynamics. We are interested in systems with complex dynamics where neglecting uncertainty or reducing it to simple Gaussian distributions is not sufficient.

In particular, this paper focuses on the problems of propagation and filtering for stochastic systems. We explore how to represent and propagate complex state distributions in the context of dynamic systems expressed as Gaussian Processes (GP), also known as GP-Bayes filtering.

The main contribution of this paper is an algorithm GP-SUM that tracks complex beliefs through a stochastic dynamic system without the need to either linearize the dynamic models or rely on Gaussian approximations of the belief. 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. The number of assumptions required to propagate in time the state distributions is kept to a minimum.

We compare our algorithm to other existing GP-filters like GP-UUKF or GP-ADF in a standard synthetic example proposed in [1] [2]. We show that GP-SUM obtains better filtering results after one and multiple filtering steps, that are also more stable and consistent. We illustrate the practicality of GP-SUM in a real pushing application by predicting how the uncertainty in a pushed object evolves over time. GP-SUM recovers the heteroscedasticity and multimodality present in some pushing behaviours.

Considering non-Gaussian and multimodal belief representations is specially important in the context of manipulation where actions can make or brake contact or go through stick/slip transitions in the contact behaviour. Ordinary tasks such as push-grasping a cup of coffee [3] are illustrative of the multimodality of the possible outcomes.

In previous work we investigated how different actions lead to different degrees of stochasticity in planar pushing. We proposed to use Heteroscedastic Gaussian Processes [4] (HGP) to capture the outcome and variance of pushing actions. This paper extends this idea—that actions determine the shape of the belief distribution—allowing multimodal and non-Gaussian representations of the state for the case where the observation and prediction models are GPs.

The paper is structured as follows. Section II reviews the main related work to GP-SUM. In Section III we provide background of Bayes filtering and Gaussian Processes to motivate the use of GP-Bayes filtering. The GP-SUM algorithm is described in Section IV where we also analyze its main assumptions and computational complexity. The applications of GP-SUM and comparisons with other GP-filters are developed in Section V. Finally, in Section VI we discuss limitations and future improvements of GP-SUM.

II. RELATED WORK

Gaussian processes have proved to be a powerful tool to model the dynamics of complex systems [5] [6] [7]. As a result, they have been applied to different contexts of robotics including planning and control [8] [9] [10], system identification [5] [11] [12], or filtering [13] [12]. In this work, we study the problem of propagating and filtering the state of a system by providing accurate distributions of the state space. When the models for the dynamics and the measurements are learned through Gaussian process regression, the filtering algorithms are referred as GP-Bayes filters. Among these algorithms, the most frequently considered are GP-EKF [13], GP-UUKF [13] and GP-ADF [1], with GP-ADF being considered the state-of-the-art for GP-filtering.

All these GP-filters rely on the assumption that the state distribution can be well captured by a single Gaussian
and require several approximations to ensure that the state distribution remains Gaussians over time. 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 unscattered 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, the GP-ADF algorithm is a sort of Assumed Density filter (ADF) that computes the first two moments of the state distribution exploiting the structure of GPs and thus returns a Gaussian distribution for the state of the system. Instead of limiting the state distribution to a single Gaussian, our algorithm GP-SUM is a GP-Bayes filter that allows non-Gaussian representations of the state distribution.

GP-SUM 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 sampled particle is associated with a weight and a covariance matrix. In the case of GP-SUM, thanks to the GP structure of the filter each sample is associated with a Gaussian. As a result, GP-SUM can be understood as a sampling algorithm that benefits from the parametric structure of Gaussians distributions to simplify its computations and represent the state distributions through weighted Gaussians. The use of sampling algorithms for GP-filtering has been studied in [13] 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 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 expressions for the state distribution. For instance, we can found some resemblances between GP-SUM and the algorithms Gaussian Mixture Filter (GMF) [14], Gaussian Sum Filter (GSF) [15], and Gaussian Sum Particle Filtering (GSPM) [16]; all using different techniques to propagate the state distributions. 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. GSP can be understood as a set of weighted EKF running in parallel. As a consequence it requires to linearize the system models while GP-SUM does not need to linearize any of the GP models considered. Finally GSPM, which has proven to be superior to GSP, is based on the sequential importance sampling filter (SIS). GSPM requires to sample from the importance function which is defined as the likelihood of an state $x$ given an observation $z$, $p(x|z)$. Our algorithm GP-SUM does not need to learn this new mapping $p(x|z)$ as it only considers the dynamics and measurement models of the system.

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 can still be applied to it. Similarly, GP-SUM can also be adapted to special types of GPs such as heteroscedastic or sparse GPs for learning the models of the system. For instance, GP-SUM can be easily extended to the case of sparse spectrum Gaussian processes (SSGPs) using the work by Pan et al. [2]. This implies that GP-SUM can be made significantly faster if desired through using sparse GP when learning the dynamics and observation models of the system.

III. BACKGROUND OF GAUSSIAN PROCESS FILTERING

This work focuses on the classical problem of Bayes filtering where 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.

A. Bayes filters

The goal of a Bayes filter is to track the state of the system, $x_t$, in a probabilistic setting. At a given 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 probability density of the state, $p(x_t)$, conditioned on the history of actions and observations obtained so far: $p(x_t|u_{t-1}, z_{1:t})$. This distribution 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 [16].

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:

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

where $p(x_{t-1}|u_{1:t-2}, z_{1:t-2})$ 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 (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 [16].

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:

$$p(x_t|u_{1:t-1}, z_{1:t}) = \frac{p(z_t|x_t)p(x_t|u_{1:t-1}, z_{1:t-1})}{p(z_t|u_{1:t-1}, z_{1:t-1})}$$ (2)

This expression usually can not be solved in a closed-form and several approximations are required to obtain a
reasonable estimation of the new belief. Linearizing the observation model or assuming Gaussianity are again common approaches [16].

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:

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

(3)

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

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 Process. This proves to be specially beneficial when the actual models are complex and parametric approaches do not provide a fair representation of the system behaviour [9, 17].

B. Gaussian processes

Gaussian processes (GPs) provide a flexible and non-parametric framework for function approximation and regression [18]. In this paper, 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 [18], the noise in the output is assumed to be Gaussian and constant over the input:

$$y(x) = f(x) + \epsilon$$

(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 $\epsilon \sim N(0, \sigma^2)$ represents zero-mean Gaussian noise with variance $\sigma^2$.

The assumption of constant Gaussian noise together with a GP prior on the latent function $f(x)$ makes analytically inference possible for GPs [5]. In practice, given a set of training points, $D = \{(x_i, y_i)\}_{i=1}^n$, and a kernel function, $k(x, x')$, is enough to learn a GP over $f(x)$. Given a new input $x_*$, the trained GP assigns a Gaussian distribution to the output $y_* = y(x_*)$ that can be expressed as:

$$p(y_*|x_*, D, \alpha) = N(y_*|a_*, c_*^2 + \sigma^2)$$

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, $\epsilon$ represents the vector of observations from the training set, and $\alpha$ is the set of hyperparameters including $\sigma^2$ 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: $\sigma^2$ and $c_*^2$ [5]. The first one, $\sigma^2$, 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 [18] 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 [2].

C. Heteroscedastic Gaussian processes

Assuming that the noise of the process $\sigma^2$ 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 [19], volatility forecasting [4] and robotic manipulation [17]. In Section V 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 (HGP). 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 HGP observations are assumed to be drawn from:

$$y(x) = f(x) + \epsilon(x)$$

(6)

where $\epsilon(x) \sim N(0, \sigma^2(x))$ explicitly depends on $x$ compared to (4) where $\epsilon$ is a random variable independent of $x$.

Our algorithm GP-SUM can be extended to the situations where the models for the transitions and measurements are given by heteroscedastic GPs. This is exemplified in Section V during the study of planar pushing where, depending on the type of push, the motion of the object is more or less noisy [17].

IV. GP-SUM Bayes filter

In this section we present our algorithm GP-SUM, discuss its main assumptions, and describe its computational complexity. Given that GP-SUM is a type of GP-Bayes filter, the first assumption is that both the transition and the measurement models are represented by trained 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 Gaussian distributed.
We are interested in the ability to keep track of complex beliefs, so rather than approximating them by Gaussians, GP-SUM is based on the weaker assumption that the prediction belief is well approximated by a mixture of Gaussians. Given that assumption, we can exploit the fact that the transition and observation models are GPs to correctly propagate the prediction belief, i.e. the pre-observation state distribution, over time without taking further assumptions. Obtaining a closed form solution of the belief at any time requires a further approximation as explained in Section IV-B.

A. Prediction update

The main idea behind GP-SUM is described in Algorithm 1 and can be derived as follows. Consider equations (1) and (3), then the belief at time $t$ can be written in terms of the prediction belief as:

$$p(x_t|u_{1:t-1}, z_{1:t-1}) \propto p(z_t|x_t) \cdot p(x_t|u_{1:t-1}, z_{1:t-1}) \quad (7)$$

If the prediction belief at time $t - 1$ can be well approximated by a finite sum of weighted Gaussians, we can write:

$$p(x_{t-1}|u_{1:t-2}, z_{1:t-2}) = \sum_{i=1}^{M_{t-1}} \omega_{t-1,i} \cdot \mathcal{N}(x_{t-1}|\mu_{t-1,i}, \Sigma_{t-1,i}) \quad (8)$$

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 the equations (1) and (7) as:

$$p(x_t|u_{1:t-1}, z_{1:t-1}) = \int p(x_t|x_{t-1}, u_{t-1})p(x_{t-1}|u_{1:t-2}, z_{1:t-2})dx_{t-1} \propto \int p(x_t|x_{t-1}, u_{t-1})p(z_{t-1}|x_{t-1})p(x_{t-1}|u_{1:t-2}, z_{1:t-2})dx_{t-1} \quad (9)$$

The prediction belief at time $t$ can be recursively computed using only the prediction belief at time $t - 1$ together with the transition and observation models. Moreover, if $p(x_{t-1}|u_{1:t-2}, z_{1:t-1})$ can be approximated as a mixture of Gaussians, then we can take $M_t$ samples from it, $\{x_{t-1,j}\}_{j=1}^{M_t}$, and approximate the previous integral by:

$$p(x_t|u_{1:t-1}, z_{1:t-1}) \propto \sum_{j=1}^{M_t} p(x_t|x_{t-1,j}, u_{t-1,j})p(z_{t-1}|x_{t-1,j}) \quad (10)$$

Because our transition model is given by a GP, $p(x_t|x_{t-1,j}, u_{t-1})$ is a Gaussian distribution that can be written as $\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$, and $p(z_{t-1}|x_{t-1,j})$ is a constant value. As a result, we can take:

$$\omega_{t,j} = \frac{p(z_{t-1}|x_{t-1,j})}{\sum_{k=1}^{M_t} p(z_{t-1}|x_{t-1,k})} \quad (11)$$

and express the prediction belief as a Gaussian mixture:

$$p(x_t|u_{1:t-1}, z_{1:t-1}) = \sum_{j=1}^{M_t} \omega_{t,j} \cdot \mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) \quad (12)$$

In the ideal case where all $M_t$ tend to infinity, 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 the previous algorithms for GP-Bayes filtering 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 (11) are directly related to the likelihood of the observations. As in most particle 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 V 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($\{\mu_{t-1,i}, \Sigma_{t-1,i}, \omega_{t-1,i}\}_{i=1}^{M_{t-1}}, u_{t-1}, z_{t-1}, M_t$):

$$\{x_{t-1,j}\}_{j=1}^{M_t} = \text{sample}(\{\mu_{t-1,i}, \Sigma_{t-1,i}, \omega_{t-1,i}\}_{i=1}^{M_{t-1}}, M_t$$

for $j \in \{1, \ldots, M_t\}$ do

$$\mu_{t,j} = \text{GP}_\mu(x_{t-1,j}, u_{t-1})$$
$$\Sigma_{t,j} = \text{GP}_\Sigma(x_{t-1,j}, u_{t-1})$$
$$\omega_{t,j} = p(z_{t-1}|x_{t-1,j})$$

end for

$$\{\omega_{t,j}\}_{j=1}^{M_t} = \text{normalize_weights}(\{\omega_{t,j}\}_{j=1}^{M_t})$$

return $\{\mu_{t,j}, \Sigma_{t,j}, \omega_{t,j}\}_{j=1}^{M_t}$

B. Measurement update: recovering the belief

After computing the prediction belief, we recover the belief of the system as another Gaussian mixture using equation (7):

$$p(x_t|u_{1:t-1}, z_{1:t}) \propto p(z_t|x_t) \sum_{j=1}^{M_t} \omega_{t,j} \cdot \mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) = \sum_{j=1}^{M_t} \omega_{t,j} \cdot p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j}) \quad (13)$$

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 mixture of Gaussians. However, in most cases $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ is not proportional to a Gaussian distribution and different approximations can be considered to convert it into a Gaussian form. For instance, the algorithm GP-EKF [13] linearizes the observation model and then uses the properties of EKF to express the previous distribution as a Gaussian. In this work, we exploit the technique proposed by Deisenroth et al. [11] to approximate $p(z_t|x_t)\mathcal{N}(x_t|\mu_{t,j}, \Sigma_{t,j})$ to a normal shape as it guarantees
that the moments of the distribution are preserved after the approximation and has proven to outperform GP-EKF results.

Algorithm 2 Belief recovery

belief_computation(\(\{\mu_{t,j}, \Sigma_{t,j}, \omega_{t,j}\}\)^{M_t}_{j=1}, z_t, M_t):
for \(j \in \{1, \ldots, M_t\}\) do
\(\mu_{t,j}, \Sigma_{t,j} = \text{Gaussian.approx}(\mu_{t,j}, \Sigma_{t,j}, z_t)\)
end for
\(\{\hat{\omega}_{t,j}\}_{j=1}^{M_t} = \{\omega_{t,j}\}_{j=1}^{M_t}\)
return \(\{\hat{\mu}_{t,j}, \hat{\Sigma}_{t,j}, \hat{\omega}_{t,j}\}_{j=1}^{M_t}\)

Following [1], to ensure the Gaussianity of the previous distribution it is required to assume that the distributions \(p(x_t, z_t|u_{1:t-1}, z_{1:t-1}) = p(z_t|x_t)p(x_t|u_{1:t-1}, z_{1:t-1})\) 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 Gaussian. However, this is true only when there exists a linear relation between \(x_t\) and \(z_t\). Given this assumption, Deisenroth et al. [1] prove that it is possible to analytically obtain the first two moments of \(p(z_t|x_t)N(\mu_{t,j}, \Sigma_{t,j})\) and thus approximate it as a Gaussian distribution.

Using this approximation technique, the GP-SUM filter can recover the belief of the system at time \(t\) as a mixture of Gaussians. Nonetheless, it is important to note that GP-SUM propagates over time the prediction beliefs instead of the beliefs. This implies that the previous approximation does not affect the computation of the following prediction beliefs. As a result, the approximation used to recover the belief will not affect the predictions for the future time steps hence reducing considerably the propagated error over time.

C. 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. 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 [18]. 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. The cost of sampling a Gaussian mixture can be considered constant, \(O(1)\), given that it is only required to select one of the Gaussians according to their weights and then sample from it. Approximating the belief does not represent an increase in \(O\)-complexity as it also implies \(O(Mn^2)\) operations [1]. Consequently GP-SUM has a time complexity that only increases linearly with the size of the mixture of Gaussians while providing a more realistic approximation of the true belief and leading to better filtering results as we discuss in Section V.

V. Results

We evaluate the performance of our algorithm in two different systems. The first one considers a 1D example taken from Deisenroth et al. [1] where our algorithm proves it can outperform the results of the previous GP-Bayes filter [1]. The second case analyses real data from a planar pushing experiment. In this second set of experiments, no measurement model is provided so GP-SUM only propagates the uncertainty of the system overtime capturing complex distributions for the position of the pushed object.

A. Synthetic task: algorithm evaluation and comparison

We compare GP-SUM with the existing GP-Bayes filters using the following 1D dynamical system:

\[
x_{k+1} = \frac{1}{2}x_t + \frac{25x_k}{1 + x_k^2} + w \quad w \sim \mathcal{N}(0, 0.2^2)
\]

and observation model:

\[
z_{k+1} = 5 \sin 2x_t + v \quad v \sim \mathcal{N}(0, 0.01^2)
\]

which correspond with the ones used by Deisenroth et al. [1] when evaluating their algorithm GP-ADF. The GP models for prediction and measurement are trained using 1000 points 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 over the state is assumed Gaussian with variance \(\sigma_0^2 = 0.5^2\) and mean \(\mu_0 \in [-10, 10]\). The mean of the prior distribution is modified to assess the filters behaviour in multiple scenarios and becomes specially interesting around \(x = 0\) where the dynamics are highly nonlinear. For each of the 200 values of \(\mu_0\) considered, the filter takes 10 steps in time. We repeat this procedure for 300 independent runs to average the performance of the algorithms evaluated: GP-SUM, GP-ADF and GP-UKF. We do not show the results for GP-EKF as they are considerably worse [1].

The error in the final state of the system is measured using 3 different metrics. The most relevant one is the negative log-likelihood, NLL, which measures how likely is the true state of the system according to the belief. We also report the root-mean-square error, RMSE, even thought the RMSE only takes into account 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 distribution. In all the metrics proposed, low values imply better performance.

From the results in Table I and Table II it is clear that GP-SUM outperforms the existing algorithms in all the metrics proposed including the RMSE and the Mahalanobis distance where only the first two moments of the belief are considered. Moreover, the performance of GP-SUM can be considered more stable as it obtains the lowest variance in

1The implementations of GP-ADF and GP-UKF are based on [1] and can be found at [https://github.com/ICL-SML/gp-adf](https://github.com/ICL-SML/gp-adf)
most of the metrics studied. For the case of GP-PF where a particle filter is used to keep track of the state density, we only computed the RMSE given that a particle filter does not provide a closed form for the belief. The number of particles considered is the same as components used in GP-SUM, \( M = 1000 \). In the first filtering step, GP-SUM and GP-ADF already outperform GP-PF and after a few steps in time, particle starvation becomes a major issue and the likelihood of the observations becomes extremely low. For this reason, we did not report a RMSE value for the GP-PF after 10 time steps.

We also show a comparison of the evolution of the NLL over time among the algorithms GP-ADF, GP-SUM and the approximation of GP-SUM as a single Gaussian (Figure 2). As the number of steps increases, all the algorithms tend to converge and GP-SUM and its Gaussian approximation coincide in value. This is because given more observations, the filter becomes more confident on the true state of the system and the belief becomes Gaussian shaped. Note that distributions can not become extremely certain because there is always some noise coming from both the dynamics and the measurements. From Figure 2 we can also observe that GP-ADF worsens its performance over time. This is due to those cases where the dynamics are highly non-linear and the variance of the GP-ADF belief does not shrink over time but increases. Just after one step in time, the Gaussian approximation of GP-SUM and GP-ADF have a very similar performance while GP-SUM is superior because it can allow non-Gaussian distributions for the belief including multimodality.

In Figure 2 it becomes clear that allowing non-Gaussian beliefs makes GP-SUM not only to assign higher likelihood to the true state of the system, but also to regress properly the shape of the true belief (computed numerically). Using a single Gaussian distribution to represent the belief and the prediction belief restricts considerably most GP-Bayes filters and is specially detrimental when the GP-models considered are highly nonlinear as observed for the case of GP-ADF. As a result, although GP-SUM is computationally more expensive, the expressiveness of its distributions and the lack of heavy assumptions makes it a good fit for those systems where multimodality and complex behaviours can not be
B. Real task: propagating uncertainty in pushing

We apply GP-SUM to the problem of uncertainty propagation for the motion of an object under planar pushing. The system considered uses an industrial ABB arm and a square object made of stainless-steel (more details in Yu et al. [20]). The end link of the robot has a cylindrical rod attached which works as the pusher and is approximated as a single contact point with the object.

In previous work, we showed that uncertainty in pushing is relevant and presents interesting properties [17]. In particular, pushing trajectories can show multimodality and are action dependent, meaning that different types of pushes lead to very distinct distributions over the object position. In particular, we are interested in characterizing different types of pushes depending on the final distributions that they provoke on the object position.

The dynamics of planar pushing were studied in [17] where the use of HGPs is proposed to capture the effect of pushing after short periods of time and regress the uncertainty created by different types of pushes. The HGPs considered take as inputs the contact point between the pusher and the object together with the pusher velocity and its direction. The outputs of the system are the displacement of the object relative to the pusher’s motion (Figure 3).

We do not include a measurement model for the system, so only uncertainty propagation over time is studied without filtering it. This implies that the prediction belief and the belief itself coincide and all the components in the Gaussian mixtures have the same weights.

Compared to all the existing algorithms for GP-filtering, GP-SUM is capable to obtain complex distributions that can be non-Gaussian shaped. This becomes clear in Figure 3b where after some time steps the shape of the distributions is no more Gaussian, but ring-shaped and by the end of the pushes the final distributions become multimodal. This result is intuitive as depending on the initial position of the object we expect the object to go to the left or to the right w.r.t. to the pushers direction. GP-Bayes filters that only provide Gaussian beliefs are incapable to regress these distributions and instead can at most provide a Gaussian wide enough to enclose the real distributions. This is specially harmful in situations where the mean of the Gaussian is placed in a low density region and taking decisions using just the expected value of a Gaussian would lead to inconsistent results. This is actually the case in Figure 3b as the mean of the final distributions would be in the middle of the two modes where almost no probability density is located.

Propagating the uncertainty of the object over time shows other interesting properties of the planar push system. In Figure 5 we can observe two different types of pushes and how the belief for the object position is propagated over time. It becomes clear that one of the pushes leads to more noisy distributions. Being able to recover these behaviours is specially useful when deciding what push actions to take. If our goal is to move the object to a specific region of the space, then it is advantageous to consider those actions that tend to provide narrower (low-variance) distributions and specially avoid those that can lead to multimodal outcomes that are hard to control.

VI. DISCUSSION AND FUTURE WORK

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 to simplify the propagation of the state distribution. In this paper, we propose the GP-SUM algorithm which considers the use of Gaussian mixtures to represent more complex distribution over the state of the system.

Our approach is sampling-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 that 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 components over time might be beneficial in order to ensure that the state space is properly covered. 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 used in sample-based algorithms can be effectively applied to improve its sampling process.

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-filter, 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 [2] would make the computations more efficient while allowing different types of kernel functions.

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.

**REFERENCES**

[1] M. Deisenroth, M. Huber, and U. Hanebeck, “Analytic moment-based gaussian process filtering,” in *ICML*, 2009, pp. 225–232.

[2] 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 *ICML*, 2017, pp. 2760–2768.

[3] M. R. Dogar and S. S. Srinivasa, “Push-Grasping with Dexterous Hands: Mechanics and a Method,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 2123–2130.

[4] M. Lazaro-Gredilla and M. Titsias, “Variational Heteroscedastic Gaussian Process Regression,” in *ICML*, 2011.

[5] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, “Dynamic systems identification with gaussian processes,” *Mathematical and Computer Modelling of Dynamical Systems*, 2005.

[6] D. Nguyen-Tuong, M. Seeger, and J. Peters, “Model learning with local gaussian process regression,” *Advanced Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[7] R. Paolini, A. Rodriguez, S. S. Srinivasa, and M. T. Mason, “A Data-Driven Statistical Framework for Post-Grasp Manipulation,” *IJRR*, vol. 33, no. 4, 2014.

[8] R. Murray-Smith and D. Sharbaro, “Nonlinear adaptive control using nonparametric gaussian process prior models,” *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 325–330, 2002.

[9] M. Deisenroth, C. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” in *Robotics: Science and Systems*, vol. 7, 2012, pp. 57–64.

[10] M. Mukadam, X. Yan, and B. Boots, “Gaussian process motion planning,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 9–15.

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

[12] K. Afzal and J. Kocijan, “Dynamical systems identification using gaussian process models with incorporated local models,” *Engineering Applications of Artificial Intelligence*, 2011.

[13] J. Ko and D. Fox, “Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models,” *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, 2009.

[14] A. S. Stordal, H. A. Karlsen, G. Nævdal, H. J. Skauge, and B. Vallès, “Bridging the ensemble kalman filter and particle filters: the adaptive gaussian mixture filter,” *Computational Geosciences*, vol. 15, no. 2, pp. 293–305, 2011.

[15] J. Kotecha and P. Djuric, “Gaussian sum particle filtering,” *IEEE Transactions on signal processing*, 2003.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[17] M. Bauza and A. Rodriguez, “A probabilistic data-driven model for planar pushing,” *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3008–3015, 2017.

[18] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[19] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, “Most likely heteroscedastic gaussian process regression,” in *ICML*, 2007.

[20] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, “More than a Million Ways to be Pushed: A High-Fidelity Experimental Data Set of Planar Pushing,” in *IROS*, 2016.