InfoSSM: Interpretable Unsupervised Learning of Nonparametric
State-Space Model for Multi-modal Dynamics

Young-Jin Park, and Han-Lim Choi

November 22, 2018

Abstract

The goal of system identification is to learn about underlying physics dynamics behind the time-series data. To model the probabilistic and nonparametric dynamics model, Gaussian process (GP) have been widely used; GP can estimate the uncertainty of prediction and avoid over-fitting. Traditional GPSSMs, however, are based on Gaussian transition model, thus often have difficulty in describing a more complex transition model, e.g. aircraft motions. To resolve the challenge, this paper proposes a framework using multiple GP transition models which is capable of describing multi-modal dynamics. Furthermore, we extend the model to the information-theoretic framework, the so-called InfoSSM, by introducing a mutual information regularizer helping the model to learn interpretable and distinguishable multiple dynamics models. Two illustrative numerical experiments in simple Dubins vehicle and high-fidelity flight simulator are presented to demonstrate the performance and interpretability of the proposed model. Finally, this paper introduces a framework using InfoSSM with Bayesian filtering for air traffic control tracking.

1 Introduction

State-space model (SSM) is one of the most general representation model that has been used on a wide range of fields (e.g. aerospace engineering, robotics, economics, and biomedical engineering, etc) for time series analysis [1]. The key idea in the state-space model is to construct the latent state space and its transition/observation model representing the sequentially observed data. In traditional researches and applications, linear Gaussian state-space models are commonly used to solve the state estimation (i.e. inference) or the system identification (i.e. model learning) problems based on Kalman filtering (KF) algorithms [2]. For the last few decades, a huge number of studies have tried to extend to nonlinear SSM such as particle filter [3], expectation maximization (EM) [4], unscented KF [5], and dual extended kalman filter [6], etc. But, most of the traditional SSM approaches assume a parametric latent model, thus they are only able to be used when we have fairly much information about the dynamics.

Alternatively, nonparametric nonlinear SSMs have been getting attention. Some recent approaches have design dynamics model and approximate inference structures based on neural networks, and could successfully represent the complex sequential data [7][10]. On the other hand, the probabilistic dynamics model are known to be more suitable in control problems because they can fully quantify the model uncertainty and enable safe and unbiased model learning [11]. As such, the probabilistic SSM approaches based on Gaussian process (GP) [12], the so-called Gaussian process state-space model (GPSSM), have been widely used in the system identification problems [13][15]. Although GPSSMs are powerful representation model, they are not still suitable to express a multi-modal dynamics model without any additional probabilistic structure since they assume Gaussian transition model. For example, if we want to learn the dynamics of airplane motions, single GP doesn’t seem to fully represent multiple behaviors (constant velocity, level turn, etc) at once. In such cases, it is more desirable to assume the latent dynamics model is comprised of multiple processes. However, GPSSM using multiple GPs has not been reported in the literature yet. The first key contribution of this paper is the extension of GPSSM framework to multi-modal state-space models, named Hybrid-GPSSM (H-GPSSM).

Meanwhile, GPSSMs are in a class of unsupervised learning that is possibly ill-posed. Although many unsupervised learning approaches believe their model will be automatically trained as human observer conceives, there is no guarantee they will learn an understandable representation without any regularization
because nonparametric models often possess too strong representation power. In particular, H-GPSSMs have difficulty to identify distinguishable multiple models without human supervision; the data does not include information regarding the types of dynamic models used. As such, it is never trivial to induce each GP dynamics to learn each mode for multi-modal dynamics. To resolve similar problems, several recent works in representation learning [16] and reinforcement learning [17][19] have proposed methods maximizing the mutual information between latent variables and the corresponding output. However, usage of mutual information regularization in the context of the probabilistic representation learning of time-series data is not yet reported in the literature. The second key contribution of this paper is the extension of GPSSM to the interpretable SSM structure by introducing mutual information regularization between the observation trajectory and the dynamics mode, where the resultant formulation is termed in this paper as InfoSSM.

Note that there have been earlier researches that utilized multiple GPs, called GP experts, to improve the regression performance [20–22]. Those approaches are based on the idea to divide the input space into subsets and assign a GP for each subset. However, such approaches can not be used directly into the GPSSM framework since the input space (i.e. latent state space) is unknown in unsupervised learning tasks. Thereby, this paper proposes a novel approach to assign data into multiple GPs by dividing the output space, not the input space, in an explainable way.

Rest of the paper is organized as follows. In section 2 we begin with a brief summary of the backgrounds about GP and GPSSM. Section 3 provides details about our algorithm; we show the mathematical formulation of H-GPSSM and the inference structures to approximate analytically intractable marginal likelihood. Section 4 provides details of InfoSSM, and describes the Bayesian filtering framework that estimates the latent state of the system. Finally in section 5 we analyze the performance of the proposed model with two system identification experiments and air traffic control (ATC) tracking simulation.

2 Backgrounds

2.1 Nomenclature

In the following derivations, subscript n denotes n\textsuperscript{th} data set, and subscript t denotes data at t\textsuperscript{th} time step. For instance, \(X_t = \{x_{t,n}\}_{n=1}^N\). Scalar, vector, and matrix values, are, represented using lowercase italic, lowercase bold italic, and capital bold italic, respectively. \(\mathcal{N}(\mathbf{X}; \mathbf{M}, \mathbf{K}, \Sigma) = \mathcal{N}(\text{vec}(\mathbf{X}); \mathbf{M}, \Sigma \otimes \mathbf{K})\) represents the probability density of matrix \(\mathbf{X}\) under the matrix normal distribution with mean matrix \(\mathbf{M}\) and two covariance matrices \(\mathbf{K}\) and \(\Sigma\), where \(\mathcal{N}\), vec(\(\mathbf{X}\)), and \(\otimes\) denotes normal distribution, the vectorization of \(\mathbf{X}\), and the Kronecker product, respectively. \(p(\mathbf{X}; \mathbf{Y}; \mathbf{Z})\) denotes the probability of random variable \(\mathbf{X}\) conditioning on random variable \(\mathbf{Y}\) and parameter \(\mathbf{Z}\).

2.2 Gaussian Process

GP is a flexible and powerful nonparametric Bayesian model to approximate the distribution over functions [12]. Consider we are given a data set of \(N\) input \(\mathbf{x}_n \in \mathbb{R}^P\) and output \(\mathbf{y}_n = f(\mathbf{x}_n) \in \mathbb{R}^Q\) pairs, \(D = \{((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N))\}\). The problem GP seeks to solve is to learn the function \(f: \mathbb{R}^P \rightarrow \mathbb{R}^Q\) that maps input space to output space of data\(^1\). GP assumes function outputs \(\mathbf{F} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{N \times Q}\) are jointly Gaussian:

\[
p(\mathbf{F}; \mathbf{X}) = \mathcal{N}(\mathbf{F}; m(\mathbf{X}), \mathbf{K}_{XX}, \Sigma)
\]

where \(\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times P}\), \(m(\cdot)\) is the mean function, \(\mathbf{K}\) is the GP kernel matrix, and \(\Sigma\) is the covariance matrix among multi-output. Affine mean function and squared exponential (SE) automatic relevance determination (ARD) kernel is commonly used for the covariance matrix between \(\mathbf{x}\) and \(\mathbf{x}'\):

\[
m(\mathbf{x}) = \mathbf{Hx} + \mathbf{b} \quad \mathbf{K}_{xx'} = \sigma_f^2 \exp \left( -\frac{1}{2} \sum_p \frac{(x_p - x'_p)^2}{\lambda_p} \right)
\]

where subscript \(p\) denotes \(p\textsuperscript{th}\) dimension of the variable (\(\mathbf{x} = \{x_p\}_{p=1}^P\)), while \(\mathbf{H} \in \mathbb{R}^{Q \times P}, \mathbf{b} \in \mathbb{R}^Q, \sigma_f, \lambda_p\) are hyperparameters of mean and covariance function to learn. Given input points \(\mathbf{X}\) and function outputs

\(^1\)In this paper, we particularly consider matrix-variate GP which further considers the covariance among multi-output.
the learned latent model. However, GPs are known to suffer from extensive $O(N^3)$ computational cost due to the matrix inversion of $N \times N$ covariance matrix, $K_{XX}$. To cope with the problem, sparse approximation is commonly used [23,24], which introduces $M$ inducing inputs $Z = [z_1, \cdots, z_M] \in \mathbb{R}^{M \times P}$ and outputs $U = [u_1, \cdots, u_M] \in \mathbb{R}^{M \times Q}$. Prior distribution of sparse GP is given by

$$p(U; Z) = MN(U; m(Z), K_{ZZ}, \Sigma).$$

Assuming inducing variables can sufficiently represent the distribution of original GP function, the true GP prediction is approximated as

$$p(f^*; F, x^*, X) \approx p(f^* | U; x^*, Z) = \mathcal{N}(f^*; r(x^* | U; Z), v(x^*; Z) \Sigma).$$

Note that inducing outputs are random variables to infer whereas inducing inputs are treated as parameters to learn. Resultant computational complexity is reduced to $O(NM^2)$. By an abuse of notation, we drop $|U$ and $; Z$ for the GP prediction term in the following derivations.

### 2.3 Gaussian Process State-Space Model

State-space model is a representation model to describe the dynamic system, consisted of latent Markovian state $x_t$ and observation output $y_t$ at time $t$, which is given by

$$\begin{align*}
x_t &= f(x_t) + \epsilon_f \\ y_t &= g(x_t) + \epsilon_g
\end{align*}$$

where $f$ and $g$ are transition and observation model, while $\epsilon_f$ and $\epsilon_g$ are process and measurement noise. $p_t$, $v_t$, and $a_t$ signify the position, velocity, acceleration vector. In this paper, we focus on the problem where the transition model is completely unknown (i.e. non-parametric) whereas observation model is fairly known (i.e. parametric)[3] In the discrete canonical state-space model with the predefined time interval $\delta t$, latent state at time $t+1$ can be recursively approximated as $x_{t+1} = x_t + \delta t \cdot \dot{x}_t$. For the sake of notational simplicity, we denote $\dot{f}_t = f(x_t)$ and $f_t = x_t + \delta t \cdot \dot{f}_t$. The key concept of GPSSM is modeling transition model of the system as GP function:

$$\begin{align*}
p(\dot{f}_t | x_t, U) &= \mathcal{N}(\dot{f}_t; r(x_t), v(x_t) \Sigma) \\ p(f_t | x_t, U) &= \mathcal{N}(f_t; x_t + \delta t \cdot r(x_t), \delta t^2 \cdot v(x_t) \Sigma).
\end{align*}$$

As shown in the graphical model of GPSSM in Fig. 1, the joint distribution of GPSSM variables for time step $t = 1, \cdots, T$ is factorized as

$$p(Y_{1:T}, X_{1:T}, F_{1:T-1}, U) = p(Y_{1:T} | X_{1:T})p(X_{1:T}, F_{1:T-1} | U)p(U) = \prod_{t=1}^{T} p(Y_t | X_t) \left[ p(X_1) \prod_{t=1}^{T-1} p(X_{t+1} | F_t)p(F_t | X_t, U) \right] p(U).$$

\footnote{It is easy to extend GPSSM to the model adopting both non-parametric transition model and observation model. However, such setting brings out the problem of severe non-identifiability between $f$ and $g$, and degrade the interpretability of the learned latent model.}
Figure 1: The graphical model of H-GPSSM. H-GPSSM reduces into GPSSM when the dynamics code $C_t$ is left out.

3 GPSSM with Multiple Transition Models

3.1 Multiple State-Space Modeling

Previous GPSSM studies \cite{13, 15} mostly targeted on the problem where the control input sequence is given. However, the control input is often unobservable and the only data we can access is the observation outputs, $y_{1:T}$. For instance, when we observe maneuvering airplanes, we can receive position (and velocity) data from radar or GPS signal but the corresponding control signals (i.e. thrust, and control surfaces, etc) are not given. Yet it is impractical to consider every possible control input sequences or infer the unknown control input. Alternatively, inspired from the interacting multiple model (IMM) algorithm \cite{25}, we propose the model, Hybrid-GPSSM, that approximates the dynamics model as the combination of $L$ numbers of motion patterns:

$$\dot{x}_t \approx \sum_{l=1}^{L} \mathbb{1}(c_t = l)[f_l(x_t) + \epsilon_{f,l}]$$  \hspace{1cm} (9)

where $\mathbb{1}(\cdot)$ is an indicator function, and $f_l$ and $\epsilon_{f,l}$ are $l^{th}$ transition model and process noise, respectively. As in the GPSSM structure, each transition model is represented by GP with $p(U_l) = \mathcal{M}(U_l; m(Z_l), K_{Z_lZ_l}, \Sigma_l)$, $Z = \{Z_1, \cdots, Z_L\}$ and $U = \{U_1, \cdots, U_L\}$:

$$p(\hat{f}_t \mid x_t, c_t, U) = \sum_{l=1}^{L} p(c_t = l)p(\hat{f}_t \mid x_t, U_l) = \sum_{l=1}^{L} p(c_t = j)\mathcal{N}(\hat{f}_t; r_l(x_t), v_l(x_t)\Sigma_l).$$  \hspace{1cm} (10)

where $r_l$ and $v_l$ is mean and variance function of $l^{th}$ GP prediction. In this paper, we call $c_t$ as a latent dynamics code at time $t$ in the way that $c_t$ determines the mode of dynamics governing the transition of latent states from time $t$ to $t+1$. We model the transition model of the latent dynamics code as a Markov chain with mode transition matrix $P$:

$$p(c_{t+1} = j|c_t = i) = P_{i,j}$$  \hspace{1cm} (11)

where $P_{i,j}$ is the $i^{th}$ row and $j^{th}$ column element of the matrix $P$, and $\sum_{j=1}^{L} P_{i,j} = 1$ for all $i \in \{1, \cdots, L\}$. Note that, when the mode transition matrix is the identity matrix, there is no mode transition along the observation sequence.\footnote{In practice, we could not find an advantage for using trainable transition matrix. Transition matrix converges to identity matrix unless the trajectory is very long.}
The joint distribution variables of H-GPSSM is then given by
\[
p(Y_{1:T}, X_{1:T}, F_{1:T-1}, C_{1:T-1}, U) = p(Y_{1:T} | X_{1:T})p(X_{1:T}, F_{1:T-1} | C_{1:T-1}, U)p(C_{1:T-1})p(U)
\]
\[
= \prod_{t=1}^{T} p(Y_t | X_t) \prod_{t=1}^{T-1} p(X_{t+1} | F_t)p(F_t | X_t, C_t, U) \prod_{t=1}^{T-2} p(C_{t+1} | C_t) \prod_{l=1}^{L} p(U_l).
\]

But, \(C_t\) unfortunately are hidden latent variables, which make the problem difficult to solve.

### 3.2 Inference Structure of Hybrid-GPSSM

The goal of GPSSM is to optimize GP hyperparameters and inducing variables maximizing the marginal likelihood \(p(Y_{1:T})\), so as to learn the latent dynamics that most represents the data. Unlike single-layer GP, however, GPSSM is no longer GP as it propagates recursively along the time. This causes a challenge for the estimation of the marginal likelihood that contains intractable integration. Instead of directly optimizing the marginal likelihood, variational inference approaches alternatively maximize the lower bound of the objective, which is called the evidence lower bound (ELBO):

\[
\log p(Y_{1:T}) = \log \int p(Y_{1:T} | X_{1:T}, F_{1:T-1}, C_{1:T-1}, U)p(X_{1:T}, F_{1:T-1}, C_{1:T-1}, U) \, dX_{1:T} \, dF_{1:T-1} \, dC_{1:T-1} \, dU
\]
\[
\geq \mathbb{E}_{q(X_{1:T}, F_{1:T-1}, C_{1:T-1}, U)} \left[ \log \frac{p(Y_{1:T} | X_{1:T}, F_{1:T-1}, C_{1:T-1}, U)p(X_{1:T}, F_{1:T-1}, C_{1:T-1}, U)}{q(X_{1:T}, F_{1:T-1}, C_{1:T-1}, U)} \right] \equiv \mathcal{L}(Y_{1:T}).
\]

This paper follows a doubly stochastic variational inference approach \[15,26\], known to give superior performance to other variational approach \[27\] or EM \[28\] for deep GP structures; it neither impose any assumptions about the independence between GP layers nor Gaussianity for GP outputs. Similar to \[15\], we factorize the variational distribution as
\[
q(X_{1:T}, F_{1:T-1}, C_{1:T-1}, U) = q(X_{1:T-1}, F_{1:T-1} | C_{1:T-1}, U)q(C_{1:T-1})q(U)
\]
\[
= \prod_{t=1}^{T-1} p(X_{t+1} | F_t)p(F_t | X_t, C_{1:T-1}, U) \prod_{t=1}^{T} q(C_{t})q(U).
\]

Remark that the gap between the log marginal likelihood and the lower bound decreases as the variational distribution \(q\) gets closer to the true posterior, hence it is very important to select a proper inference structure.
3.2.1 Approximate Inference for Latent states

Approximate inference for the posterior distribution of latent states consists of three parts: variational distribution of GP inducing points $q(U)$, initial state $q(X_1)$, and propagated states $q(X_{2:T})$. Primarily, $q(U)$ is factorized by $L$ numbers of $q(U_i)$ which is parameterized by a matrix-variate normal distribution sharing the same multi-output covariance with each GP prior:

$$q(U) = \prod_{l=1}^{L} q(U_i)$$
$$q(U_i) = MN(A_l, S_l, \Sigma_l)$$ (15)

where $A_l \in \mathbb{R}^{M \times Q}$ and $S_l \in \mathbb{R}^{M \times M}$ are variational parameters to optimize.

Secondly, for the initial state distribution, one naive approach is to parameterize $q(x_{1,n})$ per data. Though it may give fairly exact inference results, it is not only computationally expensive but impossible to generalize for unseen data. Alternatively, this paper uses amortized inference networks \cite{29, 30} to approximate the posterior distribution. We choose backward recurrent neural network (BRNN) that is designed to mimic Kalman smoothing algorithm \cite{8}; the initial hidden state of BRNN $h_1$ is expected to contain the compressed information from $y_T, \cdots, y_1$. An additional shallow neural network outputs the mean and covariance of initial states from $h_1$:

$$q(X_1) = \prod_{n=1}^{N} q(x_{1,n})$$
$$q(x_{1,n}) = N(x_{1,n}; \mu_{\phi}(y_{1:T,n}), V_{\phi}(y_{1:T,n}))$$ (16)

where $\phi$ is a set of parameters within a BRNN and a neural-network. To reduce the computational complexity, diagonal covariance is used. The inference structure is shown in Fig. [3]

![Inference Structure](image)

**Figure 3:** The inference structure for the initial latent state. $h_i$ is a $t^{th}$ hidden state

Finally, propagated states can be recursively derived from GP prediction:

$$q(X_{2:T}) = \prod_{n=1}^{N} q(x_{2:T,n})$$
$$q(x_{t+1,n} | x_{t,n}, c_{t,n}) = p(x_{t+1,n} | f_{t,n})q(f_{t,n} | x_{t,n}, c_{t,n})$$
$$q(f_{t,n} | x_{t,n}, c_{t,n}) = \int p(f_{t,n} | x_{t,n}, c_{t,n}, U)q(U)dU$$
$$= \sum_{l=1}^{L} p(c_{t,n} = l) \cdot N(\tilde{f}; \tilde{r}(x_{t,n}), \tilde{v}_l(x_{t,n})
\Sigma_l)$$ (17)

where $\tilde{r}(x) = m(x) + k_{xz}K_{zz}^{-1}(A - m(Z))$ and $\tilde{v}(x) = k_{xx} - k_{xz}K_{zz}^{-1}(K_{zz} - S)K_{zz}^{-1}k_{zx} \ [26, 31].
3.2.2 Approximate Inference for Latent Dynamics Code

The probability distribution of \( c_{1:T-1,n} \) can be ideally inferred from \( y_{1:T,n} \) although intractable. But it appears that the mode of dynamics can be distinguished from the shape of the observation trajectory. Thus, we built a neural network, \( Q_{\phi} \), that receives the observation trajectory as input and outputs the corresponding initial latent dynamics code:

\[
q(C_{1:T-1}) = \prod_{n=1}^{N} q(c_{1:T-1,n})
\]

\[
q(c_{1:T-1,n}) = q(c_{1,n}) \prod_{t=1}^{T-2} p(c_{t+1,n} | c_{t,n})
\]

\[
q(c_{1,n}) = \text{Cat}(c_{1,n}; Q_{\phi}(y_{1:T,n}))
\]

where \( \text{Cat}(x; p = \{ p_1, \cdots, p_L \}) \) is a categorical distribution with probability mass function: \( p(x = l) = p_l \) for \( l = 1, \cdots, L \). A softmax activation is used for the last layer to mimic the distribution.

3.3 Monte Carlo Objective

Based on (12 - 14), the ELBO is derived:

\[
\mathcal{L}(Y_{1:T}) = \sum_{t=1}^{T} \mathbb{E}_{q(x_t)} \left[ \log p(y_t | x_t) \right] + \mathbb{E}_{q(x_t)} \left[ \log \frac{p(x_t)}{q(x_t)} \right] + \mathbb{E}_{q(c_1)} \left[ \log \frac{p(c_1)}{q(c_1)} \right] + \mathbb{E}(U) \left[ \log \frac{p(U)}{q(U)} \right]
\]

\[
\mathcal{L} = \sum_{n=1}^{N} \left( \sum_{t=1}^{T} \mathbb{E}_{q(x_{t,n})} \left[ \log p(y_{t,n} | x_{t,n}) \right] + \mathbb{E}_{q(x_{1,n})} \left[ \log \frac{p(x_{1,n})}{q(x_{1,n})} \right] + \mathbb{E}_{q(c_{1,n})} \left[ \log \frac{p(c_{1,n})}{q(c_{1,n})} \right] \right) - \sum_{i=1}^{L} KL(q(U_i) || p(U_i)).
\]

Expectation terms can be computed by Monte Carlo sampling approach for ELBO update. Meanwhile, instead of using traditional ELBO update, this paper adopts Monte Carlo objective (MCO) update approach [32] that is known to not only increase the representation power of variational distribution but give a tighter lower bound by using multiple samples in a more effective way [33][34]. The MCO with \( K \) Monte Carlo samples is given by

\[
\mathcal{L}^{K}(Y_{1:T}) = \sum_{n=1}^{N} \mathbb{E}_{q(x^{(k)}_{1:T,n}, c^{(k)}_{1:T-1,n})} \left[ \log \frac{1}{K} \sum_{k=1}^{K} \prod_{t=1}^{T} p(y_{t,n} | x^{(k)}_{t,n}) \cdot \frac{p(x^{(k)}_{1,n})}{q(x^{(k)}_{1,n})} \cdot \frac{p(c^{(k)}_{1,n})}{q(c^{(k)}_{1,n})} \right] - \sum_{i=1}^{L} KL(q(U_i) || p(U_i)).
\]

See Appendix for detailed derivations. During the computation, reparameterization trick [29][30][35] is further used to make learning signal from the MCO be back-propagated into every inference network. As such, we construct the end-to-end training procedure in a fully differentiable manner.

4 InfoSSM

4.1 Mutual Information Regularization for Latent Dynamics Code

Ideally, we hope each GP is trained in a meaningful way even without any supervision. For example, when we have a data set of trajectories from the maneuvering airplane, we expect one GP learns constant velocity, another learns level turn, and the others learn climb and descend motions. But in the worst case, it is also possible that only one useful GP dynamics is trained to represent every motion while the others learn useless motions [36]. To avoid such circumstance, we need to introduce an additional information theoretic objective that lead the latent dynamics code to satisfies two criterions. First, the code should contain useful information determining the shape of trajectory. Second, trajectories generated from different GPs should
be distinguishable. As such, there should be the strong mutual dependence between the latent dynamics code sequence and the corresponding trajectory.

Inspired from InfoGAN [16], this paper as well adopts mutual information regularization for latent dynamics code. In the information theoretic sense, the mutual information is a measure quantifying the mutual dependence between two random variables:

\[ I(V; X) = H(V) - H(V | X) = H(X) - H(X | V). \]  

(21)

Mutual information can also be interpreted as the amount of information contained in one random variable about the other random variable. If two random variables are mutually independent, mutual information becomes zero. Finally, the InfoSSM is trained to maximize the mutual information along with the MCO:

\[ L_{info}^K(Y_{1:T}) = L^K(Y_{1:T}) + \lambda I(c_{1:T-1}; G(x_1, c_{1:T-1})) \]

(22)

where \( \lambda \) is a tunable parameter adjusting the relative scale between MCO and regularization. \( G \) is a function sampling the observation trajectory from the learned H-GPSSM model for given initial state and code sequence by using [9] and [17]. Note that, however, it is intractable to directly compute the mutual information term. Instead, we optimize the lower bound of the mutual information by variational information maximization approach [37].

\[ I(c_{1:T-1}; G(x_1, c_{1:T-1})) = H(c_{1:T-1}) - H(c_{1:T-1} | G(x_1, c_{1:T-1})) \]

\[ \geq H(c_{1:T-1}) + E_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_{1:T-1}; Y_{1:T}) \right] \]

\[ \approx H(c_1) + E_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_1; Y_{1:T}) \right]. \]

(23)

See Appendix for detailed derivations. \( H(c_1) \) is constant since the prior is fixed, thus we did not compute the term in the implementation. Unlike previous studies [16,19], we did not need to build a new auxiliary function \( q \) since we already have the approximate inference model \( Q_0 \) in (18). Expectation term can be computed by Monte Carlo estimation, and detail implementation is provided in Algorithm 1. We implement InfoSSM with Tensorflow [38], and Adaptive moment estimation (Adam) algorithm [39] is used for gradient-descent optimizer. The code is soon available at https://github.com/yjparkLiCS/InfoSSM.

4.2 Approximate Bayesian Filtering and Smoothing Using InfoSSM

By integrating the inference structure of InfoSSM with Bayesian filtering technique, the more accurate inferences of latent state trajectory become possible. Instead of simply propagating with the transition model as in the Eq. (17), it is desirable to recursively estimate the latent state for each time step by using the observation data. A Commonly used Kalman Filter, however, is not suitable for InfoSSM which contains non-linear transition models with non-Gaussian noise. Alternatively, particle filter (PF), one of the most well known non-linear data assimilation methods, is selected in this work. [4]

A PF is sample-based Bayes filter, where the state probability distribution is represented as:

\[ p(x) \approx \sum_{k=1}^{K} w^{(k)} \delta(x - x^{(k)}) \]

(24)

where \( x^{(k)} \) and \( w^{(k)} \) is the state and the weight for the \( k^{th} \) of \( K \) particles, respectively. For each time \( t \), the particle is propagated via the (learned) transition model while the weight of each particle of state \( x_i^{(k)} \) is given as its likelihood under given observation \( y_i^{(k)} \):

\[ w_i^{(k)} = w_i^{(k-1)} \cdot p(x_i^{(k)} | y_i). \]

(25)

After the propagation step, resampling step is applied when the effective sample size (ESS) drops below than the certain number (e.g. \( K/2 \)), to resolve the sample degeneracy problem. The weight of each particle

---

[4] Since Gaussian processes are analytically differentiable, InfoSSM can be also combined with the IMM-EKF filtering algorithm.
Algorithm 1 InfoSSM Implementation
1: Initialize $\lambda$.
2: for $\text{iter} = 1$ to $\text{max}\_\text{iter}$ do
3: \( L_{\text{Info}}^K \leftarrow 0. \)
4: for $n = 1$ to $N$ do
5: Select $n$-th observation data: $y_{1:T,n}$ from $\mathcal{D}$.
6: Sample $K$ initial states and code sequences: \( \{(x_{1:T,n}^{(k)}, c_{1:T-1,n}^{(k)})\}_{k=1}^K \), using (16) and (18).
7: Propagate states: \( \{x_{1:T,n}^{(k)}\}_{k=1}^K \), using (17) recursively.
8: Compute the MCO: \( L_{\text{Info}}^K (y_{1:T,n}) \), using (20).
9: \( L_{\text{Info}}^K \leftarrow L_{\text{Info}}^K + L_{\text{Info}}^K (y_{1:T,n}) \).
10: end for
11: Sample $c_{1:T-1} \sim p(c_{1:T-1})$.
12: Generate the trajectories $y_{1:T}$ from each $c_{1:T-1}$.
13: Compute the mutual information: $I$, using (23).
14: \( L_{\text{Info}}^K \leftarrow L_{\text{Info}}^K + \lambda I \).
15: Update InfoSSM using Adam.
16: end for

Algorithm 2 Target Tracking Using InfoSSM
Require: \( y_{t-T+1:t} \): past observation sequence, $K$: the number of particles
1: Sample $K$ initial states and code: \( \{(x_{t-T+1:t}^{(k)}, c_{t-T+1:t}^{(k)})\}_{k=1}^K \). // Smoothing
2: Initialize weights, \( \{w_{t-T+1}^{(k)}\}_{k=1}^K \), as $1/K$.
3: for $\tau = t-t+2$ to $t$ and $k = 1$ to $K$ do
4: Propagate \( \{x_{\tau}^{(k)}, c_{\tau}^{(k)}, w_{\tau}^{(k)}\} \) from \( \{x_{\tau-1}^{(k)}, c_{\tau-1}^{(k)}, w_{\tau-1}^{(k)}\} \), using PF (Algorithm 3). // Filtering
5: end for
6: Estimate $p(x_t) \approx \sum_{k=1}^K w_t^{(k)} \delta(x_t - x_t^{(k)})$. // Estimation

is reset to $1/K$ after resampling step. Note that the resampling step can be implemented in a differentiable way, thus it can be applied not only after the training but also during the training stage as introduced in [40]. Empirically found that, however, powerful inference through resampling step prevents the InfoSSM to learn accurate dynamics models. Thus, we used particle filtering only for the target tracking. A detail implementation of target tracking framework using InfoSSM is provided in Algorithm 2 and 3.

5 Experiments
5.1 Dubins Vehicle
First, we evaluate the proposed algorithm with the Dubins vehicle experiment. The Dubins path is commonly used approximated dynamics in the planning and control problems for wheeled vehicles and airplanes [41, 43]. The dynamics of Dubins’ vehicle is given by:
\[
\begin{align*}
\dot{p}_x &= V \cos\theta \\
\dot{p}_y &= V \sin\theta \\
\dot{\theta} &= u + \epsilon_{f,\text{true}}.
\end{align*}
\]
(26)
where $V$ is the speed of vehicle which is assumed as constant ($V = 1$), and $\epsilon_{f,\text{true}} \sim \mathcal{N}(0, 0.1)$. Depending on the control input, Dubins’ vehicle shows three motion primitives: right(R), straight(S), left(L) with $u = -1, 0, 1$, respectively. Fifty sub-trajectories are used for the training procedure with the observation noise \( \mathcal{N}(0, 0.1) \) for each dimension. In this toy example, we intentionally cut sub-trajectories in a way that there is no control input change in the middle of the sub-trajectory and fix the transition matrix as identity matrix in order to make an ideal experiment environment for the InfoSSM.
5.2 High-Fidelity Flight Simulation

To demonstrate the performance of our model for the practical situation, we evaluate the proposed algorithm with flight trajectory data generated from X-plane 11 simulator. The type of airplane was Cessna 172 skyhawk, and it flew in San Diego, California, U.S. One hundred sub-trajectories are used for the training procedure with the observation noise $\mathcal{N}(0,5)$ meters for each dimension. In this example, we considered the mode transition unlike the Dubins example, so let the transition matrix as a trainable variable initialized by the identity matrix.

5.3 Analysis of Learned Models

As baselines, we compare the InfoSSM with the state art of the art GPSSM model, PRSSM \[15\] ($L = 1$ and $\lambda = 0$), and InfoSSM without mutual information regularization ($\lambda = 0$) which is equivalent to H-GPSSM. We evaluate the models in three aspects: interpretability, model accuracy, and long-term prediction performance. We would refer the readers to Appendix C for experiment details.

5.3.1 Interpretability

Most importantly, we analyze the effect of mutual information to model interpretability. We compare the results between InfoSSM and H-GPSSM. To visualize the effect, several trajectories generated from different GP (i.e. latent dynamics code) at random initial states are plotted in Fig. 4 and Fig. 8. As shown in the results, we can’t see any interpretable meanings from H-GPSSM. We observe that the model without mutual information regularization tends to utilize only one GP; it tries to learn the motion considering overall transition, which is a very inefficient way. The InfoSSM, in contrast, successfully learns disentangled representation so that each code perfectly distinguishes L, S, and R motion pattern. Furthermore, as shown in Table 1, InfoSSM demonstrates the highest mutual information thus the most interpretable.

5.3.2 Model Accuracy

To compare the model accuracy (i.e. how well model represents the observation), we analyze two factors: the lower bound of log marginal likelihood and the reconstruction (i.e. inference) performance for 50 Dubins and 100 Cessna test trajectories which are not in training data set. To investigate the performance of the learned model itself, Bayesian filtering is not used during reconstruction processes. 30 samples are used for the reconstruction.

Primarily, note that the H-GPSSM often show worse performance than PRSSM which uses only single GP. This is due to the fact that H-GPSSM fails to use multiple GPs while the model complexity is increased and make GPs hard to train. Thanks to the mutual information regularization, on the other hand, InfoSSM successfully distinguishes data from different dynamics pattern, and gradient signal from each data group could be efficiently delivered to each GP. As a result, InfoSSM not only achieves the highest lower bound
Table 1: The Model Accuracy

|                | InfoSSM | H-GPSSM | PRSSM | InfoSSM | H-GPSSM | PRSSM |
|----------------|---------|---------|-------|---------|---------|-------|
| $L^N$          | 3727.2  | 1249.2  | 1525.5| -33110.8| -429933.9| -78155.3|
| $I$            | -0.0020 | -2.874  | -     | -0.0021 | -1.513  | -     |
| RMSE (test)    | 0.6217  | 2.2507  | 1.8430| 0.9082×10^3 | 1.512×10^3 | 1.365×10^3 |
| log-likelihood (test) | 40.119  | -3.124  | -14.996| -419.40 | -699.94 | -608.49 |

Table 2: The Tracking Accuracy

|                | InfoSSM | H-GPSSM | PRSSM | InfoSSM | H-GPSSM | PRSSM |
|----------------|---------|---------|-------|---------|---------|-------|
| Average Position Error (m) |         |       |       |         |       |       |
| East-West      | 20.2375 | 24.6978 | 23.7322| 6.1574  | 11.2264 | 9.6225 |
| North-South    | 22.7680 | 19.6529 | 25.6338| 6.4174  | 10.3157 | 10.6262|
| Horizontal     | 33.9305 | 34.7536 | 39.1680| 9.9167  | 16.9407 | 15.8674|
| Average Velocity Error (m/s) |       |       |       |         |       |       |

as shown in Table 1 but shows the smallest root mean square error and largest log-likelihood compare to H-GPSSM and PRSSM for both Dubins and Cessna airplane systems. Further reconstruction results are presented Table 4 and Fig. 10 in Appendix D.

5.3.3 Long-term Prediction

Finally, we evaluate the long-term prediction performance of InfoSSM to see whether the learned dynamics well matches with the true dynamics. From the initial state $x_1 = [0, 0, 1, 0]$, we propagate the Dubins vehicle for 100 time steps with $u = -1, 0, 1$. Using the first 20 step trajectory, the model inferred the code and latent state. Then, the latent state is propagated with the learned dynamics of corresponding code and compared with the true trajectory. As shown in Fig. 5, the InfoSSM shows highly accurate long-term prediction for each dynamics mode. Note that, however, baseline models fail to predict the future state and the uncertainty is increased rapidly as time goes.

5.4 Application in Air Traffic Control Tracking

After training InfoSSM, learned dynamics can be used for ATC tracking with filtering algorithms, as introduced in section 4.2. Since our InfoSSM only learned horizontal motion of maneuvering airplane, we additionally trained the GP dynamics for the vertical motion. For the model simplicity, we train the vertical motion by the state-space model with single GP (i.e. PRSSM approach).

ATC simulation using X-Plane 11 was performed for 1000 seconds. During the simulation, the 95% accuracy bound (i.e. measurement error for the horizontal position is under 30 meters which corresponds to the NAC$_p=9$, and vertical position is under 45 meters) is considered for the measurement model. Tracking results is provided in Table 2 and detail estimation results for InfoSSM is illustrated in Fig. 6. Although H-GPSSM shows better results for north-south position estimation, the overall horizontal estimation performance of InfoSSM is more accurate than other models. Especially, InfoSSM shows much better accuracy in velocity estimation during turning motions which supports that the dynamics of InfoSSM is the most accurate among three models. Further estimation results for H-GPSSM and PRSSM are in Fig. 11 in Appendix D.

5

[5] For Cessna airplane, we could not manipulate ground truth trajectories with constant motion pattern since the true dynamics is unknown, unlike the Dubins model.
6 Conclusion

In this paper, we presented the InfoSSM, an information theoretic extension of GPSSM that can effectively learn the multi-modal dynamical system. To describe the multi-modal dynamics, we modeled multiple dynamics by using multiple GPs assigned by the latent dynamics code. The inference of the latent state and dynamics code is performed via structured neural networks. Unlike previous inference methods, InfoSSM could learn interpretable representation for dynamics by using the mutual information regularization without any supervision from a human. The proposed model was evaluated with Dubins vehicle and high-fidelity flight simulator data and showed that the InfoSSM not only effectively represent multi-modal latent dynamics, but can reconstruct and predict the system. Combining with Bayesian filtering algorithm, InfoSSM could be applied to ATC target tracking framework and gave decent performance.
Figure 6: ATC tracking results by using InfoSSM and particle filter. Ground truth and estimated states are colored by black and red, respectively. Average altitude position and velocity error are 33.03 m and 0.9830 m/s.

References

[1] J. D. Hamilton, “State-space models,” *Handbook of econometrics*, vol. 4, pp. 3039–3080, 1994.
[2] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[3] N. J. Gordon, D. J. Salmond, and A. F. Smith, “Novel approach to nonlinear/non-gaussian bayesian state estimation,” in *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2. IET, 1993, pp. 107–113.

[4] T. Briegel and V. Tresp, “Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models,” in *Advances in Neural Information Processing Systems*, 1999, pp. 403–409.

[5] E. A. Wan and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. Ieee, 2000, pp. 153–158.

[6] T. A. Wenzel, K. Burnham, M. Blundell, and R. Williams, “Dual extended kalman filter for vehicle state and parameter estimation,” *Vehicle System Dynamics*, vol. 44, no. 2, pp. 153–171, 2006.

[7] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, “Deep variational bayes filters: Unsupervised learning of state space models from raw data,” *arXiv preprint arXiv:1605.06432*, 2016.

[8] R. G. Krishnan, U. Shalit, and D. Sontag, “Structured inference networks for nonlinear state space models.” in *AAAI*, 2017, pp. 2101–2109.

[9] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, “A disentangled recognition and nonlinear dynamics model for unsupervised learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3604–3613.

[10] J.-S. Ha, Y.-J. Park, H.-J. Chae, S.-S. Park, and H.-L. Choi, “Adaptive path-integral approach for representation learning and planning,” 2018.

[11] M. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search,” in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[12] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[13] R. Frigola, Y. Chen, and C. E. Rasmussen, “Variational gaussian process state-space models,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3680–3688.

[14] W. Sternberg and M. P. Deisenroth, “Identification of gaussian process state-space models,” 2017.

[15] A. Doerr, C. Daniel, M. Schiegg, D. Nguyen-Tuong, S. Schaal, M. Toussaint, and S. Trimpe, “Probabilistic recurrent state-space models,” *arXiv preprint arXiv:1801.10395*, 2018.

[16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[17] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim, “Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1235–1245.

[18] C. Florensa, Y. Duan, and P. Abbeel, “Stochastic neural networks for hierarchical reinforcement learning,” *arXiv preprint arXiv:1704.03012*, 2017.

[19] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, “Diversity is all you need: Learning skills without a reward function,” *arXiv preprint arXiv:1802.06070*, 2018.
[20] C. E. Rasmussen and Z. Ghahramani, “Infinite mixtures of gaussian process experts,” in Advances in neural information processing systems, 2002, pp. 881–888.

[21] T. Nguyen and E. Bonilla, “Fast allocation of gaussian process experts,” in International Conference on Machine Learning, 2014, pp. 145–153.

[22] C. Yuan and C. Neubauer, “Variational mixture of gaussian process experts,” in Advances in Neural Information Processing Systems, 2009, pp. 1897–1904.

[23] E. Sneldson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” in Advances in neural information processing systems, 2006, pp. 1257–1264.

[24] M. K. Titsias, “Variational learning of inducing variables in sparse gaussian processes,” in International Conference on Artificial Intelligence and Statistics, 2009, pp. 567–574.

[25] Y. Bar-Shalom, K. Chang, and H. A. Blom, “Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm,” IEEE Transactions on Aerospace and Electronic Systems, vol. 25, no. 2, pp. 296–300, 1989.

[26] H. Salimbeni and M. Deisenroth, “Doubly stochastic variational inference for deep gaussian processes,” in Advances in Neural Information Processing Systems, 2017, pp. 4591–4602.

[27] A. Damianou and N. Lawrence, “Deep gaussian processes,” in Artificial Intelligence and Statistics, 2013, pp. 207–215.

[28] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, “Deep gaussian processes for regression using approximate expectation propagation,” in International Conference on Machine Learning, 2016, pp. 1472–1481.

[29] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” arXiv preprint arXiv:1312.6114, 2013.

[30] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in International Conference on Machine Learning (ICML), 2015, pp. 1530–1538.

[31] Y.-J. Park, P. M. Tagade, and H.-L. Choi, “Deep gaussian process-based bayesian inference for contaminant source localization,” arXiv preprint arXiv:1806.08069, 2018.

[32] A. Mnih and D. J. Rezende, “Variational inference for monte carlo objectives,” arXiv preprint arXiv:1602.06725, 2016.

[33] Y. Burda, R. Grosse, and R. Salakhutdinov, “Importance weighted autoencoders,” arXiv preprint arXiv:1509.00519, 2015.

[34] C. Cremer, Q. Morris, and D. Duvenaud, “Reinterpreting importance-weighted autoencoders,” arXiv preprint arXiv:1704.02916, 2017.

[35] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” arXiv preprint arXiv:1611.01144, 2016.

[36] P.-L. Bacon, J. Harb, and D. Precup, “The option-critic architecture.” in AAAI, 2017, pp. 1726–1734.

[37] D. Barber and F. Agakov, “The im algorithm: a variational approach to information maximization,” in Proceedings of the 16th International Conference on Neural Information Processing Systems. MIT Press, 2003, pp. 201–208.

[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., “Tensorflow: A system for large-scale machine learning.” in OSDI, vol. 16, 2016, pp. 265–283.

[39] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.
[40] C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh, “Filtering variational objectives,” in Advances in Neural Information Processing Systems, 2017, pp. 6573–6583.

[41] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.

[42] A. Tsourdos, B. White, and M. Shanmugavel, Cooperative path planning of unmanned aerial vehicles. John Wiley & Sons, 2010, vol. 32.

[43] R. W. Beard and T. W. McLain, Small unmanned aircraft: Theory and practice. Princeton university press, 2012.

[44] S. C. Mohleji and G. Wang, “Modeling ads-b position and velocity errors for airborne merging and spacing in interval management application,” MITRE release, pp. 10–3026, 2010.
Appendix A. Derivation of the MCO

The MCO for deep Gaussian processes are previously introduced in [31]. The model likelihood $\mathcal{P} \equiv p(Y_{1:T})$ can be estimated by Monte Carlo estimator \[32\]:

$$\mathcal{P} = \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \frac{p(Y_{1:T}, x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})}{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \right]$$

(27)

where $\langle \cdot \rangle$ represents $\frac{1}{K} \sum_{k=1}^{K} \cdot$. By using [12] and [14],

$$\mathcal{P} = \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \frac{p(Y_{1:T}, x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})}{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \right]$$

$$= \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \prod_{t=1}^{T} p(Y_{t} | x_{t}^{(k)}) \cdot \frac{p(x_{t}^{(k)})}{q(x_{t}^{(k)})} \cdot \frac{p(u)}{q(u)} \cdot \frac{p(c_{t}^{(k)})}{q(c_{t}^{(k)})} \right]$$

$$= \mathbb{E}_{q(u)} \left[ \frac{p(u)}{q(u)} \right] \cdot \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \prod_{t=1}^{T} p(Y_{t} | x_{t}^{(k)}) \cdot \frac{p(x_{t}^{(k)})}{q(x_{t}^{(k)})} \cdot \frac{p(c_{t}^{(k)})}{q(c_{t}^{(k)})} \right]$$

(28)

By using Jensen’s Inequality, the lower bound of marginal log likelihood is given by

$$\log \mathcal{P} \geq \mathbb{E}_{q(u)} \left[ \log \frac{p(u)}{q(u)} \cdot \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \prod_{t=1}^{T} p(Y_{t} | x_{t}^{(k)}) \cdot \frac{p(x_{t}^{(k)})}{q(x_{t}^{(k)})} \cdot \frac{p(c_{t}^{(k)})}{q(c_{t}^{(k)})} \right] \right]$$

$$= \mathbb{E}_{q(u)} \left[ \log \frac{p(u)}{q(u)} \right] + \mathbb{E}_{q(u)} \left[ \log \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \prod_{t=1}^{T} p(Y_{t} | x_{t}^{(k)}) \cdot \frac{p(x_{t}^{(k)})}{q(x_{t}^{(k)})} \cdot \frac{p(c_{t}^{(k)})}{q(c_{t}^{(k)})} \right] \right]$$

$$\geq \mathbb{E}_{q(u)} \left[ \log \frac{p(u)}{q(u)} \right] + \mathbb{E}_{q(u)} \left[ \log \mathbb{E}_{q(x_{1:T}^{(k)}, f_{1:T-1}^{(k)}, u, e_{1:T-1}^{(k)})} \left[ \prod_{t=1}^{T} p(Y_{t} | x_{t}^{(k)}) \cdot \frac{p(x_{t}^{(k)})}{q(x_{t}^{(k)})} \cdot \frac{p(c_{t}^{(k)})}{q(c_{t}^{(k)})} \right] \right]$$

$$= \sum_{n=1}^{N} \mathbb{E}_{q(x_{1:T-1}^{(k)}, e_{1:T-1}^{(k)})} \left[ \log \mathbb{E}_{q(x_{1:n}^{(k)}, e_{1:n}^{(k)})} \left[ \prod_{t=1}^{T} p(y_{t} | x_{t,n}^{(k)}) \cdot \frac{p(x_{t,n}^{(k)})}{q(x_{t,n}^{(k)})} \cdot \frac{p(c_{t,n}^{(k)})}{q(c_{t,n}^{(k)})} \right] \right] - \sum_{t=1}^{L} KL(q(u))|p(u||).$$

(29)

Hence the MCO defined in \[20\] is the lower bound of the model marginal log-likelihood.

Remark that, the MCO becomes equivalent to the ELBO when $K = 1$. In fact, defined MCO is equivalent to the IWAE bound, thus it gets monotonically tighter as the number of sample size $K$ increases \[33\]. Hence, the MCO achieves a tighter lower bound than the traditional ELBO:

$$\mathcal{L}(Y_{1:T}) = \mathcal{L}^{1}(Y_{1:T}) \leq \mathcal{L}^{K}(Y_{1:T}) \leq \log \mathcal{P}.$$  

(30)
Appendix B. Derivation of the Variational Lower Bound of Mutual Information

\[ I(c_{1:T-1}; G(x_1, c_{1:T-1})) = H(c_{1:T-1}) - H(c_{1:T-1} | G(x_1, c_{1:T-1})) \]

\[ = H(c_{1:T-1}) + \mathbb{E}_{y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \mathbb{E}_{c'_{1:T-1} \sim p(c_{1:T-1} | y_{1:T})} \left[ \log p(c'_{1:T-1} ; y_{1:T}) \right] \right] \]

\[ = H(c_{1:T-1}) + \mathbb{E}_{y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ KL(p(\cdot | y_{1:T}) || q(\cdot | y_{1:T})) + \mathbb{E}_{c'_{1:T-1} \sim p(c_{1:T-1} | y_{1:T})} \left[ \log q(c'_{1:T-1} ; y_{1:T}) \right] \right] \]

\[ \geq H(c_{1:T-1}) + \mathbb{E}_{y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \mathbb{E}_{c'_{1:T-1} \sim p(c_{1:T-1} | y_{1:T})} \left[ \log p(c'_{1:T-1} ; y_{1:T}) \right] \right] \]

\[ = H(c_{1:T-1}) + \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_{1:T-1} ; y_{1:T}) \right] \]

\[ = \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1})} \left[ - \log p(c_{1:T-1}) \right] + \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_{1:T-1} ; y_{1:T}) - \log p(c_{1:T-1}) \right] \]

\[ \approx \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log \frac{q(c_{1:T-1} ; y_{1:T}) \prod_{t=1}^{T-2} p(c_t+1 | c_t)}{p(c_1) \prod_{t=1}^{T-2} p(c_t+1 | c_t)} \right] \]

\[ = \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_{1:T-1} ; y_{1:T}) - \log p(c_1) \right] \]

\[ = \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1})} \left[ - \log p(c_1) \right] + \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_{1:T-1} ; y_{1:T}) \right] \]

\[ = H(c_1) + \mathbb{E}_{c_{1:T-1} \sim p(c_{1:T-1}), y_{1:T} \sim G(x_1, c_{1:T-1})} \left[ \log q(c_1 ; y_{1:T}) \right] . \] (31)

Appendix C. Experiment Details

C.1. Shift and Rotation Invariant Inference Structures

In the present work, we are particularly focusing on the state-space models in the Cartesian coordinate. Dynamics in Cartesian coordinate has two general characteristics; physics are invariant to shift and rotation in the horizontal space. As such, we hope networks output the same dynamics code even if we input the shifted and rotated trajectories. To embed such characteristics, we propose a shift and rotation invariant inference structure via simple idea; we shift the initial position of the input trajectory to the origin and rotate the shifted trajectory with random angle: \[ y_{1:T,n} \leftarrow \Phi(y_{1:T,n} - y_{1,n}) \]

\[ q(c_{1,n}) = Cat \left( c_{1,n} ; Q_{\phi}(y_{1:T,n}) \right) \] (32)

where \( \Phi \) is function rotates the input trajectory with random angle in horizontal space. After the inference, this also can be interpreted as data augmentation techniques.

C.2. Dubins Airplane

For both Dubins example, canonical state-space model with \( x = [p_x, p_y, v_x, v_y] \) is constructed as

\[ [\dot{p}_x, \dot{p}_y] = [v_x, v_y] \]

\[ [\dot{v}_x, \dot{v}_y] = \sum_{i=1}^{L} p(c = i) [GP_i(v_x, v_y) + \epsilon_{f,i}] \]

\[ y = [p_x, p_y] + \epsilon_g \] (33)

where \( \epsilon_{f,i} \sim \mathcal{N}(0, \text{diag}([\sigma_{v_x,i}^2, \sigma_{v_y,i}^2])) \) and \( \epsilon_g \sim \mathcal{N}(0, \text{diag}([\sigma_{p_x}^2, \sigma_{p_y}^2])) \) that are both trainable variables. GPSSMs are constructed with \( M = 20 \), \( K = 4 \), \( T = 20 \), and \( \delta t = 0.1 \). For InfoSSM and H-GPSSM, three GPs are used \( (L = 3) \). We set \( \lambda = NT \) for InfoSSM.
C.3. High-Fidelity Flight Simulator

For both high-fidelity flight simulator example, canonical state-space model with \( \mathbf{x} = [p_x, p_y, v_x, v_y, a_x, a_y] \) is constructed as

\[
\begin{align*}
\dot{p}_x, \dot{p}_y &= [v_x, v_y] \\
\dot{v}_x, \dot{v}_y &= [a_x, a_y] \\
\dot{a}_x, \dot{a}_y &= \sum_{l=1}^{L} p(c = l) [\mathcal{GP}_l(v_x, v_y, a_x, a_y) + \epsilon_{f,l}] \\
y &= [p_x, p_y] + \epsilon_g \tag{34}
\end{align*}
\]

where \( \epsilon_{f,l} \sim \mathcal{N}(0, \text{diag}[\sigma_{a_x,l}^2, \sigma_{a_y,l}^2]) \) are trainable variables, and \( \epsilon_g \sim \mathcal{N}(0, \text{diag}[30^2, 30^2]) \) in SI units.

GPSSMs are constructed with \( M = 50, K = 8, T = 20, \) and \( \delta t = 2 \). For InfoSSM and H-GPSSM, three GPs are used (\( L = 3 \)). We set \( \lambda = 10NT \) for InfoSSM.

---

**Algorithm 3** Particle Filter Implementation

**Require:** \( \{x^{(k)}(t) c^{(k)}(t), w^{(k)}(t) \} \): latent states, and \( y_t \): observation

1. for \( k = 1 \) to \( K \) do
2. Sample \( c^{(k)}_t \) from \( c^{(k)}_{t-1} \), using (11).
3. Sample \( x^{(k)}_t \) from \( x^{(k)}_{t-1} \), using (17).
4. Compute the likelihood, \( p(x^{(k)}_t | y_t) \), using the observation model.
5. Update the weight, \( w^{(k)}_t \), using (25).
6. end for
7. Compute the ESS: \( \left( \sum_{k=1}^{K} (w^{(k)}_t)^2 \right)^{-1} \).
8. if \( \text{ESS} \leq K/2 \) then
9. for \( k = 1 \) to \( K \) do
10. Sample \( i \sim \text{Cat}(\{w^{(1)}_t, \ldots, w^{(K)}_t\}) \).
11. \( x^{(k)}_t, c^{(k)}_t \leftarrow x^{(i)}_t, c^{(i)}_t \)
12. end for
13. \( w^{(k)}_t \leftarrow 1/K \) for \( \{k = 1, \ldots, K\} \).
14. else
15. \( x^{(k)}_t, c^{(k)}_t \leftarrow x^{(k)}_t, c^{(k)}_t \) for \( \{k = 1, \ldots, K\} \).
16. end if
C.3. Air Traffic Control Tracking

Horizontal motion model is built by InfoSSM explained in the above. Similarly, state-space for vertical motion model is constructed as

\[
\begin{align*}
\dot{p}_h &= v_h \\
\dot{v}_h &= a_h \\
\dot{a}_h &= GP_h(v_h, a_h) + \epsilon_{f,h} \\
y_h &= [p_z] + \epsilon_{g,h}
\end{align*}
\]

where \( \epsilon_{f,h} \sim \mathcal{N}(0, \sigma_a^2) \) are trainable variables, and \( \epsilon_{g,h} \sim \mathcal{N}(0, 45^2) \) in SI units. GPSSMs are constructed with \( M = 50, K = 8, T = 20, \) and \( \delta t = 2 \). For PF filtering, implemented as algorithm 3, total 512 particles are used. During target tracking, transition matrix is fixed as follows:

\[
P_{i,j} = \begin{cases} 0.9, & \text{if } i = j \\ 0.05, & \text{otherwise} \end{cases}
\]

D. Further Results

![Sampled Cessna Airplane Trajectories](image)

(a) Cessna: H-GPSSM (\( \lambda = 0 \))  
(b) Cessna: InfoSSM (\( \lambda > 0 \))

Figure 8: The sampled Cessna airplane trajectories from different codes. Initial state is marked with red dot. From above, \( c = 1, 2, 3 \).

### Table 3: Dubins: The RMSE and Log-likelihood of Test Cases in Fig. 9

|          | RMSE          | log-likelihood |
|----------|---------------|----------------|
|          | InfoSSM | H-GPSSM | PRSSM | InfoSSM | H-GPSSM | PRSSM |
| Case 1. Right | 0.6767 | 2.8916 | 1.0623 | 50.0152 | -32.6587 | 31.6831 |
| Case 2. Straight | 0.4091 | 1.6435 | 2.0696 | 51.3238 | 30.6983 | 11.4390 |
| Case 3. Left | 0.7263 | 1.4932 | 1.5680 | 35.8378 | -10.6636 | -27.3243 |

### Table 4: Cessna: The RMSE and Log-likelihood of Test Cases in Fig. 10

|          | RMSE (\( \times 10^3 \)) | log-likelihood (\( \times 10^3 \)) |
|----------|---------------------------|------------------------------------|
|          | InfoSSM | H-GPSSM | PRSSM | InfoSSM | H-GPSSM | PRSSM |
| Case 1. Right | 1.015 | 1.4592 | 1.2521 | -0.4720 | -1.7942 | -0.4848 |
| Case 2. Straight | 0.7565 | 1.172  | 1.4298 | -0.1409 | -0.2579 | -0.2390 |
| Case 3. Left | 0.7157 | 2.0995 | 0.8766 | -0.1959 | -0.3853 | -1.1422 |

20
Figure 9: Dubins: Reconstruction results at three different cases. From the left, let case 1, 2, and 3. The RMSE and log-likelihood of each result are shown in Table 3.
Figure 10: Cessna: Reconstruction results at three different cases. From the left, let case 1, 2, and 3. The RMSE and log-likelihood of each result are shown in Table 4.
Figure 11: ATC tracking results by using H-GPSSM (blue) and PRSSM (green) and particle filter. Ground truth and estimated states are colored by black.