LINEAR VARIATIONAL STATE SPACE FILTERING AND MODEL PREDICTIVE CONTROL

JIANNING CUI [CUIJN@SEAS.UPENN.EDU], YILIN LI [YILLI@UPENN.EDU], ZHANGKAIWEN CHU [CHUZHKW@UPENN.EDU]

ABSTRACT. This project report proposed a method that builds a Linear Variational State Space Filter that, given the picture of a linear system, learns a latent space that mimics the state space of the dynamical system. It then applies MPC to achieve control objectives in the learned latent space and then maps it back through the decoder to generate images showing the states of the dynamical system. We show the effectiveness of our pipeline on a pendulum system and shows directions for future improvements like combining it with more advance MPC techniques like lumped System Level Synthesis Robust MPC, etc.

1. INTRODUCTION

With the success in generative representation learning techniques based on Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) in image generation, video prediction, etc.[1], it is natural to expect if the latent space learned by the generative model bears any physical meaning, such as the states of a dynamical system. If so, then whether the learned latent space can be used for control purposes. This problem is important, as it will provide one pathway to understand the latent space of the generative models, so that camera pictures can be used along with other sensor measurements like accelerometer, odometry, etc. to cross-validate and better estimate the states of the dynamical system.

1.1. **Contributions.** In this work, our main contribution is to show the possibility of using VAE to learn latent space that bears the meaning of the state space of a dynamical system as well as using this latent space for vision-based control objectives.

2. BACKGROUND

2.1. **Problem 1: Learning the latent space as the state space.** In this problem, we need to design and train a VAE such that, given an picture $\mathcal{X}_t$ of a dynamical system, it needs to learn a encoder that map that picture $\mathcal{X}_t$ into a latent space whose latent vector z_t is close to the state space vector of the dynamical system (e.g. the $\begin{bmatrix}\theta & \dot{\theta}\end{bmatrix}^T$ vector of a pendulum) so that it can filter out the current states of the dynamical system from the picture of the dynamical system. Then, from the latent space that mimics the state space, the decoder can generate another picture $\mathcal{X}_t$ that faithfully represents the current states of the dynamical system.

2.2. **Problem 2: MPC in the learned latent space.** After obtaining the latent space vector z_t, and assume we know the A and the B matrix of the linear dynamics of the latent states and we also assume that they are controllable, the problem of doing MPC in that latent space becomes the following:
Finding the series of control input u such that the latent states converge to the predefined goal (e.g. the origin), such that the constraints in the latent space is satisfied and the value of a latent space cost function (e.g. quadratic cost) is minimized. Also, we expect that the latent space solution can be mapped through the decoder back to images that correspond to the latent space.

3. RELATED WORK

3.1. **L-VSSF.** In [1], a new VAE structure is proposed to find a low dimensional embeddings suitable for control given pixel data and auxiliary low-dimensional (traditional) sensor measurements for Linear system's Variational State Space Filtering (L-VSSF). Specially, this embedding, or the latent space, tries to mimic the state space of the dynamical system.

3.2. **MPC, Robust MPC and lumped SLS Robust MPC.** In [2], the MPC problem is defined and solved. Also a robust MPC structure that can handle noise in the states is presented. In [3], a new robust MPC technique based on System Level Synthesis (SLS) is proposed that lumps both the noise and the uncertainly of the system dynamics together, give bounds to it and achieve robust and less conservative and fast computing MPC compared with Robust MPC and Tube MPC.

4. Approach

Generally, we alternatively apply L-VSSF on the photo of the dynamical system (e.g. a pendulum) to estimate the latent variables, and then use MPC to calculate the control, pass the control to the system and update the photo of the dynamical system. When the system dynamics is given, L-VSSF serves as a state estimator.

4.1. **Linear Variational State Space Filtering (L-VSSF).** Consider the state transition model

$$z_1 \sim p_\theta(z_1), z_{t+1} \sim p_\psi(z_{t+1}|z_t, u_t),$$

where z_t is the latent state obtained from the encoder, u_t is the input, and θ and ψ are the generative parameters and state transition model parameters. The observation model is given by

$$x_t \sim p_\theta(x_t|z_t).$$

where x_t here is the observation obtained from the generative model p_θ using z_t. Our goal is to estimate the the parameters θ, ψ. Given n independent input-observation trajectories $\mathcal{D} = \{(x^{(i)}_{1:T}, u^{(i)}_{1:T-1})\}_{i=1}^n$ sampled from the true data distribution $p_\mathcal{D}(\mathcal{D})$, the maximum a posteriori estimation for θ, ψ is given by

$$\hat{\theta}, \hat{\psi} = \arg\max_{\theta,\psi} p(\theta, \psi|\mathcal{D}) = \arg\max_{\theta,\psi} \frac{1}{n} \sum_{i=1}^{n} \log p_{\theta,\psi}(x^{(i)}_{1:T}|u^{(i)}_{1:T-1}).$$

The second equality is due to the independency of the trajectories. By variational inference techniques, after introducing a variational approximation distribution $q_{\phi,\psi(z_{1:T}|\mathcal{X}_{1:T},u_{1:T-1})}$, the MAP problem is equivalent to maximizing the evidence-based lower bound (ELBO) under certain conditions, where the ELBO is given by

$$\begin{aligned}\mathcal{L}(\phi, \theta, \psi, x_{1:T}|u_{1:T-1}) =& \log p_{\theta,\psi}(x_{1:T}, u_{1:T-1}) - G \\ =& -\mathrm{D}_{KL}(q_{\phi,\psi}(z_{1:T}|x_{1:T}, u_{1:T-1})||p_{\theta,\psi}(z_{1:T}||u_{1:T-1}))) \\ &+ \mathbb{E}_{q_{\phi,\psi}(z_{1:T}|x_{1:T},u_{1:T-1})}[\log p_{\theta,\psi}(x_{1:T}|z_{1:T})],\end{aligned} \tag{1}$$

where $G = \mathrm{D}_{KL}(q_{\phi,\psi}(z_{1:T}|x_{1:T}, u_{1:T-1})||p_{\theta,\psi}(z_{1:T}||u_{1:T-1})))$ is the gap between the approximated distribution and the true distribution.

4.2. **MPC.** Suppose we want to do optimal control on the following discrete linear system with A and B matrix as well as state at time t being x_t (here x_t is overloaded and should be differentiate from the observation x_t in 4.1):

$$x_{t+1} = Ax_t + Bu_t$$

with the following state and input constraints:

$$x_t \in \mathcal{X}, u_t \in \mathcal{U} \text{ for } k = 0, 1, \ldots, N-1, x_N \in \mathcal{X}_f$$

With some overloading, $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$ represent the set that x_t, x_N and u_t live in. We assume that (A, B) is controllable. We can use MPC to solve this problem. When there is no noise to the states, the problem can be formulated and solve as an MPC problem, in which at each time step we solve the following Constrained Finite Time Optimal Control (CFTOC) problem with quadratic cost functions as follows.

$$\begin{aligned}\min_{U_0}\ & x_N^T Q_f x_N + \sum_{k=0}^{N-1} (x_t^T Q x_t + u_t^T R u_t) \\ \text{subject to } & x_{t+1} = Ax_t + Bu_t, \text{ for } t = 0, 1, \ldots, N-1 \\ & x_t \in \mathcal{X}, u_t \in \mathcal{U} \text{ for } t = 0, 1, \ldots, N-1 \\ & x_N \in \mathcal{X}_f, x_0 = x(0)\end{aligned} \tag{2}$$

where $U_0 = \begin{bmatrix} u_0(x_0) & u_1(x_1) & \ldots & u_{N-1}(x_{N-1}) \end{bmatrix}$ is the N-step control sequence, $(x_t^T Q x_t + u_t^T R u_t)$ is the stage cost of each time step and $x_N^T Q_f x_N$ is the terminal cost of x_N. The $Q \succeq 0, Q_f \succeq 0, R \succ 0$ are hyper-parameters picked by the users. After solving the sequence of U_0, we just apply the first input $u_0(x_0)$, move to the next state x_1, then solve this CFTOC problem again.

For details of computing the constraint set $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$, please refer to [2].

When there is noise w_t,

$$x_{t+1} = Ax_t + Bu_t + w_t$$

we need to apply some robust MPC technique. We make the assumption that the noise w_t is bounded by some constant ϵ_w. Then, from [2] we know that, at each time step k, instead of the orignal constraint $x_t \in \mathcal{X}$ for x_t, the x_t now has to satisfy the following robust constraint:

$$x_t \in \mathcal{X} \ominus \mathcal{A}_t \mathbf{W}^t$$

In which $\ominus$ means the Pontryagin difference of 2 polytopes whose details are presented in [2].

$$\mathcal{A}_t = \begin{bmatrix} A^0 & A^1 & \dots & A^{t-1} \end{bmatrix}, \mathbf{W}^t = \begin{bmatrix} w & w & \dots & w \end{bmatrix}^\top$$

where $\mathbf{W}^t$ contains k w regions and each w region is formed by the noise.
Similarly, we have the constraints for the final state x_N as follows:

$$x_N \in \mathcal{X}_f \ominus \mathcal{A}_N \mathbf{W}^N$$

Thus, the above MPC problem can be modified into the following robust MPC problem, in which each CFTOC problem is formulated as follows:

$$\begin{aligned} \min_{U_0} \; & x_N^T Q_f x_N + \sum_{k=0}^{N-1} (x_t^T Q x_t + u_t^T R u_t) \\ \text{subject to } & x_{t+1} = A x_t + B u_t + w_t, \\ & x_t \in \mathcal{X} \ominus \mathcal{A}_i \mathbf{W}^i, u_t \in \mathcal{U}, w_t \preceq \epsilon_w \\ & x_N \in \mathcal{X}_f \ominus \mathcal{A}_N \mathbf{W}^N, x_0 = x(0) \end{aligned} \tag{3}$$

for $t = 0, 1, \dots, N-1$.
In our L-VSSF-MPC problem, the state x_t is the latent space vector z_t which is in R^2.

4.3. **Combining the L-VSSF and MPC.** In this project, we are using a linearized pendulum model. Its dynamics can be expressed in the form of

$$z_1 \sim \mathcal{N}(0, \Sigma_z), z_{t+1} = f_\psi(z_t, u_t) = A z_t + B u_t + w_t, w_t \sim \mathcal{N}(0, \Sigma_w), \tag{4}$$

where z_t is the state variable approximated by the latent space vector and u_t is the control. The observation model is modeled as a Gaussian

$$\frac{q_\phi(z_t|x_t)}{q_{\phi,\psi}(z_t)} \approx \frac{q_\phi(z_t|x_t)}{p_\theta(z_t)} \propto \mathcal{N}(h_t, H_t). \tag{5}$$

Where x_t is the observations, h_t is a neural network-parametrized mean function and H_t is a positive definite neural network-parametrized covariance matrix. Combining equation (4) and (5), the Gaussian filtering prior $q_{\phi,\psi}(z_t|x_{1:t-1}, u_{1:t-1}) \sim \mathcal{N}(p_{t|t-1}, P_{t|t-1})$ and posterior $q_{\phi,\psi}(z_t|x_{1:t}, u_{1:t-1}) \sim \mathcal{N}(p_{t|t-1}, P_{t|t-1})$ can be solved recursively in closed-form. The prior can be computed by using the Kalman filter propagation equations:

$$P_{t|t-1} = A P_{t-1|t-1} A^\top + \Sigma_z, p_{t|t-1} + B u_t, \tag{6}$$

and the posterior can be computed using the Information Filter update equations:

$$P_{t|t}^{-1} = P_{t|t-1}^{-1} + H_t^{-1}, P_{t|t}^{-1} p_{t|t} = P_{t-1|t-1}^{-1} + H_t^{-1} h_t. \tag{7}$$

At each time step, we can get the estimation for the latent variables by sampling from or using the mean of the posterior. By passing the estimated latent variable to the MPC algorithm, we can get the control u_{t+1} for the next time step. The algorithm can be summarized as:

Estimate the parameters θ, ψ by maximizing the ELBO (1).
For t = 1 to N **do**:

(1) Get a picture x_t from the environment.
(2) Update the prior estimation of the mean $p_{t|t-1}$ and the covariance $P_{t|t-1}$ using the control u_t and the dynamics f_ψ with (6).
(3) Update the posterior estimation of the mean $p_{t|t}$ and the covariance $P_{t|t}$ using $h_t(x_t, \theta)$ and $H_t(x_t, \theta)$ with (7).
(4) Get an estimation ot the latent variable $\hat{z}_t$ from the posterior.
(5) Solve the MPC problem (2) to get the control u_{t+1}.
(6) Manipulate the environment with control u_{t+1}.

---

5. Experimental Results

5.1. **VSSF.** The Datasets we used to train the VAE is generated by the following dynamics.

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$u_t = K^T x_t + \sqrt{u_\sigma}\nu$$
$$w_t \sim \mathcal{N}(0, \Sigma)$$

where $x_t \in R^2$ is the state vector, $A = \begin{bmatrix} 0.99 & 0.2 \\ 0 & 0.998 \end{bmatrix}$, $B = \begin{bmatrix} 0.03 \\ 0.3 \end{bmatrix}$, $K = \begin{bmatrix} -0.1 \\ -0.01 \end{bmatrix}$, $u_\sigma = 0.5$, $\nu \sim \mathcal{N}(0, 1)$, $\Sigma = 0.0001I$. Notice that the zeros step x_0 is drawn uniformly from its domain. We draw 10000 trajectories as the training data and each trajectory has 5 steps, where the state is stored as a 64×64 RGB image rendered with Pycairo module.

The latent spaces of two models are shown below. The left figure shows the latent space of the VAE model trained with true dynamics. Therefore, x-axis represents the angle of the pendulum and y-axis represents its angular speed. The right figure shows the latent space of the VAE model without telling the model what true dynamics are at each step. The 1st and the 2nd component x_1 and x_2 are inexplicable latent variables which fit well to the model. In the figures below, each cluster of 10 black dots is draw from the distribution of the latent space given a certain angle. One color specifies one angle from $-\pi$ to π. Notice that an interesting manifold is formed on the right figure. Both of the model's latent space can be used as the place where MPC happens. We use the model with true dynamics as the latent space for MPC because it's more explainable.

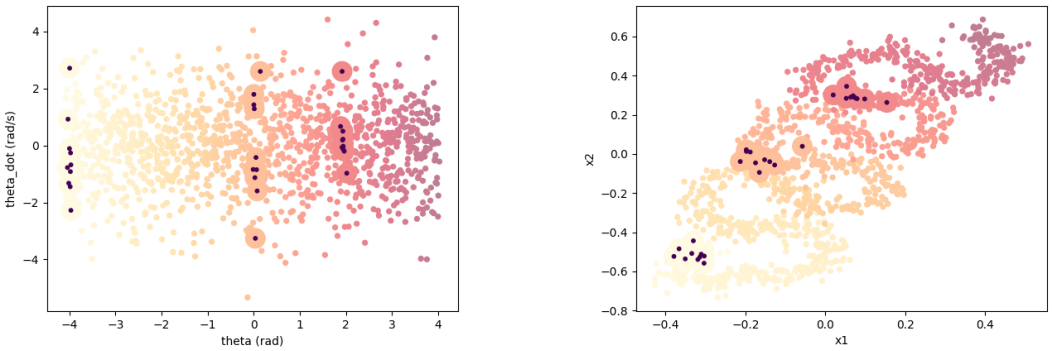

Figure 1. On the left is the latent space of VAE trained with true dynamics. On the right is the latent space of VAE which is trained by itself.

5.2. **MPC and Robust MPC.** After getting the x_t vector in 5.1, which is also the latent space vector z_t obtained from the encoder of the VSSF, the next step is to do MPC in the latent space. In the implementation, the prediction horizon is chosen to be $N = 5$. The simulation is run for 200 steps. We pick $Q_f = Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, R = 0.05$.
As for the constraint sets $\mathcal{Z}, \mathcal{Z}_f$ and $\mathcal{U}$ that z_t, z_N and u_t must be within, from the paper of [1] there is no explicit restrictions on z_t, z_N and u_t. Thus, in our implementation, we just set a trivial constraint set for $\mathcal{Z}, \mathcal{Z}_f$ and for $\mathcal{U}$ such that:

$$\begin{bmatrix} -4\pi \\ -10 \end{bmatrix} \leq z_t \leq \begin{bmatrix} 4\pi \\ 10 \end{bmatrix}, -20 \leq u_t \leq 20$$

And we compute the control invariant set for z_t based on the above constraints.
We also implemented the robust MPC. The covariance of the noise presented in [1] is:

$$\Sigma_w = \begin{bmatrix} 0.001 & 0.000237 \\ 0.000237 & 1.904 \end{bmatrix}$$

Here the variance for $\dot{\theta}$ is $\Sigma_{\dot{\theta}} = 1.904$. This high variance is to fully excite the signal for system identification purposes in case we don't know the dynamics of the system. In testing we don't have such high variance. Although the noise

during testing will have much smaller variance, it is still Gaussian with no upper bound. Yet the assumption of robust MPC requires that the noise is bounded by ϵ_w. Thus, it is not theoretically rigorous to apply Robust MPC to Gaussian noise. Thus, we just pick the standard deviation of the given noise $\sigma_{\dot{\theta}} = \sqrt{1.904} = 1.4$, as the assumed bound of the noise. Then we compute the robust control invariant set, implement and present some results of applying Robust MPC based on that noise bound to see if we can get some improvement.

5.3. **Experiment design and results.** To show the effectiveness of our method, we design the following experiments:
1. No noise for MPC/Robust MPC. In this case, no noise is added into the latent space. and we expect that the pendulum will go back to and stay at the origin from an arbitrary starting point.
2. Mild noise with $\sigma_{\dot{\theta}} = 0.4$ for MPC/Robust MPC. In this case, noise with $\sigma_{\dot{\theta}} = 0.4, \sigma_{\theta} = 0.001$ applies to the system.
3. Big noise with $\sigma_{\dot{\theta}} = 1.4$ for MPC/Robust MPC. In this case, noise with $\sigma_{\dot{\theta}} = 1.4, \sigma_{\theta} = 0.001$ applies to the system.

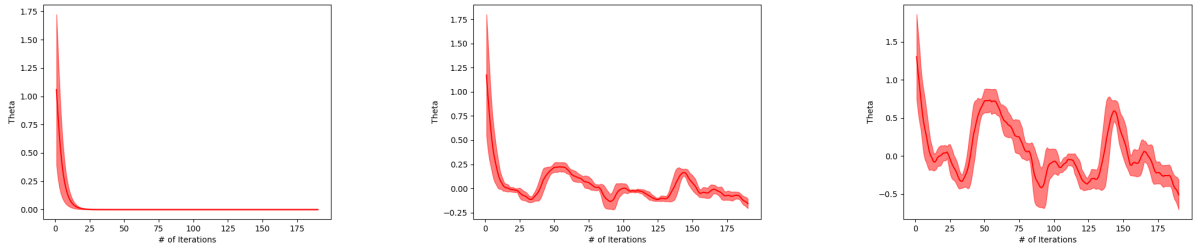

FIGURE 2. Resulting angles of pendulum from 3 experiments above. Left, Middle, Right correspond to experiment 1, 2, 3 separately.

The θ value of the above 3 cases are shown in figure results. From this figure we see that, when there is no noise, the pendulum soon converges to 0 angle and stays there. This shows the effectiveness of our VSSF+MPC pipeline. When there is noise in $\dot{\theta}$, the pendulum cannot stay at the 0 angle position. Yet it still stays within a region around the origin whose magnitude is directly related to the magnitude of the noise and never diverges. This shows that the MPC implementation has some resistance to the noise.
One interesting thing to notice is that, we also applied the Robust MPC in the simulation. Yet under the same seed of noises in the 3 cases, the results are the same as their non-robust counterpart (so that we don't even need to show extra pictures)! This phenomenon will be discussed in detail in the Discussion section.
Videos of the pendulum behavior are attached in the zip file.

6. DISCUSSION

Here we mainly discuss about why our current implementation of robust MPC leads to the same result as its counterpart of using non-robust MPC. The first thing to notice is the bounded noise assumption of robust MPC, as it assume that the max noise is bounded, but the Gaussian noise is not. Indeed we can pick our noise bound to be 3_w due to the 3σ property of normal distribution, yet this will make the controller to be too conservative and may lead to no feasible solutions of the robust control invariant set $\mathcal{Z}_f \ominus \mathcal{A}_N \mathbf{W}^N$ to satisfy the robustness constraint under such big noise.
To try to address this issue, in our implementation, we abuse the fact that there is no constraints for the latent state in the original paper, we intentionally pick the constraints for the latent state to be very big. that is why we have such loose bound for z_t as $\begin{bmatrix} -4\pi \\ -10 \end{bmatrix} \leq z_t \leq \begin{bmatrix} 4\pi \\ 10 \end{bmatrix}$ so that we let our final robust control invariant set $\mathcal{Z}_f \ominus \mathcal{A}_N \mathbf{W}^N$ to be not an empty set. Yet, as in MPC we only applies the first input and then re-plan, this means that, to let the final robust control invariant set to be non-empty, the robust constraint for x_1 has to be greatly inflated, making the robust constraint for x_1 being nonsense. That is probably why the result of using robust MPC is the same as that of using non-robust MPC as even the robust constraint set for x_1 is so loose.
As for future works, our next step direction includes, implementing the lumped SLS robust MPC, jointly learn the latent space as well as the (possibly non-linear) system dynamics of the system from scratch, fusion with (partial or full) state supervision, exploring more broader scenarios, etc.

REFERENCES

[1] D. Pfrommer and N. Matni, "Linear variational state space filtering," *arXiv preprint arXiv:2201.01353*, 2022.
[2] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
[3] S. Chen, N. Matni, M. Morari, and V. M. Preciado, "System level synthesis-based robust model predictive control through convex inner approximation," *arXiv preprint arXiv:2111.05509*, 2021.