Probabilistic Modeling and Bayesian Filtering for Improved State Estimation for Soft Robots

DongWook Kim, *Student Member*, IEEE, Myungsun Park, *Student Member*, IEEE, and Yong-Lae Park, *Member*, IEEE

***Abstract*—State estimation is one of the key requirements in robot control, which has been achieved by kinematic and dynamic models combined with motion sensors in traditional robotics. However, it is challenging to acquire accurate proprioceptive information in soft robots due to relatively high noise levels and hysteretic responses of soft actuators and sensors. In this study, we propose a method of estimating real-time states of soft robots by filtering noisy output signals and including hysteresis in the models using a Bayesian network. This approach is useful in constructing a state observer for soft robot control when both the kinematic model of the actuator and the model of the sensor are used. In our method, we regard a hysteresis function as a conditional random process model. We then introduce a dynamic Bayesian network composed of the actuator and the sensor models of the target system using distribution hysteresis mapping. Finally, we show that solving a Bayesian filtering problem is equivalent to sub-optimal state estimation of the soft system. This paper describes two ways for defining modeling and filtering; one is by Gaussian process regression combined with an extended Kalman filter, and the other is based on variational inference with a particle filter. While the first approach relaxes the uncertainty level in modeling to Gaussian, the second approach illustrates a general probability distribution. We experimentally validate the proposed methods through real-time state estimation of a sensor-integrated soft robotic gripper. The result shows significant improvement in state estimation compared to conventional estimation methods.**

***Index Terms*—Probability and Statistical Methods, Modeling, Control, and Learning for Soft Robots, Model Learning for Control, Hysteresis Analysis for Soft Robots**

I. Introduction

SOFT robots take great advantage of their structural compliance for safe interaction with environments, which enables them to address a wide range of challenges that traditional robots have not been able to deal with [1]–[4]. However, it is challenging to analyze soft robotic systems using conventional methods since they have additional properties, such as continuity and hysteresis, which have not been considered in the rigid-body kinematics and dynamics [4]–[6]. To address this issue, researchers have tried to model end-to-end relations of soft systems using black-box models based on deep learning. They have also tried to model the internal kinematics of soft structures using continuum mechanics with characterization of viscoelastic behaviors of soft materials. In deep learning, hysteresis and nonlinearity are analyzed by using time-correlated neural networks. For hysteresis, a recurrent neural network (RNN), a long-short term memory (LSTM), or a time delay artificial neural network (TDANN) has been employed to model temporal relations [7]–[17]. Meanwhile, the use of continuum mechanics has focused on modeling the internal kinematics of soft materials using approximation methods, such as viscoelastic models, the elastic rod theory, or the finite element approximation [18]–[25].

Constructing a model of a soft robot improves the accuracy of state estimation or control of the robot in real-time. In particular, rather than using a single model for estimating states, using both actuator and sensor models, based on their kinematics/dynamics and the observation model, respectively, is effective in improving the estimation accuracies, which is called a state observer (e.g., Luenberger observer [26]). However, the models cannot be always accurate, since sensor

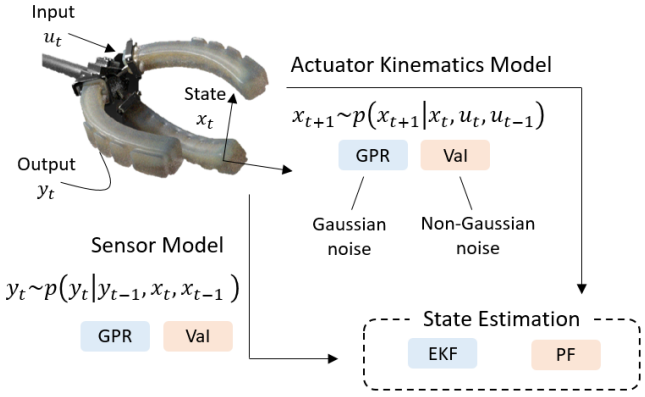

Fig. 1. Overall concept of probabilistic modeling and Bayesian filtering of soft robot. We use both models of actuator kinematics and sensor dynamics to improve the accuracy of the state estimation. The models are constructed to handle nonlinearity and temporal correlation of hysteresis in the sense of probabilistic data-driven function. We can handle both Gaussian and non-Gaussian noises using Gaussian process regression (GPR) with extended Kalman filter (EKF), and variational inference (VaI) with a particle filter (PF), respectively.

Manuscript received November 5, 2020; accepted January 21, 2021. This article was recommended for publication by Associate Editor and Editor P. Dupont upon evaluation of the reviewers' comments. This work was supported in part by the National Research Foundation of Korea (Grants NRF-2016R1A5A1938472 and NRF-2017M2A8A1092482) funded by the Korean Government (MSIT), and in part by the Technology Innovation Program (Grant 20008912) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea). *(Corresponding author: Yong-Lae Park)*

The authors are with the Department of Mechanical Engineering; the Institute of Advanced Machines and Design (IAMD); the Institute of Engineering Research, Seoul National University, Seoul, 08826, Republic of Korea. (E-mails: {shigumchis,arielrbts,ylpark}@snu.ac.kr)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Source code for this paper can be found in https://github.com/mochacoco/TRO_PMBF.

Digital Object Identifier 10.1109/TRO.2020.XXXXXXX

signals may contain noises, and actuator outputs are susceptible to perturbations. An inaccurate model for a state observer thus may lead to instability or divergence of the system. One solution to improve the model accuracy in state estimation is to use a probability distribution model, in which the model contains information on the uncertainty variance level (e.g., Kalman filter [27]). Despite various existing methods, constructing a distribution model with a state observer for a soft robot has been rarely studied, since incorporation of both nonlinearity and hysteresis in modeling is not only tedious but also complex, requiring building of distribution models.

Therefore, we propose a method of designing a state observer based on a data-driven distribution model to improve the accuracy of state estimation in physically soft systems. We have two main goals in this study. First, we develop a skeleton theory for distribution modeling of soft robots, capable of characterizing both hysteresis and nonlinearity. By expressing hysteresis as a first-order Markov chain, we derive the model of the actuator kinematics and the sensor as random processes. Second, we merge two data-driven models, the actuator model that constructs the relations between the actuator input and the state, and the sensor model that maps from the state to the sensor output. These two pieces of information are mustered to derive a dynamic Bayesian network (DBN). In this case, Bayesian filtering works as a state observer to improve the estimation accuracy.

Two ways of setting up the model and the filtering process are also presented in this work. First, we limit the class of the random process to be Gaussian. A Gaussian process regression (GPR) [28] is used to construct the actuator's kinematic model and the sensor model. Given the GPR mean and covariance information, an extended Kalman filter (EKF) is used as a filtering algorithm. Second, we consider a general random process, in which the model uncertainty may show a non-Gaussian behavior. In this case, variational inference (VaI) [29]–[31] constructed with a neural network is used as a model, and a particle filter (PF) is used as a filtering algorithm. Both methods were experimentally evaluated using a PneuNet-type soft actuator integrated with a soft sensor, and the result showed improvement in state estimation compared to estimation with previous methods. The method of GPR with an EKF allows for model brevity and reduced calculation time compared to that of VaI with a PF. In the case of complex hysteresis, such as a multi-modal behavior, the VaI with a PF is appropriate. The overall process of our work is summarized in Fig. 1, and the contributions of this work are

1) formulation of a random process to model soft robots capable of characterizing hysteresis and handling uncertainty, and
2) construction of a dynamic Bayesian network (DBN) for Bayesian filtering for improved state estimation of a soft robotic system, which makes the system robust against perturbations.

II. Background

This section reviews recent work related to modeling and state estimation of soft robot systems, followed by a short description of VaI and a discussion of uncertainty in soft robot systems focused on when and where it appears during operation.

A. Modeling and State Estimation of Soft Robots

Deep learning has recently been used to model physically soft systems [16], [32], focusing on addressing the issue of hysteresis, where certain learning techniques, such as RNNs, LSTMs, or TDANNs, are suitable for extracting temporal correlation from hysteresis. For example, RNNs and LSTMs have been used to efficiently process proprioceptive information of a soft actuator [7], or to localize contacts on tactile skin [12], [13], [15]. TDANN and its variations have been employed for high-level tactile sensing [8], [17] and model-based control [10]. In addition to end-to-end modeling approaches discussed so far, LSTMs have been used to estimate the internal kinematics of soft actuators based on proprioceptive sensor feedback [14].

Internal kinematics of soft systems based on more conventional techniques, such as nonlinear system identification [33], [34], control with expanded feature spaces [35], finite element methods (FEMs) [18], [19], [22], and locally-segmented models [25], [36], [37], have been developed. Even creep effects of non-rigid systems have been modeled using elastic, viscoelastic, and hyperelastic rod theories [21]–[24].

State estimation can be done by a state observer in control. A disturbance observer, a particular form of a state observer, is effective in handling uncertainty that is one of the most common issues in soft robot systems. A curvature model of a soft structure has been used to design a disturbance observer for correcting estimated states from learned data [38]. An FEM-based reduced state observer has been implemented to a control algorithm of a soft robotic arm [19]. A Kalman filter has beem employed for improved state estimation in dynamic modeling of a soft inflatable robot [39].

Our approach uses deep learning to construct models of actuator kinematics and sensor dynamics. By directly mapping inputs to outputs, we can not only define the task of interest easily but also reduce the number of input parameters. Although it is possible to include uncertainties in modeling by employing learning techniques, such as RNNs or ANNs, that use statistical methods of confidence test or check similarity values, these methods can handle only a certain class of uncertainties (e.g., Gaussian) and need advanced processing for handling general distribution [40]. Moreover, the input dimension of such cases is too long for Bayesian filtering; the sequence length of the LSTM used in [15] was 40 for example. In our method, the size of the network input is only 3, which can be easily handled by a DBN.

B. Variational Inference

The VaI method is used to model general probability distributions, even including non-Gaussian distributions. Let distribution $p(x)$ be a complex distribution of our interest. We define encoder $p(z \mid x)$ and decoder $p(x \mid z)$ where z is the hidden feature of x, and both of them can be designed using neural networks, as shown in Fig. 2. We assume that there

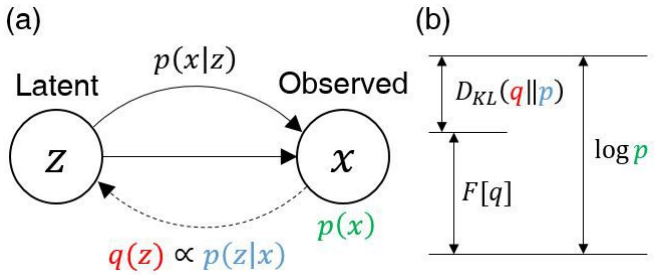

Fig. 2. (a) Latent variable model that consists of latent variable z, observable x, connected with encoder $p(z \mid x)$ and decoder $p(x \mid z)$. (b) To maximize the posterior, both Kullback-Leibler (KL) divergence and variational free energy $F[q]$ are considered.

exists $q(z)$ that can explain the encoder $p(z \mid x)$. Then, we use Kullback-Leibler (KL) divergence to express how much p and q are different, which is always positive; i.e.,

$$\mathcal{D}_{KL}(q(z) \parallel p(z \mid x)) \geq 0. \tag{1}$$

However, we cannot find the actual value of the measure of Equation 1 by direct calculation. We rather decompose the relation to:

$$\begin{aligned} &\mathcal{D}_{KL}(q(z) \parallel p(z \mid x)) = \int_{\mathcal{Z}} q(z) \log \frac{q(z)}{p(z \mid x)}\, dz \\ &= \int_{\mathcal{Z}} q(z) \log \frac{q(z)p(x)}{p(x \mid z)p(z)}\, dz \\ &= \mathcal{D}_{KL}(q(z) \parallel p(z)) + \log p(x) - E_{z\sim q(z)}[\log p(x \mid z)] \end{aligned} \tag{2}$$

where $p(z)$ is a prior distribution that can be arbitrarily determined, and $\mathcal{Z}$ is the set of all possible z. By minimizing Equation 2, we can find the approximation of the encoder $p(z \mid x)$ as $q(z)$. Then, we obtain an inequality,

$$\log p(x) \geq E_{z\sim q(z)}[\log p(x \mid z)] - \mathcal{D}_{KL}(q(z) \parallel p(z)) =: F[q] \tag{3}$$

where the right side of the inequality is defined as a variational free energy ($F[q]$) or an evidence lower bound (ELBO). Maximizing $F[q]$ minimizes $\mathcal{D}_{KL}(q(z) \parallel p(z \mid x))$ so that $p(z \mid x)$ can be approximated as $q(z)$. As a result, if we sample z from $p(z)$, it will be similar to $q(z)$ and also $p(z \mid x)$, and the decoder $p(x \mid z)$ can express the complex distribution. Details on VaI can be found in [29].

C. Uncertainty in Soft Robot System

There exist various sources that induce uncertainties in modeling soft robots. There are two types of uncertainties. One is epistemic uncertainty in the model induced by limited dataset or prior knowledge, while the other is aleatoric (i.e., statistical) uncertainty due to external disturbances or system noises, some of which cannot be inferred from training data [41]. In our system, both types of uncertainties exist.

Uncertainty can occur by change of the physical properties of the materials. Polymer materials are prone to aging. When an actuator or a sensor is made of polymer, it needs time to fully cure and to be chemically stable, and its property gradually changes [42]. The internal bonding structure can also be transformed if used over time [43]. Moreover, there always exist variations in materials and manufacturing tolerances.

Uncertainty also exists in the operating hardware of soft robots. A control system needs different hardware drivers depending on the types of actuation mechanisms, which may show different noise and accuracy levels. When a human user comes in the loop of a soft robot system, as often can be seen in wearable robots [8] and haptic devices [44], [45], the issue of uncertainty becomes especially serious.

III. Probabilistic Modeling of Hysteresis Function

In this section, we construct a probability model to represent hysteresis, $y_t = f(x_{t-k:t})$, where $x, y \in \mathbb{R}$ are defined in every time t, and $x_{t-k:t} := [x_{t-k}, x_{t-k+1}, ..., x_t]^\top$. Rather than directly implementing previous methods of deep learning or continuum mechanics to modeling of hysteresis, which typically requires a large volume of input variables, we reduce the number of inputs to lessen the computational burden during the filtering process in our approach.

A. Mathematical Definitions and Assumptions

Throughout the paper, we express the system (or function) in a continuous space but in a discrete-time domain. If the system is defined in a continuous-time domain, it can be converted to a system in a discrete-time domain by sampling, provided that aliasing does not occur. Let us think of a simple example of a one-dimensional hysteresis curve between input $x \in \mathbb{R}$ and output $y \in \mathbb{R}$, where x_t denotes the value of x at discretized time t in this problem. Hysteresis is a causal mapping in which the output y is dependent on both the entire past inputs of x, which is $x_{0:t}$, and the initial condition $y_t = f(x_{0:t}, y_0)$ [46], [47]. However, in this setting, the size of the input dimension is different in every timestep. Thus, for modeling with simplicity but without losing the generality, we made the following assumptions to relax the condition.

Assumption 1: Hysteresis output y is dependent only on the past $k+1$ inputs of x satisfying $t \geq k$, i.e., there exists a function $f : \mathbb{R}^{k+1} \to \mathbb{R}$ such that

$$y_t = f(x_{t-k:t}). \tag{4}$$

Assumption 2: The difference between two time-adjacent inputs is bounded by some positive real-valued number $\eta \in \mathbb{R}^+$,

$$\Delta x_t := \|x_{t+1} - x_t\| \leq \eta. \tag{5}$$

Assumption 3: Hysteresis function $y_t = f(x_{t-k:t})$ is locally Lipschitz with respect to $x_{t-k:t}$. In other words, for all $x^A_{t-k:t}, x^B_{t-k:t} \in \mathbb{R}^{k+1}$,

$$\|f(x^A_{t-k:t}) - f(x^B_{t-k:t})\| \leq L\|x^A_{t-k:t} - x^B_{t-k:t}\| \tag{6}$$

with Lipschitz constant $L \in \mathbb{R}^+$.

Assumption 1 is necessary for modeling with simplicity that limits the size of the input vector to $k+1$, in which it is appropriate for general data-driven methods (i.e., neural

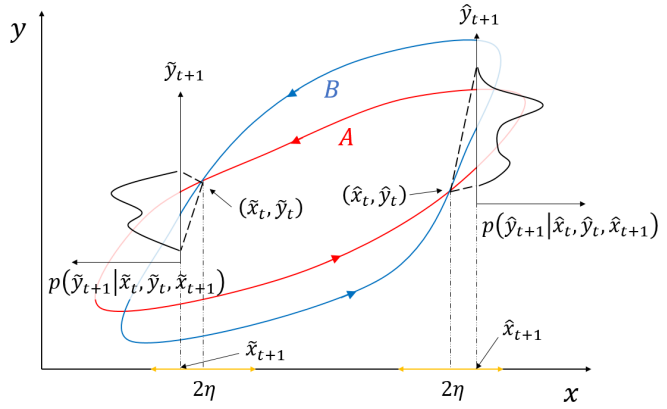

Fig. 3. Example trajectories of a hysteresis loops generated by the same system, and its one-step evolution for two cases: increasing input ($\hat{x}_{t+1} > \hat{x}_t$) and decreasing input ($\tilde{x}_{t+1} < \tilde{x}_t$). If two different solutions for different input sequences meet at a single point, i.e., $(\hat{x}_t, \hat{y}_t)$ or $(\tilde{x}_t, \tilde{y}_t)$, the next input-output tuple $(\hat{x}_{t+1}, \hat{y}_{t+1})$ or $(\tilde{x}_{t+1}, \tilde{y}_{t+1})$ lies within a certain interval.

networks) to fix the size of the input. Assumption 2 limits the maximum number of evolutions of the input variable, which is the dynamic bandwidth. Assumption 3 confines the type of the hysteresis function to a locally-Lipschitz type only.

B. Modeling of Hysteresis Function as Probability Box

We can visualize the above three assumptions on a two-dimensional xy plane with two output trajectories from the same system, as shown with A and B in Fig. 3. Each curve contains the history of the inputs and the corresponding outputs from the hysteresis function $y_t = f(x_{t-k:t})$. If the two trajectories intersects at time t (i.e., $x_t^A = x_t^B$ and $y_t^A = y_t^B$), the next inputs x_{t+1}^A and x_{t+1}^B should be defined in the interval $[x_t^A - \eta, x_t^A + \eta]$ or $[x_t^B - \eta, x_t^B + \eta]$, based on Assumption 2. Then, even if the output values of A and B at t are the same (i.e., $y_t^A = y_t^B$), and the next inputs are the same again (i.e., $x_{t+1}^A = x_{t+1}^B$), the outputs of A and B at $t+1$ can be different (i.e., $y_{t+1}^A \neq y_{t+1}^B$), since the inputs at the time interval $[t-k, t-1]$ still influence the outputs. Before we consider the difference between the two outputs y_{t+1}^A and y_{t+1}^B, we first define the difference of the inputs over the $k+1$ time interval:

$$\mathbf{\Delta x_t} := x_{t-k+1:t+1} - x_{t-k:t} \in \mathbb{R}^{k+1}. \tag{7}$$

We can show that the difference between y_{t+1} and y_t for any trajectory is bounded, using the multi-dimensional mean value theorem:

$$\begin{aligned} &\|y_{t+1} - y_t\| = \|f(x_{t-k+1:t+1}) - f(x_{t-k:t})\| \\ &= \|\nabla f(x_{t-k:t} + \alpha \mathbf{\Delta x_t}) \cdot \mathbf{\Delta x_t}\|, \ \exists \alpha \in [0,1] \\ &\leq \|\mathbf{\Delta x_t}\| \|\nabla f(x_{t-k:t} + \alpha \mathbf{\Delta x_t})\| \\ &= \eta\sqrt{k+1} \|\nabla f(x_{t-k:t} + \alpha \mathbf{\Delta x_t})\|, \end{aligned} \tag{8}$$

where ∇f is the gradient of f. Finally, the difference between the two outputs y_{t+1}^A and y_{t+1}^B can be derived by the result of Equation 8 and Assumption 3 as

$$\begin{aligned} &\|y_{t+1}^A - y_{t+1}^B\| \\ &= \|f(x_{t-k+1:t+1}^A) - f(x_{t-k+1:t+1}^B)\| \\ &= \|f(x_{t-k:t}^A + \mathbf{\Delta x_t^A}) - f(x_{t-k:t}^B + \mathbf{\Delta x_t^B})\| \\ &= \|f(x_{t-k:t}^A + \mathbf{\Delta x_t^A}) - f(x_{t-k:t}^A) \\ &- f(x_{t-k:t}^B + \mathbf{\Delta x_t^B}) + f(x_{t-k:t}^B)\| \\ &\leq \eta\sqrt{k+1}(\|\nabla f(x_{t-k:t}^A + \alpha \mathbf{\Delta x_t^A})\| + \\ &\|\nabla f(x_{t-k:t}^B + \beta \mathbf{\Delta x_t^B})\|) \\ &\leq \eta\sqrt{k+1}(L+M), \ \exists (\alpha, \beta) \in [0,1], \end{aligned} \tag{9}$$

where $L, M \in \mathbb{R}^+$ are the Lipschitz constants corresponding to the trajectories A and B, respectively. The result shows that if values of x_t, y_t, and x_{t+1} are identically defined for two different solutions A and B of the hysteresis system $y_t = f(x_{t-k:t})$, the distance between the two outputs $\|y_{t+1}^A - y_{t+1}^B\|$ is bounded by the function of the maximum state evolution η and the time-dependency k. If $k = 1$ or $k = 2$, or in other words, if there is no hysteresis (i.e., $y_t = f(x_t)$) or a weak time-dependency (i.e., $y_t = f(x_t, x_{t-1})$), fixing x_t, y_t, and x_{t+1} leads to the identical solution, which is trivial. As a result, a hysteretic system $y_t = f(x_{t-k:t})$ can be expressed as a function of x_t, x_{t-1}, and y_{t-1} with a residual term ϵ_t which is bounded by $\eta\sqrt{k+1}(L+M)$. While the residual term ϵ_t is bounded, we can say that the hysteresis can be expressed as a probability box or a random process, which means that the reachable set of y_{t+1} with given y_t, x_t, and x_{t+1} is compact (Fig. 3):

$$y_t \sim p(\cdot \mid x_t, x_{t-1}, y_{t-1}). \tag{10}$$

This probability distribution can be modeled with a function approximator f_θ based on supervised learning methods. For example, the most common probability distribution is Gaussian, as represented by using a weight parameter θ with a mean μ and a variance σ:

$$(\mu_{y_t}, \sigma_{y_t}) = f_\theta(x_t, x_{t-1}, y_{t-1}). \tag{11}$$

The final result infers that our proposed probabilistic settings can be used as an *a priori* knowledge for modeling hysteresis. This method is useful in that we can model hysteresis with a single-input single-output (SISO) relation by using only three input parameters and one output parameter for a probabilistic point of view, compared to RNNs or TDANNs that generally have more than three inputs to model temporal correlations.

C. Modeling of Soft Robot with Dynamic Bayesian Network

We now start modeling of a soft robot system that consists of a soft actuator and a proprioceptive soft sensor. As we defined the hysteresis function as a random process, the actuator kinematics and the sensor outputs can be both modeled with probability distributions like the expression in Equation 10, given that both the actuator and the sensor shows hysteresis in their outputs. The kinematics model of the actuator defines the relation of the input u_t to the state x_t of the actuator, and

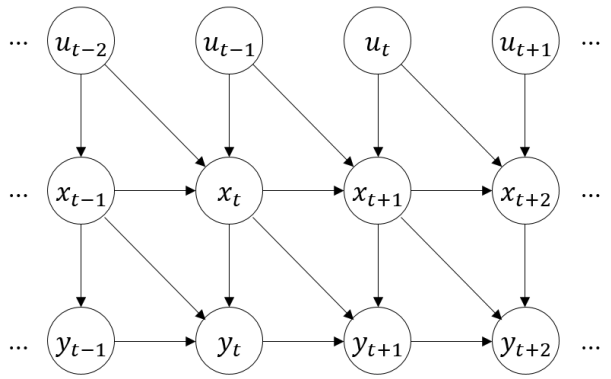

Fig. 4. Forward kinematics and its sensor relation can be represented by using a Bayesian network. Inputs u_t are under the constraint of Assumption 2. The goal of filtering is to estimate x_t given all the past inputs u_* and outputs y_*.

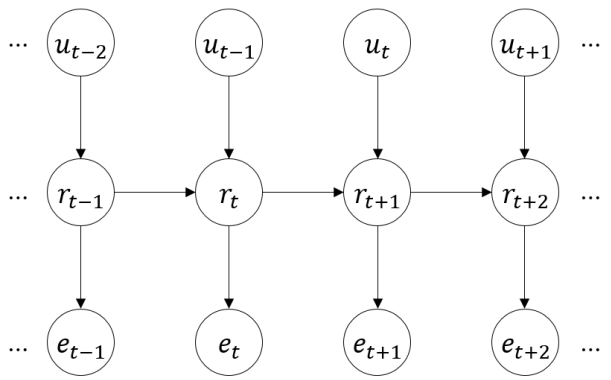

Fig. 5. Forward kinematics and its sensor relation can be represented by using a Bayesian network, but with introduction of the new state and the evidence variable, r and e, respectively.

the sensor model defines the relation of the state x_t to the sensor output y_t. The mathematical representations of the two relations are

$$\begin{aligned} x_{t+1} &\sim p(\cdot \mid x_t, u_t, u_{t-1}), \\ y_t &\sim p(\cdot \mid y_{t-1}, x_t, x_{t-1}) \end{aligned} \tag{12}$$

by reflecting Equation 10.

We can visualize these relations in a DBN, as shown in Fig. 4. This network employs the assumption of a first-order Markov chain, where each node of the Bayesian network depends only on the previous nodes, not the nodes earlier than the previous ones. Then, the joint distribution $p(\tau) := p(u_{0:T-1}, x_{0:T}, y_{0:T})$ can be expressed as:

$$\begin{aligned} p(\tau) = p(x_0)p(y_0 \mid x_0)p(x_1 \mid x_0, u_0) \prod_{i=0}^{T-1} p(u_i) \cdots \\ \prod_{i=2}^{T} p(x_i \mid x_{i-1}, u_{i-1}, u_{i-2}) \prod_{i=1}^{T} p(y_i \mid y_{i-1}, x_i, x_{i-1}). \end{aligned} \tag{13}$$

All the mathematical tools for modeling a soft robot using probabilistic representation are now ready. Obtaining each probability density function is data-driven. In other words, we operate the system with random inputs, collect data, and then approximate the probability density functions through deep learning. Two different methods of modeling and filtering were considered in this study, and they will be introduced in the next two sections. In Section IV, we first study a case in which the uncertainty is limited to Gaussian and model the relation using GPR. We then estimate the states by applying an EKF. In Section V, we study a general probability distribution in which the uncertainty is non-Gaussian. In this case, VaI is used as a model to express a general histogram, and a PF to estimate the state.

IV. Modeling and Filtering for Gaussian-class Uncertainty

We consider a special case in which uncertainties both in the actuator and the sensor are Gaussian distributed. Then, we can rearrange the variables in Equation 12 into new state variables r_t and e_t, which are defined as

$$\begin{aligned} \begin{bmatrix} r_{1,t+1} \\ r_{2,t+1} \\ r_{3,t+1} \\ r_{4,t+1} \end{bmatrix} &= \begin{bmatrix} u_t \\ p(r_{2,t+1} | r_{1,t}, r_{2,t}, u_t) \\ r_{2,t} \\ p(r_{4,t+1} | r_{2,t}, r_{3,t}, r_{4,t}) \end{bmatrix}. \\ e_t &= r_{4,t} \end{aligned} \tag{14}$$

This leads to another DBN, shown in Fig. 5, still representing the same system as Equation 12. The complex DBN in Fig. 4 has been significantly reduced to what we generally call a dynamics model with an input and an output. Then, our new actuator model and the sensor model are re-defined as:

$$\begin{aligned} r_{t+1} &\sim p(\cdot \mid r_t, u_t), \\ e_t &\sim p(\cdot \mid r_t). \end{aligned} \tag{15}$$

A benefit of this new reduced model is the brevity in the filtering equation as following:

$$\begin{aligned} &p(r_{t+1} | e_{1:t+1}, u_{0:t}) \\ &= \alpha p(e_{t+1} | r_{t+1}, e_{1:t}, u_{0:t}) p(r_{t+1} | e_{1:t}, u_{0:t}) \\ &= \alpha p(e_{t+1} | r_{t+1}, u_{0:t}) p(r_{t+1} | e_{1:t}, u_{0:t}) \\ &= \alpha p(e_{t+1} | r_{t+1}) \int_{\mathcal{R}} p(r_{t+1} | r_t, e_{1:t}, u_{0:t}) p(r_t | e_{1:t}, u_{0:t}) \, dr_t \\ &= \alpha \underbrace{p(e_{t+1} | r_{t+1})}_{\text{Sensor Model}} \int_{\mathcal{R}} \underbrace{p(r_{t+1} | r_t, u_t)}_{\text{Transition Model}} \underbrace{p(r_t | e_{1:t}, u_{0:t-1})}_{\text{Filtering}} \, dr_t, \end{aligned} \tag{16}$$

where $\mathcal{R}$ is a reachable set of r_t, and α is a normalization coefficient.

If the current distribution, given with all the evidence $p(r_t \mid e_{1:t})$ and the actuator model $p(r_{t+1} \mid r_t, u_t)$, is linear Gaussian, the prediction given by

$$\underbrace{p(r_{t+1} \mid e_{1:t}, u_{0:t})}_{\text{Prediction}} = \int_{\mathcal{R}} p(r_{t+1} \mid r_t, u_t) p(r_t \mid e_{1:t}, u_{0:t-1}) \, dr_t \tag{17}$$

is also a Gaussian distribution. Likewise, if the prediction is Gaussian and the sensor model $p(e_t \mid r_t)$ is linear Gaussian, the filtering result

$$p(r_{t+1} \mid e_{1:t+1}, u_{0:t}) = \alpha p(e_{t+1} | r_{t+1}) p(r_{t+1} | e_{1:t}, u_{0:t}) \tag{18}$$

is Gaussian as well. Therefore, the assumption that the actuator and the sensor models to be (linear) Gaussian distributions can simplify the filtering algorithm, which is called an EKF. To achieve this, the actuator and the sensor relations are modeled with GPR to express a nonlinear function with a Gaussian-distributed variance.

The most important step in the filtering is linearization to use an EKF. By implementing derivatives of the GPR result in each corresponding state and linearization, we obtain the following linearized system equation in a matrix form:

$$\begin{bmatrix} r_{1,t+1} \\ r_{2,t+1} \\ r_{3,t+1} \\ r_{4,t+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ A_{21,t} & A_{22,t} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & A_{42,t} & A_{43,t} & A_{44,t} \end{bmatrix} \begin{bmatrix} r_{1,t} \\ r_{2,t} \\ r_{3,t} \\ r_{4,t} \end{bmatrix} + \begin{bmatrix} 1 \\ B_{2,t} \\ 0 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} 0 \\ w_{2,t} \\ 0 \\ 0 \end{bmatrix}, \tag{19}$$

$$e_t = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,t} \\ r_{2,t} \\ r_{3,t} \\ r_{4,t} \end{bmatrix} + v_t \tag{20}$$

where $A_{*,t}$ and $B_{2,t}$ are the first derivatives of GPR modeling of Equation 14 by their corresponding states. Here, w_2 and v are model uncertainties induced from GPR modeling, and they are originated from the variance level of the GPR output. This variance value contains the modeling error of $p(r_{2,t+1} \mid r_{1,t}, r_{2,t}, u_t)$ and $p(r_{4,t+1} \mid r_{2,t}, r_{3,t}, r_{4,t})$. Now, we can infer that the GPR variance values show us how inaccurate the model is in a certain region so that the EKF can compensate the error by its best filtering, even if there exists a modeling error from our proposition. The uncertainty level of $r_{4,t}$ induced by the probability distribution is moved to the sensor model in order to perform the EKF.

V. Modeling and Filtering for non-Gaussian Uncertainty

We now consider a case in which uncertainty can be non-Gaussian distributed. In general, the probability density equation of a given probability box thus can have any complex form, such as a multi-modal distribution (i.e., the posterior distribution of y_{t+1} shown in Fig. 3). However, the global function approximator (e.g., ANN) of the probability density equation can only represent a limited number of probability classes (e.g., Gaussian or Bernoulli distributions). Since VaI with a new latent variable is useful in complex statistical models, we introduce a latent variable z following a well-known distribution, such as a normal distribution, as shown in Fig. 6-(a), and a structured conditional distribution $p(y_t \mid x_t, x_{t-1}, y_{t-1}, z)$ parameterized by θ [48]. This parameterized distribution can be user-defined, and we choose a Gaussian distribution in this paper. By marginalization, the integral of the two well-known distributions can produce a more complex

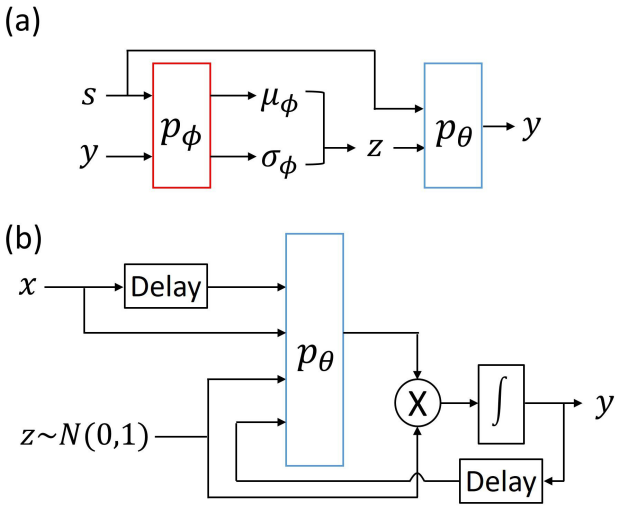

Fig. 6. (a) Given input $s_t := [x_t, x_{t-1}, y_{t-1}]^\top$ and target y_t, encoder network parameterized by ϕ and decoder network parameterized by θ can be trained by maximizing ELBO equation. (b) In real-time testing, encoder network is removed and latent feature z is replaced by a pre-defined distribution.

probability density function as

$$p(y_t \mid x_t, x_{t-1}, y_{t-1}) = \int_z \underbrace{p_\theta(y_t \mid x_t, x_{t-1}, y_{t-1}, z)}_{\text{Gaussian distribution}} p(z \mid x_t, x_{t-1}, y_{t-1})\, dz. \tag{21}$$

If the probability distribution is trained by the maximum likelihood optimization, we can make the probability distribution $p(z \mid x_t, x_{t-1}, y_{t-1})$ akin to the prior $p(z)$, which can be our choice, a normal distribution. A fully trained latent variable distribution is then

$$p(y_t \mid x_t, x_{t-1}, y_{t-1}) \approx \int_z \underbrace{p_\theta(y_t \mid x_t, x_{t-1}, y_{t-1}, z)}_{\text{Gaussian distribution}} \underbrace{p(z)}_{\mathcal{N}(0,1)}\, dz. \tag{22}$$

Training this latent variable model uses general VaI learning. For simplicity of notation, let $s_t := [x_t, x_{t-1}, y_{t-1}]^\top$. Then, the ELBO of the log likelihood $\log p(y_i \mid s_i)$ is defined as [48]

$$\log p(y_i \mid s_i) \geq \underbrace{\mathcal{L}_i}_{\text{ELBO}} \tag{23}$$

with the ELBO of the i^{th} data given as

$$\begin{aligned} \mathcal{L}_i &:= E_{z\sim p_\phi(z|s_i,y_i)}[\log p_\theta(y_i \mid s_i, z) + \log p(z \mid s_i)] \\ &\quad + \mathcal{H}(p_\phi(z \mid s_i, y_i)) \\ &= E_{z\sim p_\phi}[\log p(y_i \mid s_i, z)] - \mathcal{D}_{KL}(p_\phi(z \mid s_i, y_i) \parallel p(z)) \end{aligned} \tag{24}$$

where $E[\cdot]$ and $\mathcal{H}$ are the expectation and the entropy of the probability distribution, respectively. By applying gradient ascent to the ELBO cost function, we can now obtain the maximum likelihood solution for the network parameterized by θ and ϕ (Fig. 6-(a)). At the time of testing, we should sample z from $z \sim p(z)$ and infer y from $y \sim p(y \mid s_i, z)$ and then calculate the marginal equation (Fig. 6-(b), Equation 22).

Algorithm 1 Particle Filter for DBN

Initialize parameter settings B and W.
N is the number of samples to be queued.
$B_0 \leftarrow$ sample N particles from the prior $p(r_0)$
for each timestep t **do**
  Get the new input u_t and current evidence y_t
  for $i = 1$ to N **do**
    $B_t[i] \leftarrow$ sample x_{t+1} from $p(x_{t+1} \mid B_{t-1}[i], u_t, u_{t-1})$
    $W_t[i] \leftarrow$ likelihood of $p(y_t \mid B_t[i], B_{t-1}[i], y_{t-1})$
  end
  Resample B_t to generate a new population using W_t, in which each sample is selected from the current B_t.
end

Now, we can derive the filtering equation for the generally distributed model. First, the filtering equation can be divided into two probability distributions using the Bayes' theorem as following:

$$\begin{aligned} &p(x_t \mid u_{0:t-1}, y_{0:t}) \\ &= \alpha p(y_t \mid x_t, u_{0:t-1}, y_{0:t-1}) \underbrace{p(x_t \mid u_{0:t-1}, y_{0:t-1})}_{\text{Prediction } \beta(x_t)} \end{aligned} \tag{25}$$

where α is a normalization coefficient.

Second, the prediction probability $\beta(x_t)$ can be derived as

$$\begin{aligned} \beta(x_t) &= p(x_t | u_{0:t-1}, y_{0:t-1}) \\ &= \int_{\mathcal{X}} p(x_t | x_{t-1}, u_{0:t-1}, y_{0:t-1}) p(x_{t-1} | u_{0:t-1}, y_{0:t-1})\, dx_{t-1} \\ &= \int_{\mathcal{X}} \underbrace{p(x_t | x_{t-1}, u_{t-1}, u_{t-2})}_{\text{Transition model}} \underbrace{p(x_{t-1} | u_{0:t-2}, y_{0:t-1})}_{\text{Filtering}}\, dx_{t-1} \end{aligned} \tag{26}$$

and the measurement update policy $p(y_t | x_t, u_{0:t-1}, y_{t-1})$ as

$$\begin{aligned} &p(y_t | x_t, u_{0:t-1}, y_{0:t-1}) \\ &= \int_{\mathcal{X}} p(y_t | x_t, x_{t-1}, u_{0:t-1}, y_{0:t-1}) \\ &\qquad p(x_{t-1} | x_t, u_{0:t-1}, y_{0:t-1})\, dx_{t-1} \\ &= \int_{\mathcal{X}} \alpha_1 p(y_t | x_t, x_{t-1}, y_{t-1}) \\ &\qquad p(x_{t-1}, x_t | u_{0:t-1}, y_{0:t-1})\, dx_{t-1} \\ &= \int_{\mathcal{X}} \alpha_1 \underbrace{p(y_t | x_t, x_{t-1}, y_{t-1})}_{\text{Sensor model}} \underbrace{p(x_t | x_{t-1}, u_{t-1}, u_{t-2})}_{\text{Transition model}} \\ &\qquad \underbrace{p(x_{t-1} | u_{0:t-2}, y_{0:t-1})}_{\text{Filtering}}\, dx_{t-1} \end{aligned} \tag{27}$$

where $\mathcal{X}$ is the reachable set of x_{t-1}, and α_1 is a normalization coefficient. Although derivation of the filtering is straightforward, it is difficult to implement the direct integration in real-time. We thus use a sampling-based filtering algorithm, called a PF.

The PF algorithm is given in Algorithm 1. Let $N_S(x_t \mid u_{0:t-1}, y_{0:t})$ be the number of samples occupying the state x_t given the evidence $y_{0:t}$ and the input $u_{0:t-1}$. Then, for a large number N, the following property holds:

$$\frac{1}{N} N_S(x_t \mid u_{0:t-1}, y_{0:t}) \approx p(x_t \mid u_{0:t-1}, y_{0:t}). \tag{28}$$

The first step is to sample N particles from the transition model and the given previously filtered sample:

$$\begin{aligned} &N_S(x_{t+1} \mid u_{0:t}, y_{0:t}) \\ &\quad = \sum_{x_t} p(x_{t+1} \mid x_t, u_t, u_{t-1}) N_S(x_t \mid u_{0:t-1}, y_{0:t}). \end{aligned} \tag{29}$$

The next step is to give a weight to each samples by the likelihood of the sensor model:

$$\begin{aligned} &W(x_{t+1} \mid u_{0:t}, y_{0:t+1}) \\ &\quad = p(y_{t+1} \mid y_t, x_{t+1}, x_t) N_S(x_{t+1} \mid u_{0:t}, y_{0:t}). \end{aligned} \tag{30}$$

Then, the following relation holds for the resampling step:

$$\begin{aligned} &\frac{1}{N} N_S(x_{t+1} \mid u_{0:t}, y_{0:t+1}) \\ &\quad = \alpha W(x_{t+1} \mid u_{0:t}, y_{0:t+1}) \\ &\quad = \alpha p(y_{t+1} \mid y_t, x_{t+1}, x_t) N_S(x_{t+1} \mid u_{0:t}, y_{0:t}) \\ &\quad = \alpha \underbrace{p(y_{t+1} \mid \cdots)}_{\text{Sensor Model}} \sum_{x_t} \underbrace{p(x_{t+1} \mid \cdots)}_{\text{Transition Model}} N_S(x_t \mid u_{0:t-1}, y_{0:t}) \\ &\quad \approx \alpha N p(y_{t+1} \mid \cdots) \sum_{x_t} p(x_{t+1} \mid \cdots) p(x_t \mid u_{0:t-1}, y_{0:t}) \\ &\quad = \hat{\alpha} p(y_{t+1} \mid \cdots) \sum_{x_t} p(x_{t+1} \mid \cdots) p(x_t \mid u_{0:t-1}, y_{0:t}) \\ &\quad = p(x_{t+1} \mid u_{0:t}, y_{0:t+1}) \end{aligned} \tag{31}$$

where α and $\hat{\alpha}$ are the normalization coefficients. This proves the consistency of the PF method for the DBN.

VI. Application

We validated the proposed modeling and its probabilistic inference method of filtering using a soft robotic system composed of a pneumatic bending actuator and an embedded proprioceptive soft strain sensor.

A. Materials and Fabrication

A PneuNet-type bending actuator [49], [50] was fabricated for experimentally validating the proposed modeling and filtering algorithms. For estimating the tip position of the actuator, a microfluidic soft strain sensor was embedded on the top side of the actuator. For fabrication, the extensible and the inextensible layers of the PneuNet actuator were separately molded first. Both layers were made from room-temperature-vulcanizing (RTV) silicone elastomer (Dragon-Skin 30, Smooth-On) by mixing parts A and B with a weight ratio of 1:1 using a centrifugal mixer (ARE, Thinky) and cured in a 60°C oven for 20 minutes. The inextensibility of the bottom layer was achieved by simply increasing the layer thickness of the same material in the prototype instead of adding a stiffer layer. The two layers were attached together by pouring uncured silicone in between, and cured for 20 minutes. For sensing, another layer of a microfluidic strain sensor with eutectic gallium-indium (eGaIn, Gallium 75.5% and Indium 24.5% by weight) was fabricated [51]. The sensing layer was made from

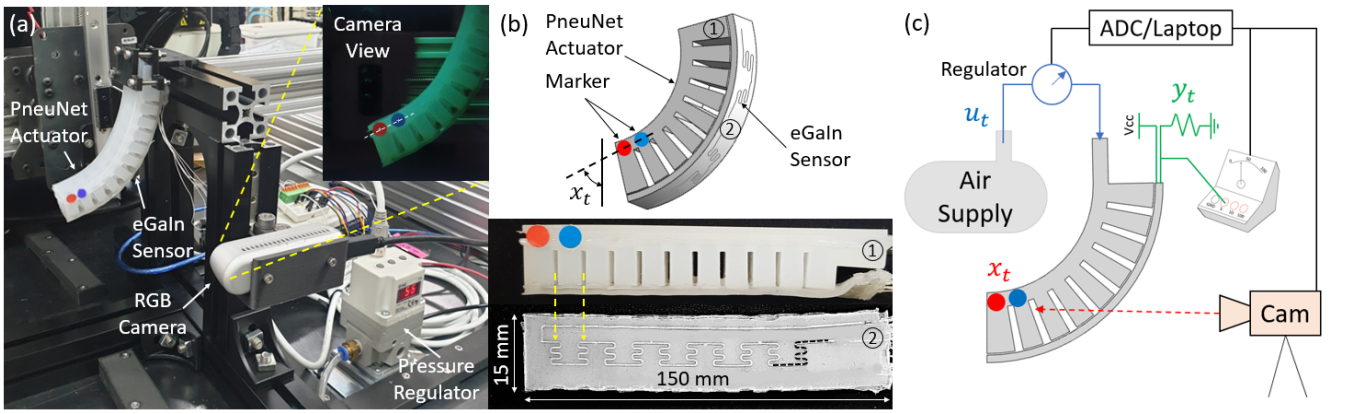

Fig. 7. (a) Experimental setup that consists of a pressure regulator, a PneuNet soft actuator with an embedded eGaIn soft sensor, and an RGB camera for tracking the motion of the actuator. The camera tracks the position of two markers to calculate the orientation. (b) Side and top views of the actuator with an embedded soft sensor. The camera collects the position of the markers and calculates the reference state x_t as an angle of the end-effector orientation relative to the vertical axis. The black dashed line in the bottom figure indicates the configuration of an eGaIn channel. The yellow dashed lines show the locations of the eGaIn channels aligned between air chambers. (c) Schematics of data acquisition setup for experiment. Compressed air is injected with predefined guide u_t, and the camera records the state x_t, and the voltmeter receives the sensor output y_t.

RTV silicone elastomer (Ecoflex 00-30, Smooth-on), whose stiffness was about eight times smaller than that of the actuator so that the performance of the actuator is not affected by the sensor. For fabrication of the sensor, uncured silicone was poured in a 3D-printed (Objet30 Pro, Stratasys) mold with a microchannel pattern and cured in the oven. The microchannel was then sealed by a thin flat silicone layer on top and then filled with eGaIn. Fabrication details for eGaIn soft sensors can be found in the previous work [51]. The complete sensing layer was finally attached to the top extensible layer of the PneuNet actuator by aligning the eGaIn channels with the empty spaces between air chambers to maximize the strain during actuation. When the chambers of the actuator are pressurized, the entire structure bends, and the sensing layer stretches, since it is placed off the neutral axis (Fig. 7-(b)).

B. Experiments

The actuator prototype was fixed on a test bench and connected to a pressure regulator (ITV2011-212BS5, SMC Corp.), then connected to an air reservoir that can supply compressed air up to 100 kPa. A red-green-blue (RGB) camera (LifeCam Studio, Microsoft) was installed in front of the actuator for recording the tip position of the actuator. The eGaIn soft sensor was connected to an external 50 Ω resistor with a 0.5 V power source. Then, an analog-to-digital converter (ADC, USB-6211, National Instruments) was installed between the eGaIn sensor and the external resistor as a voltage divider, and its output voltage was recorded. All the data were gathered and saved online, passing through the ADC, and finally to the laptop. A digital low-pass filter with a Gaussian kernel of dynamic bandwidth 0.1 was applied to the soft sensor output. The operating frequency of the overall system was 25 Hz, and all the data were processed through MATLAB (2019a, Mathworks) and Python. A MATLAB support package for the visual tracking and the Python Tensorflow framework (Tensorflow 1.0, with Graphics Processing Unit NVIDIA GTX 1050) were additionally installed for validation. The input u_t is the pressure value of the regulator, the state x_t is the bending angle of the actuator, and the output y_t is the sensor signal from the soft sensor. The experimental setup with the actuator prototype and its simplified schematic are shown in Fig. 7.

Before operating state estimation, the actuator model $p(x_{t+1} \mid x_t, u_t, u_{t-1})$ and the sensor model $p(y_t \mid y_{t-1}, x_t, x_{t-1})$ need to be trained. A dataset was collected by implementing a random sequence input to the actuator system. The training and validation sequence input consists of multiple inflation and deflation commands in the range of input pressures between 0 and 75 kPa. First, periodic actuation commands were given to the actuator with periods of 4,6,8 and 10 seconds. Second, inflation and deflation were made randomly every 3 seconds targeting a pressure level of 40, 20, 60, 40, and 75 kPa. These combinations were applied to the actuator five times in each command, and the final epoch was used as the validation data (Fig. 8-(a)). For variation in the training dataset, 1% random noise was applied to the training inputs. The overall volume of the training dataset was 28,000.

The training datasets satisfy the two conditions that we derived in the modeling section. First, the difference between two time-adjacent inputs should be bounded according to Assumption 2, in which its specific value of η was 0.018 for the actuator model and 1.128 for the sensor model. Second, each model should be locally Lipschitz by Assumption 3, and its Lipschitz constant L was 2.874 and 0.143 for the actuator kinematics and the sensor model, respectively.

Then, the actuator kinematics and the sensor dynamics were trained by two different types of learning methods: GPR and VaI. GPR was for the Gaussian assumption, and the VaI was for the general case.

The performance of each algorithm was evaluated by estimating the state given the test input signal, which consists of three parts: a triangle wave with varying peak amplitudes, a mixture of sine waves with different frequencies, and a sine wave with a perturbation offset (Tests 1, 2, and 3 in Fig. 8-(b), respectively). The test input signal was also applied to existing calibration and filtering methods. First, a simple polynomial

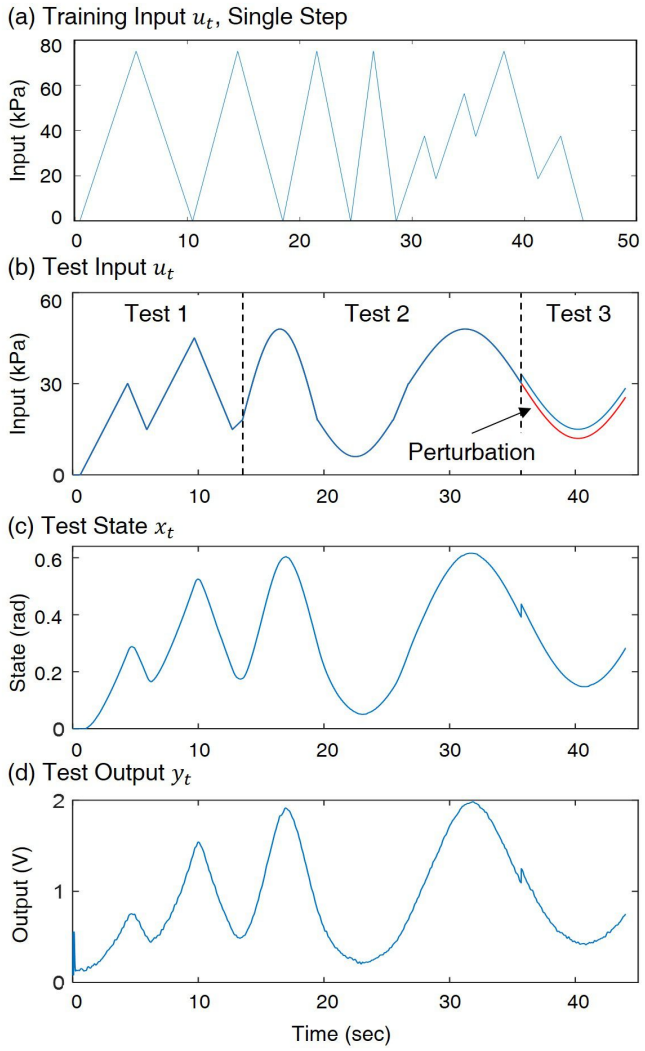

Fig. 8. Single loop of train and test data. (a) Train input of a single step. (b) Test input pressure to the soft actuator. (c) Actuator state as a curvature angle. (d) Output of the eGaIn sensor. The red curve in (b) indicates the nominal input. While the actual input was the blue line, the red colored line is given to the user.

fitting that maps between the input u_t and the state x_t was used. Second, a conventional state-space (SS) approach was used with an EKF. The actuator model $x_t = f(x_{t-1}, u_t)$ and the sensor model $y_t = h(x_t)$ was constructed with a GPR and an EKF was implemented to filter the signal. Third, an RNN was used to express the hysteresis, previously proposed by other groups [12], [14], [15]. Fourth, a GPR mapping from $\{u_t, u_{t-1}, x_{t-1}\}$ to x_t, a feedforward estimation of our proposed hysteresis mapping, was applied to compare the single-model estimation and the Bayes' filtering. The evaluation criteria for the test input is the normalized root-mean-squared error (nRMSE), which is normalization of the root-mean-squared difference of the reference state x_{ref} from the estimated state x_{est} via filtering, i.e.,

$$nRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{i,est} - x_{i,ref})^2}}{\max(x_{ref}) - \min(x_{ref})}. \tag{32}$$

Fig. 8 shows a single loop of training and test case, each representing the input, the state, and the output.

C. GPR and VaI Settings

The input pressure, the tip position of the actuator from the camera, and the sensor output were recorded at a sampling rate of 25 Hz. After gathering the training dataset, they were reclassified to train the actuator and the sensor models.

The GPR model training was first implemented by optimizing the target hyperparameters of the kernel function and the noise level. A Gaussian process is a collection of finite joint Gaussian-distribution random variables, where a real process $f(x)$ is given as

$$\begin{aligned} &f(x) \sim \mathcal{GP}(m(x), k(x, x')) \\ &m(x) = E[f(x)] \\ &k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \end{aligned} \tag{33}$$

where k is the kernel function. The posterior distribution, given the training data D, can be written as

$$\begin{aligned} &f \mid D \sim \mathcal{GP}(m_D(x), k_D(x, x')) \\ &m_D(x) = k(X, x)^T (K + \sigma_n^2)^{-1}(y - m) \\ &k_D(x, x') = k(x, x') - k(X, x)^T (K + \sigma_n^2)^{-1} k(X, x') \end{aligned} \tag{34}$$

where σ_n is the designated noise covariance, X is the training input dataset, and x is the test input. m_D and k_D are the mean and the covariance functions, respectively, when the dataset D is given. K is equivalent to $k(X, X)$, and $K(X, \cdot)$ is the kernel covariance matrix, given all the training input dataset X. Therefore, the estimated value Y' for the new input X' is the value obtained by plugging X' into the mean function of the input, and the uncertainty is the value assigned to the covariance function. Our goal is to maximize the log likelihood of the marginal given the constant basis, i.e.,

$$\begin{aligned} \log p(y \mid X) = &-\frac{1}{2} y^\top (K + \sigma_n^2 I)^{-1} y \\ &- \frac{1}{2} \log \mid K + \sigma_n^2 I \mid - \frac{n}{2} \log 2\pi. \end{aligned} \tag{35}$$

For the GPR setting, we used an automatic relevance determination (ARD) squared exponential function as a kernel with a constant basis. The optimization of Equation 35 was conducted by the quasi-Newton convex optimization algorithm.

Training a latent variable model was implemented next by optimizing the ELBO equation of the given dataset. The ELBO consists of two parts: a log likelihood of the decoder network (shown in Fig. 9, which generates the output x_{t+1} given the inputs $s_t := [x_t, u_{t-1}, u_t]$ and the latent variable z sampled from the normal distribution (i.e., $N(0, 1)$) and the Kullback-Leibler (KL) divergence between the encoder probability network and the nominal $p(z)$, as shown below:

$$\begin{aligned} \mathcal{L}_i = E_{z \sim p_{d,I}(z \mid s_{t,i}, x_{t+1,i})} [&\underbrace{\log p_{d,I}(x_{t+1} \mid s_{t,i}, z))}_{\text{Decoder Network Loss}} \\ &\underbrace{- \mathcal{D}_{KL}(p_{e,I}(z \mid s_{t,i}, x_{t+1,i}) \parallel p(z \mid s_{t,i}))}_{\text{Encoder Network Loss}}]. \end{aligned} \tag{36}$$

Maximization of the log probability of $p_{d,I}$, which is from the generative network, is supervised learning given the input $s_{t,i}$

TABLE I
RESULT OF THE VALIDATION EXPERIMENT IN AVERAGE (STANDARD DEVIATION)

nRMSE(%)	Comparison				DBN + Filtering	
	PolyFit	SS+EKF	RNN	GPR (DBN)	GPR+EKF	VaI+PF
Test 1	8.77 (0.03)	5.50 (0.04)	3.53 (0.18)	2.45 (0.01)	0.69 (0.04)	1.22 (0.05)
Test 2	8.39 (0.04)	4.68 (0.03)	2.57 (0.29)	1.34 (0.01)	0.67 (0.09)	0.83 (0.04)
Test 3	6.08 (0.02)	3.88 (0.06)	4.13 (0.11)	2.86 (0.02)	2.09 (0.17)	1.15 (0.07)
Overall	8.10 (0.03)	4.81 (0.05)	3.21 (0.18)	2.05 (0.01)	1.12 (0.14)	1.01 (0.02)

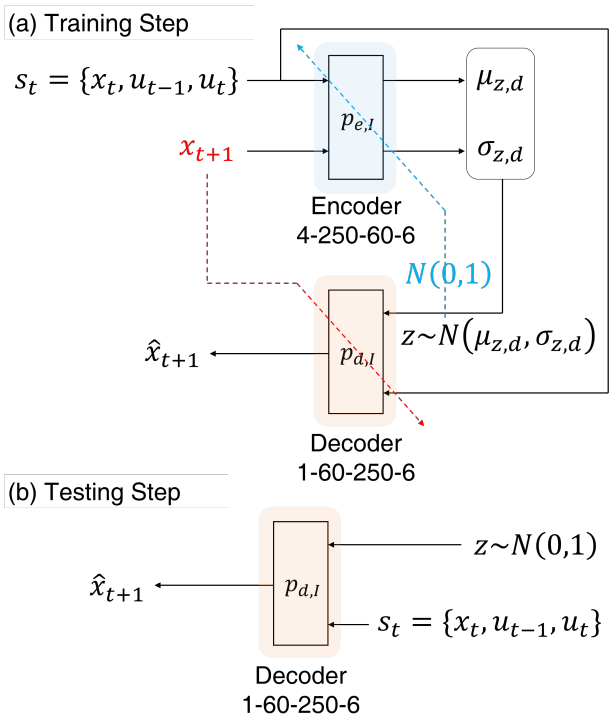

Fig. 9. (a) Training process of VaI for system dynamics model. The blue-shaded box is an encoder, which attempts to find the distribution of the latent feature and adjust weights to make the latent feature distributed to be normal distribution. The red-shaded box is a decoder, which gathers the training input (s_t) and the sampled feature z and returns the estimated training output (x_{t+1}) and adjust weights for appropriate prediction. The network consists of two hidden layers having 250 and 60 hidden features, respectively. (b) During the test time, the feature z is sampled from the normal distribution since the distribution of z is trained to be the same as $\mathcal{N}(0, 1)$ due to Equation 3.

and the output $x_{t+1,i}$, where minimizing the KL divergence is an encoding process with the latent variable similar to the normal distribution. A reparameterization trick was used to sample the latent variable z from the output of the inference network. The dimension of the latent variable was 10, which was normal-distributed. The graphical representation of the training dynamics network is shown in Fig. 9. Both the actuator and the sensor networks were trained using minibatch gradient descent (batch size = 256) in Adam optimizer (i.e., a stochastic gradient descent method capable of dynamic gradient rescaling and bias correction [52]) for 2,500 epochs with checking its performance of the validation dataset, and conditionally activate early stopping to avoid overfitting. The network consists of two hidden layers, in which the hidden units were 250 and 60, respectively, each followed by a rectified linear unit (ReLU) activation layer. The network's base learning rate was 10^{-4}, and it decayed every 100 epochs with a ratio of 0.99. All the hyperparameter settings for neural networks were determined by the brute force (i.e., attempt to train with various combinations of hyperparameters and choose the best one).

The same procedure was also applied to train the sensor model, in which the input and the output were $\hat{s}_t := [x_t, y_t, x_{t+1}]$ and y_{t+1}, respectively.

D. Results

Fig. 10 shows the test result. The test input signal consists of a linear (Test 1) and a sinusoidal (Test 2) inputs, and the input signal was perturbed with an offset of 8% (equal to 5 kPa, which is the maximum offset level that does not exceed the variance level of the state refers to a non-perturbed results) from the current value. Figs. 10-(a) through 10-(f) show the output states estimated by six different methods and their corresponding hysteresis curves. The nRMSE values, summarized in Table I, indicate the error levels of the six cases.

The result shows that the two proposed methods, GPR with an EKF and VaI with a PF, achieved better performances than the results of the comparisons. One of the major benefits of the DBN with filtering is that both the actuator and the sensor models are used for estimation with real-time sensor feedback, while the curve fitting and the RNN use only either of them with a forward model. Moreover, rather than using only the forward actuator model (Fig. 10-(d)), the filtering process combined with an EKF or a PF using the sensor model improved the accuracy. Finally, our approach showed improved result compared to the conventional SS model with the EKF. There are three specific reasons that the DBN with filtering outperformed the others. First, the curve fitting did not model the hysteretic behavior of the system; it only mapped the input to the current output state without including any property of history. Second, although the RNN was able to model the hysteresis, sudden unwanted peaks appeared in the estimation mostly at the boundaries where the input signal changed its direction, as shown in Fig. 10-(c). Third, use of both the actuator and the sensor models helped decreasing the error levels using GPR and VaI. Especially, introducing previous input u_{t-1} in the actuator model and tuple $\{x_{t-1}, y_{t-1}\}$ in the sensor model efficiently expressed the hysteresis than conventional SS filtering method. Finally, the DBN reflected the uncertainty of the system in the model when used with filtering as demonstrated with the perturbation offset. Even though the perturbation might disturb the actuator model, the sensor model was able to cancel the effect with filtering.

Compared to the forward DBN model with GPR, the accuracy was improved in GPR and VaI when filtering tech-

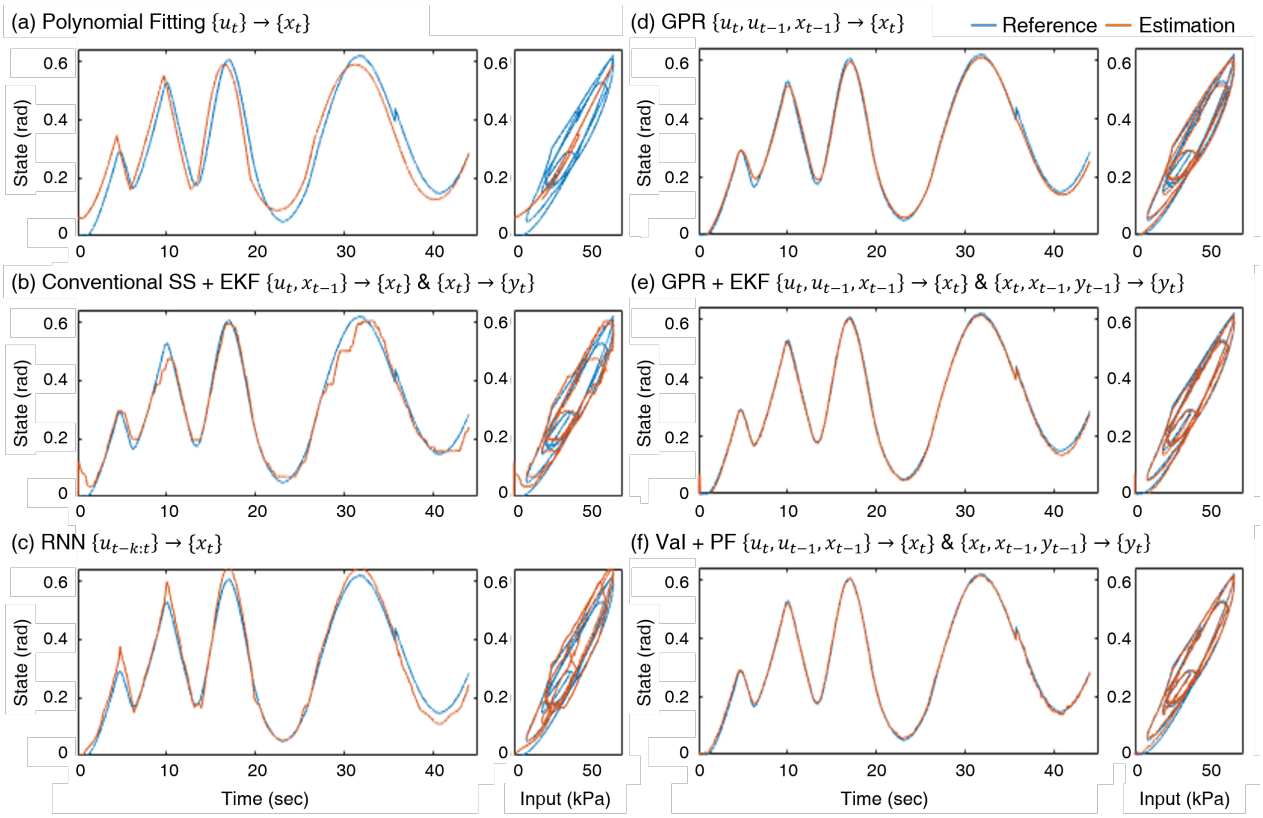

Fig. 10. Results showing the representative state signals from five tests, estimated by five different methods. (a) Polynomial fitting that maps directly from u_t (input) to x_t (state), (b) comparison study using conventional state-space approach with an EKF (c) RNN that maps the past inputs $u_{t-k:t}$ to state x_t, (d) GPR that uses our hysteresis mapping $\{u_t, u_{t-1}, x_{t-1}\}$ to x_t, (e) GPR model with EKF, and (f) VaI model with PF.

niques, such as an EKF and a PF were used, since they successfully followed the peak values, as shown in Figs. 10-(d) through 10-(f). In Test 3, both filtering processes were able to capture the offset in the input and the error in the actuator model was recovered by the filtering. GPR-EKF showed a better performance in Tests 1 and 2, while VaI-PF showed better performance in Test 3. As the PF uses the Monte-Carlo sampling approach, its filtering may not be absolutely optimal. Nevertheless, VaI-PF was more powerful, capturing the perturbations by the resampling process.

The probability box model of the hysteresis function in our experimental setting was similar to a Gaussian model, as shown in Fig. 11 which is the probability distribution of the curvature of the actuator in the histogram. The sampled values were smoothed by a nonparametric regression technique, showing that the actual model would be similar to a Gaussian distribution, with a weak evidence of a multi-modal behavior. This means that GPR-EKF can be an alternative to VaI-PF when the hysteresis level is not dominant.

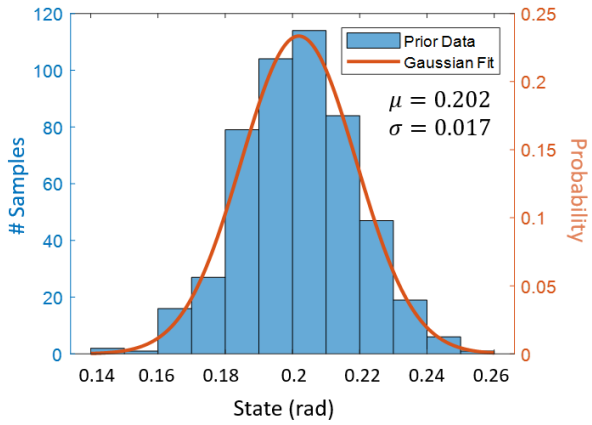

Fig. 11. Histogram of the sampled population of the variational inference in the dynamics model with the input signal from 20 second in Fig. 10-(f).

Finally, the effect of the PF is shown in Fig. 12. The PF procedure consists of three steps: sampling from the dynamics model $p(x_{t+1} \mid x_t, u_t, u_{t-1})$, weighting from the sensor model $p(y_t \mid x_t, x_{t-1}, y_{t-1})$, and resampling using the previous results, as shown in Algorithm 1. In this procedure, the tip position x_{t+1} is first sampled from the previous filtering data and the new input pressure data, as shown in Fig. 12-(a). Given the sensor output value, each sample's weight was calculated next proportional to the likelihood of the sensor data of the corresponding samples, as visualized in Fig. 12-(b). The weight results show that the values between 0.49 and 0.495 yield the highest probability of 0.43. The samples are then resampled by reflecting the weights. The result is shown in Fig. 12-(c). The posterior value after filtering has recovered its reference value correctly.

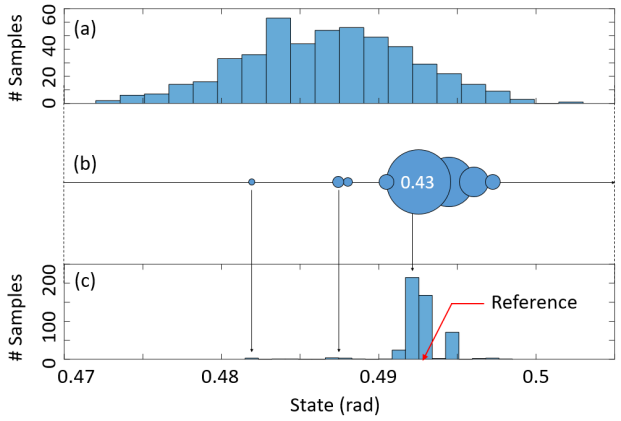

Fig. 12. Resampling procedure of PF with testing signal from 35.5 second in Fig. 10-(f). (a) Result of the generative network of the dynamics model (B_t in Algorithm 1) in histogram. (b) Corresponding likelihood of the sensor model (W_t in Algorithm 1). (c) Resampling result in histogram.

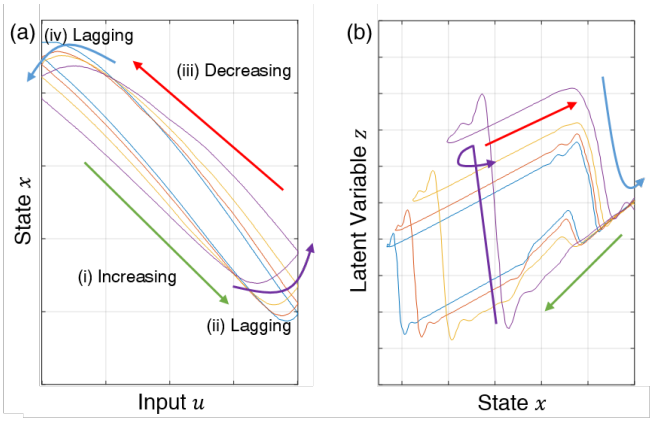

Fig. 13. (a) Hysteresis curve between input u and state x. (b) Temporal flows of one of the hidden variables z from the encoder part of the dynamics model $p(x_{t+1} \mid x_t, u_{t-1}, u_t)$.

VII. Discussion

This section provides analysis on the trained latent variables to check if the effect of the hysteresis has been effectively modeled in our approach. In addition, issues in filtering, stability for implementation, and expansion to multiple inputs and outputs are discussed.

A. Temporal Flow and Analysis

Fig. 13 shows the temporal flow of the first latent variable z among three of the trained latent variables. Different colors in Fig. 13-(a) indicate different input speeds, linearly increasing or decreasing, and the labels (i)-(iv) indicate the four parts in the hysteresis curve. Fig. 13-(b) shows that the magnitude of the latent variable was capable of classifying the four phases in the hysteresis curve. The latent variable also catched the increasing speed of the input, where the system under the highest increasing input had the largest magnitude. As a result, the latent variable clearly classified the property of the directionality in the hysteresis loop as well as the rate-dependency.

B. Real Time Filtering

It is important to use the state estimator on-line to perform a complex task, such as feedback control. However, there are issues in on-line implementation of the proposed method.

When we model the system in a discrete-time domain, the sampling frequency of the operating system is critical, and we need to use the same sampling frequency for gathering training dataset and also for real-time state estimation.

Due to the delay in the microcontroller and the computing time of the software, the sampling frequency for training might be lower than that for testing.

In GPR-EKF, GPR prediction includes vector calculations, and the EKF needs a matrix inversion process. In VaI-PF, VaI processes need iterations of sampling and calculation. The average time consumed for real-time filtering for a single step was 0.028 s and 0.039 s for GPR-EKF and VaI-PF, respectively. As a result, we need to set a fixed time delay when gathering the training dataset to match the sampling frequencies for both training and testing.

C. Stability

The stability of the filtering algorithm is also an issue when implementing the DBN to an actual robot system. For example, let's assume a situation of a large perturbation (i.e., a perturbation that exceeds the variance level of the trained model) in which a new input or output state is given in a form that has never been trained before. In this case, an ANN that especially uses the Adam optimizer may return strange output values that do not match the real behavior of the robot. This may induce divergence of the filtered values, which can be problematic to the system. Therefore, the training dataset should contain possible perturbation or noise issues that could occur during testing. In this case, an advanced exploration techniques, such as entropy adaptation, sampling, or bootstrapping, may be useful to improve the quality of the training dataset additionally [53], [54].

D. Expansion to Multi-Input Multi-Output Case

In our application, we showed the feasibility of implementing filtering techniques to a SISO soft system. If the number of actuators increases, the first thing to consider is the state observability. Since the proprioceptive feedback of an actuator is provided in general by the corresponding sensor integrated in the actuator, the solution is to make a neural network for each actuator-sensor combination. However, if the system is not fully-observable, state estimation is not achievable.

If the system is observable, there are two ways of collecting data and estimating the state. First, if the data is individually collected in the actuator level (i.e., collect individual input-state-output tuples), the task state can be determined by concatenation of each internal state in which the whole model shows an additive behavior. Second, if the data is globally collected in the system level, the global input dimension should be three times the number of the actuators. This can be done by the previous data-driven work and applicable to our setting [10], [14].

VIII. Conclusion

We proposed a method of deriving probabilistic models for soft robot systems in a DBN. Our approach assumes that the distribution of the hysteresis output is dependent on its current and previous inputs as well as on the previous output. This concept was expanded to the actuator and the sensor models in a soft robot system, and a Bayesian network was established. The real-world application results supported the reliability of the proposed algorithm, and the Gaussian modeling and the latent variable modeling techniques were used for constructing and filtering the probability box model. This method is beneficial to the field of soft robotics in three folds. First, we can analyze hysteresis by using only three inputs to determine the next state, which was not possible with previous methods based on machine learning. Second, our method provides information on the uncertainty level that can be used for safety verification or stochastic control. Finally, we used both the kinematic model of the actuator and the proprioceptive sensor model to improve the performance of state estimation of the soft system, which are both data-driven. In this case, we have two options, GPR-EKF and VaI-PF, depending on the conditions of the target system. If nonlinearity and hysteresis are small or limited, we can use GPR-EKF for quick learning and filtering. Otherwise, VaI-PF is suitable for harsh conditions in which VaI can model a more complex structure, and a PF can deal with high nonlinearity. Introducing a probabilistic approach to modeling of the behavior of a soft robot may circumvent ramifications raised in modeling based on deep learning. We believe that this study will benefit researchers who are interested in controlling or estimating the states of a soft system where hysteresis and uncertainty are major limiting factors in real-world implementation.

Future work will be focused on designing a control system of a soft robot using the proposed DBN and also considering the multi-input multi-output (MIMO) systems. Control policies will be added to the DBN, and its structure will be slightly changed. By introducing the stochastic control theory in the sense of reinforcement learning (RL), we can develop optimal control strategies while guaranteeing that the state of the robot system is sub-optimal by the EKF or the PF process in terms of a Markov decision process with the hysteresis setting [55]. In a more complex system, collecting datasets may be a difficult or tedious process, and thus a hybrid RL method, such as guided policy search, may also be considered [56], [57]. Since the Gaussian process dynamics of stochastic states (x_t) and inputs (u_t) may lead to non-Gaussian state prediction (x_{x+1}) when a Gaussian model is used, advanced techniques, such as moment matching, is prerequisite [58]. Finally, to be ready for mass production of soft robots or soft sensors, calibration of those systems based on transfer learning will be useful for modeling the actuator and the sensor models [8].

Acknowledgement

The authors thank to Mr. Junghan Kwon for his technical support for the experimental setups and feedback on the theory, and Mr. Myungjun Jeon for his support on illustrations.

References

[1] J. Tapia, E. Knoop, M. Mutný, M. A. Otaduy, and M. Bächer, "Makesense: Automated sensor design for proprioceptive soft robots," *Soft Rob.*, 2019.

[2] M. A. Mcevoy and N. Correll, "Shape-changing materials using variable stiffness and distributed control," *Soft Rob.*, vol. 5, no. 6, pp. 737–747, 2018.

[3] C. Duriez and T. Bieze, "Soft robot modeling, simulation and control in real-time," *Soft Robotics: Trends, Applications and Challenges*, vol. 17, pp. 103–109, 2017.

[4] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[5] C. Schumacher, E. Knoop, and M. Bächer, "Simulation-ready characterization of soft robotic materials," *IEEE Rob. Autom. Lett.*, vol. 5, no. 3, pp. 3775–3782, 2020.

[6] P. Polygerinos, N. Correll, S. A. Morin, B. Mosadegh, C. D. Onal, K. Petersen, M. Cianchetti, M. T. Tolley, and R. F. Shepherd, "Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human-robot interaction," *Adv. Eng. Mater.*, vol. 19, no. 12, p. 1700016, 2017.

[7] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Sci. Rob.*, vol. 4, no. 26, p. eaav1488, 2019.

[8] D. Kim, J. Kwon, B. J. Jeon, and Y.-L. Park, "Adaptive calibration of soft sensors using optimal transportation transfer learning for mass production and long-term usage," *Adv. Intell. Syst.*, p. 1900178, 2020.

[9] G.-Z. Yang, "Robot learning - Beyond imitation," *Sci. Rob.*, vol. 5, no. 41, p. eaaw3520, 2019.

[10] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans. Rob.*, vol. 35, no. 1, pp. 124–134, 2018.

[11] F. Reinhart and J. J. Steil, "Hybrid mechanical and data-driven modeling improves inverse kinematic control of a soft robot," *Procedia Technol.*, vol. 26, 2016.

[12] T. Hellebrekers, N. Chang, K. Chin, M. Ford, O. Kroemer, and C. Majidi, "Soft magnetic tactile skin for continuous force and location estimation using neural networks," *IEEE Rob. Autom. Lett.*, 2020.

[13] S. H. Kim, Y. Kwon, K. Kim, and Y. Cha, "Estimation of hand motion from piezoelectric soft sensor using deep recurrent network," *Appl. Sci.*, vol. 10, no. 6, p. 2194, 2020.

[14] R. L. Truby, C. Della Santina, and D. Rus, "Distributed proprioception of 3d configuration in soft, sensorized robots via deep learning," *IEEE Rob. Autom. Lett.*, vol. 5, no. 2, pp. 3299–3306, 2020.

[15] S. Han, T. Kim, D. Kim, Y.-L. Park, and S. Jo, "Use of deep learning for characterization of microfluidic soft sensors," *IEEE Rob. Autom. Lett.*, vol. 3, no. 2, pp. 873–880, 2018.

[16] B. Shih, D. Shah, J. Li, T. G. Thuruthel, Y.-L. Park, F. Iida, Z. Bao, R. Kramer-Bottiglio, and M. T. Tolley, "Electronic skins and machine learning for intelligent soft robots," *Sci. Rob.*, vol. 5, no. 41, p. eaaz9239, 2020.

[17] D. Kim and Y.-L. Park, "Contact localization and force estimation of soft tactile sensors using artificial intelligence," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Syst. (IROS2018)*, 2018, pp. 7480–7485.

[18] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A geometrically exact model for soft continuum robots: The finite element deformation space formulation," *Soft Rob.*, vol. 6, no. 6, pp. 790–811, 2019.

[19] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, "Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer," in *Proc. IEEE Int. Conf. Soft Rob. (RoboSoft2019)*, 2019, pp. 717–724.

[20] S. M. Mustaza, Y. Elsayed, C. Lekakou, C. Saaj, and J. Fras, "Dynamic modeling of fiber-reinforced soft manipulator: A visco-hyperelastic material-based continuum mechanics approach," *Soft Rob.*, vol. 6, no. 3, pp. 305–317, 2019.

[21] G.-Y. Gu, U. Gupta, J. Zhu, L.-M. Zhu, and X. Zhu, "Modeling of viscoelastic electromechanical behavior in a soft dielectric elastomer actuator," *IEEE Trans. Rob.*, vol. 33, no. 5, pp. 1263–1271, 2017.

[22] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Trans. Rob*, vol. 34, no. 6, pp. 1565–1576, 2018.

[23] N. N. Goldberg, X. Huang, C. Majidi, A. Novelia, O. M. O'Reilly, D. A. Paley, and W. L. Scott, "On planar discrete elastic rod models for the locomotion of soft robots," *Soft rob.*, vol. 6, no. 5, pp. 595–610, 2019.

[24] J. Till, V. Aloi, and C. Rucker, "Real-time dynamics of soft and continuum robots based on cosserat rod models," *Int. J. Rob. Res.*, vol. 38, no. 6, pp. 723–746, 2019.

[25] C. Della Santina and D. Rus, "Control oriented modeling of soft robots: the polynomial curvature case," *IEEE Rob. Autom. Lett.*, vol. 5, no. 2, pp. 290–298, 2019.

[26] M. Zeitz, "The extended luenberger observer for nonlinear systems," *Systems & Control Letters*, vol. 9, no. 2, pp. 149–156, 1987.

[27] R. J. Meinhold and N. D. Singpurwalla, "Understanding the kalman filter," *The American Statistician*, vol. 37, no. 2, pp. 123–127, 1983.

[28] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[29] S. Levine and V. Koltun, "Variational policy search via trajectory optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS2013)*, 2013, pp. 207–215.

[30] M. Titsias and N. D. Lawrence, "Bayesian gaussian process latent variable model," in *Proc. Int. Conf. Artif. Intell. Stat. (NeurIPS2010)*, 2010, pp. 844–851.

[31] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 2746–2754.

[32] K. Chin, T. Hellebrekers, and C. Majidi, "Machine learning for soft robotic sensing and control," *Adv. Intell. Syst.*, p. 1900171, 2020.

[33] B. K. Johnson, V. Sundaram, M. Naris, E. Acome, K. Ly, N. Correll, C. Keplinger, J. S. Humbert, and M. E. Rentschler, "Identification and control of a nonlinear soft actuator and sensor system," *IEEE Rob. Autom. Lett.*, vol. 5, no. 3, pp. 3783–3790, 2020.

[34] R. K. Katzschmann, C. Della Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, "Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model," in *Proc. IEEE Int. Conf. Soft Rob. (RoboSoft2019)*, 2019, pp. 454–461.

[35] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using koopman operator theory," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA2019)*, 2019, pp. 6244–6250.

[36] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: Trajectory tracking and interaction with the environment," *Int. J. Rob. Res.*, vol. 39, no. 4, pp. 490–513, 2020.

[37] C. Della Santina, A. Bicchi, and D. Rus, "On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control," *IEEE Rob. Autom. Lett.*, vol. 5, no. 2, pp. 1001–1008, 2020.

[38] C. Della Santina, R. L. Truby, and D. Rus, "Data–driven disturbance observers for estimating external forces on soft robots," *IEEE Rob. Autom. Lett.*, vol. 5, no. 4, pp. 5717–5724, 2020.

[39] M. T. Gillespie, C. M. Best, and M. D. Killpack, "Simultaneous position and stiffness control for an inflatable soft robot," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA2016)*, 2016, pp. 1095–1101.

[40] T. J. DiCiccio and B. Efron, "Bootstrap confidence intervals," *Statistical science*, pp. 189–212, 1996.

[41] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS2017)*, 2017, pp. 5574–5584.

[42] J. Jung, M. Park, D. Kim, and Y.-L. Park, "Optically sensorized elastomer air chamber for proprioceptive sensing of soft pneumatic actuators," *IEEE Rob. Autom. Lett.*, vol. 5, no. 2, pp. 2333–2340, 2020.

[43] Y. Gao, X. Fang, D. Tran, K. Ju, B. Qian, and J. Li, "Dielectric elastomer actuators based on stretchable and self-healable hydrogel electrodes," *R. Soc. Open Sci.*, vol. 6, no. 8, p. 182145, 2019.

[44] J.-B. Chossat, D. K. Chen, Y.-L. Park, and P. B. Shull, "Soft wearable skin-stretch device for haptic feedback using twisted and coiled polymer actuators," *IEEE Trans. Haptics*, vol. 12, no. 4, pp. 521–532, 2019.

[45] Y. Lee, M. Kim, Y. Lee, J. Kwon, Y.-L. Park, and D. Lee, "Wearable finger tracking and cutaneous haptic interface with soft sensors for multi-fingered virtual manipulation," *IEEE/ASME Trans. Mech.*, vol. 24, no. 1, pp. 67–77, 2018.

[46] V. Hassani, T. Tjahjowidodo, and T. N. Do, "A survey on hysteresis modeling, identification and control," *Mech. Syst. Signal Process.*, vol. 49, no. 1-2, pp. 209–233, 2014.

[47] I. D. Mayergoyz, *Mathematical models of hysteresis*. Springer Science & Business Media, 2012.

[48] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Am. Stati. Assoc.*, vol. 112, no. 518, pp. 859–877, 2017.

[49] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Adv. Funct. Mater.*, vol. 24, no. 15, pp. 2163–2170, 2014.

[50] M. Park, Y. Ohm, D. Kim, and Y.-L. Park, "Multi-material soft strain sensors with high gauge factors for proprioceptive sensing of soft bending actuators," in *Proc. IEEE Int. Conf. Soft Rob. (RoboSoft2019)*, 2019, pp. 384–390.

[51] Y.-L. Park, B.-R. Chen, and R. J. Wood, "Design and fabrication of soft artificial skin using embedded microchannels and liquid conductors," *IEEE Sens. J.*, vol. 12, no. 8, pp. 2711–2718, 2012.

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

[53] J. I. Kim, M. Hong, K. Lee, D. Kim, Y.-L. Park, and S. Oh, "Learning to walk a tripod mobile robot using nonlinear soft vibration actuators with entropy adaptive reinforcement learning," *IEEE Rob. Autom. Lett.*, vol. 5, no. 2, pp. 2317–2324, 2020.

[54] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS 2016)*, 2016, pp. 4026–4034.

[55] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[56] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Mach. Learn. (ICML2013)*, 2013, pp. 1–9.

[57] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. Int. Conf. Mach. Learn. (ICML2011)*, 2011, pp. 465–472.

[58] M. P. Deisenroth, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Proc. Rob. Sci. Syst.*, 2011, pp. 57–64.

DongWook Kim received the B.S. and M.S. degrees in Mechanical and Aerospace Engineering at Seoul National University, Seoul, Korea, in 2018 and 2020, respectively. He has been working toward a Ph.D. degree in Mechanical Engineering at Seoul National University. His research interests include data-driven control systems of soft robots, machine intelligence, and nonlinear system identifications.

Myungsun Park received the B.S. degree in Mechanical and Aerospace Engineering at Seoul National University, Seoul, Korea, in 2018. She has been working toward a Ph.D. degree in Mechanical Engineering at Seoul National University. Her research interests include identification, designing and controlling soft robot systems.

Yong-Lae Park received the M.S. and Ph.D. degrees in Mechanical Engineering from Stanford University, Stanford, CA, USA, in 2005 and 2010, respectively. He is currently an Associate Professor in the Department of Mechanical Engineering at Seoul National University, Seoul, Korea. Prior to joining SNU, he was an Assistant Professor in the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, USA (2013-2017) and a Technology Development Fellow in the Wyss Institute for Biologically Inspired Engineering at Harvard University, Cambridge, MA, USA (2010-2013). His research interests include soft robots, artificial skin sensors and muscle actuators, and soft wearable robots.