Nurali Virani[1]
Department of Mechanical and Nuclear Engineering,
The Pennsylvania State University,
University Park,
State College, PA 16802
e-mail: nurali.virani88@gmail.com

Devesh K. Jha[2]
Department of Mechanical and Nuclear Engineering,
The Pennsylvania State University,
University Park,
State College, PA 16802
e-mail: devesh.dkj@gmail.com

Zhenyuan Yuan
Department of Electrical Engineering,
The Pennsylvania State University,
University Park,
State College, PA 16802
e-mail: zqy5086@psu.edu

Ishana Shekhawat
Department of Mechanical and Nuclear Engineering,
The Pennsylvania State University,
University Park,
State College, PA 16802
e-mail: ibs5048@psu.edu

Asok Ray
Fellow ASME
Department of Mechanical and Nuclear Engineering;
Department of Electrical Engineering,
The Pennsylvania State University,
University Park,
State College, PA 16802
e-mail: axr2@psu.edu

Imitation of Demonstrations Using Bayesian Filtering With Nonparametric Data-Driven Models

This paper addresses the problem of learning dynamic models of hybrid systems from demonstrations and then the problem of imitation of those demonstrations by using Bayesian filtering. A linear programming-based approach is used to develop nonparametric kernel-based conditional density estimation technique to infer accurate and concise dynamic models of system evolution from data. The training data for these models have been acquired from demonstrations by teleoperation. The trained data-driven models for mode-dependent state evolution and state-dependent mode evolution are then used online for imitation of demonstrated tasks via particle filtering. The results of simulation and experimental validation with a hexapod robot are reported to establish generalization of the proposed learning and control algorithms. [DOI: 10.1115/1.4037782]

Keywords: learning from demonstrations, dynamic data-driven systems, nonparametric density estimation

1 Introduction

Traditionally, plant models of intelligent systems are developed based on the knowledge of underlying physics. As the system complexity grows, physics-based modeling and the associated task of decision and control synthesis become increasingly difficult, if not infeasible. However, advancements in statistical learning theory [1] allow to handle various complex dependencies that can be used for synthesis of decision and control laws. This paper aims to investigate this alternative technique of learning dynamic data-driven models [2] and controllers for hybrid systems by making use of density estimation techniques [1,3].

Based on the data from several demonstrations by an expert using teleoperation for robotic systems, statistical models are inferred for imitating the tasks with the objective that these models finally learn the expert-level decision and control laws. The models developed in this work are represented as nonparametric conditional probability density functions using ratios of kernel-based mixture models. Unlike parametric approaches for density estimation, these nonparametric models have no restriction of structure of the relationship between states and modes and require no assumptions about the noise statistics. The density modeling approach in Refs. [3] and [4] is only suitable when the conditioning variable is a discrete random variable. In contrast, the conditioning variable associated with the dynamic model is continuous-valued in this work. Thus, using Ref. [4] with a special type of product kernel, a dynamic modeling framework has been developed for hybrid systems. The optimal dynamic policy for mode selection is obtained by using Bayesian filtering with a data-driven model of plant dynamics for prediction and a measurement model for correction.

1.1 Prior Work.

Learning from demonstration (LFD) is a class of machine learning algorithms, where a teacher provides examples or trajectories for the robot to perform specified tasks [5]. In this context, an example is a sequence of observation–

[1]Present address: GE Global Research, Niskayuna, NY 12309.
[2]Present address: Mitsubishi Electric Research Laboratories, Cambridge, MA 02139.

Contributed by the Dynamic Systems Division of ASME for publication in the Journal of Dynamic Systems, Measurement, and Control. Manuscript received February 15, 2017; final manuscript received June 22, 2017; published online November 8, 2017. Assoc. Editor: Prashant Mehta.

action pairs that are recorded during a demonstration. Learning from demonstration is closely related to reinforcement learning [6], as model learning or direct policy search algorithms are commonly used in both these areas [5]. Model learning techniques use examples to learn a model of the environment (or of its own dynamics) to find a policy for controlling the robot (e.g., see Refs. [7–9]). The notion of direct policy search is to learn the policies directly and bypass the associated model learning and policy synthesis steps (e.g., see Refs. [10–12]). However, reinforcement learning algorithms are very difficult to optimize for composite tasks which are easily solved by expert humans. Thus, the idea in LFD is to be able to learn the policy used by the expert supervisor directly. Unlike reinforcement learning, usually in LFD algorithms the robot does not receive any reward for performing any action, rather it attempts to infer the policy being followed by the supervisor (assuming it is the optimal policy) using a classification or regression algorithm. In this paper, an LFD technique is presented where the robot is shown demonstrations by teleoperation [13]. The data from these demonstrations are used to learn a transition model over a finite set of predefined modes using a novel conditional density estimation technique. A particle filter is then used with these data-driven dynamic models to track the states as well as select an operation mode, which most likely imitates the actions used by the human supervisor.

Classification-based LFD approaches [5,10] learn a mapping using supervised learning approaches, such as K-nearest neighbor (KNN) or Gaussian mixture models (GMM), to predict the mode using only the current observation. However, the proposed method uses a filter that is built upon the history as well as current measurements of the states to predict and switch the mode of a robotic system. Moreover, in contrast to the classification or regression-based approaches for LFD in robots, the theme of the current paper is to infer the transition model, which was used by the supervisor, for deciding an optimal mode-selection strategy for the robot. Such an approach has the advantage that the underlying structure present in the training data can be exploited, which may not be available directly in the classification approach. For example, a human can make decisions for a task during the training based on certain environmental features that may not be directly available from the state information of the robot. Finding the transition model for making these decisions can improve the accuracy of imitation actions, which is demonstrated in this work through an experiment.

1.2 Contributions. The major contribution of this paper is development and validation of a data-driven method of modeling behavior of hybrid systems using kernel-based mixture models. This paper presents an extension of the density estimation technique developed in Ref. [4] to solve the problem of learning dynamic models for state-dependent mode evolution and mode-dependent state evolution for hybrid systems. Another contribution of this work is the dynamic mode-selection policy for imitation, which is based on Bayesian filtering and estimation. The particle filter is used for tracking states as well as for prediction of modes using the joint state–mode evolution models that are learned from demonstrations.

The proposed procedure of learning theory-based decision and control has been validated via both simulation and laboratory experimentation. In example 1, a simulated one-dimensional (1D) robot learns to slow down at points of interest in a pipeline inspection scenario. In example 2, a walking hexapod robot, which is an over-actuated high-dimensional robotic system, learns to inspect a power plant by following demonstrations given by a human operator using teleoperation. The experimental results have been compared with classification-based approaches. A few videos from the experiment are available online.[2]

[2]https://goo.gl/g7cgHs

The paper is organized as follows: Section 2 formulates the learning and imitation problems mathematically. In Sec. 3, the justification for using Bayesian filtering and the technical approach, including density estimation, data-driven modeling, and particle filtering, is explained in detail. Section 4 illustrates the proposed framework with a 1D simulation example. The experimental setup for validation and the associated results have been described in Sec. 5. Section 6 provides "Concluding Remarks" and future research directions.

2 Problem Formulation

The system of interest is a discrete-time switched system with few distinct modes. Each mode of the system activates an atomic dynamic behavior, which is already programmed into the system. A task (e.g., completing an inspection routine in a power plant for a walking robot) is modeled as a sequence of state-dependent and/or observation-dependent mode transitions. The objective is to learn a dynamic model for the system from demonstrations and then use the dynamic model to imitate that demonstration autonomously. These two problems are formulated ahead.

2.1 Problem of Model Learning From Demonstrations. Let X_t be a continuous random variable denoting the state of the system at a time step $t \in \mathbb{N}$, which can take values from a state set $\mathcal{X} \subseteq \mathbb{R}^n$. The state of the system can be observed using sensors, and let Z_t be a continuous random variable for the observation obtained at a time step $t \in \mathbb{N}$, which can take values from an observation set $\mathcal{Y} \subseteq \mathbb{R}^m$. The true underlying state of the system is not needed in this approach, so the observation need not capture the full state. The discrete set of operating modes of the system are given by a finite set $\mathcal{M}$, and a discrete random variable M_t denotes the mode at time step t, which takes values from the set $\mathcal{M}$. The state evolution dynamics of this switched system are mode-dependent; thus, the conditional density $p(X_{t+1}|X_t, M_t)$ captures the state evolution dynamics. The mode-switching behavior is represented by the state-dependent mode-evolution model $p(M_{t+1}|X_t, M_t)$, which is encoded in the sequence of teleoperation commands given by the operator to the system. Since the state evolution model can be assumed to be independent of the mode at the next time step, we get $p(X_{t+1}|X_t, M_t) = p(X_{t+1}|X_t, M_t, M_{t+1})$. Thus, using the chain rule of probability, we obtain the joint state and mode evolution model as follows:

$$p(X_{t+1}, M_{t+1}|X_t, M_t) = p(X_{t+1}|X_t, M_t)p(M_{t+1}|X_t, M_t), \tag{1}$$

which is learned using the training data from demonstrations. The demonstrations are recorded and are made available as a sequence of tuples of observations z_t^i and an input mode from the user m_t^i as $d^i = (\{z_1^i, m_1^i\}, \{z_2^i, m_2^i\}, \ldots, \{z_{n_i}^i, m_{n_i}^i\})$ for the ith demonstration, where n_i is the length of that demonstration. The finite set of all recorded successful demonstrations from the teacher is given by $\mathcal{D}$. Thus, the objective is to solve the conditional density estimation problem from estimation of the models $p(X_{t+1}|X_t, M_t)$ and $p(M_{t+1}|X_t, M_t)$ from the demonstration sequences given in set $\mathcal{D}$.

2.2 Problem of Imitating the Demonstrations. Assuming that the first problem of model learning can be solved, this second problem can use the joint state–mode evolution dynamics to obtain a predictive model for the mode in the next time step. In essence, the problem is to learn an observation-feedback dynamic mode-selection policy $\mu = (\mu_t : \mu_t(I_t) = \widehat{m}_{t+1},\ \widehat{m}_{t+1} \in \mathcal{M}'(I_t))$, where I_t is the information vector denoting all previous modes and observations, i.e., $I_t = (I_{t-1}, z_t, m_t)$ and $I_0 = (m_0)$. Here, the information-dependent admissible mode set is given as

$$\mathcal{M}'(I_t) = \begin{cases} \varnothing, & \text{if } m_t = \varnothing \\ \mathcal{M}, & \text{otherwise} \end{cases} \tag{2}$$

where $\varnothing$ implies that the task has been completed; hence, mode selection is not needed. The completion of task is determined by the operator or when the state enters a predefined goal set. The overall objective of this LFD approach is to solve an optimization problem, which minimizes the total expected cost over the set of all admissible policies, in order to determine the optimal policy μ^*. This optimization problem is stated as

$$\mu^* = \arg\min_{\mu \in \Pi} \left[\sum_{t \in \mathbb{N}} E_{m_{t+1} \sim \tilde{p}(\cdot|I_t)} (\mathbb{1}(\mu_t(I_t) \neq m_{t+1})) \right] \tag{3}$$

where E is the expectation operator, $\mathbb{1}(\cdot)$ is the indicator function, Π is the set of all admissible policies, and $\tilde{p}(\cdot|I_t)$ is probability distribution representing the uncertainty/variability over the modes in the next time step in the demonstrations after the information I_t was observed. If the teacher would have shown the demonstration of the task in several different ways, then variability would be high; whereas if the demonstrator was confident about the next mode, then variability at that time step would be low. Since we want to obtain the policy being executed by a human operator, the expectation in the cost function is computed over the distribution induced by that operator over the modes in the next time step. Although this distribution is not directly available to solve this optimization problem, it is captured in the demonstrations shown by the teacher. The objective here is to learn the underlying distribution from data, so that an imitation policy can be obtained by the aforementioned optimization. This optimization problem is solved in Sec. 3.

3 Technical Approach

This section presents the technical approach by first giving an overview of the proposed approach and then explains the optimization-based density estimation process to learn the dynamic models from data. The Bayesian filtering-based state tracking and mode prediction process is also explained in Sec. 3.1.

3.1 Overview. The training and the operation phases of the problem address the learning and imitation tasks, respectively. During training, the data from demonstrations are used to obtain the dynamic model $p(X_{t+1}, M_{t+1}|X_t, M_t)$ by nonparametric density estimation [3,4]. The observation model $p(Z_t|X_t)$ is assumed to be given; else, it can also be learned from data [14]. The optimization problem, as given in Eq. (3), is simplified and restated as

$$\mu^* = \arg\min_{\mu \in \Pi} \left[\sum_{t \in \mathbb{N}} (1 - \tilde{p}(M_{t+1} = \mu_t(I_t)|I_t)) \right]$$

Since the sequence of mode estimates from the policy can be chosen independently at every time step, the optimal estimate for any $t \in \mathbb{N}$ should satisfy

$$\begin{aligned} \widehat{m}_{t+1} &= \arg\min_{m \in \mathcal{M}'(I_t)} (1 - \tilde{p}(M_{t+1} = m|I_t)) \\ \Rightarrow \widehat{m}_{t+1} &= \arg\max_{m \in \mathcal{M}'(I_t)} \tilde{p}(M_{t+1} = m|I_t) \end{aligned} \tag{4}$$

Thus, the optimal policy which minimizes the cost in Eq. (3) is given by the most likely next mode that would have been executed by the teacher. Although the distribution $\tilde{p}$ is not directly available to obtain the optimal policy, that distribution has been encoded in the demonstrations shown by the teacher. Now, an estimate of this probability distribution is obtained by using state–mode evolution model, sequential update from Bayesian filtering, and observations. Assuming that initially a prior distribution is available at time t, i.e., $p(X_t, M_t|I_t)$, prediction with the state–mode evolution model yields

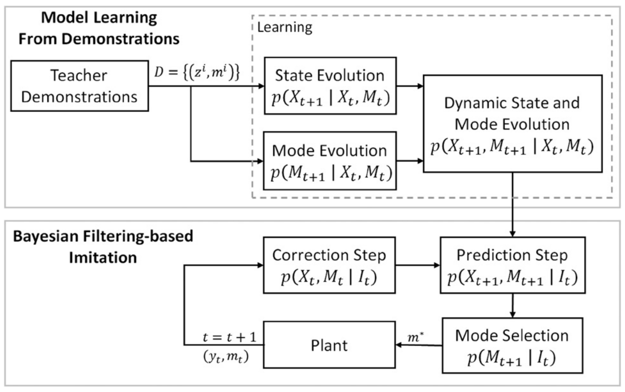

Model Learning From Demonstrations
Learning
Teacher Demonstrations
$D = \{(z^i, m^i)\}$
State Evolution $p(X_{t+1} \mid X_t, M_t)$
Mode Evolution $p(M_{t+1} \mid X_t, M_t)$
Dynamic State and Mode Evolution $p(X_{t+1}, M_{t+1} \mid X_t, M_t)$
Bayesian Filtering-based Imitation
Correction Step $p(X_t, M_t \mid I_t)$
Prediction Step $p(X_{t+1}, M_{t+1} \mid I_t)$
$t = t+1$
(y_t, m_t)
Plant
m^*
Mode Selection $p(M_{t+1} \mid I_t)$

Fig. 1 Schematic for proposed model learning and imitation

$$p(X_{t+1}, M_{t+1}|I_t) = \int_{x_t \in \mathcal{X}} \sum_{m_t \in \mathcal{M}} p(X_{t+1}, M_{t+1}|x_t, m_t) p(x_t, m_t|I_t) dx_t \tag{5}$$

This predicted state–mode distribution yields the posterior density with the new observation by using the Bayes' rule in the correction step as

$$p(X_t, M_t|I_t) = \frac{p(Z_t|X_t) p(X_t, M_t|I_{t-1})}{\int_{x_t \in \mathcal{X}} \sum_{m_t \in \mathcal{M}} p(Z_t|x_t) p(x_t, m_t|I_{t-1}) dx_t}$$

The prediction and correction steps of Bayesian filtering will be used sequentially to obtain the posterior density and the predicted density as shown earlier. The predicted density from Eq. (5) will also used to obtain the predicted mode density

$$p(M_{t+1}|I_t) = \int_{x \in \mathcal{X}} p(X_{t+1} = x, M_{t+1}|I_t) dx \tag{6}$$

which is an estimate of the desired probability distribution to obtain the mode selection policy as shown in Eq. (4). The proposed overall learning and task execution process is shown in Fig. 1, which shows the Bayesian filter for state estimation and mode prediction during execution. The learning and imitation problems are explained in Sec. 3.2.

3.2 Model Learning. The problem of model learning from demonstration data uses nonparametric density estimation, since it does not restrict the structure of the relationship or the noise statistics. The kernel-based density estimation framework used in this work has been developed in Ref. [4] and it is succinctly explained here for completeness. This density estimation framework is then used to obtain the relevant data-driven models.

3.2.1 Kernel-Based Density Estimation. In nonparametric regression techniques, such as support vector regression and regularization networks [15], the estimate of the unknown function f is obtained as

$$f(y) = \sum_{i=1}^{N} \pi_i K(y, y_i) \tag{7}$$

where $K(\cdot,\cdot)$ is the kernel function. In the problem of probability density estimation, the training dataset includes only the measurements $\{y_1, y_2, \ldots, y_N\}$, with $y_i \in \mathcal{Y}$ for all i, and the desired regression output is unknown. However, from the Glivenko–Cantelli (GC) theorem of statistical learning theory [1], it is known that the empirical cumulative density function (CDF) uniformly converges almost surely to the true CDF as $N \to \infty$. That is,

$\sup_{y\in\mathcal{Y}}|F(y)-F_N(y)| \to 0$ almost surely, for all CDFs F, where F_N is the empirical CDF with N measurements, which is defined by $F_N(y)=\sum_{i=1}^{N}(\prod_{d=1}^{m} 1_{[y_i^d,\infty)}(y^d))$ for all $y=(y^1,\ldots,y^m)\in\mathcal{Y}$. Here, $1_S(\cdot)$ is the indicator function on a set S, and $y_i=(y_i^1,\ldots,y_i^m)$ for $i=1,\ldots,N$. Recall that the CDF of a random vector $F(y)=f(Y\le y)$ is computed by considering the inequality elementwise [1]. Although the original GC theorem assumes that the measurement sequence is i.i.d., generalization of the GC theorem for strictly stationary measurement sequences [16], and for data coming from uniformly ergodic Markov chains [17], is also available in the literature. In order to leverage the GC theorem, the problem of density estimation is formulated as that of CDF estimation, where an empirical CDF is used in place of the true CDF [1,3]. Thus, the regression function is estimated as $\tilde{f}(y)=\sum_{i=1}^{N}\pi_i\tilde{K}(y,y_i)$ where $\tilde{K}(y,y_i)=\int_{-\infty}^{y}K(y',y_i)dy'$ for $y\in\mathcal{Y}$ and for $i=1,\ldots,N$. The set of tuples $\{(y_1,F_N(y_1)),\ldots,(y_N,F_N(y_N))\}$ is then used as the training set to obtain $\pi_1,\ldots,\pi_N$. Here, the requirement is that the kernel function should satisfy $\int_{\mathcal{Y}}K(y',y)dy'=1$ (integration to 1) and $0\le K(\cdot,y)\le k<\infty$ (boundedness) for any $y\in\mathcal{Y}$. An example of kernel function that satisfies this requirement is the Gaussian probability density function.

The optimization problem for density estimation is formulated next. Let $\tilde{y}=[F_N(y_1)\ldots F_N(y_N)]^{\mathrm{T}}$ and $\pi=[\pi_1\cdots\pi_N]^{\mathrm{T}}$ be column vectors with N elements. Let $\tilde{\mathbf{K}}=[\tilde{K}(y_i,y_j)]$ be the $N\times N$ matrix whose entry (i,j) is equal to $\tilde{K}(y_i,y_j)$ for $i,j=1,\ldots,N$ and $\tilde{K}_i$ is the ith row of $\tilde{\mathbf{K}}$. The optimization for density estimation is then formulated as a constrained empirical risk minimization problem as: $\pi=\arg\min_{\pi\in\mathcal{H}}(1/N)\sum_{i=1}^{N}\max(0,|\tilde{y}_i-\tilde{K}_i\pi|-\sigma)$, where the constrained hypothesis set $\mathcal{H}=\{\pi\in\mathbb{R}^N:\sum_{i=1}^{N}\pi_i=1,\pi\ge 0\}$ satisfies that each estimate is a well-defined density function. However, the acceptable error margin σ in the hinge loss function need not be known a priori for a general dataset; thus, the following problem is formulated [4]:

$$(\pi,\sigma)=\underset{\pi\in\mathcal{H},\sigma\ge 0}{\arg\min}\ \nu\sigma+\frac{1}{N}\sum_{i=1}^{N}\max\left(0,|\tilde{y}_i-\tilde{K}_i\pi|-\sigma\right) \tag{8}$$

where $\nu\in(0,1]$ is the given weight for the error margin σ. The cost function includes a penalty on the size of margin σ and a loss term which penalizes the absolute error in estimation only if it is larger than that margin. This constrained optimization problem is convex, but nonlinear, due to the presence of $\max(\cdot,\cdot)$ and $|\cdot|$ functions in the cost function. However, it can be converted to a linear program (LP) by adding more optimization variables. The LP for density estimation [4], corresponding to Eq. (8), is given as

$$\left(\pi,\sigma,\xi^+,\xi^-\right)=\underset{\substack{\pi\in\mathcal{H},\sigma\ge 0\\(\xi^+,\xi^-)\in\Xi}}{\arg\min}\ \nu\sigma+\frac{1}{N}\sum_{i=1}^{N}\left(\xi_i^++\xi_i^-\right) \tag{9}$$

where the convex set $\Xi=\{(\xi^+,\xi^-)\in\mathbb{R}^N:\tilde{y}_i-\tilde{K}_i\pi\le\sigma+\xi_i^+,\ -\tilde{y}_i+\tilde{K}_i\pi\le\sigma+\xi_i^-,\xi_i^+\ge 0,\xi_i^-\ge 0,\forall i=1,2,\ldots,N\}$. This LP can be efficiently solved using the interior point method [18] available in standard LP solvers. The computational complexity for learning the model using the interior point method (Karmakars algorithm) is $O(N^{3.5}L)$, where N is the number of observations and L is the number of bits of input to the solver [18]. This linear program generates sparse mixture models; thus, the resulting models are concise [4]. An upper bound (σ) of the error in estimation is also available from the optimization; thus, accurate models can be obtained by selecting appropriate kernel parameters by using cross-validation.

3.2.2 Nonparametric Data-Driven Models. Using the chain rule and marginalization from probability theory, the state evolution model is represented as

$$p(X_{t+1}|X_t,M_t)=\frac{p(X_{t+1},X_t|M_t)}{\displaystyle\int_{x_{t+1}\in\mathcal{X}}p(x_{t+1},X_t|M_t)dx_{t+1}}$$

and the mode evolution model is represented as

$$p(M_{t+1}|X_t,M_t)=\frac{p(X_t|M_{t+1},M_t)p(M_{t+1}|M_t)}{\displaystyle\sum_{m_{t+1}\in\mathcal{M}}p(X_t|m_{t+1},M_t)p(m_{t+1}|M_t)}$$

The densities $p(X_{t+1},X_t|M_t)$, $p(X_t|M_{t+1},M_t)$, and $p(M_{t+1}|M_t)$ are estimated from the training data available in the set of all successful demonstrations $\mathcal{D}$. The density estimation approach described in Secs. 3.2 and 3.2.1 is used to obtain the density estimates for $p(X_{t+1},X_t|M_t)$, $p(X_t|M_{t+1},M_t)$. Whereas $T_{l,m}=p(M_{t+1}=m|M_t=l)$ is estimated by counting the occurrences of mode m after mode l, i.e., by frequency counting and normalization, which is the maximum likelihood estimate [19] of that density.

Training sets $\mathcal{Y}_m=\{y_t^i=(z_{t+1}^i,z_t^i),\forall t\in\{1,2,\ldots,n_i\},\forall d^i\in\mathcal{D}:m_t^i=m\}$ are constructed using all demonstrations for all modes in the mode set, i.e., $m\in\mathcal{M}$. An estimate of the density $p(X_{t+1},X_t|M_t=m)$ is obtained by solving the LP in Eq. (9) using the training set $\mathcal{Y}_m$ and a special type of product kernel

$$K^m([z_{t+1},z_t]^{\mathrm{T}},[z'_{t+1},z'_t]^{\mathrm{T}})=K_2^m(z_{t+1},z'_{t+1})K_1^m(z_t,z'_t)$$

where both K_1^m and K_2^m individually satisfy the integration to 1 and the boundedness requirement. The obtained density estimate is given as

$$p(X_{t+1},X_t|M_t=m)=\sum_{i=1}^{n_m}\pi_i^mK_2^m(X_{t+1},z_{t+1}^i)K_1^m(X_t,z_t^i)$$

where $n_m\ll|\mathcal{Y}_m|$ is the number of nonzero coefficients in π^m. The marginalization of this density is trivial, due to the special choice of kernels, which gives

$$p(X_t|M_t=m)=\sum_{i=1}^{n_m}\pi_i^mK_1^m(X_t,z_t^i)$$

In Ref. [14], it was proven that the ratio of joint density and marginal density estimates, obtained as shown previously, are bounded and well-defined everywhere, if $K_1^m(\cdot,z)>0$ for all z in the domain of the problem. Using this result, the data-driven state evolution model is given as

$$p(X_{t+1}|X_t,M_t=m)=\frac{\displaystyle\sum_{i=1}^{n_m}\pi_i^mK_2^m\left(X_{t+1},z_{t+1}^i\right)K_1^m\left(X_t,z_t^i\right)}{\displaystyle\sum_{i=1}^{n_m}\pi_i^mK_1^m\left(X_t,z_t^i\right)}$$

On the other hand, the training set for mode evolution model is obtained using all the demonstrations as: $\mathcal{Y}_{l,m}=\{y_t^i=(z_t^i),\forall t\in\{1,2,\ldots,n_i\},\forall d^i\in\mathcal{D}:m_t^i=l,m_{t+1}^i=m\}$. This training set and a kernel function $K^{l,m}(\cdot,\cdot)$ are used to solve the LP in Eq. (9). The resulting density estimate has $n_{l,m}\ll|\mathcal{Y}_{l,m}|$ number of nonzero coefficients in $\pi^{l,m}$. This density is used to obtain the mode evolution model, which is a probability mass function given as follows:

$$p(M_{t+1}=m|X_t,M_t=l)=\frac{T_{l,m}\displaystyle\sum_{i=1}^{n_{l,m}}\pi_i^{l,m}K^{l,m}\left(X_t,z_t^i\right)}{\displaystyle\sum_{\tilde{m}\in\mathcal{M}}T_{l,\tilde{m}}\sum_{i=1}^{n_{l,\tilde{m}}}\pi_i^{l,\tilde{m}}K^{l,\tilde{m}}\left(X_t,z_t^i\right)}$$

where $T_{l,m}$ is the mode transition probability from mode l to mode m. Since all terms in the numerator and denominator are positive, it is readily verified that the probability mass function estimate is well defined, i.e., non-negative and adds up to 1, if $K^{l,m}(\cdot, z) > 0$ for all z in the domain of the problem.

This part shows the procedure to derive the nonparametric data-driven models of state evolution and mode evolution. As shown in Eq. (1), the product of these two models serves as the desired dynamic model for joint evolution of state and mode for the system under consideration.

3.3 Bayesian Filtering-Based Imitation. We know from Sec. 3.1 that the predicted mode density can be obtained from data-driven model and observations using sequential updates of a Bayesian filter. We also know that the optimal policy μ^* selects the mode with the highest probability from the predicted density. The dynamic motion model is available as a nonparametric density function in this paper, which can only generate point estimates and do not have a representation as a deterministic function with additive noise. Thus, one cannot directly use the Kalman filter and its variants for sequential updates and filtering. The particle filter [20] is an appropriate choice for working with density models. The implementation of this Bayesian filter for imitation using the concept of particle filtering is presented ahead in this section.

Algorithm 1: Particle Filter with Mode Prediction

```
1 Initialize by distributing $N_p$ particles in the domain
2 for $t = 1 : MAX$ do
3     for $i = 1 : N_p$ do
4         Calculate $p(z_t|x_t^i)$ for measurement $z_t$
5         Update Weights: $w_t^i = w_{t-1}^i p(z_t|x_t^i)$
6     Normalization: $w_t^i = w_t^i / \sum_{i=1}^{N_p} w_t^i$ for all particles
7     $[\{x_t^i, m_t^i, w_t^i\}_{i=1}^{N_p}]$ = Resample using Algorithm 2
8     for $i = 1 : N_p$ do
9         $x_{t+1}^i, m_{t+1}^i$ = Move $x_t^i, m_t^i$ using Algorithm 3
10    Mode selection: $\widehat{m}_{t+1} = \text{mode}\left(\{m_t^i\}_{i=1}^{N_p}\right)$
```

Algorithm 2: Low-Variance Resampling Algorithm [21]

```
1 Set $l = 1, c_1 = w_t^1$
2 Sample $u_1$ from $\mathcal{U}[0, N_p^{-1}]$
3 for $j = 1 : N_p$ do
4     $u_j = u_1 + \frac{j-1}{N_p}$
5     while $u_j > c_l$ do
6         $l = l + 1, c_l = c_{l-1} + w_t^l$
7     $\tilde{x}_t^j = x_t^l, \tilde{m}_t^j = m_t^l, \tilde{w}_t^j = \frac{1}{N_p}$
8 return $[\{\tilde{x}_t^j, \tilde{m}_t^j, \tilde{w}_t^j\}_{j=1}^{N_p}]$
```

Algorithm 3: Propagation via Importance Sampling

```
1 Generate $N_q$ samples of $\tilde{x}^q$ from $\mathcal{B}_\eta(x_t^i)$
2 Generate $N_q$ samples of $\tilde{m}^q$ using $p(m_{t+1}|m_t^i)$
3 for $q = 1 : N_q$ do
4     Evaluate $p(\tilde{m}^q|m_t^i, x_t^i)$ and $p(\tilde{x}^q|m_t^i, x_t^i)$
5     Calculate importance: $p(\tilde{x}^q, \tilde{m}^q|m_t^i, x_t^i)$
6 $q^* = \arg\max_q p(\tilde{x}^q, \tilde{m}^q|m_t^i, x_t^i)$
7 return $\tilde{x}^{q^*}, \tilde{m}^{q^*}$
```

In particle filtering, a set of particles is used to approximate the representation of joint state and mode densities $p(X_t, M_t)$. The prediction step propagates the particles to the next step, while the correction step accounts for the measurement by performing resampling of the particles to obtain the posterior density. The state evolution model $p(X_{t+1}|X_t, M_t)$, the measurement model $p(Z_t|X_t)$, and the mode switching dynamics $p(M_{t+1}|X_t, M_t)$ are used along with all the previous information I_t to obtain the posterior density. The particle filter estimates the posterior filtered density by first dispersing a finite number of particles within the joint domain of states and modes, i.e., $\mathcal{X} \times \mathcal{M}$. Each particle is assigned a state, a mode, and an associated uniform weight $\{x_t^i, m_t^i, w_t^i\}$, where i represents the ith particle and t is the iteration number representing the discrete time step. Here, nonuniform weights can capture the effects of a specific prior distribution. It is shown in Ref. [20] that the posterior filtered density can then be approximated as

$$p(X_t, M_t|I_t) \approx \sum_{i=1}^{N} w_t^i \delta(X_t - x_t^i)\delta(M_t - m_t^i) \tag{10}$$

where w_t^i is proportional to $w_{t-1}^i p(z_t|x_t^i)$ and $\delta(\cdot)$ is the Dirac measure.

The particle filter for prediction, as shown in Algorithm 1, begins with the correction step (lines 3–7). In this step, the state of each particle x_t^i and the measurement z_t is used to obtain the likelihood value from the measurement model, $p(z_t|x_t^i)$, which is then used to update the particle weights. The particles with low weights are eliminated and those with high weights are duplicated by using resampling. The low-variance resampling algorithm [20,21], as explained in Algorithm 2, is chosen in this work, since it is among the most time-efficient methods for resampling.

The propagation step requires moving the particle according to the learnt dynamics. The state and mode transition dynamics is available only as a density function. Hence, importance sampling is employed to propagate the particles using this model, as shown in Algorithm 3. A sample of N_q number of next states $\{\tilde{x}^q\}$ is generated uniformly from a ball $\mathcal{B}_\eta(x_t^i)$ around the state x_t^i, whereas a sample of N_q number of next modes $\{\tilde{m}^q\}$ is generated randomly from $p(M_{t+1}|m_t^i)$, for each particle (x_t^i, m_t^i). Here, the set $\mathcal{B}_\eta(x_t^i) \triangleq \{x \in \mathcal{X} : d(x_t^i, x) \leq \eta\}$, where the function $d(\cdot, \cdot)$ is a distance metric defined on the space $\mathcal{X}$. An importance measure is assigned to each candidate particle according to the probability from the joint state–mode dynamic model. The state–mode pair with the highest importance value is selected as the next state and mode of that particle. This sampling is repeated for each of the N_p particles, as shown in line 8–9 in Algorithm 1. The set of propagated particles represent the predicted state–mode density. In order to obtain the predicted mode density, the state has to be marginalized. This is achieved by counting the number of particles for each mode and then the most frequent mode is used as the estimated mode. After execution of this mode, a new measurement is sampled and the correction, prediction, mode estimation, and execution steps are repeated.

It is noted that the time complexity of executing this filtering-based imitation mode-selection policy depends upon the sparsity of the data-driven models obtained from the proposed technique. If the model with the highest number of components in the mixture model has S components, then the time complexity is governed by the propagation step as $O(N_p N_q S)$, where N_p is the number of particles, N_q is the number of next states explored for each particle, and S is the highest number of components in the data-driven model which is evaluated for each next state.

4 Illustration With Simulation

A one-dimensional simulation case is presented to illustrate and visualize the working concept of the proposed method. In this example, a pipe inspection robot moves along a straight path with high velocity except near points of interest (e.g., welded joints),

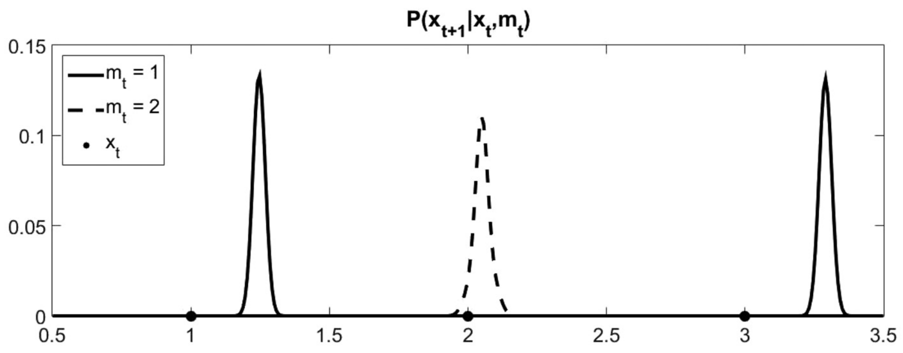


Fig. 2 Probability distribution of the next step

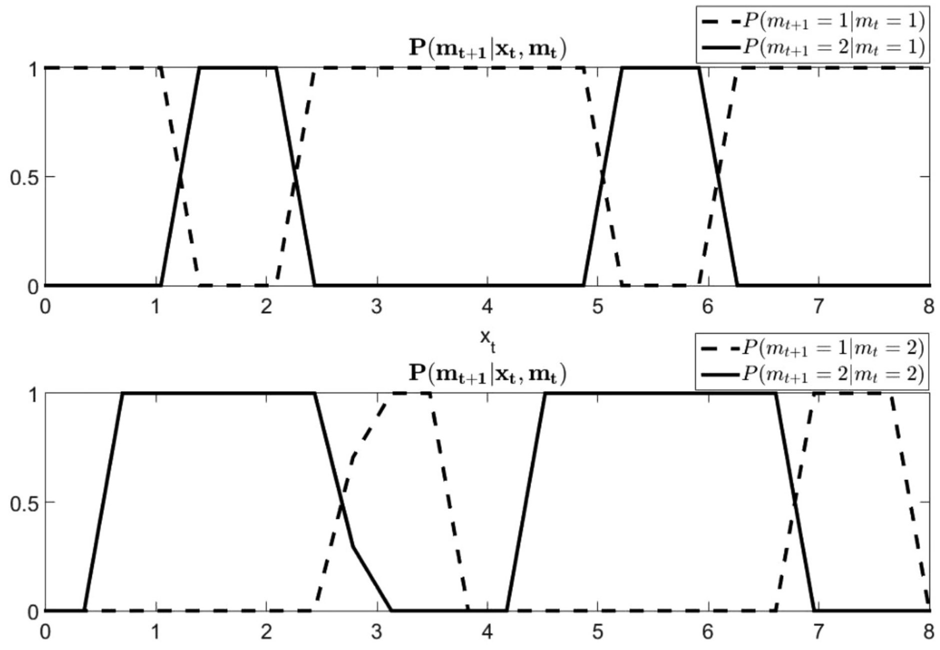


Fig. 3 Probability distribution of the next mode (top: $m_t = 1$, bottom: $m_t = 2$)

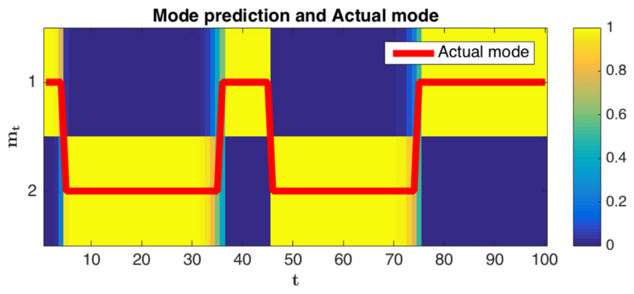
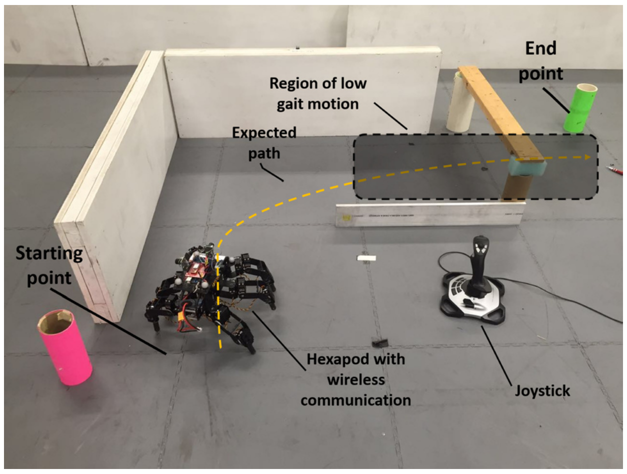


Fig. 4 Probabilities of next mode as given by the particles and the actual mode given as input during demonstration



End point
Region of low gait motion
Expected path
Starting point
Hexapod with wireless communication
Joystick

Fig. 5 Hexapod robot experiment setup

where it slows down for better inspection. The simulated state evolution is given as

$$X_{t+1} = X_t + v(M_t)\Delta T + \varepsilon_x$$

where $M_t \in \{1, 2\}$ is the mode, $v(M_t)$ is the mode-dependent speed, X_t is the displacement along the path, ε_x is the (additive) process noise, and ΔT is the discretization time step. The demonstration has also been simulated using a statistical rule, which generates mode inputs. The points of interest are chosen to be $x = 2$ and $x = 6$. The dataset, consisting of observation-mode pairs, to train the evolution models is obtained through multiple runs of this simulated model.

Figure 2 shows the probability distribution of next state $p(X_{t+1}|X_t, M_t)$ that peaks farther from X_t, if $M_t = 1$, and it peaks closer to X_t, if $M_t = 2$, because the speed is higher in mode 1.

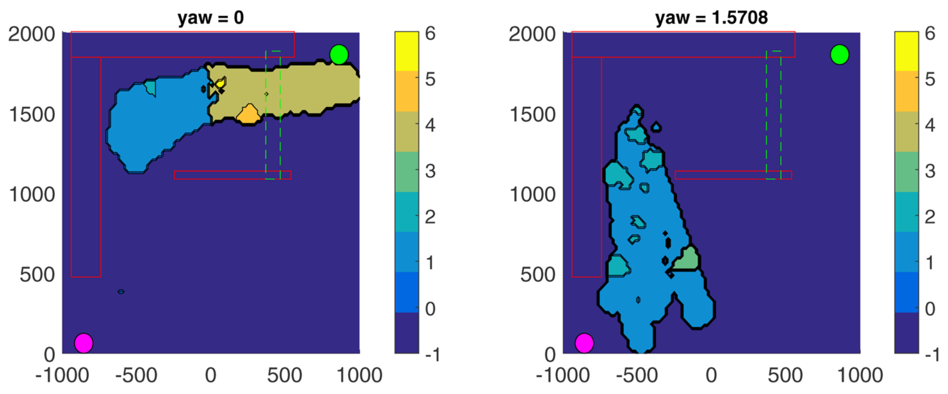


Fig. 6 Visualization of state-to-mode mapping for KNN classifier

Thus, this model captures the dependence of state evolution dynamics on the mode. The mode evolution model in Fig. 3 shows the mode-switching behavior around the points of interest. The robot initially starts in mode 1, and then around $x = 1.5$

$$P(M_{t+1} = 2 | X_t = x, M_t = 1) > P(M_{t+1} = 1 | X_t = x, M_t = 1)$$

makes the robot switch to mode 2. Once it crosses $x = 2$, it will soon switch back to mode 1. Similar behavior can be seen around $x = 6$. Thus, a predictive model of state-dependent mode-switching is also obtained from the learning approach.

During imitation, the true motion of the robot is simulated for the mode estimated from the Bayesian approach, as discussed in Sec. 3.3, and a new measurement is generated from the measurement model for the true state. Specifically, the particles are initially dispersed on the state space and the known mode. The particles are propagated to obtain the predicted state–mode density using the dynamic state and mode evolution model. The most-likely mode is chosen to be executed on the simulated system. The marginal density over the modes is shown in Fig. 4 as a heat map. The estimated mode is compared with the actual mode from a recorded demonstration. It is seen that the proposed method is able to correctly predict the next mode. Thus, the particle filter is capable of not only localizing the robot, but also enabling mode selection with data-driven models. The code for demonstration on the one-dimensional case is available at the website.[3]

5 Experimental Validation

This section presents experimental validation of the proposed method. A simple inspection scenario, where a hexapod robot is shown to move around an obstacle course via teleoperation, is used to experimentally validate this method. The details of the experiment setup and comparison of results with two reference techniques are presented in this section.

5.1 Experimental Setup.

Emulation of a power-plant inspection scenario with walls, overhead obstacles, and low-height obstacles has been constructed in a laboratory setting. The inspection is to be carried out by an 18 degrees-of-freedom Hexapod robot as seen in Fig. 5. Although the true state of the 18 degrees-of-freedom robot would be higher, the state (x, y, θ) was considered, which can be observed using a motion capture system, where (x, y) is the location of the center of the mass of the robot in the ground frame of reference, and θ is the yaw angle of the robot. The robot is 36 cm long and 33 cm wide, and it can stretch as long as 45 cm when walking under the tripod gait [22]. In the tripod gait, it can walk straight, turn left, turn right, and switch between two heights. It is noted that the low-height mode is necessary to go under the overhead obstacle. The mode set for the robot is given as

$$\mathcal{M} = \{HS, HR, HF, LS, LF, LR, -1\}$$

where H (respectively, L) means the robot is high (respectively, low); similarly, S-walk straight, R-turn right, and F-turn left, and -1 is the *call for help* mode. For example, the robot starts in mode HS, which implies the robot is walking straight in high gait. In this setting, the robot is instructed by a human operator using joystick input over Bluetooth. For each input from human, the robot would execute one step of the gait and then wait. It takes about 4 s for the robot to finish one action after receiving an input. The teaching process is repeated for ten times, and the information about position, orientation, and mode has been used as training data.

5.2 Model Training.

The proposed Bayesian approach is compared with two supervised learning methods—KNN and Gaussian mixture model [19]. Both KNN and GMM classifiers make use of the state (x, y, θ) as input and next mode as the output. For KNN, the distance metric is selected as the weighted ℓ_2-norm, due to the different units of the input features. If the distance of the nearest neighbor is more than 1000, the mode output is given as -1. For GMM, the component for each class is determined to be $[25, 19, 4, 36, 8, 7]$, respectively, according to the Akaike information criterion [19]. The output of -1 is assigned when the output mode probability is less than 10^{-10}. The KNN and GMM classifiers are visualized in Figs. 6 and 7, respectively, as filled contour plots. These plots have been generated for the $X-Y$ domain with yaw angles at 0 rad and $\pi/2$ rad.

The system evolution model is learned in terms of $p(X_{t+1}, X_t | M_t)$ and $p(X_t | M_{t+1}, M_t)$ using the kernel-based density estimation. The kernel parameter γ and ν are chosen to be 0.005 and 0.01, respectively, for solving the LP. The density $p(X_{t+1}, X_t | M_t = 1)$ is trained with 558 observations and is represented by 138 components; similarly, the density $p(M_{t+1} = 1 | X_t, M_t = 1)$ is trained with 516 observations and represented by 254 components. The learning framework is capable of giving a compressed model give a concise model; however, the compression is lower here as the runs are too few and dissimilar. In the proposed approach, if less than 25% of the particles form the majority, then the classifier outputs -1 mode to ask for human intervention.

5.3 Results and Inferences.

The experiments have been conducted with the inputs being generated by the classifier/policy instead of human input. Ten runs with each approach have been considered for comparison. If the robot could finish without

[3]https://goo.gl/9zZDbd

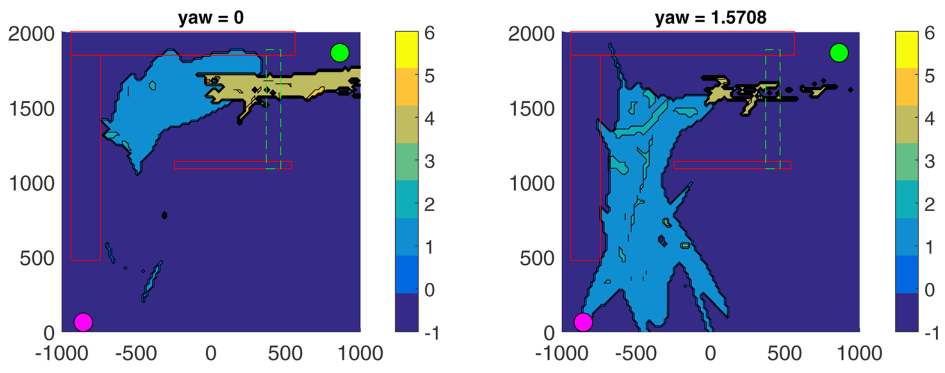


Fig. 7 Visualization of state-to-mode mapping for GMM classifier

Table 1 Performance comparison

Metric	Result	KNN	GMM	Bayesian
Number of runs	Success	3	0	6
	Collision	7	1	0
	Call	0	9	4
Distance traveled (m)	Success	2.15	—	2.44
	Failure	2.03	1.29	1.72

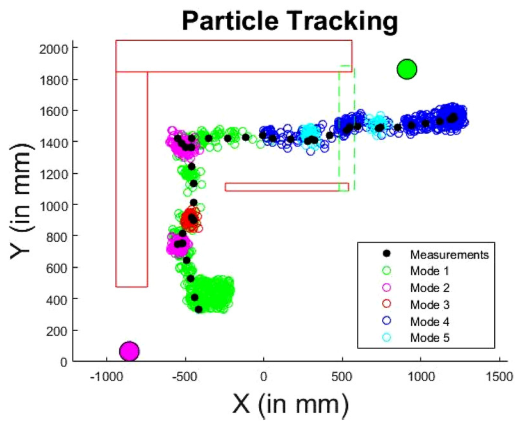


Fig. 8 Tracking and prediction by particle filters

collision or calling for help, the experiment is counted as success. The results are shown in Table 1 and are discussed below.

Experiments with KNN and GMM classifiers have received a low success rate probably due to scarcity of data. KNN has three successful runs and the rest have failed due to collision. For GMM, none of the ten experiments have succeeded and majority of the runs ended up in the -1 mode. Furthermore, the Bayesian approach travels more distance before failing than GMM, since the robot asks for help whenever it enters a -1 mode region with GMM classifier. The Bayesian technique is better than the other approaches in this test, as six out of ten runs succeeded while the other runs concluded with asking for help. One result of experimentation in Fig. 8 shows how the particles track the measured position closely and also select the mode to be executed. The robot travels less before failing with Bayesian approach than with KNN, since with Bayesian classifier it fails by inaction (due to lack of information), whereas the failure in KNN is due to action which result in collision. For successful runs, the robot travels a larger distance with Bayesian than KNN. In the Bayesian method, the robot was seen to apply corrective action to compensate for navigation errors. This shows better generalizability of the proposed algorithm as compared to the other two classification techniques. The videos for several of these runs are available online.[4]

6 Conclusion and Future Work

This paper presents a novel data-driven modeling technique for switched systems from observation-mode data that are obtained from demonstrations with the system. A kernel-based density estimation framework is used to develop the state and mode evolution models. These models are then used in a Bayesian framework for tracking as well as for obtaining a dynamic observation-feedback mode-selection policy, which enables the system to imitate the demonstrations. The Bayesian framework for data-driven models is implemented with particle filters, where an importance sampling-based particle propagation has been used.

This paper has illustrated the developed approach with simulation data showing that it can indeed represent dynamics and mode-switching behaviors. The feasibility of the proposed method has been tested with experiments of learning from demonstrations, where the test with a hexapod robot shows that the method is comparable to or even better than existing classification-based methods. As a topic of future research, the kernel bandwidth selection process will be further explored to make the proposed method more suitable for real-life systems. The incorporation of human feedback via sequential learning is another topic of future research. Another important topic of future research is to explore the proposed idea for systems with discontinuous behavior which are difficult to be modeled directly by a classification or regression-based approach [23]. The implementation of an alternative Bayesian filter, such as Rao-Blackwellized particle filter [24], and the comparison with the presented particle filters should also be considered in future.

Acknowledgment

The authors are thankful to Yusuke Seto and Noboru Takahashi for doing the initial programming of the walking gaits for the hexapod robot. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

Funding Data

- U.S. Air Force Office of Scientific Research (AFOSR) (Grant No. FA9550-15-1-0400).

References

[1] Vapnik, V. N., 1998, *Statistical Learning Theory*, Wiley, New York.
[2] Darema, F., 2005, "Dynamic Data Driven Applications Systems: New Capabilities for Application Simulations and Measurements," Fifth International Conference on Computational Science (ICCS), Atlanta, GA, May 22–25, pp. 610–615.

[4]https://goo.gl/g7cgHs

[3] Vapnik, V., and Mukherjee, S., 2000, "Support Vector Method for Multivariate Density Estimation," *Advances in Neural Information Processing Systems*, Vol. 12, S. A. Solla, T. K. Leen, and K.-R. Muller, eds., MIT Press, Cambridge, MA, pp. 659–665.
[4] Virani, N., Lee, J.-W., Phoha, S., and Ray, A., 2016, "Information-Space Partitioning and Symbolization of Multi-Dimensional Time-Series Data Using Density Estimation," American Control Conference (ACC), Boston, MA, July 6–8, pp. 3328–3333.
[5] Argall, B. D., Chernova, S., Veloso, M., and Browning, B., 2009, "A Survey of Robot Learning From Demonstration," Rob. Auton. Syst., **57**(5), pp. 469–483.
[6] Sutton, R. S., and Barto, A. G., 1998, *Reinforcement Learning: An Introduction*, Vol. 1, MIT Press, Cambridge, MA.
[7] Smart, W. D., and Kaelbling, L. P., 2002, "Effective Reinforcement Learning for Mobile Robots," IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, May 11–15, pp. 3404–3410.
[8] Stolle, M., and Atkeson, C. G., 2007, "Knowledge Transfer Using Local Features," IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL), Honolulu, HI, Apr. 1–5, pp. 26–31.
[9] Kuniyoshi, Y., Inaba, M., and Inoue, H., 1994, "Learning by Watching: Extracting Reusable Task Knowledge From Visual Observation of Human Performance," IEEE Trans. Rob. Autom., **10**(6), pp. 799–822.
[10] Chernova, S., and Veloso, M., 2007, "Confidence-Based policy Learning From Demonstration Using Gaussian Mixture Models," Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Honolulu, HI, May 14–18, p. 233.
[11] Schaal, S., Ijspeert, A., and Billard, A., 2003, "Computational Approaches to Motor Learning by Imitation," Philos. Trans. R. Soc., B, **358**(1431), pp. 537–547.
[12] Saunders, J., Nehaniv, C. L., and Dautenhahn, K., 2006, "Teaching Robots by Moulding Behavior and Scaffolding the Environment," First ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI), Salt Lake City, UT, Mar. 2–3, pp. 118–125.
[13] Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E., 2006, "Autonomous Inverted Helicopter Flight Via Reinforcement Learning," *Experimental Robotics IX*, Springer, Berlin, pp. 363–372.
[14] Trezza, A., Virani, N., Wolkowicz, K., Moore, J., and Brennan, S., 2015, "Indoor Mapping and Localization for a Smart Wheelchair Using Measurements of Ambient Magnetic Fields," ASME Paper No. DSCC2015-9915.
[15] Smola, A. J., and Schlkopf, B., 2004, "A Tutorial on Support Vector Regression," Stat. Comput., **14**(3), pp. 199–222.
[16] Tucker, H. G., 1959, "A Generalization of the Glivenko-Cantelli Theorem," Ann. Math. Stat., **30**(3), pp. 828–830.
[17] Zou, B., Zhang, H., and Xu, Z., 2009, "Learning From Uniformly ergodic Markov Chains," J. Complexity, **25**(2), pp. 188–200.
[18] Karmarkar, N., 1984, "A New Polynomial-Time Algorithm for Linear Programming," 16th Annual ACM Symposium on Theory of Computing (STOC), Washington, DC, Apr. 30–May 2, pp. 302–311.
[19] Bishop, C. M., 2006, *Pattern Recognition and Machine Learning* (Information Science and Statistics), Springer-Verlag, New York.
[20] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T., 2002, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," IEEE Trans. Signal Process., **50**(2), pp. 174–188.
[21] Thrun, S., Burgard, W., and Fox, D., 2005, *Probabilistic Robotics*, MIT Press, Cambridge, MA.
[22] Seto, Y., Takahashi, N., Jha, D. K., Virani, N., and Ray, A., 2016, "Data-Driven Robot Gait Modeling Via Symbolic Time Series Analysis," American Control Conference (ACC), Boston, MA, July 6–8, pp. 3904–3909.
[23] Kroemer, O., Van Hoof, H., Neumann, G., and Peters, J., 2014, "Learning to Predict Phases of Manipulation Tasks as Hidden States," IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, May 31–June 7, pp. 4009–4014.
[24] Grisetti, G., Stachniss, C., and Burgard, W., 2007, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," IEEE Trans. Rob., **23**(1), pp. 34–45.