

# Unsupervised Learning for Nonlinear PieceWise Smooth Hybrid Systems

Gilwoo Lee<sup>\*</sup>, Zita Marinho<sup>\*\*</sup>, Aaron M. Johnson<sup>\*\*</sup>,

Geoffrey J. Gordon<sup>\*\*</sup>, Siddhartha S. Srinivasa<sup>\*</sup>, and Matthew T. Mason<sup>\*\*</sup>

<sup>\*</sup> Paul G. Allen School of Computer Science & Engineering  
University of Washington  
Seattle, WA 98195

<sup>\*\*</sup>Robotics Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213

**Abstract**—This paper introduces a novel system identification and tracking method for PieceWise Smooth (PWS) nonlinear stochastic hybrid systems. We are able to correctly identify and track challenging problems with diverse dynamics and low dimensional transitions. We exploit the composite structure system to learn a simpler model on each component/mode. We use Gaussian Process Regression techniques to learn smooth, nonlinear manifolds across mode transitions, *guard-regions*, and make multi-step ahead predictions on each mode dynamics. We combine a PWS non-linear model with a particle filter to effectively track multi-modal transitions. We further use synthetic oversampling techniques to address the challenge of detecting mode transition which is sparse compared to mode dynamics. This work provides an effective form of model learning in complex hybrid system, which can be useful for future integration in a reinforcement learning setting. We compare multi-step prediction and tracking performance against traditional dynamical system tracking methods, such as EKF and Switching Gaussian Processes, and show that this framework performs significantly better, being able to correctly track complex dynamics with sparse transitions.

## I. INTRODUCTION

Having a good model of the interaction between a robot and the environment is critical in reinforcement learning, as it depends on expected reward based on the probability of predicted trajectories [1]. Poor models lead to false approximations of those probabilities, which in turn lead to poor policies. In receding horizon control we also rely on good model estimates to compute locally optimal policies within short timeframes [2]. While reinforcement learning and optimal control are active fields of research, most existing work assumes that the underlying dynamics is smooth, and application to Hybrid Systems has been limited[3, 4]. In the context of robotics, deterministic hybrid models are often used for locomotion [5], object manipulation. Although some non-parametric, data-driven approaches to estimate hybrid systems have been proposed [6, 7], they assume overly simplified simplified models such as Gaussian Mixture Model or neglect the sparsity of mode transitions in the data.

In the field of hybrid systems, many system identification models has been studied for hybrid systems, but most of them are focused on Switched Systems [8, 9, 10, 11], in which

mode transition is independent of continuous state. These approaches for Switched Systems often perform poorly on PWS-HS, and identification of PWS-HS has been limited to those with affine dynamics for subsystems or linearly separable subsystems [12]. Others assume system dynamics can be provided in a closed form solution [13], which is limited in its application when we need to learn complex systems with little prior knowledge about the underlying models.

The main contribution of this paper is to present a unsupervised, nonparametric framework that can learn *nonlinear* PWS-HS systems with little prior knowledge of the underlying subsystems. Doing so is challenging in two ways. First, the domains of the subsystems are unknown, which makes mode assignment and function approximation difficult. Second, mode transitions happen very rarely compared to the non-transitioning counterparts, which makes identification of mode transition condition challenging. In order to address these challenges, we heavily exploit the structure of PWS-HS, i.e. piecewise smoothness and the existence of guard regions, which allow us to use simple non-parametric clustering such as Spectral Clustering[14] and function regression methods, such as Gaussian Processes[15]. To address the challenge in the sparsity of transition points, we use oversampling techniques in the transition region. By iterative mode assignment and subsystem estimation, our system converges to a close approximation of the underlying hybrid system.

The rest of this paper is structured as following. First we define our problem and explain PWS-HS. Then we introduce how our learning system of PWS-HS, and how it uses a particle filter to predict future observations and track them. We evaluate our system in two experiments, in which we compare it with existing methods such as Single GP, Switching GPs[16], and Extended Kalman Filter [17].

## II. PROBLEM STATEMENT

The goal of our framework is to learn a nonlinear PWS-HS from data when little prior model knowledge is available.

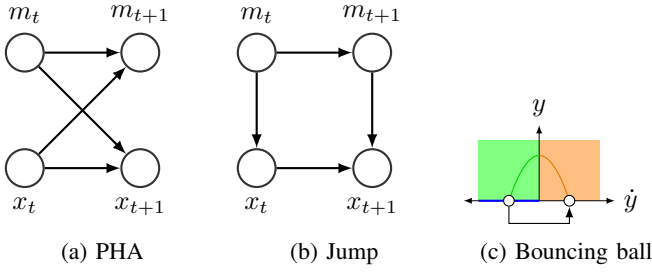


Figure 1: Probabilistic Hybrid Automata PHA (left). Switched System (middle). model structure:  $m_t$  is a discrete mode,  $x_t$  is a continuous state undergoing dynamics defined by  $m_t$ . Probability of mode transition at time  $t$  is defined both by the current mode  $m_t$  and continuous state  $x_t$ .

### A. PieceWise Smooth Hybrid Systems

A state in a PWS-HS is defined as a tuple  $(m, x)$ . Here,  $m \in M$  is a *mode* chosen from a set of discrete modes  $M$ . Each mode is associated with a continuous, connected subspace  $F_m \subseteq \mathbb{R}^d$ , and  $x \in F_m$  is an element of that subspace. We assume the subspaces do not overlap, so  $F_m \cap F_n = \emptyset$  if  $m \neq n$ .

Each mode  $m$  is also associated with a discrete-time continuous function  $f_m \in \mathcal{F} : F_m \rightarrow \mathbb{R}^d$ , which determines the evolution of  $x$  within the mode  $m$ .

We also define a *guard region*, denoted as  $G_{m,m'} \subseteq F_m$  in which transition occurs deterministically from mode  $m$  to a different mode  $m'$ . When  $x_t$  in mode  $m$  reaches  $G_{m,m'}$ , it teleports to  $x_{t+1}$  through a *reset function*  $r_{m,m'} \in \mathcal{F}$ .

In short, the continuous state propagates in the following form:

$$x_{t+1} = \begin{cases} r_{m_t, m_{t+1}}(x_t) + w_t & \text{if } x_t \in G_{m_t, m_{t+1}} \\ f_{m_t}(x_t) + w_t & \text{if } x_t \in F_{m_t} \end{cases} \quad (1)$$

where  $w_t \sim \mathcal{N}(0, \sigma_w)$  is additive process noise. Although our model does not explicitly model control input  $u$ , it is straightforward to incorporate.

Consider, for example, a perfectly elastic bouncing ball (Figure 1c). The ball undergoes a nonsmooth transition of state when it hits the ground and its velocity changes sign instantaneously. We can describe this system as having two modes, with transition between the modes upon collision.<sup>1</sup> The bouncing ball clearly illustrates our challenge. Although the dynamics of each mode can be easily learned, it is challenging to learn the guard region, since these are typically low dimensional or sparse. In this case the switching point is instantaneous and the probability of a discrete time step hitting exactly the bouncing moment is approximately zero. Due to this challenging property of mode transitions, many existing SHS identification tools often fail to learn the mode transitions or handle it incorrectly. This in turn makes the model fail to

<sup>1</sup>We can interpret this model as having a single mode with self-reset at the bouncing point. For simplicity, we disallow self-resets in our model and view this as two modes with one mode transition.

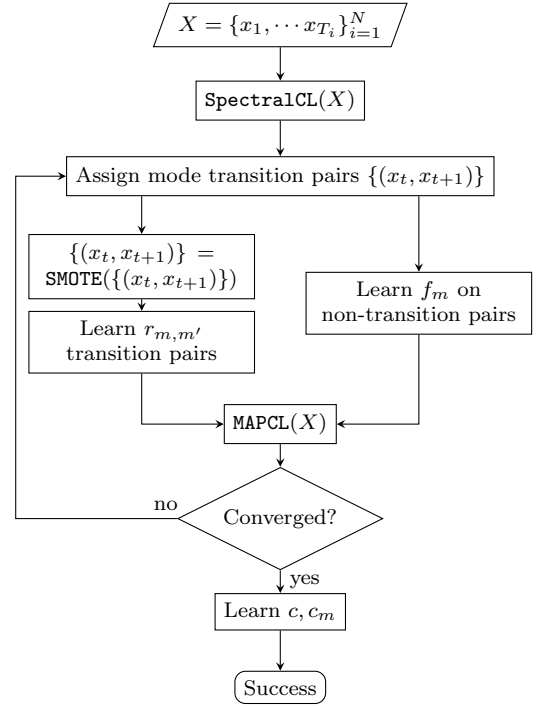


Figure 2: Flowchart for system identification and learning §III.

make any reasonable prediction near mode transitions. In this work we propose to exploit the structure of PWS systems, and design a framework that addresses this challenge.

We now formally state our problem:

We observe the continuous state with some additive noise  $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon)$  in observation. We then collect a set of  $n$  trajectories, with each trajectory defined as  $X_i = \{x_1^i, \dots, x_{T_i}^i\}$ . Our goal is to learn a PWS-HS, i.e., the number of modes, the smooth system dynamics within a mode, and the guard regions and reset function for each transition.

### III. LEARNING PWS HYBRID SYSTEMS

Assuming that the number of modes is given, learning a PWS Hybrid System can be decomposed into:

- 1) **Mode identification:** Determine subsets of state space governed by each mode.
- 2) **Learn a model per mode:** Estimate the dynamics of each mode.
- 3) **Guard estimation:** Identify mode transition regions.
- 4) **Learn a reset mapping:** Estimate reset dynamics for mode transitions.
- 5) **Iteratively mode assignment:** Compute MAP estimates of mode for each observed dynamics.
- 6) **Classification:** Learn classifiers for each mode prediction and transition.

Our approach is focused on constructing a model that closely matches the definition of a PWS-HS, including explicit mode of guard regions and reset dynamics, which many existing approaches fail to address. By doing so, we are able to

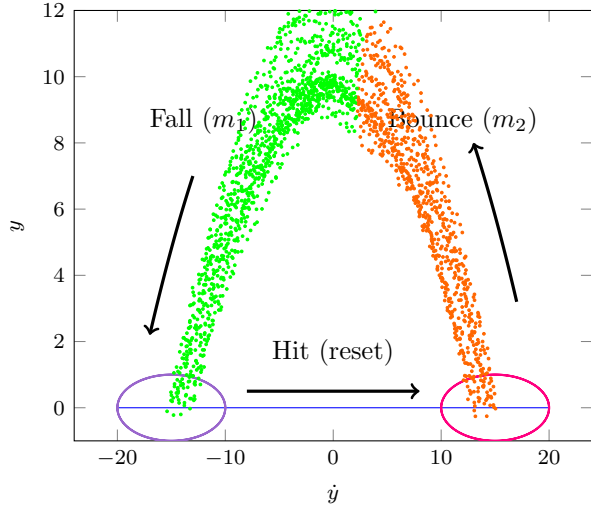


Figure 3: Initial mode assignment (§III-A) for bouncing ball trajectories (§V-A). The ball falls ( $m_1$ ), reaches the **guard region**, jumps to **post-guard region**, and bounces up ( $m_2$ ).

leverage the structure of the model and learn more effectively with less prior information and data. We learn each component and each transition separately, and hope to achieve a more accurate PWS hybrid model. We design an iterative process that predicts maximum a posteriori mode estimate for each point in the dataset, in a fully unsupervised approach.

We start with  $n$  trajectories, with each trajectory defined as  $X_i = \{x_1^i, \dots, x_{T_i}^i\}$ . We first assign modes to each trajectory of points using a spectral clustering algorithm (§III-A). Then, for each sequence, we identify mode transition pairs and oversample them (§III-B). We then learn both mode dynamics (§III-C) and the reset mapping (§III-D) through Gaussian Process (GP), and reassign the modes based on the predictions of each GP (§III-E). Once this process converges, we train the mode and guard classifiers using Logistic Regression (§III-F). In the following section, we describe the details of this procedure.

#### A. Initial Mode Assignment

Spectral Clustering (SC) is a powerful clustering technique that can be used for clustering nonlinear, smooth manifolds [18]. We assume each mode dynamics is smooth and non-intersecting. In this case SC serves as an effective initial mode assignment tool in our model. Refer to [14], [19] for more detail. Given a set of points  $X = \cup_{i=1}^n X_i$ , we build an affinity matrix using a metric of our choice, in our case given by a Radial Basis Function kernel.<sup>2</sup> From this affinity matrix, we apply K-means algorithm to get the mode assignment.<sup>3</sup> The result is a set of ordered pairs of a continuous state and its respective mode  $\tau_i = \{(m_1^i, x_1^i), \dots, (m_{T_i}^i, x_{T_i}^i)\}$  for each sequence  $X_i$ .

<sup>2</sup> $k(x_i, x_j) = \exp(-\beta_0 \|x_i - x_j\|^2)$

<sup>3</sup>We assume the number of modes is known a priori, but it can be estimated using an eigengap heuristic [20, 21].

Figure 3 shows the initial clustering result given by SC with the number of modes set to 2. The algorithm clusters most of the falling trajectory as one mode and the bouncing trajectory as another. Note that at the top where the ball changes its velocity from positive to negative, the mode transition is smooth, and thus mode assignment in this region is slightly different every time we cluster. This different initialization does not affect the rest of our framework, as we only require the dynamics within each mode to be smooth.

#### B. Guard Identification

After mode assignments are complete, we identify mode transitions. We create pairs  $(x_t, x_{t+1})$  along all trajectories  $X_i$  and create an augmented matrix

$$X_i^{-+} = [X_i^- \quad X_i^+] = \begin{bmatrix} x_1^i & x_2^i \\ \vdots & \vdots \\ x_{T_i-1}^i & x_{T_i}^i \end{bmatrix}$$

Note that from the previous step of mode assignment, each of these points has assigned modes  $(m_t^i, m_{t+1}^i)$ . Hence, for each mode  $m$ , we collect all pairs with the same mode ( $m_t^i = m, m_{t+1}^i = m$ ) in  $\hat{F}_m^{-+}$ , and collect all pairs with different modes ( $m_t^i = m, m_{t+1}^i = m'$ ) in  $\hat{G}_{m,m'}^{-+}$ . For example, in figure 3, all green points are in  $m_1$  and all orange points are in  $m_2$ , and all pairs of points from  $m_1$  to  $m_2$  are in  $\hat{G}_{m_1,m_2}^{-+}$ . At the end of this step, we have  $\hat{F}_m^{-+}, \hat{G}_{m,m'}^{-+}$  for all modes and mode transitions.

#### C. Learning Mode Dynamics through Gaussian Process Regression

From the paired points in  $\hat{F}_m^{-+}$ , we can learn the dynamics  $f_m$  of subsystem  $m$ . Recall that  $\hat{F}_m^{-+}$  is a set of tuples of  $(x_t, x_{t+1})$ , both in mode  $m$ . Let  $\hat{F}_m^-$  be the first column of  $\hat{F}_m^{-+}$ , and  $\hat{F}_m^+$  be second column. Now we use Gaussian Processes (GP) [22] to learn the mapping  $\hat{F}_m^- \rightarrow \hat{F}_m^+$ . A GP models the conditional probability of  $F_m^+$  given  $F_m^-$  as a normal distribution. Once the GP is trained, given a new test point  $\tilde{x}_t$  in mode  $m$ , it produces the probability distribution of  $\tilde{x}_{t+1}$  as a normal distribution  $\mathcal{N}(\mu_m(\tilde{x}_t), \Sigma_m(\tilde{x}_t))$ , with parameters:

$$\begin{aligned} \mu_m(\tilde{x}_t) &= K(\tilde{x}_t, \hat{F}_m^-)^\top K_{FF}^{-1} \hat{F}_m^+ \\ \Sigma_m(\tilde{x}_t) &= K_{\tilde{x}\tilde{x}} - K(\tilde{x}_t, \hat{F}_m^-)^\top K_{FF}^{-1} K(\tilde{x}_t, \hat{F}_m^-) \end{aligned} \quad (2)$$

where  $K(a, b)$  is the Gram matrix defined by the kernel function of our choice<sup>4</sup>,  $K_{\tilde{x}\tilde{x}} = K(\tilde{x}_t, \tilde{x}_t)$ , and  $K_{FF} = K(\hat{F}_m^-, \hat{F}_m^-)$ . Throughout the rest of the paper we will denote discrete mode prediction and continuous dynamics during test time as  $\tilde{m}_t, \tilde{x}_t$ , while  $x_t, m_t$  is reserved for training samples. We choose kernel parameters  $\beta = (\beta_0, \beta_1)$  to maximize the likelihood of  $P(\hat{F}^+ | \hat{X}^-; \beta)$ . See [23] for a full derivation.

Note that GP is in fact capable of learning non-smooth, discontinuous functions using complex, non-stationary kernels.

<sup>4</sup>We use RBF kernel with additional Kronecker delta term for signal variance:  $k(x_i, x_j) = \exp(-\beta_0 \|x_i - x_j\|^2) + \beta_1 \delta_{ij}$

However, this requires stronger prior knowledge of the underlying model or large amounts of data. On the contrary, by exploiting the structure of PWS-HS, we can use GP with simple kernels such as RBF to learn each mode and combine them to learn a complicated system as a whole. This approach is similar to the Mixture of Gaussian Processes introduced in [15].

#### D. Learning Reset Dynamics with Oversampling

Ideally, we would like to learn reset dynamics across mode transitions from  $G_{m,m'}^{-+}$  in a similar manner as we used  $F_m^{-+}$  for mode dynamics (§III-C). However, this is challenging because these mode transitions are sparse. For example, in the case of the bouncing ball (figure 3) only one mode transition occurs per cycle of ball bounce. To address this challenge, we use a common Machine Learning technique called *oversampling*.

Extending the Synthetic Minority Over-sampling TEchnique (SMOTE) [24] algorithm originally proposed for classification, we generate synthetic samples of tuples in  $G_{m,m'}^{-+}$ . First, we randomly choose two tuples  $(x_{t_1}, x_{t_1+1}), (x_{t_2}, x_{t_2+1})$  from  $G_{m,m'}^{-+}$ . Then, we generate a new synthetic point by convex combination of the two tuples:

$$\begin{bmatrix} x_t \\ x_{t+1} \end{bmatrix} = r \begin{bmatrix} x_{t_1} \\ x_{t_1+1} \end{bmatrix} + (1-r) \begin{bmatrix} x_{t_2} \\ x_{t_2+1} \end{bmatrix} \quad (3)$$

where  $r$  is randomly chosen from the unit interval  $[0, 1]$ .

Once enough synthetic tuples are generated, we can learn  $r_{m,m'}$  using the same process as §III-C, where instead we regress the second column of  $G_{m,m'}^{-+}$  on the first column.

#### E. Iterative Mode Assignment and Model Learning

When mode boundaries overlap due to smooth transitions or noise, initial mode assignments may be incorrect for some of the points, especially because we do not take into account time-continuity of points along a trajectory at the initial clustering stage. Similar to [12], we can fix these mis-assignments by re-assigning modes based on the trained GPs' predictions. After training all GPs, we revisit all points in  $X$  and re-assign modes as:

$$m_t^{i,new} = \arg \max_m p(x_{t+1}^i | x_t^i; \mu_{g_m}(x_t), \Sigma_{g_m}(x_t))$$

where  $g_m \in \{f_m\} \oplus \cup_{m'} r_{m,m'}$  is a GP from the set of all dynamics related to mode  $m$ , i.e., the set of all reset dynamics from  $m$  and the mode dynamics  $f_m$ . We refer to this algorithm as MAPCL in figure ??.

We iterate the process of guard identification, oversampling, GP training, and mode assignment until mode assignment does not change.

#### F. Mode and Guard Classification

Once the learning process has converged, we finally train classifiers for modes and guards. These can be learned by one classifier, but to explicitly encode the dependency of the current mode and state in the mode prediction for the next time step, we train two level of classifiers. The first level of

classification being for mode prediction and then the second level being for guards within each mode. All classifiers return probabilities across possible choices.

The first level mode classifier is defined as  $c : \mathbb{R}^d \rightarrow \Delta^{|M|-1}$ , and is trained using all points in  $X$  and their assigned modes. The second level classifier is defined per mode:

$$c_m : x \in F_m \rightarrow \Delta^{|M|-1},$$

and we use all points in the guard regions of  $m$ , i.e.  $\cup G_{m,m'}^- \forall m'$ , as well as the synthetic points generated in §III-D. We use balanced Logistic Regression [25] for both levels, but the choice of classifier is independent of our framework as long as it provides probabilistic mode estimation and is capable of learning unbalanced data sets.

#### IV. MULTI-STEP PREDICTION AND TRACKING

Our learned PWS-HS can be used for multi-step prediction and tracking. We combine our learned hybrid model with a Sequential Importance Sampling Particle Filter [17]. While closed form updates of mean and covariance for multi-step predictions of a single GP are available [26], these are often insufficient in modeling complex systems. In the case of multi-modal PWS-Hybrid Systems closed form updates are often inaccurate, hence we opt for a Monte Carlo approach. Figure 4 shows a flowchart of our tracking algorithm. For each particle, we first sample the next mode, according to the mode to the learn classifiers, and then propagate the continuous state using the corresponding dynamics function (GP). If an observation is made, it is used to re-weight the particles, and approximate better the target distribution. Within each mode, we expect the propagation to be unimodal, so we approximate each mode's distribution to be Gaussian, producing a multi-modal Gaussian Mixture Model prediction at the end of each iteration. At the end of the process we resample particles according to the updated mode distribution.

#### V. EXPERIMENTS

We evaluate our hybrid learning system (Hybrid)<sup>5</sup> by comparing its prediction and tracking performance with three baseline methods, Single Gaussian Process (GP), a Switching Gaussian Process (Switching) [16], and an Extended Kalman Filter (EKF) [17]. GP assumes that state dynamics is unimodal. Switching assumes that state dynamics is drawn from multiple modes, but assumes that mode evolution is independent of the continuous dynamics. EKF requires knowledge of the model dynamics equation in closed form, and makes mode transitions deterministically based on its state estimation.

For prediction, we first qualitatively compare  $n$ -step ahead prediction of our model with all the other baselines. We show that our hybrid learning system is the only method that makes correct predictions near mode transition. We then quantitatively compare the log likelihood of  $n$ -step ahead predictions and the posterior distribution of predictions given observation for tracking. We show that our model performs

<sup>5</sup>Our system is implemented in Python 2.7.12. For balanced Logistic Regression and Spectral Clustering we used Scikit-learn 0.17.14 [27].

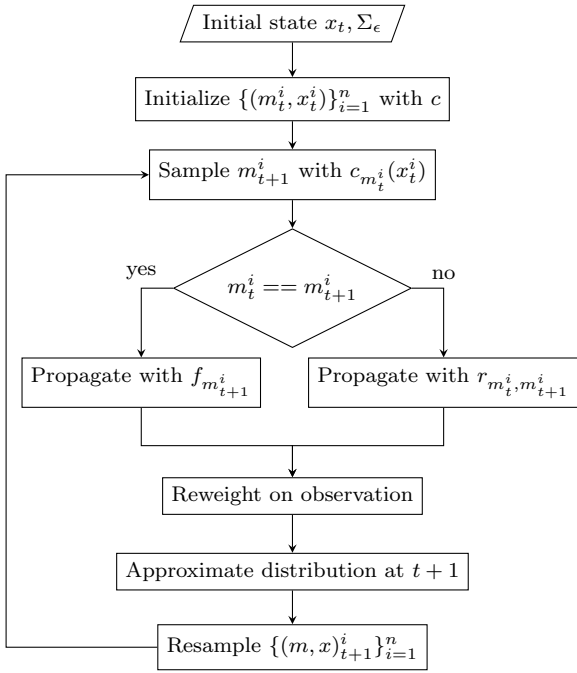


Figure 4: Flowchart for tracking with sequential importance sampling (§IV)

as well as the remaining methods, when the state is far from transitioning, and has the best performance near transition. Both GP and Switching use a sequential importance sampling particle filter similarly to our Hybrid model. We use two synthetic experiments as illustrative examples, but we plan to extend it to real robot experiments in the near future.

#### A. Bouncing ball

We simulate a one-dimensional bouncing ball experiment. We consider state to be  $[y \ \hat{y}]^\top$ , where  $y$  is the height of the ball and  $\hat{y}$  is the velocity, with positive direction pointing upward. We assume the ground is at  $y = 0$  and that collisions are perfectly elastic. Except for the collision with ground, the state is updated by the following equation of motion:

$$\begin{bmatrix} y_{t+1} \\ \hat{y}_{t+1} \end{bmatrix} = \begin{bmatrix} y_t \\ \hat{y}_t \end{bmatrix} + \begin{bmatrix} \hat{y}_t t + gt^2/2 \\ gt \end{bmatrix} + w_t$$

We add process noise  $w_t \sim \mathcal{N}(0, \text{diag}([0.01, 0.01]))$  during the data generation and observation noise of the same magnitude to the training set. We additionally generate 5 trajectories of 100 steps, and use this as test set.

1) *Prediction and Tracking*: We evaluate tracking performance near the transition region, see Figure 5. We compare four stages of the bouncing ball experiment out of collision in one mode (ball going down), immediately before collision (guard region), immediately after collision (post-guard), and out of collision in the next mode (ball going upwards). Hybrid correctly predicts that the ball will continue to fall until collision. At this point, the algorithm predicts that either the ball will continue to fall or bounce, with stronger belief for bouncing in the next step. Note that this bi-modal prediction is

an accurate representation of how the bouncing ball behaves near collision given noisy observations. This multi-modality is particularly useful in manipulation planning. Once collision is made, the algorithm predicts that the ball will continue to move upward. In all four stages our method correctly predicts the velocity of the ball, giving as well a correct bi-modal distribution at the collision points (guard and post-guard). EKF and GP make unimodal predictions, and Switching makes incorrect multimodal predictions in the post-guard region, not being able to correctly map to the different dynamics in the new mode.

We also compare the prediction performance of our method, by evaluating the log likelihood of 2-steps ahead prediction and compare with the other baselines, in Figure 6. We show log-likelihood of observations after updating the probability distribution given the observation. When there is no mode transition, all methods make similar predictions, though Hybrid and Switching perform better than the other two methods, since they learn the system’s dynamics of each mode separately. EKF performs best during this no transition region, which is expected since it is given the correct equations of motion. However, near transition, EKF performs poorly, since the initial prediction is often too far from what is observed. During transition, Switching performs better in tracking than in pure prediction, see Figure 6. This is due to an increased weight for particles in the correct mode after the observation update. Nonetheless, Hybrid performs better than any other methods during mode transition, as none of the others correctly identifies the reset (bouncing).

#### B. Box pushing

In this second experiment, we consider a simplified robot pushing a box, depicted in Figure 7. Here, a circular robot of radius  $2\text{cm}$ , and a box of size  $4\text{cm} \times 4\text{cm}$  sitting  $6\text{cm}$  apart from the robot. The robot moves in the positive  $x$  direction for  $1\text{s}$ , with velocity  $4\text{cm/s}$ , and stays still for the next  $1\text{s}$ . 30 trajectories were generated for training with additive observation noise  $\mathcal{N}(0, 0.1)$ , and 6 of them were used for testing. The state space has 5 dimensions,  $[x_o, v_{x,o}, x_r, y_r, v_{x,r}]$ , where  $x_o, v_{x,o}, x_r, y_r, v_{x,r}$  describe object and robot’s pose and velocity, respectively. Both the robot and object move only in the  $x$  direction. The robot selects initial height from normal distribution, i.e.,  $y_r \sim \mathcal{N}(6\text{cm}, 0.5)$ , which results in successfully pushing the box half of the time.

For EKF, we use the following equations of motion:

$$\begin{bmatrix} x_o \\ x_r \end{bmatrix}^{(t+1)} = \begin{bmatrix} x_o \\ x_r \end{bmatrix}^{(t)} + \begin{bmatrix} v_{x,o} \\ v_{x,r} \end{bmatrix}^{(t)} t_{step}$$

$$\begin{cases} v_o^{(t+1)} = v_{x,r} & \text{if } x_o - x_r \geq 4, y_r \geq 6 \\ v_o^{(t+1)} = 0 & \text{otherwise} \end{cases}$$

where  $t_{step}$  is the step size,  $0.2\text{s}$ .

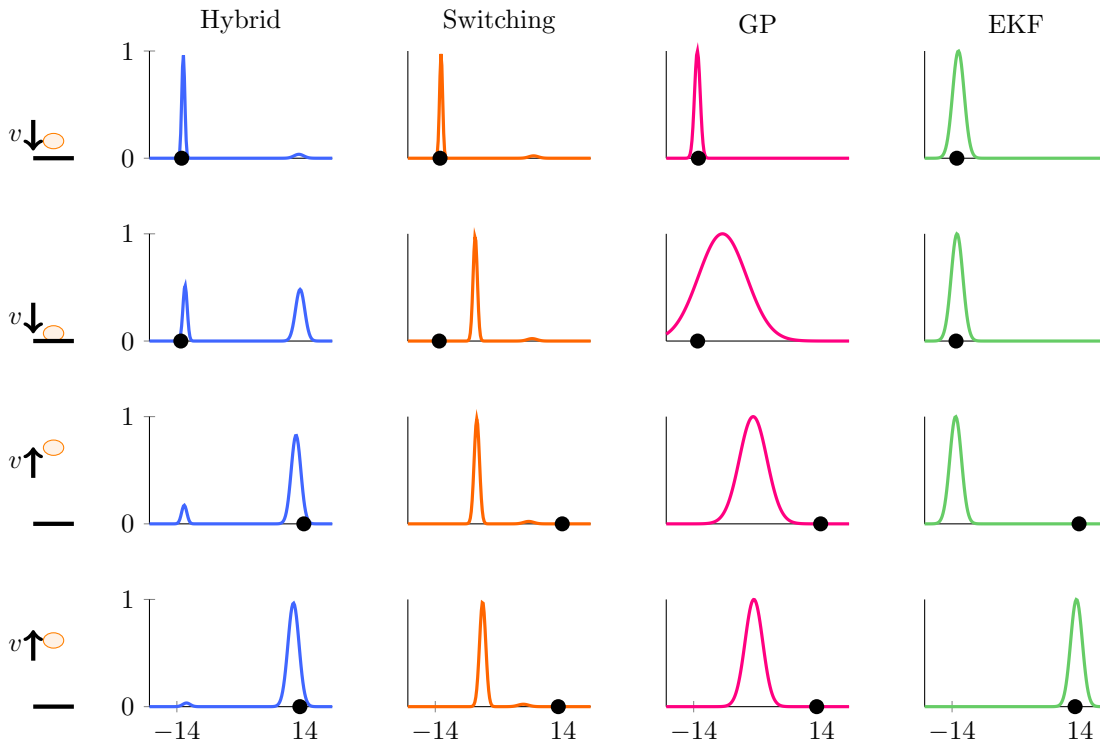


Figure 5: Probability distribution for  $v_{t+1}$  prediction, made by Hybrid, Switching, GP and EKF (from left to right). Dot on each x-axis represents the observation. Falling ball (top), immediately before collision (middle top), immediately after collision (middle bottom), ball going up (bottom).

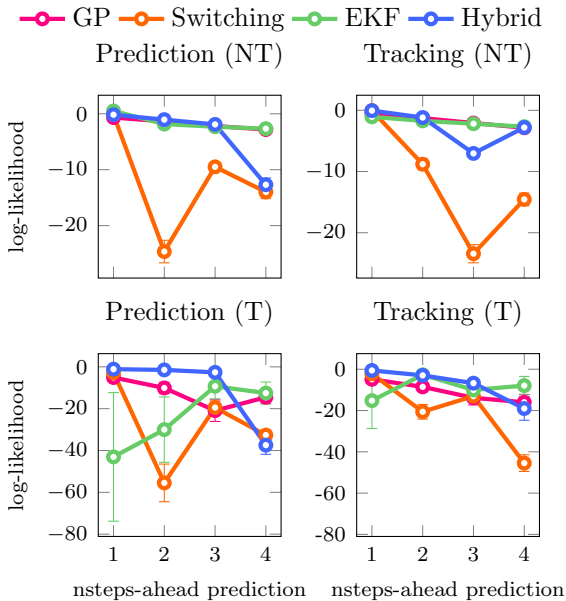


Figure 6: Bouncing ball log-likelihood of velocity prediction vs. n-time steps ahead for prediction (left) and tracking (right). Hybrid (blue), GP (red), Switching (orange) and EKF (purple). On each mode subspace—no transition (top), at the transition region (bottom).

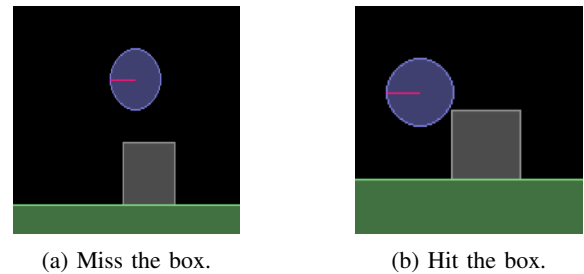


Figure 7: The box is dragged by the manipulator 50% of the time, and slows down over until it stops (right). The other half the box remains at rest (left).

1) *Prediction and tracking*: Figure 8 compares the log likelihood of 1 and 2-step ahead predictions made by the four algorithms. As before, we distinguish predictions near transition from the rest. When there is no mode transition Hybrid, GP, and Switching are perform almost equally well. However near transition, Hybrid outperforms the other methods. We do not report EKF's performance since it is much worse than other methods because it's equation of motion is deterministic on what it believes is the true mode, and so it is more prone to errors. Moreover, as in the previous experiments, we are able to observe successful multi-modal predictions near transition for this task, similarly to Figure 5. These synthetic experiments show the effectiveness of our

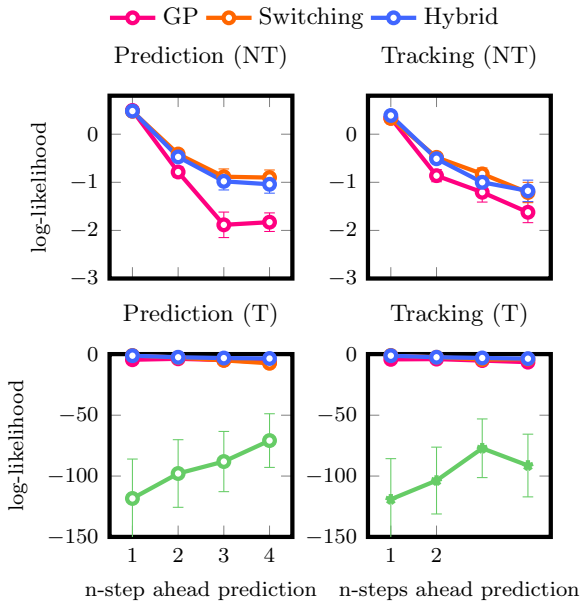


Figure 8: Pushing box log-likelihood of velocity prediction vs. n-time steps ahead for prediction (left) and tracking (right). Hybrid (blue), GP (red), Switching (orange) and EKF (purple). On each mode subspace—no transition (top), at the transition region (bottom).

method, we are currently working on experiments with more realistic robotic manipulation experiments. Nevertheless, with our hybrid system we are able to make multi-modal predictions near transition, capturing the actual distribution of trajectories near mode transitions, and outperforming current state of the art tracking algorithms.

## VI. DISCUSSION

We have presented an unsupervised learning framework for PieceWise Smooth Hybrid Systems, which can be used to model the underlying dynamics of many robot systems in which the robot interacts with environment. This work addresses a complex robotic problem and a challenging learn task. We provide an algorithm that is able to learn hybrid dynamics in an unsupervised manner, we exploit the structure of the system to detect guard regions and learn mode and reset dynamics separately, in an iterative process. We have shown through our experiments that this model captures the multimodality of the dynamics near mode transitions, which many existing techniques fail to capture. We believe that having a proper hybrid systems model is crucial in generating policies for robots interacting with the environment, and we are excited to extend this work in such direction in the near future.

## REFERENCES

[1] J. Kober and J. Peters, “Reinforcement learning in robotics: A survey,” in *Reinforcement Learning*. Springer, 2012, pp. 579–610.

[2] R. M. Murray, “Optimization-based control.”

[3] M. Posa and R. Tedrake, “Direct trajectory optimization of rigid body dynamical systems through contact,” in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 527–542.

[4] Y. Tassa, “Stochastic complementarity for local control of discontinuous dynamics.”

[5] A. M. Johnson, S. A. Burden, and D. E. Koditschek, “A hybrid systems model for simple manipulation and self-manipulation systems,” *The International Journal of Robotics Research*, p. 0278364916639380, 2016.

[6] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, “Towards learning hierarchical skills for multi-phase manipulation tasks,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1503–1510.

[7] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[8] F. Lauer and G. Bloch, “Switched and piecewise nonlinear hybrid system identification,” in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2008, pp. 330–343.

[9] F. Lauer, G. Bloch, and R. Vidal, “Nonlinear hybrid system identification with kernel models,” in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 696–701.

[10] F. Lauer and G. Bloch, “Piecewise smooth system identification in reproducing kernel hilbert space,” in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6498–6503.

[11] G. Bloch, F. Lauer *et al.*, “Reduced-size kernel models for nonlinear hybrid system identification,” *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2398–2405, 2011.

[12] P. Santana, S. Lane, E. Timmons, B. C. Williams, and C. Forster, “Learning hybrid models with guarded transitions,” in *AAAI*, 2015, pp. 1847–1853.

[13] A. Núñez, B. De Schutter, D. Sáez, and I. Škrjanc, “Hybrid-fuzzy modeling and identification,” *Applied Soft Computing*, vol. 17, pp. 67–78, 2014.

[14] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[15] C. E. Rasmussen and Z. Ghahramani, “Infinite mixtures of gaussian process experts,” 2002.

[16] J. Chen, M. Kim, Y. Wang, and Q. Ji, “Switching gaussian process dynamic models for simultaneous composite motion tracking and recognition,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2655–2662.

[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, 2005.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in*

- Neural Information Processing Systems 14*. MIT Press, 2001, pp. 849–856.
- [19] M. I. Jordan, F. R. Bach, and F. R. Bach, “Learning spectral clustering,” in *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.
- [20] M. Soltanolkotabi, E. Elhamifar, and E. J. Candes, “Robust subspace clustering,” *The Annals of Statistics*, vol. 42, no. 2, pp. 669–699, 04 2014.
- [21] B. Mohar, *Some applications of Laplace eigenvalues of graphs, Graph Symmetry: Algebraic Methods and Applications*. Springer Netherlands, 1997.
- [22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
- [23] N. D. Lawrence, “Gaussian process latent variable models for visualisation of high dimensional data,” 2004.
- [24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
- [25] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.
- [26] A. Girard, C. E. Rasmussen, J. Quinonero-Candela, and R. Murray-Smith, “Gaussian process priors with uncertain inputs? application to multiple-step ahead time series forecasting,” 2003.
- [27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.