2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Learning Predictive Models of a Depth Camera & Manipulator from Raw Execution Traces

Byron Boots Arunkumar Byravan Dieter Fox

***Abstract*— In this paper, we attack the problem of learning a predictive model of a depth camera and manipulator directly from raw execution traces. While the problem of learning manipulator models from visual and proprioceptive data has been addressed before, existing techniques often rely on assumptions about the structure of the robot or tracked features in observation space. We make no such assumptions. Instead, we formulate the problem as that of learning a high-dimensional controlled stochastic process. We leverage recent work on nonparametric predictive state representations to learn a generative model of the depth camera and robotic arm from sequences of uninterpreted actions and observations. We perform several experiments in which we demonstrate that our learned model can accurately predict future depth camera observations in response to sequences of motor commands.**

I. Introduction

One of the most fundamental challenges in robotics is the *general identification problem* [1][1]: a robot, capable of performing a set of actions $a \in \mathcal{A}$ and receiving observations $o \in \mathcal{O}$, is placed in an unknown environment. The robot has no interpretation for its actions or observations and no knowledge of the structure of the environment (Fig. 1). The problem is to program the robot to learn about its observations, actions, and environment well enough to make predictions of future observations given sequences of actions. In other words, the goal is to learn a *generative model* of the observations directly from raw execution traces.

In this paper we investigate an instance of the general identification problem: A robot observes a manipulator under its control with a Kinect RGB-D camera. The goal is to learn a generative model of RGB-D observations as the robot controls its manipulator (Figure 2).

While the problem of learning manipulator models, or body schemas, from visual and proprioceptive modalities has been addressed before, existing techniques rely critically on assumptions about the kinematic structure of the robot and tracked features in observation space [2]–[5]. Here, we address this problem in its most challenging instance: The observations are streams of raw depth images (1.2 million pixels), and the robot has *no* prior knowledge about what it is controlling.

Byron Boots, Arunkumar Byravan, and Dieter Fox are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 {bboots,barun,fox}@cs.washington.edu
This work was supported in part by ONR MURI grant number N00014-09-1-1052 and by the National Science Foundation under contract NSF-NRI 1227234.

[1]The general identification problem was first proposed by Ron Rivest in 1984 and originally called the *Critter Problem*

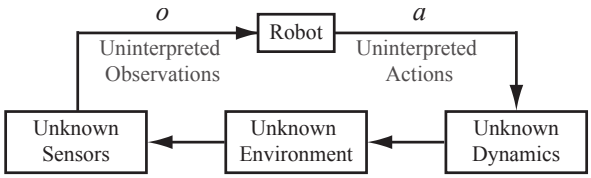

Fig. 1. The problem setup. The robot has access to uninterpreted streams of continuous observations and actions. We do not make *any* assumptions about the observations that the robot receives, the actions the robot can execute, or the structure of the environment.

We approach this difficult problem from a machine learning perspective. We dispense with problem-dependent geometric and physical intuitions and instead model the sensorimotor data as a *Predictive State Representation (PSR)* [6], [7], a general probabilistic modeling framework that can represent a wide variety of stochastic process models including Kalman filters (KFs) [8], input output hidden Markov models (IO-HMMs) [9], [10], and nonparametric dynamical system models [11].

The main contribution of our work is to show that a recent nonparametric variant of PSRs, called Hilbert Space Embeddings of PSRs, can learn a generative model of the RGB-D camera and robotic arm directly from sequences of uninterpreted actions and observations. This problem is *far* more difficult than the simulated problems explored in previous PSR work [10], [12]–[14]. Additionally, the manipulator used in our experiments has many additional degrees of freedom compared to the systems considered in recent work on bootstrapping in robotics [15]–[17].

We run several experiments that show qualitative examples of our learned model *tracking* the current state of the system and *predicting* future RGB-D images given motor commands. We also provide rigorous quantitative results which demonstrate that our learned model is more accurate than competing nonparametric methods at tracking and predicting RGB-D observations given previously unseen sequences of motor commands. To the best of our knowledge this is the first work to learn a model of a depth camera and manipulator directly from raw execution traces.

II. Related Work

Variations of the general identification problem are central to several fields of research, although the assumptions made by different communities are often very different.

In the controls community, the problem of inferring the unknown parameters of an input-output system is called *system identification* [18]–[20]. The system can be deter-

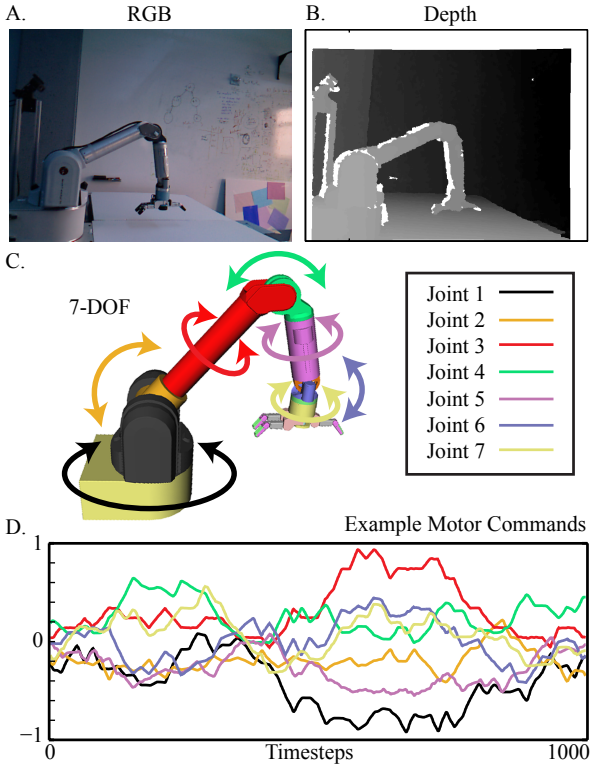

Fig. 2. Observations and Actions. The robot receives sense data from a Kinect RGB-D depth camera, but has no knowledge of the geometry of the scene or the physics of the camera. The robot can control a 7–degree-of-freedom Barrett WAM arm, but has no a priori knowledge of the geometry, kinematics, or any aspects of the action space of the arm. (A) An example $640 \times 480 \times 3$ RGB image. (B) An example 640×480 depth map. Darker indicates increased distance. Together, the RGB-D observation vector has 1228800 elements. (C) The 7 degree-of-freedom arm. Each action is a continuous-valued 7-dimensional vector. (D) Examples of actions executed: 1000 continuous motor encoder values from the training data set. (See Section V for details)

ministic, stochastic, or combined. Usually, either the form of the system (linear, bilinear, etc.), the state space, or the dynamics are assumed to be known in advance. If the system has a stochastic component, the noise is generally assumed to be sampled from a Gaussian distribution.

The robotics community has typically approached the problem of learning a generative model of sensor data, called *bootstrapping*, from a deterministic viewpoint, focusing on the geometry and kinematics of the observation and action spaces. Proposed solutions have focused on the *topology* of the environment [21], the *manifold* structure of observation space [22], or *diffeomorphisms* that describe the effect of taking actions [17]. Although these approaches have been developed specifically for robotic systems, they often make very strong assumptions about the form of (or lack of) stochasticity, the observability of the state space, or the geometric structure of the observations, actions, or environment.

The bootstrapping problem can also be viewed as an extreme form of system identification or self-calibration [15]. And, over the last several years, substantial progress has been made on these problems. For example: learning complex nonlinear *forward kinematics* to predict the consequences of actions [23], [24], or learning *body schemas* that integrate the visual and proprioceptive modalities to discover the kinematic structure of a robot [2]–[4]. While these techniques are able to learn accurate models, they rely heavily on assumptions about the kinematic structure of the robot and tracked features in visual space.

In the machine learning community, where probabilistic models play a central role, the identification problem is typically concerned with learning the parameters of a *controlled stochastic process* that generated the observations [11], [25], [26]. Predictive State Representations (PSRs) [6], [7] are an example of such a stochastic process model.

PSRs were originally developed in the reinforcement learning community and explicitly designed to solve the general identification problem. They have several advantages over previous approaches to bootstrapping in robotics. First, PSRs are more *expressive* than methods like the Spatial Semantic Hierarchy [21] and Diffeomorphism models of sensorimotor cascades [17], and the theory behind PSRs is mature: their subsumption of popular classes of latent variable models is well understood [7]. Second, PSR models are easy to learn. Popular latent variable models of stochastic processes are often learned using heuristics such as Expectation Maximization (EM), which suffer from bad local optima and slow convergence rates. Recent PSR learning algorithms rely on spectral methods [10], [27] and kernel methods [11] which are statistically consistent.

Unfortunately, evaluation of PSRs have long been restricted to learning fairly simple "grid-world" type simulated environments [10], [12]–[14], a fact which has lead to the perception that PSRs are not flexible enough, or that PSR learning algorithms are not efficient enough, to represent high-dimensional raw sensorimotor data [17]. However, in the last few years PSRs and PSR-like models have experienced a marked resurgence as increasingly powerful learning algorithms have been developed [10], [11], [27]–[29].

III. Predictive State Representations

We begin by providing a brief overview of predictive state representations [7], [10] which we use as a generic framework for tackling the general identification problem.

A. Predictive State

A PSR represents the state of a dynamical system as a set of predictions of experiments or *tests* that can be performed in the system. A test of length N is an ordered sequence of future action-observations pairs $\tau = a_1, o_1, \ldots a_N, o_N$ that can be selected and observed at any time t.

A test τ_i is *executed* at time t if we intervene [30] to select the sequence of actions specified by the test $\tau_i^{\mathcal{A}} = a_1, \ldots, a_N$. A test is said to *succeed* at time t if it is executed and the sequence of observations in the test $\tau_i^{\mathcal{O}} = o_1, \ldots, o_N$ matches the observations generated by the system. The *prediction* for test i at time t is the probability of the test succeeding given a history h_t and given that we execute it:[2]

$$\mathbb{P}\left[\tau_{i,t}^{\mathcal{O}} \mid \tau_{i,t}^{\mathcal{A}}, h_t\right] \tag{1}$$

[2]For simplicity, we assume that all probabilities involving actions refer to our PSR as controlled by an arbitrary *blind* or *open-loop* policy [31].

The key idea behind a PSR is that if we know the expected outcomes of executing *all* possible tests, then we know everything there is to know about the state of a dynamical system [7]. In practice we will work with the predictions of some *set* of tests. Let $\mathcal{T} = \{\tau_i\}$ be a set of d tests, then

$$s(h_t) = \left(\mathbb{P}\left[\tau_{i,t}^{\mathcal{O}} \mid \tau_{i,t}^{\mathcal{A}}, h_t\right]\right)_{i=1}^{d} \tag{2}$$

is the *prediction vector* of success probabilities for the tests $\tau_i \in \mathcal{T}$ given a history h_t.

Knowing the success probabilities of some tests may allow us to compute the success probabilities of other tests. That is, given a test τ_l and a prediction vector $s(h_t)$, there may exist a *prediction function* f_{τ_l} such that $\mathbb{P}\left[\tau_l^{\mathcal{O}} \mid \tau_l^{\mathcal{A}}, h_t\right] = f_{\tau_l}(s(h_t))$. In this case, we say $s(h_t)$ is a *sufficient statistic* for $\mathbb{P}\left[\tau_l^{\mathcal{O}} \mid \tau_l^{\mathcal{A}}, h_t\right]$. A *core* set of tests is a set whose prediction vector $s(h_t)$ is a sufficient statistic for the predictions of *all* tests τ_l at time t. Therefore, $s(h_t)$ is a *state* for a PSR.

B. Bayes Filtering

After taking action a and seeing observation o, we can update the predictive state $s(h_t)$ to the state $s(h_{t+1})$ using Bayes' rule. The key idea is that the set of functions $\mathcal{F}$ allows us to predict *any* test from our core set of tests.

The state update proceeds as follows: first, we predict the success of any core test τ_i prepended by an action a and an observation o, which we call $ao\tau_i$, as a function of our core test predictions $s(h_t)$:

$$\mathbb{P}\left[\tau_{i,t+1}^{\mathcal{O}}, o_t{=}o \mid \tau_{i,t+1}^{\mathcal{A}}, a_t{=}a, h_t\right] = f_{ao\tau_i}(s(h_t)) \tag{3}$$

Second, we predict the likelihood of any observation o given that we select action a (i.e., the test ao):

$$\mathbb{P}\left[o_t = o \mid a_t = a, h_t\right] = f_{ao}(s(h_t)) \tag{4}$$

After executing action a and seeing observation o, Equations 3 and 4 allow us to find the prediction for a core test τ_i from $s(h_t)$ using Bayes' Rule:

$$s_i(h_{t+1}) = \frac{f_{ao\tau_i}(s(h_t))}{f_{ao}(s(h_t))} \tag{5}$$

This recursive application of Bayes' rule to the predictive belief state is an instance of a *Bayes filter*.

The predictive state and the Bayes' rule state update together provide a very general framework for modeling dynamical systems. In the next section we show how a recent variant of PSRs can be used to learn models of dynamical systems with high-dimensional continuous actions and observations.

IV. Hilbert Space Embeddings of PSRs

PSRs generally either assume small discrete sets of actions $\mathcal{A}$ and observations $\mathcal{O}$ along with linear prediction functions $f_\tau \in \mathcal{F}$ [10], or if the actions and observations are continuous, they assume Gaussian distributions and linear functions [8]. Researchers have relied heavily on these assumptions in order to devise computationally and statistically efficient learning algorithms. Unfortunately, such restrictions can be particularly unsuitable for robotics applications.

Instead, we consider a recent generalization of PSRs for continuous actions and observations called *Hilbert Space Embeddings of PSRs* (HSE-PSRs) [11]. The essence of the method is to represent probability distributions of tests, observations, and actions as elements in a Hilbert space of functions, defined through a chosen kernel. The distributions are learned nonparametrically from samples and no assumptions are made about the shape of the underlying distributions. This results in an extremely flexible model. A HSE-PSR is capable of modeling non-linear dynamics and estimating multi-modal distributions for continuous or discrete random variables without having to contend with problems such as density estimation and numerical integration. During filtering these points are conceptually updated entirely in Hilbert space using a kernel version of Bayes' rule. In practice, the "kernel trick" is leveraged to represent the state and required operators implicitly and to maintain a state vector with length proportional to the size of the training dataset.

In the following subsections, we provide a very brief overview HSE-PSRs. We ask the reader to refer to [11] for a more complete treatment.

A. Hilbert Space Embeddings of Distributions

Let $\mathcal{F}$ be a reproducing kernel Hilbert space (RKHS) associated with kernel $K_X(x, x') \stackrel{\text{def}}{=} \left\langle \phi^X(x), \phi^X(x') \right\rangle_{\mathcal{F}}$ for $x \in \mathcal{X}$. Let $\mathcal{P}$ be the set of probability distributions on $\mathcal{X}$, and X be a random variable with distribution $\mathbb{P} \in \mathcal{P}$. Following Smola et al. [32], we define the mean map (or the embedding) of $\mathbb{P} \in \mathcal{P}$ into RKHS $\mathcal{F}$ to be $\mu_X \stackrel{\text{def}}{=} \mathbb{E}\left[\phi^X(X)\right]$. A *characteristic* RKHS is one for which the mean map is injective: that is, each distribution $\mathbb{P}$ has a unique embedding [33]. This property holds for many commonly used kernels including the Gaussian RBF kernel when $\mathcal{X} = \mathbb{R}^d$. Given *i.i.d.* observations x_t, $t = 1 \dots T$, an estimate of the mean map is:

$$\hat{\mu}_X \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=1}^{T} \phi^X(x_t) = \frac{1}{T}\Upsilon^X \mathbf{1}_T \tag{6}$$

where $\Upsilon^X \stackrel{\text{def}}{=} (\phi^X(x_1), \dots, \phi^X(x_T))$ is the linear operator which maps the tth unit vector of $\mathbb{R}^T$ to $\phi^X(x_t)$.

Below, we'll sometimes need to embed a joint distribution $\mathbb{P}[X, Y]$. It is natural to embed $\mathbb{P}[X, Y]$ into a tensor product RKHS: let $K_Y(y, y') = \left\langle \phi^Y(y), \phi^Y(y') \right\rangle_{\mathcal{G}}$ be a kernel on $\mathcal{Y}$ with associated RKHS $\mathcal{G}$. Then we write μ_{XY} for the mean map of $\mathbb{P}[X, Y]$ under the kernel $K_{XY}((x, y), (x', y')) \stackrel{\text{def}}{=} K_X(x, x')K_Y(y, y')$ for the tensor product RKHS $\mathcal{F} \otimes \mathcal{G}$. We also define the *uncentered* covariance operator $\mathcal{C}_{XY} \stackrel{\text{def}}{=} \mathbb{E}_{XY}\left[\phi^X(X) \otimes \phi^Y(Y)\right]$. Both μ_{XY} and $\mathcal{C}_{XY}$ represent the distribution $\mathbb{P}[X, Y]$. One is defined as an element of $\mathcal{F} \otimes \mathcal{G}$, and the other as a linear operator from $\mathcal{G}$ to $\mathcal{F}$, but they are isomorphic under the standard identification of these spaces [34], so we abuse notation and write $\mu_{XY} = \mathcal{C}_{XY}$. Given T *i.i.d.* pairs of observations (x_t, y_t), define $\Upsilon^X = \left(\phi^X(x_1), \dots, \phi^X(x_T)\right)$ and $\Upsilon^Y = \left(\phi^Y(y_1), \dots, \phi^Y(y_T)\right)$. Write Υ^* for the adjoint of Υ. Analogous to (6), we can estimate

$$\widehat{\mathcal{C}}_{XY} = \frac{1}{T}\Upsilon^X \Upsilon^{Y*}. \tag{7}$$

B. Kernel Bayes' Rule

We now define the kernel mean map implementation of Bayes' rule (called the Kernel Bayes' Rule, or KBR). In particular, we want the kernel analog of $\mathbb{P}[X \mid y, z] = \mathbb{P}[X, y \mid z] / \mathbb{P}[y \mid z]$. In deriving the kernel realization of this rule we need the kernel mean representation of a conditional joint probability $\mathbb{P}[X, Y \mid z]$. Given Hilbert spaces $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ corresponding to the random variables X, Y, and Z respectively, $\mathbb{P}[X, Y \mid z]$ can be represented as a mean map $\mu_{XY|z} \stackrel{\text{def}}{=} \mathbb{E}\left[\phi^X(X) \otimes \phi^Y(Y) \mid z\right]$ or the corresponding operator $\mathcal{C}_{XY|z}$. Under some assumptions [34], this operator satisfies:

$$\mathcal{C}_{XY|z} = \mu_{XY|z} \stackrel{\text{def}}{=} \mathcal{C}_{(XY)Z}\mathcal{C}_{ZZ}^{-1}\phi(z) \tag{8}$$

Here the operator $\mathcal{C}_{(XY)Z}$ represents the covariance of the random variable (X, Y) with the random variable Z. We now define KBR in terms of conditional covariance operators [34]:

$$\mu_{X|y,z} = \mathcal{C}_{XY|z}\mathcal{C}_{YY|z}^{-1}\phi(y) \tag{9}$$

To use KBR in practice, we need to estimate the operators on the RKHS of (9) from data. Given T *i.i.d.* triples (x_t, y_t, z_t) from $\mathbb{P}[X, Y, Z]$, write $\Upsilon^X = \left(\phi^X(x_1), \ldots, \phi^X(x_T)\right)$, $\Upsilon^Y = \left(\phi^Y(y_1), \ldots, \phi^Y(y_T)\right)$, and $\Upsilon^Z = \left(\phi^Z(z_1), \ldots, \phi^Z(z_T)\right)$. We can now estimate the covariance operators $\widehat{\mathcal{C}}_{XY|z}$ and $\widehat{\mathcal{C}}_{YY|z}$ via Equation 8 and then apply KBR via Equation 9. We express this process with Gram matrices, using a ridge parameter λ that goes to zero at an appropriate rate with T [34]:

$$\Lambda_z = \text{diag}((G_{Z,Z} + \lambda T I)^{-1}\Upsilon^{Z*}\phi^Z(z)) \tag{10}$$

$$\widehat{\mathcal{W}}_{X|Y,z} = \Upsilon^X(\Lambda_z G_{Y,Y} + \lambda T I)^{-1}\Lambda_z \Upsilon^{Y*} \tag{11}$$

$$\widehat{\mu}_{X|y,z} = \widehat{\mathcal{W}}_{X|Y,z}\phi^Y(y) \tag{12}$$

where $G_{Y,Y} \stackrel{\text{def}}{=} \Upsilon^{Y*}\Upsilon^Y$ has (i,j)th entry $K_Y(y_i, y_j)$, and $G_{Z,Z} \stackrel{\text{def}}{=} \Upsilon^{Z*}\Upsilon^Z$ has (i,j)th entry $K_Z(z_i, z_j)$. The diagonal elements of Λ_z weight the samples, encoding the conditioning information from z.

C. Nonparametric Representation of PSRs

We now use Hilbert space embeddings to represent predictive states and kernel Bayes' rule to update the distributions given a new action and observation.

1) Parameters: HSE-PSR models are represented nonparametrically as Gram matrices of training data. Given $T+1$ *i.i.d.* tuples of actions, observations, and histories $\{(a_t, o_t, h_t)\}_{t=1}^T$ generated by a controlled stochastic process, we denote

$$\Upsilon^{\mathcal{A}} \stackrel{\text{def}}{=} \left(\phi^{\mathcal{A}}(a_1), \ldots, \phi^{\mathcal{A}}(a_T)\right) \tag{13}$$

$$\Upsilon^{\mathcal{O}} \stackrel{\text{def}}{=} \left(\phi^{\mathcal{O}}(o_1), \ldots, \phi^{\mathcal{O}}(o_T)\right) \tag{14}$$

along with Gram matrices $G_{\mathcal{A},\mathcal{A}} = \Upsilon^{\mathcal{A}*}\Upsilon^{\mathcal{A}}$ and $G_{\mathcal{O},\mathcal{O}} = \Upsilon^{\mathcal{O}*}\Upsilon^{\mathcal{O}}$. We also define test embeddings

$$\Upsilon^{\mathcal{T}} \stackrel{\text{def}}{=} \left(\phi^{\mathcal{T}}(h_1), \ldots, \phi^{\mathcal{T}}(h_T)\right) \tag{15}$$

$$\Upsilon^{\mathcal{T}'} \stackrel{\text{def}}{=} \left(\phi^{\mathcal{T}}(h_2), \ldots, \phi^{\mathcal{T}}(h_{T+1})\right) \tag{16}$$

along with Gram matrices $G_{\mathcal{T},\mathcal{T}} = \Upsilon^{\mathcal{T}*}\Upsilon^{\mathcal{T}}$ and $G_{\mathcal{T},\mathcal{T}'} = \Upsilon^{\mathcal{T}*}\Upsilon^{\mathcal{T}'}$. Here primes indicate tests shifted forward in time by one step. The Gram matrices are the parameters for our nonparametric dynamical system model. We will use them below in order to create an initial feasible state as well as update the state with KBR.

2) Estimating a Feasible State: We estimate an initial feasible state $\mathcal{S}_*$ for the HSE-PSR as the mean map of the stationary distributions of tests $\Upsilon^{\mathcal{T}}\alpha_{h_*}$ where

$$\alpha_{h_*} = \frac{1}{T}\mathbf{1}_T \tag{17}$$

Therefore, the initial state is the vector α_{h_*} with length equal to the size of the training dataset.

3) Gram Matrix State Updates: Given a HSE-PSR state α_t, kernel Bayes' rule is applied to *update* state given a new action and observation. Updating consists of several steps.

The first step is *extending* the test distribution [11]. A transition function which accomplishes this is computed $(G_{\mathcal{T},\mathcal{T}} + \lambda T I)^{-1}G_{\mathcal{T},\mathcal{T}'}$. The transition is applied to the state

$$\hat{\alpha}_t = (G_{\mathcal{T},\mathcal{T}} + \lambda T I)^{-1}G_{\mathcal{T},\mathcal{T}'}\alpha_t \tag{18}$$

resulting in a weight vector $\hat{\alpha}_t$ encodes the embeddings of the extended test predictions at time t.

Given a diagonal matrix $\Lambda_t = \text{diag}(\hat{\alpha}_t)$, and a new action a_t, we can condition the embedded test predictions by right-multiplying

$$\alpha_t^a = \Lambda_t(G_{\mathcal{A},\mathcal{A}} + \lambda T I)^{-1}\Upsilon^{\mathcal{A}*}\phi^{\mathcal{A}}(a_t) \tag{19}$$

The weight vector α_t^a encodes the embeddings of the extended test predictions at time t given action a_t.

Next, given a diagonal matrix $\Lambda_t^a = \text{diag}(\alpha_t^a)$, and a new observation o_t, we apply KBR to calculate the next state:

$$\alpha_t^{ao} = (\Lambda_t^a G_{\mathcal{O},\mathcal{O}} + \lambda T I)^{-1}\Lambda_t^a \Upsilon^{\mathcal{O}*}\phi^{\mathcal{O}}(o_t) \tag{20}$$

This completes the state update. The nonparametric state at time $t+1$ is represented by the weight vector $\alpha_{t+1} = \alpha_t^{ao}$. We can continue to filter on actions and observations by recursively applying Eqs. 18–20.

V. Modeling a Depth Camera & Manipulator

A. The Experimental Setup

In this work, we seek to enable a robotic system to autonomously learn a generative model of RGB-D images collected from a depth camera that observes the robot's manipulation space. Our robot consists of a Kinect depth camera observing a Barrett WAM arm located approximately 1.5 meters away. The robot can execute actions and receive observations at a rate of 30 frames per second.

At each point in time, the robot executes a motor command to each of the 7 active joints in the arm (see Figure 2(B)).

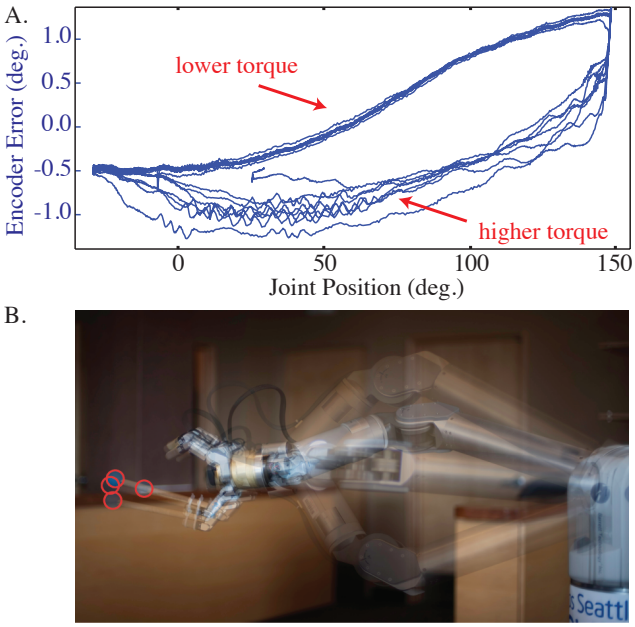

Fig. 3. Motor encoder and kinematic error. (A) Motor encoder error caused by cable stretch in Joint 4 of the WAM arm. The motor encoder returns joint position estimates that deviate from the true joint positions as a stochastic function of torque. The higher the torque, the more the motor encoders err. (B) The arm in four configurations. In each configuration, the encoders and forward kinematics erroneously predict the same hand pose. To show the deviation, a ball is attached to the end effector. The center-to-center distance between the two farthest ball positions is approximately 8 cm. (Figure in (B) from [35])

For each joint, the motor command specifies a desired joint configuration. The exact movement is a function of the commanded target configuration and the controller's estimate of the current joint position as provided by the arm's motor encoders. After executing a motor command, the robot receives an RGB-D observation from the depth camera. Each observation is a vectorized $640 \times 480 \times 3$ pixel RGB image and a time-aligned 640×480 pixel depth map (see Figure 2(A)).

If the motor encoders and RGB-D images were accurate enough, then it would be possible to precisely specify a generative model of the RGB-D images given the true configuration of the arm joints and known geometry and kinematics. Unfortunately, this is not the case. Both the actions and observations contain error due to unmodeled physics in the arm's movements, inaccuracies in the motor encoders, and limitations of the depth camera.

An important example of unmodeled physics is *cable stretch*. The WAM am is driven by cables which wind and unwind as the arm moves. Under differing torques, the cables are wound with differing tensions causing inaccuracies in the joint angles reported by the motor encoders (Figure 3(A)). This results in hysteresis in the reported angles and ultimately in inaccurate predictions of the arm's location in 3D space (Figure 3(B)).

Many of the factors contributing to inaccuracies in the sensor and robot arm can be mitigated by building higher precision parts. However, for many cheaper robots, at least some form of error is likely to affect actions and observations. Modeling a robot as a stochastic process is a natural framework for contending with these errors.

B. Learning the Model

1) Data Collection: The training data consisted of vectorized RGB-D observations in response to *motor babbling*: we randomly moved the arm at different velocities to positions randomly sampled from a uniform distribution in the 7D configuration space (with some velocity and joint-limit constraints). We collected a long execution trace of 30,000 actions and observations; or roughly 16 minutes of data. This data was used as training data for our HSE-PSR algorithm. A sequence of 2000 similarly collected actions and observations were held out as test data.

This is a very large quantity of training data. Previous work on learning HSE-PSRs learned models from ~ 500 training data samples [11]. The quantity of training data was kept low in these prior experiments due to the computational complexity in learning, predicting, and filtering, each of which is $O(T^3)$ in the number of samples. Given the physical complexity of the robot considered here, it would be very difficult to learn an accurate model from so few training examples (500 samples is ~ 15 seconds of data). To overcome this problem, we use a standard trick for computing a *sparse* representation of Hilbert space embeddings via an incomplete Cholesky approximation [36], [37]. This reduced the computational complexity of our state updates from an intractable 30000^3 to a more reasonable 1000^3.

2) State: The core component of our dynamical system model is the *predictive state*. We model the robot with 1-step tests. That is, each test is an action-observation pair $\tau = a_1, o_1$ that can be executed and observed at each time t. The state of the robot is, therefore, the probability distributions of the next RGB-D images in response to motor commands: $\mathbb{P}[o_t \mid a_t, h_t]$.

The predictive distributions are represented nonparametrically as Hilbert space embeddings. The Gram matrices $G_{\mathcal{O},\mathcal{O}}, G_{\mathcal{A},\mathcal{A}}, G_{\mathcal{T},\mathcal{T}}$ and $G_{\mathcal{T},\mathcal{T}'}$ were computed using Gaussian RBF kernels and bandwidth parameters set by the median of squared distance between training points (the "median trick") [28]. Finally, the initial state was set to the stationary distribution of observations given our data collection policy: $\alpha_{h_*} = \frac{1}{T}\mathbf{1}_T$ (Eq. 17). Given these parameters and Eqs. 18–20, we can filter and predict observations from our model.

3) Predicting: We have described above how to implicitly maintain the PSR state nonparametrically as a set of weights on training data. However, our ultimate goal is to make *predictions* about future observations. We can do so with mean embeddings: for example, given the extended state $\hat{\alpha}_t$ (Eq. 18) at some history h_t, we fill in an action using Eq. 19 to find the mean embedding of the distribution of observations:

$$\mu_{\mathcal{O}|h_t,a_t} = \Upsilon^{\mathcal{O}} \alpha_t^a \tag{21}$$

Once we have the embedding of the predictive distribution, we have two options for efficiently computing a prediction.

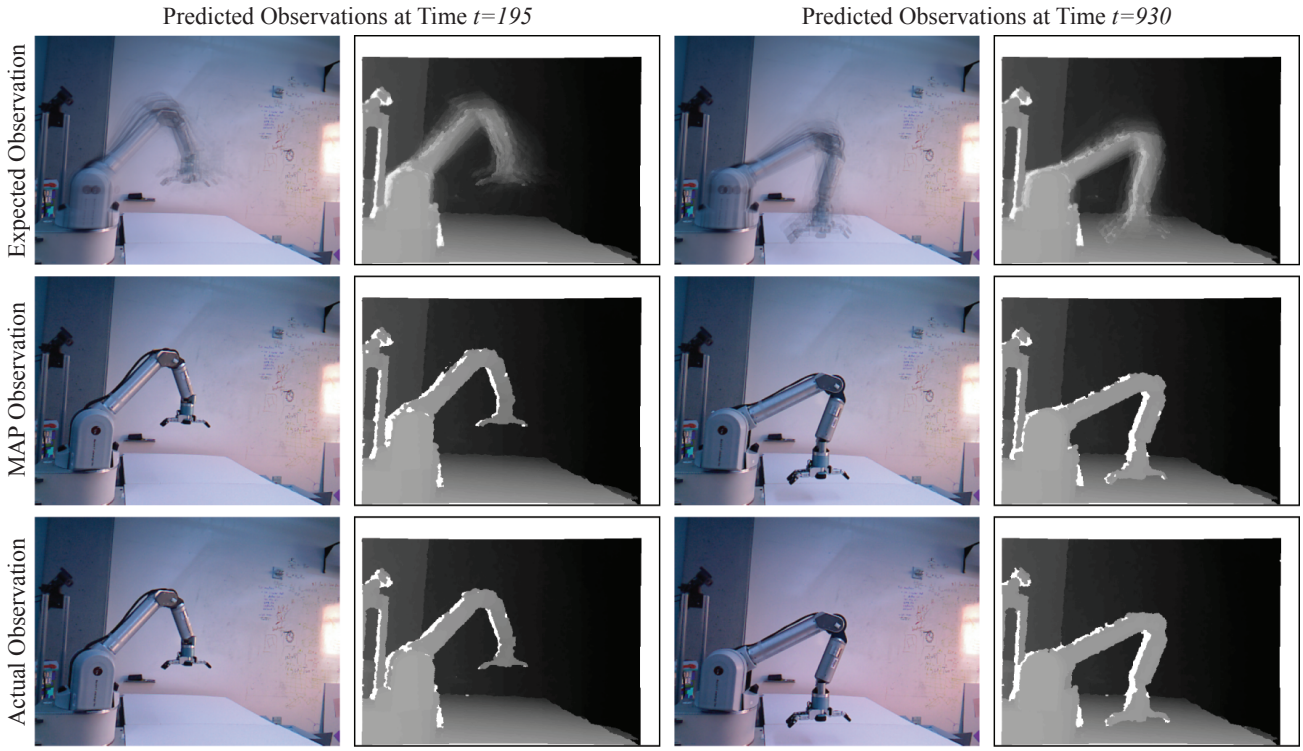

Fig. 4. Example predictions from the learned HSE-PSR model. We can calculate the *expected observation* or the *Maximum A Posteriori observation* from an embedding of the probability distribution over the next observation given that we take a specific action. The two columns on the left show the two predictions after filtering for 195 time steps. The two columns on the right show the two predictions after filtering for 930 time steps. The bottom row shows the actual observation. The expected observation is the weighted average of many images in the training data set. The MAP observation is the the highest probability observation in the training data set. Both are able to predict the actual observation well.

We can either compute the *maximum a posteri (MAP)* observation from the embedded distribution or we can compute the expected observation. The MAP observation is computed:

$$\hat{o} = \arg\max_{o} \left\langle \mu_{\mathcal{O}|h,a}, \phi^{\mathcal{O}}(o) \right\rangle$$

However, the number of possible observations for our robot is very large, so this maximization is not tractable in practice; instead, we approximate it by using the standard approach of maximizing over all observations in the training set [11].

We can also compute the expectation of our embedded distribution of observations. Since the mean embedding μ_X satisfies $\mathbb{E}_X[f(x)] = \langle f, \mu_X \rangle$ for any f in our RKHS, we can write $\pi_i(o_t)$ for the function which extracts the ith coordinate of an observation. If these coordinate projections are in our RKHS, we can compute $\mathbb{E}[o_t|h_t, a_t]$, the expected observation, by computing the inner product $\left\langle \pi_i, \mu_{O|h_t,a_t} \right\rangle$ for all i. Examples of MAP and expected observations calculated from embedded tests are shown in Figure 4.

VI. Quantitative Results

We designed several experiments to illustrate the behavior of the HSE-PSR and to rigorously evaluate the learned model's predictive accuracy. All evaluations are performed on heldout data consisting of random trajectories that were *never* observed in the training data.

Specifically, we studied the *filtering* or tracking performance of the model as the robot executes motor commands and receives RGB-D observations. We also studied the long-range *predictive* accuracy of the model in response to sequences of motor commands.

Finally, we compared the learned HSE-PSR model to several nonparametric function approximation methods for mapping motor commands directly to observations. We show that the learned dynamical system model greatly outperforms the non-dynamic methods by learning to accurately track the state of the robot.

A. Filtering Accuracy

First we studied the filtering performance of the HSE-PSR. As the learned model executes actions and receives observations, the model's prediction accuracy should increase. Additionally, the process of filtering should help to overcome error in the reported joint angles and observations (see Section V-A), leading to more accurate predictions than models which do not take history into account.

To test this hypothesis, we performed filtering over sequences of 100 actions and observations, comparing the predictive accuracy of the model as measured by mean squared error (MSE) in the prediction of the next observation given the current action. We then compared to a baseline provided by kernel regression from motor commands to observations. We trained kernel regression on the same dataset as the HSE-PSR and used Gaussian RBF kernels. The squared error of the predictions was averaged over 1000 trials (Figure 5(A)).

As expected, the model quickly incorporates information from the actions and observations to accurately track the state of the system. 1-step predictions indicate that the model

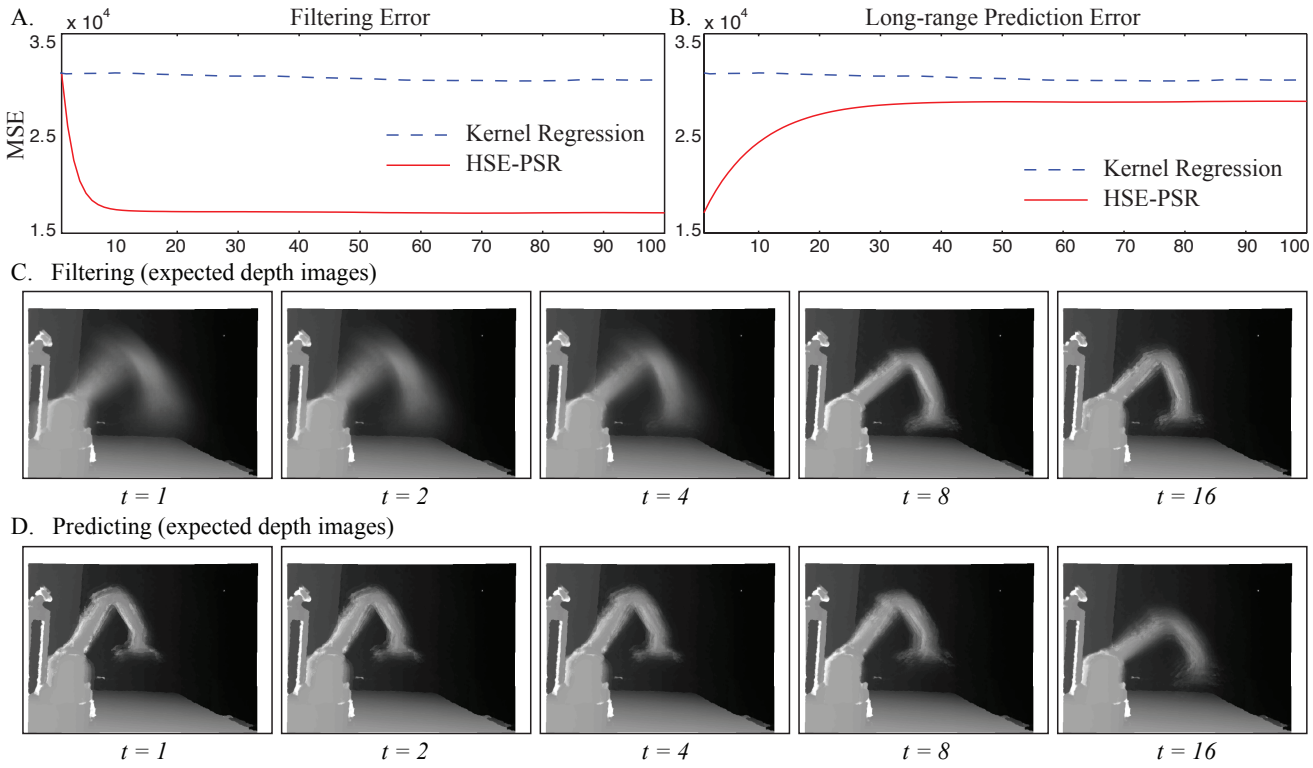

Fig. 5. Accuracy of the learned model. Mean Squared Error (MSE) is computed by taking the squared difference between predicted and true depth maps (at the pixel level) for 1000 experiments. (A.) Filtering for 100 time steps starting from the stationary distribution. The graph shows the mean squared error in the prediction of the next observation given that we take a specified action. The HSE-PSR model decreases its prediction error over time, and, once it is accurately tracking the system, is able to substantially outperform kernel regression which does not model dynamics. (B.) Predicting forward 100 time steps. After filtering, we used the learned model to predict 100 time steps into the future using only actions (no observations). The graph shows the mean squared error of these predictions. Prediction error increases over time until the prediction accuracy is close to the accuracy of kernel regression. This shows that long rang predictions are *no worse* than kernel regression and short term predictions are much more accurate. (C.) Example of filtering: The distribution of observations becomes more concentrated as the robot begins to accurately track its state. (D.) Example of long-range predictions: The distribution of observations becomes more uniform as the robot's uncertainty increases over time.

soundly outperforms kernel regression while tracking. An example of expected depth maps during filtering is shown in Figure 5(C). The sharpening of the predicted images indicates that the variance of the embedded distribution is shrinking.

B. Long-range Prediction Accuracy

Next we consider the motivating problem of this paper: *Can we make accurate long range predictions of what the depth camera will see given that the robot executes a sequence of motor commands?* We expect the predictive performance of the model to degrade over time, but long range prediction performance should not be worse than non-parametric regression models which do not take history or dynamics into account.

To test this hypothesis, we performed filtering for 1000 different extents $t_1 = 101, ..., 1100$, and then predicted observations a further t_2 steps in the future, for $t_2 = 1, ..., 100$, using the given sequence of actions. We then averaged the squared prediction error over all t_1. Again, we compared to kernel regression with Gaussian RBF kernels learned on the training data set. The squared error of the predictions was averaged over 1000 trials (Figure 5(B)).

The prediction accuracy of the learned model degrades over time, as expected. However, the model continues to produce predictions that are more accurate than kernel regression at 100 time steps into the future. An example of expected depth map during prediction is shown in Figure 5(D). The blurring of the expected images indicates that the variance of the embedded distribution is growing.

C. MAP vs. Expectation

In the previous experiments we measured prediction accuracy by looking at the *expected* observation given the HSE-PSR state. We then compared this prediction with the result of kernel regression which can be interpreted as the expected observation given a motor command.

It often makes sense to consider the MAP observation instead of the expected observation. (For a visual comparison, see Figure 4). For example, if the predictive distribution is multimodal, then the MAP observation may result in a more accurate prediction. Or, if we require a visualization of the predictions, then MAP observations may provide a qualitatively better looking prediction.

We compared four methods, the expected and MAP observation from our model as computed by Section V-B.3, as well as their nonparametric regression counterparts: kernel regression and nearest neighbor regression. The results are shown in Figure 6. First, the results indicate that the HSE-PSR produces much better predictions than the nonparamet-

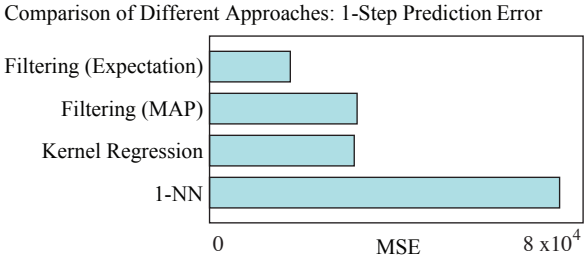

Fig. 6. Comparison of nonparametric approaches to predicting RGB-D images. Mean Squared Error (MSE) computed by taking the squared difference between predicted and true depth maps (at the pixel level) for 1000 experiments.

ric regression approaches. This result is likely attributable to inacuracies in the motor commands. Second, the expected observations have higher predictive accuracy than MAP observations. This is likely due to the fact that the action and observation spaces are high-dimensional and (approximately) continuous. Since the MAP approaches are calculated with a limited set of training samples, we cannot expect to always have access to an observation in the training data set that is very close to the observation that we wish to predict.

VII. Conclusion

In this paper we consider the problem of learning a predictive model of a depth camera and manipular directly from execution traces of RGB-D observations and motor commands. We make as few assumptions about the system as possible: the robot has no knowledge of its manipulator and its goal is to predict raw observations. The fundamental idea is to formulate the problem as learning a controlled stochastic process and leverage recent work on Hilbert space embeddings of predictive state representations in order to learn the model. In real-world experiments, we showed that our approach was able to handle high-dimensional data, to learn complex nonlinear dynamics, and to overcome error in the motor controller and depth camera to make accurate predictions.

References

[1] B. Kuipers, "The map-learning critter," Tech. Rep., 1985.

[2] M. Hersch, E. L. Sauser, and A. Billard, "Online learning of the body schema." *I. J. Humanoid Robotics*, vol. 5, no. 2, pp. 161–181, 2008.

[3] J. Sturm, C. Plagemann, and W. Burgard, "Unsupervised body scheme learning through self-perception." in *ICRA*. IEEE, 2008, pp. 3328–3333.

[4] ——, "Body schema learning for robotic manipulators from visual self-perception," *Journal of Physiology-Paris*, vol. 103, no. 3-5, pp. 220–231, Sept. 2009, neurorobotics.

[5] M. Deisenroth and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Proc. of Robotics: Science and Systems (RSS)*, 20e11.

[6] M. Littman, R. Sutton, and S. Singh, "Predictive representations of state," in *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[7] S. Singh, M. James, and M. Rudary, "Predictive state representations: A new theory for modeling dynamical systems," in *Proc. UAI*, 2004.

[8] M. Rudary, S. Singh, and D. Wingate, "Predictive linear-Gaussian models of stochastic dynamical systems," in *Proc. UAI*, 2005.

[9] Y. Bengio and P. Frasconi, "An Input Output HMM Architecture," in *Advances in Neural Information Processing Systems*, 1995.

[10] B. Boots, S. M. Siddiqi, and G. J. Gordon, "Closing the learning-planning loop with predictive state representations," in *Proceedings of Robotics: Science and Systems VI*, 2010.

[11] B. Boots, A. Gretton, and G. J. Gordon, "Hilbert Space Embeddings of Predictive State Representations," in *Proc. UAI*, 2013.

[12] D. Wingate and S. Singh, "On discovery and learning of models with predictive representations of state for agents with continuous actions and observations," in *Proc. AAMAS*, 2007.

[13] S. C. Ong, Y. Grinberg, and J. Pineau, "Mixed observability predictive state representations," 2013.

[14] W. L. Hamilton, M. M. Fard, and J. Pineau, "Modelling sparse dynamical systems with compressed predictive state representations," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds., vol. 28, no. 1. JMLR Workshop and Conference Proceedings, 2013, pp. 178–186.

[15] A. Censi and R. M. Murray, "Bootstrapping bilinear models of robotic sensorimotor cascades," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[16] ——, "Bootstrapping sensorimotor cascades: a group-theoretic perspective," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, September 2011.

[17] ——, "Learning diffeomorphism models of robotic sensorimotor cascades," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, May 2012.

[18] L. Ljung, *System Identification: Theory for the user*, 2nd ed. Prentice Hall, 1999.

[19] P. Van Overschee and B. De Moor, *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer, 1996.

[20] T. Katayama, *Subspace Methods for System Identification: A Realization Approach*. Springer, 2005.

[21] B. Kuipers, R. Browning, B. Gribble, M. Hewett, and E. Remolina, "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, pp. 191–233, 2000.

[22] J. Modayil, "Discovering sensor space: Constructing spatial embeddings that explain sensor correlations," in *International Conference on Development and Learning*, 2010.

[23] A. M. Dearden and Y. Demiris, "Learning forward models for robots," in *IJCAI*, 2005, pp. 1440–1445.

[24] S. Ulbrich, V. Ruiz, T. Asfour, C. Torras, and R. Dillmann, "Kinematic bzier maps," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 4, pp. 1215–1230, 2012.

[25] H. Jaeger, "Observable operator models for discrete stochastic time series," *Neural Computation*, vol. 12, pp. 1371–1398, 2000.

[26] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models," in *In NIPS*. MIT Press, 2006, pp. 1441–1448.

[27] B. Boots and G. Gordon, "An online spectral learning algorithm for partially observable nonlinear dynamical systems," in *Proc. of the 25th National Conference on Artificial Intelligence (AAAI-2011)*, 2011.

[28] L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola, "Hilbert space embeddings of hidden Markov models," in *Proc. 27th Intl. Conf. on Machine Learning (ICML)*, 2010.

[29] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *CoRR*, vol. abs/1210.7559, 2012.

[30] J. Pearl, *Causality: models, reasoning, and inference*. Cambridge University Press, 2000.

[31] M. Bowling, P. McCracken, M. James, J. Neufeld, and D. Wilkinson, "Learning predictive state representations using non-blind policies," in *Proc. ICML*, 2006.

[32] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Algorithmic Learning Theory*, ser. Lecture Notes on Computer Science, E. Takimoto, Ed. Springer, 2007.

[33] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf, "Injective Hilbert space embeddings of probability measures," 2008.

[34] K. Fukumizu, L. Song, and A. Gretton, "Kernel bayes' rule," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 1737–1745.

[35] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *Int. J. Rob. Res.*, vol. 30, no. 11, pp. 1311–1327, Sept. 2011.

[36] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[37] S. Grunewalder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton, "Modelling transition dynamics in MDPs with RKHS embeddings," *CoRR*, vol. abs/1206.4655, 2012.