Learning Predictive State Representation for In-Hand Manipulation

Johannes A. Stork Carl Henrik Ek Yasemin Bekiroglu Danica Kragic

***Abstract*—We study the use of Predictive State Representation (PSR) for in-hand manipulation. PSR is used to model a hand-object dynamical system for achieving a specific configuration of the object in the hand using a series of object movements induced by pushing the object against a table. For planning pushing motions, the representation is learned using haptic data. We show how to deal with the ambiguity inherit to the data in terms of state identification and how visual data can be used to support the identification problem.**

The model maintains sufficient statistics of the object state and allows for planning of action sequences to achieve specific hand-object configurations. We introduce new kernel-based features that operate on a discrete action space and analyze variance in belief states, resulting in a system that plans action sequences when the observations are ambiguous. We show that the learned representation is geometrically meaningful by embedding labeled action-observation traces. Suitability for planning is demonstrated by post-grasp manipulation sequences that change object state to multiple specified target configurations.

I. Introduction

A model that avoids explicit latent space formulations referred to as Predictive State Representation (PSR) has been proposed recently [1]. PSRs capture world state as probability distributions over sets of observable events and can directly be learned from observed data. The formalism provides means to overcome learning problems associated with small amounts of ambiguous, high dimensional data.

However, applications of PSRs have so far mainly been demonstrated on synthetic data [1]–[3], relying on a large amount of training data which is impractical for the manipulation domain [1]. Furthermore, PSRs have been applied in the discrete domain only [4], [5] thus not considering advanced planning and control where it is non-trivial to extend the formalism to consider continous data. Results have been obtained based on observations that omit actions [1], which deprives a PSR model of the possibility to disambiguate state when observations are highly ambiguous.

This work, for the first time, applies PSRs in an in-hand manipulation scenario using sensory data constrained by the limited amount of training data available for learning and planning. We consider a simple task of changing the in-hand pose of a marker-pen grasped by a parallel gripper. Since the manipulator is not dexterous, pushing the marker against external objects—edge of a table—enables the robot to achieve hand-object configurations that are otherwise unobtainable, see Fig. 1. Our goal is to learn an interaction model between the hand, object and the environment which can be used for planning.

The authors are with the Computer Vision and Active Perception Lab, Centre for Autonomous Systems, School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden, {jastork,chek,yaseminb,dani}@kth.se.

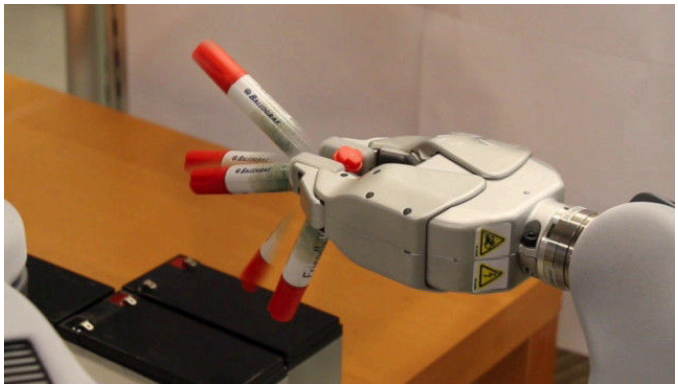


Fig. 1. In-hand manipulation using resources extrinsic to the robot requires representations that describe the hand-object dynamical system including the task-relevant parts of the environment. Here, the robotic gripper was placed in different positions and moved down to push the marker-pen against an edge.

The robot used in our experiments is equipped with tactile and vision sensing that constitute a high dimensional observation space. We rely on visual measurements to reason about semantics and state uncertainty in the PSR but learn the model from tactile data only. This is motivated by the fact that vision feedback comes at a high cost since the hand and the marker need to be placed in the camera's field of view at all times during the interaction. Furthermore, as images are highly redundant a relevant representation needs to be extracted. On the contrary, tactile feedback is cheaper to gather, more robust to noise and for this specific task much less redundant compared to vision. Therefore, we rely on vision sensing only during identification of the model. The drawback of tactile feedback is that it cannot alone disambiguate the pose as different poses result in the same sensor readings. To circumvent this, we integrate the tactile readings with the actions resulting in state disambiguation over time. This is enabled by proposing a new kernel based structure for PSRs that directly relates to the sequential structure of the training data. It can represent symbolic actions jointly with continous data, which are otherwise difficult to integrate. This way, the learning-planning loop is closed by planning and executing sequential actions based on visual and tactile observations.

The reminder of the paper is structured as follows, in Section II we describe the related work. As PSRs are a relatively new discovery we provide a detailed introduction to these models in Section III. We then proceed to detail the extensions and alterations proposed in this paper in Section IV. The results and the conclusions are found in Section V and VI.

II. Related Work

The task of changing a pose of an object in a robot's hand is a challenging problem and it requires online planning based on high dimensional obeservations when both visual and tactile measurements are taken into account. The problem has not been studied extensively in the literature but it is one of the cornerstones towards making robots more dextreous. In this section, we briefly review relevant works and focus on representation learning for this problem.

Back in 80's, it was shown how a precision grasp can be used for a similar task if fine control of the fingertips is available [6]. More recently, a planner for dexterous manipulation has been introduced where finger movements are controlled to follow a trajectory without breaking contacts, given an initial grasp and the desired trajectory of the object in task space [7]. Another way to obtain in-hand manipulation is to use compliant and underactuated robot hands [8], [9] with simpler control. In-hand manipulation with simple hands requires the use of resources besides the manipulator, such as a surface for external contact, gravity, inertia of the object, and acceleration of the arm [10]. In our work, the robot relies on the extrinsic dexterity [10] by learning actions that control the movement of the object in a hand through interaction with a table.

The ability to exploit the environment is an essential characteristic of human manipulation. In [11], the authors investigate how humans exploit environmental constraints through motion contact for counteracting uncertainty. This has led to the design of robotic hands with the inherent capacity to exploit environmental constraints and to *a priori* definition of manipulation primitives with open-loop execution [10].

The ability to perform successful planning for in-hand manipulation requires a suitable representation. A traditional approach in robotics is to represent the world state in terms of a latent variable. In this way, dynamical systems can be formalized as *Partially Observable Markov Decision Processes* (POMDPs), which allows reasoning about uncertainty in action effects and state observability. In [12] the authors address a very similar task to the one we use in this paper by using a latent state space approach. In contrast to our work, they consider fewer goal configurations and use vision information to initially estimate the object state. We only employ visual sensing in the training phase. While their tactile feedback consists of force-torque sensing in the three fingers, we build on pressure readings.

The work of [13] employs the POMDP framework to track pose belief of a known-shaped object and localizes the object by planning grasping and information-gathering trajectories. Similar to our work, the robot acquires new information through tactile feedback and the object moves when actions are performed. However, as opposed to pushing the object, grasping at a certain relative pose is the goal.

The works discussed above operate on *a-priori* defined latent variable representations of state. However, for many scenarios defining such a representation *a-priori* is very challenging and it would be beneficial if the representation could be automatically inferred from the data [1].

In [14] the agent learns task-relevant state representations from high-dimensional observations. Structure is imposed by using five different physically motivated robot-specific priors. We also consider interaction with the physical world, but refrain from geometric and physical intuitions. In general, learning POMDPs from *exploration data* without such priors has only had limited success, because of the *curse of dimensionality* (of the observations) and the *curse of history* (of required training sequences). It involves high-dimensional (observation) spaces and for increasingly longer horizon an exponential amount of different sample sequences.

Predictive State Representation [15] (PSR) is a different approach to model high-dimensional dynamical systems that represent world states as predictions about future events called tests and therefore does not rely on hidden unobservable latent states. Since these models are rooted directly in observable quantities they can be constructed without expectation maximization type heuristics. Learning a PSR fundamentally consists of two challenges: i) discovery of a sufficient set of future events to form a sufficient state, and ii) learning state update parameters. The discovery problem has been addressed through information theoretic notions [2], transformation by spectral projection methods (TPSRs) [1], [16], [17], and compression with random bases (CPSRs) [3]. Successful approaches to parameter learning are a formalization that relates parameters directly to observables for compressed and transformed [1], [3], [17] PSRs, which allows learning parameters as empirical estimation, or by Hilbert space embedding of nonparametric distributions [4], [5] (HSE-PSRs).

PSRs have been extended to operate in continuous domains [1], [2], [4], which is required for applications in robotic context. While [2] learned exponential family distributions of tests by kernel density estimation, [1] extends TPSRs to a feature-based formulation and using RBF kernel, both times for a simulated autonomous navigation problem using vision observations. HSE-PSRs naturally handle continuous domains and have been employed to model depth images and joint data of a robot arm [4], [5].

III. Predictive State Representations

In this section, we describe the *learning algorithm* for transformed predictive state representations (TPSR) with *features for continuous domains*. How discrete PSRs are extended to continuous domains is described steps-by-step in [1], where all proofs are given. We omit details that are only of theoretical value, such as the observable representation for *discrete* TPSRs.

A. Linear PSRs

Predictive State Representations [15] are a general probabilistic framework for controlled stochastic processes that can represent latent-space models like *Partially Observable Decision Processes* [18] (POMDPs) or *Hidden Markov Models* [19] (HMMs). At the same time PSR models are at least as compact as POMDPs and still provide greater representational capacity.

In contrast to POMDPs and HMMs, PSRs are designed to describe dynamical systems directly in relation to observable events. For that purpose, a predictive state representation consists of probability distributions over future events (called *tests*) conditioned on past events (called *histories*).

The components of a PSR are described by the tuple $(O, A, Q, F, \tau(\varepsilon))$, where O and A describe the set of observations and actions respectively. In our example, action will be pushing motion and observations will be haptic features. The variable Q denotes a finite set of test, F denotes a set of prediction functions relating tests in Q to *all* possible tests, and $\tau(\varepsilon)$ is a vector of prior probabilities (i.e. predictions). A test is a sequence of future actions and observations, denoted $\tau = a_1^\tau o_1^\tau a_2^\tau o_2^\tau \dots a_n^\tau o_n^\tau$, histories refer to sequences in the past, denoted $h = a_1^h o_1^h a_2^h o_2^h \dots a_k^h o_k^h$. We refer to the actions of a test τ as $\tau^A = a_1^\tau a_2^\tau \dots a_n^\tau$ and to the observations as $\tau^O = o_1^\tau o_2^\tau \dots o_n^\tau$.

Maintaining probabilities in form of *test predictions* is the eponymous task in the PSR formalism and is formulated for a test τ, given a history h by $P(\tau^O|\mathrm{do}(\tau^A), h)$, intervening [20] with the test actions τ^A. Knowing the predictions of *all* tests for *all* possible histories completely characterizes the dynamical system. However, even for discrete systems there are too many different tests and histories to estimate the probabilities for all instances. For a compact representation, we refer only to predictions of tests $\tau_i \in Q$ recorded in the PSR prediction vector:

$$\tau(h) = \left(P(\tau_1^O|\mathrm{do}(\tau_1^A), h), \dots, P(\tau_{|Q|}^O|\mathrm{do}(\tau_{|Q|}^A), h)\right)^\intercal \quad (1)$$

The prediction vector for the empty history is denoted as $\tau(\varepsilon)$.

For a linear PSR, Q is *linear sufficient* (also called *core*) if and only if all test predictions can be expressed as linear projection of the prediction vector, $P(\tau^O|\mathrm{do}(\tau^A), h) = \mathbf{m}_\tau^\intercal \tau(h)$, for any history. The probabilities collected in the prediction vector $\tau(h)$ thus form a sufficient statistic of history and all other tests τ can be represented by a function $\mathbf{m}_\tau$ of predictions for tests in Q. To update the prediction vector for new action and observation pairs ao we use the prediction functions $\mathbf{m}_{ao\tau_i}$ for each $\tau_i \in Q$, arranged in rows to form the matrix $\mathbf{M}_{ao}$. This notation refers to the tests prefixed by action a and observation o: $P(\tau_i^O, o|\mathrm{do}(\tau_i^A, a), h) = \mathbf{m}_{ao\tau_i}^\intercal \tau(h)$. The new prediction vector $\tau(hao)$ is computed compactly by:

$$\tau(hao) = \frac{\mathbf{M}_{ao}\tau(h)}{\mathbf{m}_\infty^\intercal \mathbf{M}_{ao}\tau(h)} \quad (2)$$

where the normalizing factor $\mathbf{m}_\infty$ satisfies $\mathbf{m}_\infty^\intercal \tau(h) = 1$ for all histories h.

B. Learning PSRs

The parameters of a linear PSR are the normalizing vector $\mathbf{m}_\infty$ and matrices $\mathbf{M}_{ao}$ of dimension $|Q| \times |Q|$ for each $ao \in A \times O$. However, these parameters are only defined up to a similarity transform $\mathbf{W}$, which allows a reformulation of Eq.(2) with a change of basis:

$$\mathbf{W}\tau(hao) = \frac{\mathbf{W}\mathbf{M}_{ao}\mathbf{W}^{-1}\mathbf{W}\tau(h)}{((\mathbf{W}^{-1})^\intercal \mathbf{m}_\infty)^\intercal \mathbf{M}_{ao}\mathbf{W}^{-1}\mathbf{W}\tau(h)} \quad (3)$$

This allows us to describe the same PSR using transformed parameters $\mathbf{B}_{ao} = \mathbf{W}\mathbf{M}_{ao}\mathbf{W}^{-1}$, $\mathbf{b}_\infty = (\mathbf{W}^{-1})^\intercal \mathbf{m}_\infty$, and transformed prediction vector $\mathbf{b}(h) = \mathbf{W}\tau(h)$. Instead of solving the discovery problem by finding a minimal sufficient set of test, we can choose a redundantly large Q such that it almost certainly contains a sufficient subset. Selecting an appropriate transform $\mathbf{W}$ reduces size and dimensions of the transformed parameters and prediction vector which yields a low dimensional representation.

Features for Continuous Domains: Observation spaces of real-world robotic systems are too complex to directly use tests and histories, e.g. by discretizing continuous multivariate variables. The TPSR formalism has therefore been generalized by features of test outcomes (*characteristic features*) and histories (*indicative features*) [1]. The parameters of the generalized formalization are directly based on observable quantities which allows statistically consistent learning as empirical estimation.

Formally, we introduce a redundantly large sufficient set of test $\mathcal{T}$ and partition the space of all histories into *indicative events* $\mathcal{H} = \bigcup_i H_i$. Characteristic features (which are linear combinations of test outcomes) are declared as $n_c \times 1$ vectors $\phi^\mathcal{T}$ and indicative features $\phi^\mathcal{H}$ as $n_i \times 1$ vectors. Examples of domain-specific features are based on RBF kernels on test and histories [1] or randomized Fourier methods [17], the latter requiring many features to reach reliable predictions in some cases. Differently from that, we use new sequence kernel-based feature described in Sec. IV-B.

The parameters are formulated as observable matrices, $\Sigma_{\mathcal{T},\mathcal{H}}$, $\Sigma_{\mathcal{T}^+,ao,\mathcal{H}}$, and the expected indicative features as $\Sigma_\mathcal{H}$ in form of a $n_i \times 1$ vector:

$$(\Sigma_\mathcal{H})_i = \mathbb{E}(\phi_i^\mathcal{H}) \quad (4)$$

$\Sigma_{\mathcal{T},\mathcal{H}}$ is defined as a $n_c \times n_i$ matrix which relates tests and histories by expected products of characteristic and indicative feature values:

$$(\Sigma_{\mathcal{T},\mathcal{H}})_{i,j} = \mathbb{E}(\phi_i^\mathcal{T} \phi_j^\mathcal{H}|\mathrm{do}((\phi_i^\mathcal{T})^A)) \quad (5)$$

where $(\phi_i^\mathcal{T})^A$ refers to the intervention actions of tests used to form characteristic feature $\phi_i^\mathcal{T}$. The matrix $\Sigma_{\mathcal{T}^+,ao,\mathcal{H}}$ represents similar kind of information, however, the action and observation pair ao is defined for the time step between the indicative and the characteristic features:

$$(\Sigma_{\mathcal{T}^+,ao,\mathcal{H}})_{i,j} = \mathbb{E}(\phi_{i,t+1}^\mathcal{T}, \mathbb{I}(o_t = o), \phi_{j,t}^\mathcal{H}|\mathrm{do}(a_t = a, (\phi_{i,t+1}^\mathcal{T})^A)) \quad (6)$$

where t and $t+1$ are used to indicate to which relative time steps observations and features belong. In the three equations above, we used j as index for indicative features and i for characteristic features.

The observable matrices can be used to describe TPSR parameters by defining the transform $\mathbf{W}$ which yields a statistically consistent learning algorithm as a practical result.

Proofs for the equations below are presented in [1].

$$\mathbf{b}_* = \mathbf{U}^\intercal \Sigma_{\mathcal{T},\mathcal{H}} \mathbf{e} \tag{7}$$
$$= (\mathbf{U}^\intercal \Phi^{\mathcal{T}} \mathbf{\Gamma}) \mathbf{m}_* \tag{8}$$
$$\mathbf{b}_\infty^\intercal = \Sigma_{\mathcal{H}}^\intercal (\mathbf{U}^\intercal \Sigma_{\mathcal{T},\mathcal{H}})^\dagger \tag{9}$$
$$= \mathbf{m}_\infty^\intercal (\mathbf{U}^\intercal \Phi^{\mathcal{T}} \mathbf{\Gamma})^{-1} \tag{10}$$
$$\mathbf{B}_{ao} = \mathbf{U}^\intercal \Sigma_{\mathcal{T}^+,ao,\mathcal{H}} (\mathbf{U}^\intercal \Sigma_{\mathcal{T},\mathcal{H}})^\dagger$$
$$= (\mathbf{U}^\intercal \Phi^{\mathcal{T}} \mathbf{\Gamma}) \mathbf{M}_{ao} (\mathbf{U}^\intercal \Phi^{\mathcal{T}} \mathbf{\Gamma})^{-1} \tag{11}$$

The normalizing vector $\mathbf{e}$ is defined by $\mathbf{1}_{n_i} = (\Phi^{\mathcal{H}})^\intercal \mathbf{e}$ and the similarity transform is of the form $\mathbf{W} = \mathbf{U}^\intercal \Phi^{\mathcal{T}} \mathbf{\Gamma}$. The matrix $\mathbf{U}$ can be used for dimension reduction, as long as $\mathbf{W}$ stays invertible and $\mathbf{\Gamma}$ is a $|\mathcal{T}| \times |Q|$ matrix with rows consisting of the projection functions for all tests in $\mathcal{T}$. $\Phi^{\mathcal{T}}$ is sized $n_c \times |\mathcal{T}|$ with entries being weights of test outcomes in calculating the specific characteristic feature. $\Phi^{\mathcal{H}}$ is of size $n_i \times |\mathcal{H}|$ and directly contains the expected value of indicative features for indicative events. Consequently, the semantics of $\mathbf{W}$ is to lift from a $|Q|$-dimensional prediction vector to a $|\mathcal{T}|$-dimensional prediction vector (by $\mathbf{\Gamma}$), project to feature space (by $\Phi^{\mathcal{T}}$), and change basis to reduce dimensions (by $\mathbf{U}$).

Choosing $\mathbf{U}$ as the transpose of the thin SVD of $\Sigma_{\mathcal{T},\mathcal{H}}$ and retaining only the $|Q|$ most significant singular vectors has been demonstrated to work well and provides that $\mathbf{U}^\intercal \Sigma_{\mathcal{T},\mathcal{H}}$ has full rank [1] .

Training Procedure: To learn TPSR parameters $\mathbf{b}_*$,$\mathbf{b}_\infty$, and $\mathbf{B}_{ao}$ from estimates of the observable matrices, it is formally required to repeatedly sample execution sequences from an initial state of the dynamical system. If a robot interacts with the environment, this process can be hardly realized and instead one long sequence is divided into shorter and overlapping sequences of fixed length using the suffix-algorithm [21]. The result is a set of training sequences $S = \{s_1, s_2, \ldots, s_w\}$ of length $2m$, where each sequence overlaps the following sequence by all but the last time step. For sequence s_t, indicative features $\phi_t^{\mathcal{H}}$ are evaluated on the first m time steps and characteristic feature $\phi_t^{\mathcal{T}}$ on steps $m+1$ to $2m$. The action and observation pair in the middle of a sequence is referred to as $a_t o_t$.

The process shown in Alg. 1 assumes that the samples represent an underlying distribution and begins by computing the empirical expectation of the indicative feature vector $\hat{\Sigma}_{\mathcal{H}} = \frac{1}{w} \sum_{t=1}^{w} \phi_t^{\mathcal{H}}$. Likewise, the empirical estimate $\hat{\Sigma}_{\mathcal{T},\mathcal{H}} = \sum_{t=1}^{w} \alpha_t \phi_t^{\mathcal{T}} \phi_t^{\mathcal{H}}$ is computed using the weighting factor α_t. Its low rank approximation in the form of $\hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^\intercal$ is used to compute $\hat{\mathbf{b}}_*$ and $\hat{\mathbf{b}}_\infty^\intercal$:

$$\hat{\mathbf{b}}_* = \hat{\mathbf{\Sigma}} \hat{\mathbf{V}}^\intercal \mathbf{e} \tag{12}$$
$$\hat{\mathbf{b}}_\infty^\intercal = \hat{\Sigma}_{\mathcal{H}}^\intercal (\hat{\mathbf{V}} \hat{\mathbf{\Sigma}}^{-1}) \tag{13}$$

where the rank can be chosen by inspecting singular values in $\hat{\mathbf{\Sigma}}$. If one indicative feature is constant the normalizing vector $\mathbf{e}$ can directly be determined. Instead of obtaining the empirical estimates $\hat{\Sigma}_{\mathcal{T},ao,\mathcal{H}}$ for each pair ao, we use the smaller matrices $\hat{\mathbf{U}}^\intercal \hat{\Sigma}_{\mathcal{T},ao,\mathcal{H}}$. However, in a continuous domain we cannot restrict the sample set to sequences where the middle action-observation pair is $ao = a_t o_t$. Instead we use a kernel distance measure on observations $\mathcal{K}(.,.)$ to represent $\mathbb{I}(o_t = o)$ in Eq. (6) and estimate based only on sequences with the same action, indexed by $T_a = \{t \in 1, \ldots, w | a_t = a\}$. The TPSR parameters $\hat{\mathbf{B}}_{ao}$ are consequently defined for each discrete action $a \in A$ and each observation kernel center $o \in O_{center} = \{o^{(1)}, o^{(2)}, \ldots, o^{(l)}\}$:

$$\hat{\mathbf{B}}_{ao} = \sum_{t \in T_a} \alpha_t^a (\hat{\mathbf{U}}^\intercal \phi_{t+1}^{\mathcal{T}} \frac{1}{Z_t} \mathcal{K}(o_t, o) \phi_t^{\mathcal{H}}) (\hat{\mathbf{V}} \hat{\mathbf{\Sigma}}^{-1}) \tag{14}$$

where Z_t is the kernel partition function computed over all kernel centers. The weighting factors α_t and α_t^a are constant and cancel out since we assume that the samples are spread uniformly.

Algorithm 1: Training algorithm for TPSRs in continuous domains

Data: Features $\phi_{1:w}^{\mathcal{T}}$, $\phi_{1:w}^{\mathcal{H}}$, observations $o_{1:w}$, actions $a_{1:w}$, kernel centers O_{center}, normalizer $\mathbf{e}$
Result: Change of basis $\hat{\mathbf{U}}$, parameters $\hat{\mathbf{b}}_*$, $\hat{\mathbf{b}}_\infty$, $\hat{\mathbf{B}}_{ao}$

$\hat{\Sigma}_{\mathcal{H}} \leftarrow \frac{1}{w} \sum_{t=1}^{w} \phi_t^{\mathcal{H}}$
$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) \leftarrow \mathsf{SVD}(\sum_{t=1}^{w} \alpha_t \phi_t^{\mathcal{T}} \phi_t^{\mathcal{H}})$
$(\hat{\mathbf{U}}, \hat{\mathbf{\Sigma}}, \hat{\mathbf{V}}) \leftarrow$ low rank approximation of $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$
$\hat{\mathbf{b}}_* \leftarrow \hat{\mathbf{\Sigma}} \hat{\mathbf{V}}^\intercal \mathbf{e}$
$\hat{\mathbf{b}}_\infty^\intercal \leftarrow \hat{\Sigma}_{\mathcal{H}}^\intercal \hat{\mathbf{V}} \hat{\mathbf{\Sigma}}^{-1}$
$Z_t \leftarrow \sum_{i=1}^{l} \mathcal{K}(o_t, o^{(i)})$ f.a. $t \in \{1, 2, \ldots, w\}$
foreach $(a, o) \in A \times O_{center}$ **do**
  $T_a \leftarrow \{t \in 1, \ldots, w | a_t = a\}$
  $\hat{\mathbf{B}}_{ao} \leftarrow \sum_{t \in T_a} \alpha_t^a (\hat{\mathbf{U}}^\intercal \phi_{t+1}^{\mathcal{T}}) \frac{1}{Z_t} \mathcal{K}(o_t, o) (\phi_t^{\mathcal{H}})^\intercal \hat{\mathbf{V}} \hat{\mathbf{\Sigma}}^{-1}$
end

C. Inference

By learning a predictive representation of a system as described in the previous section (Sec. III-B), we obtain the TPSR parameters $\hat{\mathbf{b}}_*$, $\hat{\mathbf{b}}_\infty$, and $\hat{\mathbf{B}}_{ao}$ and a set of l observations as kernel centers O_{center}. To close the learning-planning loop we need to reason about effects of sequences of actions and like for POMDPs, plan a policy that maps (belief) states to actions $\pi \colon \mathcal{B} \rightarrow A$. The PSR space $\mathcal{B}$ is similar to a belief space since every state $\mathbf{b}(h) \in \mathcal{B}$ represents a distribution over tests, but unlike the belief spaces of POMDPs it lacks inhered semantics like an underlying state space. Therefore we populate the space $\mathcal{B}$ with labeled states by iterative application of Eq. (3). In practice this means filtering (tracking the state) for sequences of actions and observations $s_t = a_1 o_1 a_2 o_2 \ldots a_m o_m$ from the training set and computing the PSR state $\mathbf{b}(s_t)$ by:

$$\mathbf{b}(s_t) = \frac{\mathbf{B}_m \mathbf{B}_{m-1} \ldots \mathbf{B}_1 \hat{\mathbf{b}}_*}{\hat{\mathbf{b}}_\infty^\intercal \mathbf{B}_m \mathbf{B}_{m-1} \ldots \mathbf{B}_1 \hat{\mathbf{b}}_*} \tag{15}$$

Each local operator $\mathbf{B}_i$ is a convex combination of the learned local operators $\mathbf{B}_{ao}$ at the different observation kernel center

positions which have the same action, mixed according to kernel response:

$$\mathbf{B}_j = \sum_{i=1}^{l} \frac{1}{Z_j} \mathcal{K}(o_j, o^{(i)}) \hat{\mathbf{B}}_{ao} \tag{16}$$

This leads to a set of states $B = \{\mathbf{b}(s_1), \mathbf{b}(s_2), \ldots, \mathbf{b}(s_w)\}$ for which the generating sequences of actions and observations are known. We can annotate each state in B with, e.g. the last observation and use this label to define rewards for a specific task or goal. In this case, policies for the complete space $\mathcal{B}$ can be found using approximate point-based planning by value iteration as described in [1], [22].

IV. Methodology and implementation

To learn a PSR representation we need to define a set of actions and collect sensory information as observations associated with the performed actions. In this section we describe the observation spaces and introduce new kernel-based features for the PSR model. Furthermore, we explain how the different parts of the learned state representation are analyzed to be used as a basis for planning policies.

A. Actions and Observations

We use the Willow Garage PR2 robotic platform [23], equipped with one-DoF parallel-jaw grippers, cameras, and pressure sensor arrays on the gripper fingertips. To manipulate a grasped marker-pen, we define six different manipulation actions $A = \{a_1, a_2, \ldots, a_6\}$, each of which consists of a sequence of arm motions. We initially place the gripper so that fingers and a table edge are orthogonal and that during an upward or downward motion, the pen hits the table edge as seen in Fig. 1. Then, we put the gripper in one of the angles depicted in Fig. 2(a) and move the arm vertically towards the edge until change of pressure indicates contact. With 0.5 sec delay we stop the arm and return to the initial position. This way, actions $1-3$ turn the marker clock-wise, while actions $4-6$ turn the marker in the opposed direction and any marker angle can be changed by at least one action.

Back at the initial pose, we use a forearm camera to acquire an image of the gripper-marker configuration and collect the pressure readings from the fingertips. In the images we find the orientation of the marker by blob detection—which describes the configuration unambiguously for our purposes—and refer to them as the space of vision data $\mathcal{O}_V$ (Fig. 2(b) *left*). This data is costly to acquire since the gripper and camera need to move to the exact same locations and illumination conditions need to be addressed. From the pressure cell readings, a set of haptic features are calculated [24]. The resulting space $\mathcal{O}_H$ consists of 2nd order image moments that describe the distribution of the pressure in the vertical and horizontal directions, separately for each sensor. This data is quick to acquire and robust to noise if the sensors are accurately calibrated. The negative aspect of the haptic features is that they do not disambiguate different marker states (marker orientation relative to the gripper) since different angles produce similar pressure readings, see Fig. 2(b) *right*.

B. String Kernel Feature

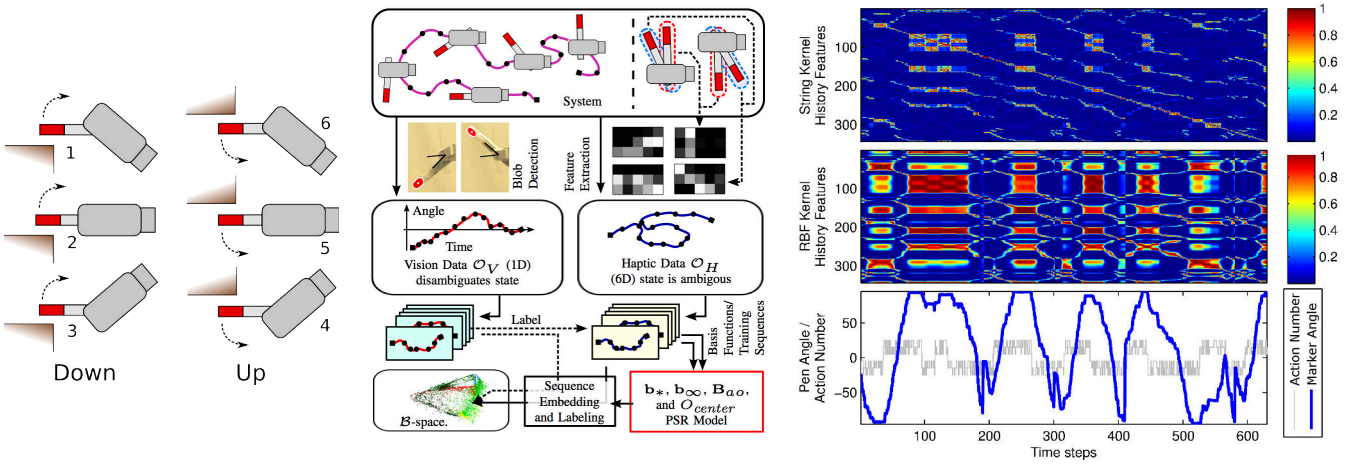
In [1], both the indicative and characteristic features are defined by RBF kernels and kernel centers (basis functions) consisting of observations from sample sequences. Actions are not represented by these features because it is difficult to intuitively define a distance measure on a set of discrete actions. For instance, in our case actions a_2 and a_5 are similar when the marker points up since both of them will not push it, if the marker points left, the actions are dissimilar either pushing the marker up or down.

However, excluding actions from features deprives the model of valuable information if sequences of observations do *not* disambiguate the state. In our case, sequences of entirely different configurations result in similar observations. For instance, pushing the marker counter clock-wise results in the same observations as clock-wise rotation when the marker either points up or down, respectively (Fig. 2(b) *right*). In Fig. 2(c) we show RBF feature vectors for training data which show strong similarity between the sequences from step 40 and step 100 even though the marker angle is different. Other examples are observed around steps 450 and steps at 510. Even though the marker moves in different direction, features of all sequences around step 100 show strong similarity.

To address this problem we borrow methodology from text analysis and adopt a string kernel [25] approach. String kernels were initially introduced as a means of representing sequences of text where no natural notion of similarity exists on a character level. The string kernel works by comparing "gaps" in discrete sequential data [26] and creates an implicit feature mapping capable of relating discrete symbol sequences of different length. The string kernel $\mathcal{K}_S(.,.)$ allows us to compare sequences of symbols without defining a distance measure on the set of symbols. We exploit this fact to define indicative and characteristic features that take actions into account. To this end, we form an alphabet $\Omega = \{\bar{a}_1, \bar{a}_2, \ldots, \bar{o}_1, \bar{o}_2, \ldots\}$ consisting of a symbol for each action and symbols from a discretization of the haptic data $\mathcal{O}_H$. Observation symbols are defined by k-means clustering into 12 clusters. Finally, the training set is represented by sequences of alternating action and observation symbols resulting in the features presented in the top diagram of Fig. 2(b). The feature vectors are more sparse but similarities between step 40 and step 100 are removed due to the actions involved. Around step 100, the phases of moving clock-wise and counter clock-wise when the marker points down can now be clearly distinguished.

C. Learning, Planning, and Execution

Using the TPSR training algorithm (Sec. III-B) together with the features from the last section (Sec. IV-B), we can learn a PSR from training data consisting of sequences of action and haptic observation symbols $s_i = \bar{a}_1\bar{o}_1\bar{a}_2\bar{o}_2 \ldots \bar{a}_m\bar{o}_m$, aligned sequences of vision data $s_i^* = o_1^* o_2^* \ldots o_m^*$, i.e. $o_i^* \in \mathcal{O}_V$, and a set of observation kernel centers O_{center}. Some sample sequences with $m = 5$ action and haptic observation symbols are selected as basis functions in feature computation



(a) (b) (c)

Fig. 2. *(a):* Side-view sketch of the 6 actions $A = \{a_1, a_2, \ldots, a_6\}$ that we define by different gripper angles. Actions 1-3 move the gripper down to push the marker clock-wise utilizing marker-table contact, while action 4-6 move the gripper upwards *(b):* The different spaces involved in learning and executing in-hand manipulation: System state (not accessible), vision data (expensive but ambiguating), haptic data (cheap and reliable but not disambiguating), PSR state space (like a belief space). *(c):* Comparison of indicative feature vectors (as columns). The string kernel features disambiguate (history) sequences at time steps where the RBF kernel features cannot.

and we select a small, suitable low-rank approximation of the matrix $\hat{\Sigma}_{\mathcal{T},\mathcal{H}}$ based on its singular values.

When populating the $\mathcal{B}$-space as described in Sec. III-C, we track sequences step-wise form 2 to 10 steps recording all intermediate states, instead of filtering only for the length of feature basis functions, as in [1]. This process additionally embeds states, where the model has not yet converged, which allows us to investigate which $\mathcal{B}$-space regions describe uncertainty. For this, we label embedded states by the last vision data collected for its generating sequence: $\ell(\mathbf{b}(s_i)) = o^*_m$.

We also use these labels to define reward $R\colon \mathcal{B} \times A \to \mathbf{R}$ and find a policy $\pi^*\colon \mathcal{B} \to A$ to maximize the expected sum of discounted rewards. Value iteration [27] algorithms represent the optimal policy by the optimal value $V^*(b) = \max_{a \in A}(R(b,a) + \gamma \sum_{o \in O} P(o|b,a)V^*(b'))$, where the state obtained from choosing action a in state b and subsequently observing o is denoted b'. Given the optimal value function, the optimal policy is determined by maximizing the argument instead of the value in the formula above. Under the assumption that reward is linear in state, we can define $R(b,a) = \eta_a^\intercal b$ and directly employ algorithms which have been extended from POMDPs to PSRs [1], [22]. Linear reward can be ensured by adding reward to the observations when learning the PSR, however we want to avoid re-learning representations for different planning goals and therefore refer to a planning approach that only relies on the embedded states.

When populating the $\mathcal{B}$-space with states, we record the traces and for each action save the pair of states. In this way, we approximately encode dynamics by exploiting that similar states are geometrically close neighbors in $\mathcal{B}$. Based on this finite representation we use value interation per embedded state and determine a greedy policy.

Since the feature basis functions represent histories of finite length, the PSR cannot distinguish all histories longer than the longest basis function. Therefore, the PSR becomes uncertain of the system state if long past information is required for proper representation. This poses a problem to belief-space planning since there always exists a path to the target state that returns to the initial state of maximal uncertainty in between. We want that actions never induce unnecessary uncertainty which can lead to such flawed policies. For this reason we analyze the labels of every state's local neighborhood $N(b)$. The spread of labels $r(b) = \frac{1}{|N(b)|} \sum_{b' \in N(b)} |\ell(b) - \ell(b')|$ describes how different the neighbor points are from the state b and therefore higher values relate to higher belief uncertainty. For high values of $r(b)$ we define reward -1000 and for target states identified by their vision data label we define reward 100. Fig. 3 shows how homogeneous neighborhoods are by color coding vision labels.

When executing a policy, we continuously filter actions and observations to maintain the current state $\mathbf{b}(h)$ and find the closest neighbor of the embedded states to determine the next action.

V. Experimental Evaluation

In this section we test how our PSR model performs on a realistic problem. First we inspect the learned model's representation geometrically, then we study the filtering (i.e. tracking the state) performance as the robot executes actions and perceives haptic observations. Finally, we exemplify manipulation planning and execution. Evaluations are performed on held-out data consisting of sequences different from the observed training data.

A. Representation Learning

To compare RBF features without actions and string kernel features proposed by us, we learn two PSRs from sample sequences of length $2m = 10$ and select the same 342 of the 1312 training sequence as kernel centers for features (with

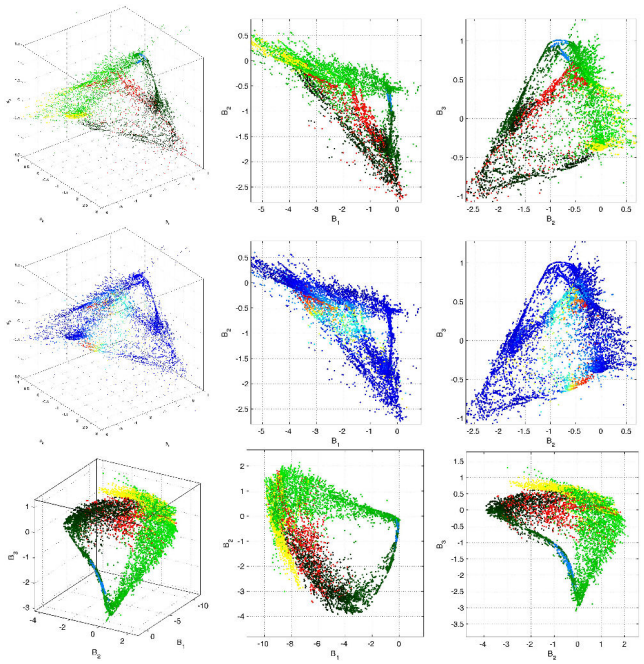


Fig. 3. Projections of the populated 5-dimensional PSR belief space where points represent filtered sequences. *Top:* PSR learned with string kernel features. Colors are angle labels from vision with pointing down (*red*), forward (*blue*), up (*yellow*), and angles in between (shades of *green*). Dimensions 2 and 3 separate *red* and *yellow* states. *Middle:* Same space, light colors indicate high uncertainty. *Bottom:* PSR learned with RBF kernel features. *Red* and *yellow* states are placed adjacently.

length $m = 5$). There are 12 observation kernel centers and for the string kernel we use 12 observation symbols, the number of dimensions is in both cases 5. For visual inspection, we populate both PSR spaces and color every state according to its angle label. In Fig. 3 we can see a gradual change in angle labels, visible by different shades of *green*, for both representations. That means the PSRs correctly arrange states relative to each other (i.e. according to angle) around some empty space. Small angles in *blue* are all located in a particular region and from there gradual changes lead to the two extreme angle positions in *red* and *yellow*. However, the string kernel features allow the model to use dimensions 2 and 3 to separate the most extreme marker states (pointing up and down) while RBF kernel features always place these states adjacently. The representation provided by the string kernel describes the discontinuity between pointing up and pointing down by establishing a large (uncertainty) region. This difference is prime if the representation's geometry shall subsequently be exploited (in form of distances). For instance in planning we can apply distance-based heuristics or use the uncertainty region to define insuperable barriers.

B. Filtering Accuracy

To see if the string kernel PSR captures the system dynamics, we execute and track 4 different action and observation sequences that are not part of the training data. After each action, we collect the vision data for ground truth labels.

The results presented in the top 4 diagrams of Fig. 4 show the filtering performance with ground truth labels in the background and the angle label of the next neighbour embedded state for every step in *blue*. When a sequence starts with large angles, the PSR requires more time steps to arrive at the correct state because of ambiguous observations (Fig. 4, *2nd row*). We can also see the implications of length 5 feature basis functions in that more than 5 steps in a large angle area leads to divergence from ground truth (Fig. 4, *2nd row, right side*).

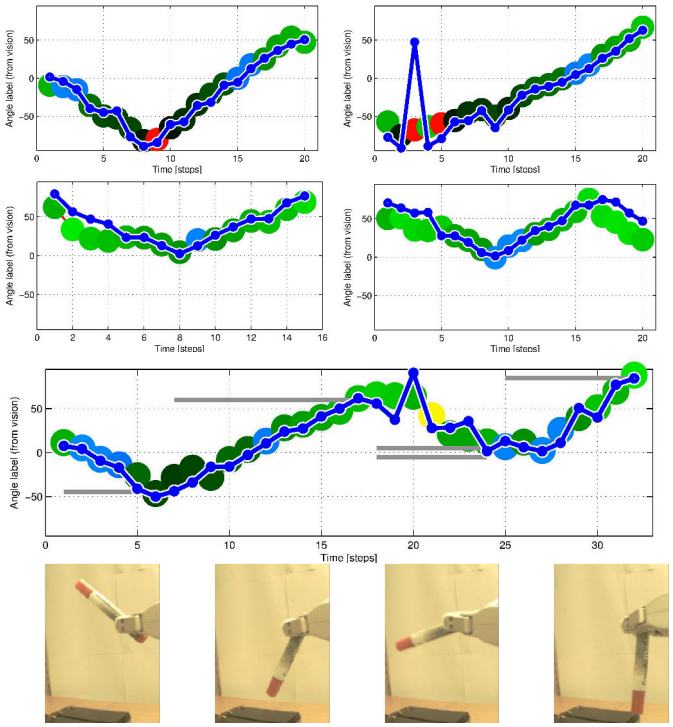


Fig. 4. Filtering of sequences of actions and observations. The colored dots in the background indicate (ground truth) labels from vision data while the *blue* dots show the label of the closest training state used to populate the PSR space. *Bottom two rows:* Filtering results from executing policies where goal labels were: Go to below $-45°$, go over $+60°$, got between -5 and $+5°$, go over $+85°$. Goal labels are indicated by *gray* bars and the photos were taken when ever a target was reached.

C. Planning and Execution

In this experiment, we use the learned PSR model to close the learning-planning loop and perform manipulations. Our goal is to move the marker to vision labels below $-45°$, over $+60°$, between -5 and $+5°$, and over $+85°$ in the listed order. For this we apply value iteration on the set of embedded states as described in Sec. IV-C and execute the resulting greedy policy by filtering each step and referring to the closest embedded state for the next action.

Fig. 5 shows the value functions for each of the commanded goals with *red* and *yellow* for high and *green* and *blue* for low rewards. *Pink* color is used to mark states of negative value. This shows how the different states of different angle labels are distributed in the PSR space and where the states with high uncertainty are located since their negative reward leads to negative value. Execution results are displayed in the

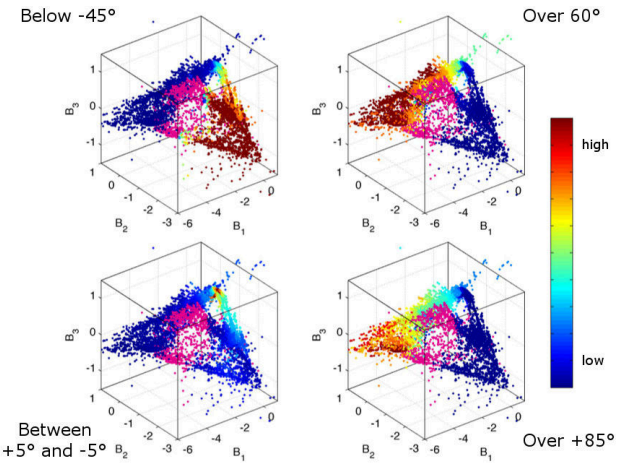


Fig. 5. Value function after value iterations for each embedded state for the experiment described in Sec. V-C. The goal states were defined as below $-45°$, over $+60°$, between -5 and $+5°$, and over $+85°$. Pink color indicates *negative* value resulting from negative reward defined for uncertainty regions. In each plot the goal region can be identified by *red* color. The executed policy greedily follows this value function.

bottom two rows of Fig. 4 where *gray* bars mark goal angles. The policies moved the marker directly to the commanded angles. Because of execution noise, the marker cap directly met the table edge at step 19 and 20 unexpectedly resulting in no rotation (see the supplementary video sequence). This type of incidences are so infrequent that they are not present in training data, which clearly shows the dependency on representative but also completely covering training data.

VI. Conclusion

We have presented work that utilizes Predictive State Representations (PSRs) to perform in-hand manipulation using tactile data. Our approach extends previous work on PSRs to continuous domains where observations cannot ambiguate the state of the system. We have presented a new kernel based approach which allows us to integrate both actions and observations into a single feature space. Further, this allows the PSR framework to be extended such that the elements of the core set are allowed to be of different length, effectively removing a free parameter of the original model. We have shown real robot experiments where the representation learned from cheap but ambiguous tactile observations is geometrically meaningful and suitable for planning in a post-grasp manipulation scenario. In future work we will focus on including non-linear latent variables to learn an even more reduced core set and how to incorporate supervision to learn representations suitable for planning.

References

[1] B. Boots, S. Siddiqi, and G. Gordon, "Closing the learning planning loop with predictive state representations," *I. J. Robotic Research*, vol. 30, pp. 954–956, 2011.

[2] D. Wingate and S. Singh, "On discovery and learning of models with predictive representations of state for agents with continuous actions and observations," in *International joint conference on Autonomous agents and multiagent systems*. ACM, 2007, p. 187.

[3] W. L. Hamilton, M. M. Fard, and J. Pineau, "Modelling sparse dynamical systems with compressed predictive state representations," in *International Conference on Machine Learning*, 2013, pp. 178–186.

[4] B. Boots, A. Gretton, and G. J. Gordon, "Hilbert space embeddings of predictive state representations," in *Intl. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[5] B. Boots, A. Byravan, and D. Fox, "Learning predictive models of a depth camera & manipulator from raw execution traces," in *IEEE ICRA*, 2014.

[6] J. K. Salisbury and J. J. Craig, "Articulated hands force control and kinematic issues," *The International Journal of Robotics Research*, vol. 1, no. 1, pp. 4–17, 1982.

[7] J. Corrales Ramon, V. Perdereau, and F. Torres Medina, "Multi-fingered robotic hand planner for object reconfiguration through a rolling contact evolution model," in *IEEE ICRA*, 2013, pp. 625–630.

[8] R. Deimel and O. Brock, "A novel type of compliant, underactuated robotic hand for dexterous grasping," in *Robotics: Science and Systems*, 2014.

[9] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *International Journal of Robotics Research*, vol. 33, p. 768–782, 2014.

[10] N. Chavan-Dafle, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *IEEE ICRA*, 2014.

[11] R. Deimel, C. Eppner, J. lvarez Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," in *International Symposium on Robotics Research (ISRR)*, 12 2013.

[12] R. Paolini, A. Rodriguez, S. S. Srinivasa, and M. T. Mason, "A data-driven statistical framework for post-grasp manipulation," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 600–615, 2014.

[13] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Robust grasping under object pose uncertainty," *Autonomous Robots*, vol. 31, no. 2-3, pp. 253–268, 2011.

[14] R. Jonschkowski and O. Brock, "State representation learning in robotics: Using prior knowledge about physical interaction," in *Robotics: Science and Systems*, 2014.

[15] S. Singh, M. R. James, and M. R. Rudary, "Predictive state representations: A new theory for modeling dynamical systems," in *Conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 512–519.

[16] M. Rosencrantz, G. Gordon, and S. Thrun, "Learning low dimensional predictive representations," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 88.

[17] B. Boots and G. J. Gordon, "An online spectral learning algorithm for partially observable nonlinear dynamical systems." in *AAAI*, 2011.

[18] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[19] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[20] J. Pearl, *Causality: models, reasoning and inference*. Cambridge University Press, 2000, vol. 29.

[21] B. Wolfe, M. R. James, and S. Singh, "Learning predictive state representations in dynamical systems without reset," in *International conference on Machine learning*. ACM, 2005, pp. 980–987.

[22] M. T. Izadi and D. Precup, "Point-based planning for predictive state representations," in *Advances in Artificial Intelligence*. Springer, 2008, pp. 126–137.

[23] A. Oyama, K. Konolige, S. Cousins, S. Chitta, K. Conley, and G. Bradski, "Come on in, our community is wide open for robotics research!" in *The 27th Annual conference of the Robotics Society of Japan*, 09/2009 2009.

[24] Y. Bekiroglu, D. Song, L. Wang, and D. Kragic, "A probabilistic framework for task-oriented grasp stability assessment," in *IEEE ICRA*, 2013, pp. 3040–3047.

[25] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *The Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.

[26] C. Saunders, H. Tschach, and J. Shawe-Taylor, "Syllables and other string kernel extensions," *International Conference on Machine Learning*, 2002.

[27] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.