Learning Grounded Finite-State Representations from Unstructured Demonstrations

Scott Niekum[1,2], Sarah Osentoski[3], George Konidaris[4], Sachin Chitta[5], Bhaskara Marthi[6], and Andrew G. Barto[2]

[1]The Robotics Institute, Carnegie Mellon University
sniekum@cmu.edu

[2]School of Computer Science, University of Massachusetts Amherst

[3]Robert Bosch Research and Technology Center, Palo Alto, CA

[4]MIT CSAIL

[5]SRI International, Menlo Park, CA

[6]Vicarious, Union City, CA

Abstract

Robots exhibit flexible behavior largely in proportion to their degree of knowledge about the world. Such knowledge is often meticulously hand-coded for a narrow class of tasks, limiting the scope of possible robot competencies. Thus, the primary limiting factor of robot capabilities is often not the physical attributes of the robot, but the limited time and skill of expert programmers. One way to deal with the vast number of situations and environments that robots face outside the laboratory is to provide users with simple methods for programming robots that do not require the skill of an expert.

For this reason, learning from demonstration (LfD) has become a popular alternative to traditional robot programming methods, aiming to provide a natural mechanism for quickly teaching robots. By simply *showing* a robot how to perform a task, users can easily demonstrate new tasks as needed, without any special knowledge about the robot. Unfortunately, LfD often yields little knowledge about the world, and thus lacks robust generalization capabilities, especially for complex, multi-step tasks.

We present a series of algorithms that draw from recent advances in Bayesian non-parametric statistics and control theory to automatically detect and leverage *repeated structure* at multiple levels of abstraction in demonstration data. The discovery of repeated structure provides critical insights into task invariants, features of importance, high-level task structure, and appropriate skills for the task. This culminates in the discovery of a finite-state representation of the task, comprised of grounded skills that are flexible and reusable, providing robust generalization and transfer in complex, multi-step robotic tasks. These algorithms are tested and evaluated using a PR2 mobile manipulator, showing success on several complex real-world tasks, such as furniture assembly.

1 Introduction

The emerging field of *personal robotics* aims to fulfill the promise of intelligent robots that can safely serve, assist, and cooperate with humans in the home and workplace. Such robots could revolutionize the modern factory, help to take care of the disabled and elderly, or simply perform monotonous chores, possibly becoming as ubiquitous as the personal computer. Great leaps have been made toward this vision in recent years, to the point that a wide range of robotic behaviors can be programmed, given enough effort; robots now exist that can fold laundry [Miller et al., 2012], play catch [Bauml et al., 2010], bake cookies [Bollini et al., 2011], and play pool [Pastor et al., 2011a]. However, despite these impressive developments, general purpose personal robots that can operate competently in human environments are conspicuously missing. There are two central reasons for this. First, most robotic feats are performed in controlled conditions in the laboratory, whereas personal robots must be able to adaptively function in many different environments. Second, many robot behaviors are carefully hand-coded by experts to provide the robot with task-specific functionalities and knowledge. While effective on a small scale, this is not a sustainable model for programming personal robots to perform the myriad of tasks desired by end-users.

One way to deal with the vast number of tasks and environments that personal robots must face outside the laboratory is to provide end-users with simple, user-friendly methods for programming robots. Thus, in recent years, learning from demonstration (LfD) [Argall et al., 2009] has become a popular alternative to traditional robot programming methods, aiming to provide a natural mechanism for quickly teaching robots. By simply *showing* a robot how to perform a task, users can demonstrate new tasks without any specialized knowledge.

LfD has had many successes in teaching robots tennis swings [Schaal, 2003], walking gaits [Nakanishi et al., 2004], pick-and-place tasks [Mühlig et al., 2011], and even complex helicopter maneuvers [Abbeel and Ng, 2004]. However, demonstrations are often simply treated as trajectories to be mimicked (inverse reinforcement learning [Abbeel and Ng, 2004; Ziebart et al., 2008] is a notable exception to this), providing little knowledge about the task or the environment. This, in turn, leads to brittle policies that often cannot generalize well to new situations—especially in the case of complex, multi-step tasks. We argue that demonstration data, in fact, contains *deep, exploitable structure* that can be discovered from a small number of examples and leveraged to provide robust generalization of complex tasks.

This leads us to ask: what information is necessary for strong generalization in complex robotic tasks? Generalization requires robots to adapt to new placements of objects in the world, identify and recover from possible contingencies, abstract away unimportant details, and recognize user intentions, among other things. All of these requirements can be summarized more simply as being able to identify what invariants must hold in a task, what is unimportant or allowed to vary, and why. The answers to these questions provide *explanatory power*, allowing the robot to explain the data it has observed and reason about what to do in new situations. However, to answer these questions, the robot requires an understanding of the world—in our case, a collection of data that describes actions, objects, and their relationships. We present a series of algorithms that look for *repeated structure* across multiple demonstrations to discover and exploit these vital task semantics.

The core of the presented algorithms are formed by a Bayesian nonparametric model—a model that does not have a fixed size, but instead infers an appropriate complexity in a fully Bayesian manner without overfitting the data or requiring model selection. This model is used to discover repeated structure in the demonstration data, identifying primitive motions that best explain the demonstrations and that can be recognized across different demonstrations and tasks. This process converts noisy, continuous demonstrations into a simpler, coherent discrete representation. This discrete representation is then leveraged to find additional structure, such as appropriate coordinate frames for actions, task-level sequencing information, and higher-level skills that are *grounded* by the robot's observations. Finally, this collection of data is combined to construct a finite-state representation of the task, comprised of grounded skills that leverage environmental feedback to adaptively perform complex, multi-step tasks. These algorithms are evaluated on several difficult real-world tasks, including furniture assembly, using a PR2 mobile manipulator.

2 Background and Related Work

We now provide a survey of relevant related work and background material. First, a short overview of learning from demonstration algorithms is provided, with a focus on segmentation and skill learning. Next, Bayesian nonparametric methods for time series analysis are introduced conceptually, including the BP-AR-HMM. Finally, we end with a mathematical discussion of Dynamic Movement Primitives.

2.1 Skill Learning from Demonstration

Learning from demonstration (LfD) [Argall et al., 2009; Billard et al., 2008] is an approach to robot programming in which users demonstrate desired skills to a robot. Ideally, nothing is required of the user beyond the ability to demonstrate the task in a way that the robot can interpret. Example demonstration trajectories are typically represented as time-series sequences of state-action pairs that are recorded during the teacher's demonstration. The set of examples collected from the teacher is then often used to learn a policy (a state to action mapping) or to infer other useful information about the task that allows for generalization beyond the given demonstrations.

A variety of approaches have been proposed for LfD, including supervised learning [Akgun et al., 2012; Atkeson and Schaal, 1997; Calinon and Billard, 2007; Chernova and Veloso, 2007; Dong and Williams, 2011; Grollman and Jenkins, 2008], (inverse) reinforcement learning [Abbeel and Ng, 2004; Konidaris et al., 2012; Smart and Kaelbling, 2002; Ziebart et al., 2008], and behavior based approaches [Nicolescu and Matarić, 2003]. However, this work has generally been limited to tasks that can be represented with a single monolithic policy or that are broken up into subtasks by hand. In a recent example, Pastor et al. [2011a] use Dynamic Movement Primitives (DMPs) [Ijspeert et al., 2003] to acquire single motor skills from structured demonstrations of a complex billiards shot. In their framework, multiple imperfect demonstrations of a skill are used to learn the initial parameters of a DMP controller, which is then improved using reinforcement learning. In addition, the statistics collected from the examples are used to predict the outcome of new executions of the same skill, to allow early termination if failure seems likely.

While many approaches enable the learning of a single policy or skill from data, some approaches perform automatic segmentation of the demonstrations into multiple skills. Jenkins and Matarić [2004a; 2004b] introduced Spatio-Temporal Isomap in order to find the underlying low-dimensional manifolds within a set of demonstrated data. This work extends the dimensionality reduction technique Isomap to include temporal information and allows the discovery of repeated motion primitives. However, segmentation in this model is performed with a heuristic. Dixon and Khosla [2004] demonstrate that generalizable motions can be parameterized as linear dynamical systems. This algorithm also uses heuristic segmentation and cannot recognize repeated instances of skills.

Gienger et al. [2010] segment skills based on co-movement between the demonstrator's hand and objects in the world and automatically find appropriate task-space abstractions for each skill. Their method can generalize skills by identifying task frames of reference, but cannot describe skills like gestures or actions in which the relevant object does not move with the hand. Rather than explicitly separate skills into separate policies, work by Cederborg et al. [2010] represents a flexible number of skills as a single policy using the Incremental Local Online Gaussian Mixture Regression algorithm. These skills are implicitly separated through the use of a local Gaussian mixture representation, where each mixture is localized in a different part of the state space.

Other recent work has examined using principled statistical techniques to segment example trajectories into multiple skills. Grollman and Jenkins [2010a] introduce the Realtime Overlapping Gaussian Expert Regression (ROGER) model to estimate the number of subtasks and their policies in a way that avoids *perceptual aliasing*, a condition in which perceptual information alone is not sufficient to choose the correct next action. Butterfield et al. [2010] extend the Hierarchical Dirichlet Processes Hidden Markov Model (HDP-HMM) to handle perceptual aliasing and automatically discover an appropriate number of skills. Although we introduce a Bayesian mechanism to parse demonstration trajectories, rather than inferring policies directly, we discover repeated dynamical systems which are considerably simpler to model than policies and generally require less data for inference.

The CST algorithm [Konidaris et al., 2012, 2010] uses an online changepoint detection method to segment example trajectories and then merges the resulting chains of skills into a skill tree. This approach simultaneously segments the trajectories and discovers abstractions, but cannot recognize repeated skills to assist with the segmentation process. Kulic et al. [2009] demonstrate an online method that can recognize repeated motion primitives to improve segmentation as additional data is collected by assuming that data points from the same primitive are generated by the same underlying distribution. Ciappa and Peters [2010] model repeated skills as being generated by one of a set of possible hidden trajectories, which is rescaled and noisy. To guide segmentation, they define an upper bound on the number of possible skills and explicitly constrain segment lengths.

Several less traditional approaches to LfD exist as well. One approach by Rozo et al. [2013] does not consider trajectory learning at all, but instead learns to cooperate with a human by learning appropriate stiffness and impedance policies from haptic information gathered from two-person task demonstrations. Kjellstrom and Kragic [2011] eschew traditional policy learning, and instead use visual data to watch the demonstrator's hand to learn the affordances of objects in the environment, leading to a notion of object-centric skills. Ekvall and Kragic [2008] use multiple examples of a task to learn task constraints

and partial orderings of primitive actions so that a planner can be used to reproduce the task.

A special case of learning from demonstration is that of inverse reinforcement learning (IRL) [Ng and Russell, 2000], in which the agent tries to infer an appropriate reward or cost function from demonstrations. Thus, rather than try to infer a policy directly from the demonstrations, the inferred cost function allows the agent to learn and improve a policy from experience to complete the task implied by the cost function. This is typically done via reinforcement learning methods. IRL techniques typically model the problem as an Markov Decision Process (MDP) and require an accurate model of the environment [Abbeel and Ng, 2004; Neu and Szepesvári, 2007; Ramachandran and Amir, 2007], but some recent methods have been proposed to circumvent this requirement by creating local control models [Tang et al., 2010], and by using an approach based on KL-divergence [Boularias et al., 2011], respectively. Maximum entropy methods have also been suggested as a way to deal with ambiguity in a principled probabilistic manner [Ziebart et al., 2008]. Michini and How [2012] present a Bayesian Nonparametric method to find an appropriate number of reward functions from demonstration data, but do not then use them for policy learning.

2.2 Bayesian Nonparametric Time Series Analysis

Hidden Markov models (HMMs) are generative Bayesian models that have long been used to make inferences about time series data. An HMM models a Markov process with discrete, unobservable hidden states, or modes[1], which generate observations through mode-specific emission distributions. A transition function describes the probability of each mode at time $t+1$ given the mode at time t, but observations are limited to being conditionally independent given the generating modes. Given a set of observations, the forward-backward and Viterbi algorithms can be used to efficiently infer parameters for the model and determine the most likely sequence of modes that generated the data. Unfortunately, the number of modes must be specified *a priori* or chosen via model selection, which can be difficult if the number of modes is not approximately known. This limits the usefulness of HMM inference when dealing with unstructured data. However, recent work in Bayesian nonparametrics offers a principled way to overcome these limitations.

The Hierarchical Dirichlet Process HMM (HDP-HMM) [Teh et al., 2006] uses a Hierarchical Dirichlet process prior over the transition and emission distributions of an HMM. The HDP-HMM allows for a potentially infinite number of latent modes to be discovered in a fully Bayesian manner, reducing the risk of overfitting and the need for prior knowledge. Since an HDP-HMM models an underlying Markov process, the probability of staying in the same mode for n consecutive steps is multiplicative, implying geometric mode durations, which can cause rapid mode switching. This assumption is incorrect for many real-world data sets, motivating the Sticky HDP-HMM [Fox et al., 2008], which adds a learned self-transition bias for each mode. An HDP-HMM also assumes that a mode emits observations that are independent and identically distributed, but this is clearly not true in the case where the observations exhibit temporal dependencies. Autoregressive HMMs [Poritz, 1982] have been proposed to represent these dependencies by adding causal links between observations in the HMM, but it can be difficult to represent the transition structure

[1]We refer to hidden states as *modes*, as to not confuse them with the RL concept of states.

when the observation space is large or continuous.

The Beta Process Autoregressive HMM (BP-AR-HMM) [Fox et al., 2009], shown in Figure 1, is a recent model that integrates many of the advantages of the aforementioned models, while extending them and providing principled solutions to some of their shortcomings. Like the Sticky HDP-HMM, it learns duration distributions from data. Improving on the HDP-HMM mechanism for sharing modes across time series, the BP-AR-HMM uses a beta process prior that leverages an infinite feature-based representation, in which each time series can exhibit a subset of the total number of discovered modes and switch between them in a unique manner. Thus, a potentially infinite library of modes can be constructed in a fully Bayesian way, in which modes are flexibly shared between time series, and an appropriate number of modes is inferred directly from the data, without the need for model selection. The BP-AR-HMM is also autoregressive and can describe temporal dependencies between continuous observations as a Vector Autoregressive (VAR) process, a special case of a linear dynamical system (LDS).

The generative model for the BP-AR-HMM can be summarized as follows [Fox et al., 2011]:

$$
\begin{aligned}
&B|B_0 \sim \mathrm{BP}(1, B_0) \\
&X_i|B \sim \mathrm{BeP}(B) \\
&\pi_j^{(i)}|\boldsymbol{f_i}, \gamma, \kappa \sim \mathrm{Dir}([\gamma, ..., \gamma+\kappa, \gamma, ...] \otimes \boldsymbol{f_i}) \\
&z_t^{(i)} \sim \pi_{z_{t-1}^{(i)}}^{(i)} \\
&\mathbf{y}_t^{(i)} = \sum_{j=1}^{r} A_{j, z_t^{(i)}} \mathbf{y}_{t-j}^{(i)} + \mathbf{e}_t^{(i)}(z_t^{(i)}).
\end{aligned}
$$

First, a draw B from a Beta Process (BP) provides a set of global weights for the potentially infinite number of states. Then, for each time series, an X_i is drawn from a Bernoulli Process (BeP) parameterized by B. Each X_i can be used to construct a binary vector $\boldsymbol{f_i}$ indicating which of the global features, or states, are present in the i^{th} time series. Thus, B encourages sharing of features amongst multiple time series, while the X_i leave room for variability. Next, given the features that are present in each time series, for all states j, the transition probability vector $\pi_j^{(i)}$ is drawn from a Dirichlet distribution with self transition bias κ. A state $z_t^{(i)}$ is then drawn for each time step t from the transition distribution of the state at the previous time step. Finally, given the state at each time step and the *order* of the model, r, the observation is computed as a sum of state-dependent linear transformations of the previous r observations, plus mode-dependent noise. This model will be used in Sections 3 and 4 to discover an appropriate number of dynamical systems that describe, and can be shared across, multiple observed robot demonstration trajectories.

2.3 Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) [Ijspeert et al., 2003] provide a framework in which dynamical systems can be described as a set of nonlinear differential equations in which a linear point attractive system or limit cycle oscillator is modulated by a nonlinear function.

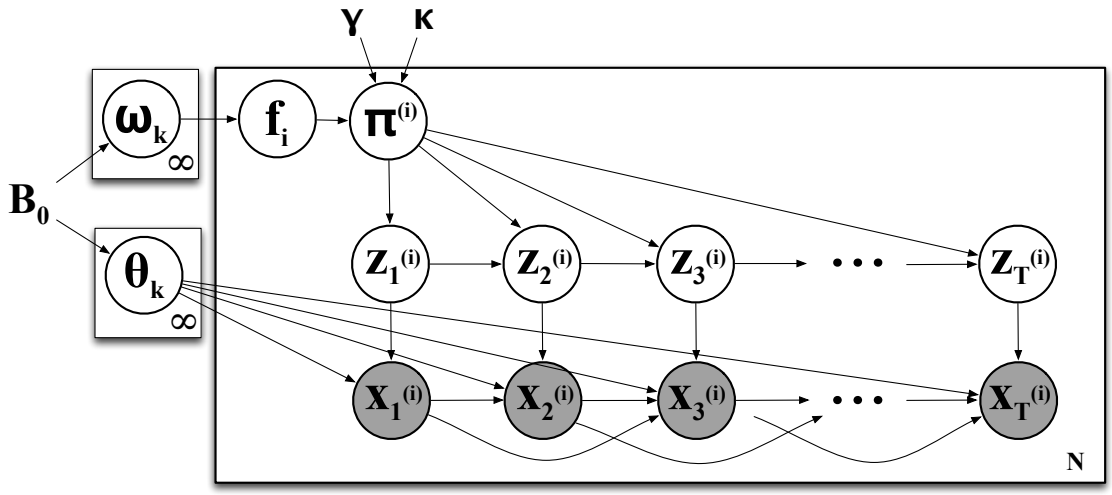


Figure 1: The Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM). Here, the Beta Process draw B has been separated into masses $\boldsymbol{\omega}$ and parameters $\boldsymbol{\theta}$. Each parameter θ_k consists of r transition matrices $\mathbf{A}$ and a covariance term $\mathbf{e}$. Based on Figure 1 from Fox et al. [2009].

Stability and convergence are guaranteed by introducing an additional canonical system, governed by linear equations that control a 0 to 1 phase variable that attenuates the influence of the nonlinear function over time. DMPs provide simple mechanisms for LfD, RL policy improvement, and execution, which scale easily in time and space and can support discrete or oscillatory movements [Schaal et al., 2004]. In this work, we focus on the use of point attractive systems for implementing discrete movements with DMPs.

A discrete movement DMP can be described by the transformation system,

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) \tag{1}$$

$$\tau \dot{x} = v, \tag{2}$$

and the canonical system,

$$\tau \dot{s} = -\alpha s, \tag{3}$$

for spring constant K, damping constant D, position x, velocity v, goal g, phase s, temporal scaling factor τ, and constant α [Pastor et al., 2009]. The nonlinear function f can be represented as a linear combination of basis functions $\psi_i(s)$, scaled by the phase variable, s:

$$f(s) = \sum_{i=1}^{N} w_i \psi_i(s) s. \tag{4}$$

In this work, we use the univariate Fourier basis [Konidaris et al., 2011] for a function approximator, though others have used normalized radial basis functions [Pastor et al.,

2009]. The spring and damping constants can be set to ensure critical damping, but we still must find appropriate weights w_i for the nonlinear function f.

Given a demonstration trajectory $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ with duration T, we can use LfD to learn a set of values for these weights [Schaal et al., 2004]. Rearranging equation 1, integrating equation 3 to convert time to phase, and substituting in the demonstration for the appropriate variables, we get:

$$f_{\text{target}}(s) = \frac{-K(g - x(s)) + D\dot{x}(s) + \tau\,\ddot{x}(s)}{g - x_0}. \tag{5}$$

Setting the goal to $g = x(T)$, and choosing τ such that the DMP reaches 95% convergence at time $t = T$, we obtain a simple supervised learning problem to find the weights w_i for the basis functions. We use standard linear regression for this task. This LfD procedure provides us with weights for a baseline controller that can be further improved through practice using RL [Schaal et al., 2004], though we do not do so in this work.

3 Learning from Unstructured Demonstrations

Ideally, an LfD system should learn to perform and generalize complex tasks given a minimal number of demonstrations without requiring specialized knowledge about the robot. Much LfD research has focused on the case in which the robot learns a monolithic policy from a demonstration of a simple task with a well-defined beginning and end [Abbeel and Ng, 2004; Ijspeert et al., 2003; Pastor et al., 2011a]. This approach often fails for complex tasks that are difficult to model with a single policy. Thus, structured demonstrations are often provided for a sequence of subtasks, or *skills*, that are easier to learn and generalize than the task as a whole, and which may be reusable in other tasks.

However, a number of problems are associated with segmenting tasks by hand and providing individual skill demonstrations. Since the most natural way to demonstrate a task is by performing it from start to finish, dividing a task into component skills is not only time-consuming, but often difficult—an effective segmentation may require knowledge of the robot's kinematic properties, internal representations, and existing skill competencies. Since skills may be repeated within and across tasks, defining skills also requires qualitative judgements about when two segments can be considered a single skill, or in deciding the appropriate level of granularity at which to perform segmentation. Users cannot be expected to manually manage this collection of skills as it grows over time.

For this reason, recent work has aimed at automating the segmentation process [Butterfield et al., 2010; Chiappa and Peters, 2010; Grollman and Jenkins, 2010a; Konidaris et al., 2012, 2010; Kulic et al., 2009]. Collectively, this body of work has addressed four key issues that are critical to any system that aims to learn increasingly complex tasks from unstructured demonstrations (i.e. demonstrations that are unsegmented, possibly incomplete, and may originate from multiple tasks or skills). First, the robot must be able to recognize repeated instances of skills and generalize them to new settings. Second, segmentation should be able to be performed without the need for *a priori* knowledge about the number or structure of skills involved in a task. Third, the robot should be able to identify a broad, general class of skills, including object manipulation skills, gestures, and

goal-based actions. Fourth, the representation of skill policies should be such that they can be improved through practice.

Although many of these issues have already been addressed individually, no system that we are aware of has jointly addressed them all in a principled manner. Our contribution is a framework that addresses all of these issues by integrating a principled Bayesian nonparametric approach to segmentation with state-of-the-art LfD techniques as a step towards a natural, scalable system that will be practical for deployment to end users. Segmentation and recognition are achieved using a Beta-Process Autoregressive HMM [Fox et al., 2009], while Dynamic Movement Primitives [Ijspeert et al., 2003] are used to address LfD, policy representation, and generalization.

The combination and extension of these techniques allows a robot to segment and identify repeated skills in unstructured human demonstrations, create baseline skill policies from demonstration segments, leverage previous skill learning, and expand its skill library as needed. We demonstrate this approach with experimental results using the PR2 mobile manipulator on both a simulated and physical task.

3.1 Approach

We now introduce a framework which integrates four major capabilities critical for the robust learning of complex tasks from unstructured demonstrations [Niekum et al., 2012]. First, the robot must be able to recognize repeated instances of skills and generalize them to new settings. Given a set of demonstrations for a task, we use the BP-AR-HMM to parse the demonstrations into segments that can be explained by a set of latent skills, represented as Vector Autoregressive (VAR) processes. The BP-AR-HMM enables these skills to be shared across demonstrations and tasks by employing a feature-based representation in which each skill corresponds to a feature that may or may not be present in a particular trajectory. Furthermore, this representation allows each trajectory to transition between skills in a unique manner, so that skills can be identified flexibly in a variety of situations, while still retaining globally shared properties.

Segmentation of trajectories into VAR models allows for tractable inference over the time-series dependencies of observations and provides a parameterization of each skill so that repeat instances can be recognized. This representation models how state changes over time, based on previous state values, potentially allowing instances of the same underlying skill to be recognized, even when performed with respect to different coordinate frames. The BP-AR-HMM also models skill-dependent noise characteristics to improve the identification of repeated skills. By recognizing repeated skills, a skill library can be incrementally constructed over time to assist in segmenting new demonstrations. Additionally, skill controllers that have been previously learned and improved through practice can be reused on new tasks. Thus, recognition of repeated skills can reduce the amount of demonstration data required to successfully segment and learn complex tasks. Similarly, if we have multiple examples of a skill we can discover invariants, such as a relevant coordinate frame, that allow us to generalize the skill to new situations robustly.

Second, segmentation must be able to be performed without the need for *a priori* knowledge about the number or structure of skills involved in a task. The BP-AR-HMM places a beta process prior over the matrix of trajectory-feature assignments, so that a poten-

tially infinite number of skills can be represented; the actual finite number of represented skills is decided upon in a principled, fully Bayesian way. Skill durations are modeled indirectly through a learned self-transition bias, preventing skills from being over-segmented into many small components unnecessarily. The BP-AR-HMM also provides reliable inference, having only a few free parameters that are robust to a wide range of initial settings and hyperparameters that conform to the data as inference progresses. Thus, little tuning should be necessary for varying tasks for a given robotic platform.

Third, our system must be able to identify a broad, general class of skills. Since our segmentation method is based upon state changes, rather than absolute state values, we are able to identify a wide array of movement types ranging from object manipulation skills to gestures and goal-based actions. Furthermore, by identifying the relevant coordinate frame of repeated skills, we can discover specific objects and goals in the world that skills are associated with.

Fourth, the representation of skill policies should be such that they can be easily generalized and improved through practice. To accomplish this, we represent skill controllers in the DMP framework. The spring-damper mechanics of a DMP allow for easy generalization, since the start and goal set-points can be moved, while still guaranteeing convergence and maintaining the "spirit" of the demonstration through the output of the nonlinear function. In addition to affording a simple LfD algorithm, the linear function approximator used to represent the nonlinear forcing function allows for skill improvement through practice by modifying the weights through an RL algorithm such as the Natural Actor-Critic [Peters et al., 2005], though we do not do so in this work.

3.2 Algorithm

Figure 2 and Algorithm 1 provide an overview of the algorithm, each step of which is described in detail in the following sections.

3.2.1 Demonstrations

First, demonstrations are captured by receiving input from the user, such as a kinesthetic demonstration, or motion capture data. The only restriction on the demonstration modality is that two main types of data must be able to be recorded at points equally spaced in time—the 6-DOF cartesian position of the end effector of the demonstrator (i.e. a human hand in the case of motion capture, and the robot's gripper in a kinesthetic demonstration), along with any other relevant data about the end effector (such as the open/closed status of the gripper), and the 6-DOF cartesian position of all task-relevant objects in the world. Subsequently, the recorded cartesian trajectory of the end effector will be referred to as the *demonstration trajectory*, and recorded object positions as the *object pose observations*.

3.2.2 Segmentation

Next, the BP-AR-HMM (based on an implementation made available by Emily Fox[2]) is used to segment the set of collected demonstration trajectories. The demonstrations are

[2] `http://stat.wharton.upenn.edu/~ebfox/software`

preprocessed so that the variance of the first differences of each dimension of the data is 1, as in Fox et al. [2011], and adjusted to be mean zero. We choose an autoregressive order of 1 and use identical parameters as those used by Fox on a human exercise motion capture dataset [Fox et al., 2011], with one exception—in the simulated experiments, we adjust the matrix-normal inverse-Wishart prior on the dynamic parameters, since the simulated data have significantly different statistical properties from that in Fox et al. [2011]. To segment the demonstrations, we run the combined Metropolis-Hastings and Gibbs sampler 10 times for 1000 iterations each, producing 10 segmentations. Qualitatively, the segmentations across runs were very consistent, but to ensure good results, the segmentation from the 10 runs with the highest log likelihood of the feature settings is selected.

3.2.3 Coordinate Frame Detection

After the demonstrations are segmented, each segment is examined to infer the coordinate frame that it is occurring in. Even though segments assigned to the same skill correspond to similar movements, they may be happening in different frames of reference. For example, a repeated reaching motion may be classified as being generated by the same skill, but be reaching toward several different objects. In order to robustly replay tasks in novel configurations, it is desirable to determine which coordinate frame each segment is associated with, so that DMP goals can be generalized correctly.

We define a coordinate frame centered on each known object, along with one centered at the torso of the robot. Other frames could be used as well if desired, such as a frame relative to the gripper, or a world frame. Then, the final point of each segment is plotted separately in each of the coordinate frames, and clusters are found in each frame by identifying points within a Euclidean distance threshold of each other. The reasoning is that clusters of points indicate that multiple segments have similar endpoints in a particular coordinate frame, suggesting that the skill often occurs in that frame of reference.

After the points are clustered in each frame, all the singleton clusters are discarded. If any remaining segment endpoint belongs only to a cluster in a single coordinate frame, then the evidence is unambiguous, and that segment is assigned to that coordinate frame. Otherwise, if a segment endpoint belongs to clusters in multiple frames, it is simply assigned to the frame corresponding to the largest cluster. Figure 3 shows an example of this clustering process. It should be emphasized that any coordinate frame inference method could be used in place of ours, and many other skill invariants could be exploited. The purpose of this method is primarily to demonstrate the utility of being able to segment and recognize repeated skills.

3.2.4 Task Replay

To perform a task in a novel configuration, we first determine the poses and identities of objects in the new scene. The position of each object is then examined to find the demonstration that begins with the objects in a configuration that is closest to the current one in a Euclidean sense. We only consider demonstrations that have an identified coordinate frame for every segment, so that the task will generalize properly. A DMP is then created and trained using the LfD algorithm from section 2.3 for each segment in the demonstration. However, rather than using the final point of a segment as the goal of a DMP, each goal

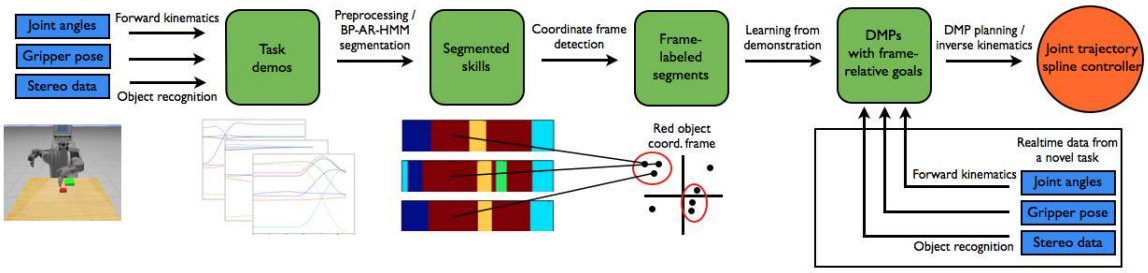


Figure 2: Overview of the framework used in the experiments, as described in section 3.2

Algorithm 1 Learning from Unstructured Demonstrations

Input: Demonstration trajectories $\mathbf{T} = (T_1, \ldots, T_N)$, corresponding object pose observations $\mathbf{O} = (O_1, \ldots, O_N)$, and current object poses $O_{current}$

1. *Segment the demonstrations with the BP-AR-HMM:*
   (segments, labels) = $Segmentation(\mathbf{T})$
2. *Find coordinate frame assignments:*
   coord_frames = $CoordinateFrameDetection$(segments, labels, $\mathbf{O}$)
3. *Learn params of a DMP for each segment:*
   dmps = $LearnDmpParams$(segments, coord_frames)
4. *Execute sequence of DMPs corresponding to the segment ordering of the most similar demonstration:*
   demo_num = $NearestFirstObservation(O_{current}, \mathbf{T})$
   $ExecuteSequence$(dmps, demo_num)

is adjusted based on the coordinate frame that the segment takes place in. If the segment is associated with the torso frame, it requires no adjustment. Otherwise, if it is associated with an object frame, the goal is adjusted by the difference between the object's current position and its position in the demonstration. Finally, the DMPs are executed in the sequence specified by the demonstration. A plan is generated by each of the DMPs until the predicted state is within a small threshold of the goal. Each plan is a Cartesian trajectory (plus a synchronized gripper state) that is converted into smooth joint commands using inverse kinematics and spline interpolation.

3.3 Experiments

For the first two experiments, we use a simulated Willow Garage PR2 mobile manipulator and the ROS framework; the final experiment uses a real PR2. The PR2 personal robot is a two-armed robot platform with an omnidirectional base designed for mobile-manipulation tasks. The PR2 is equipped with 7-degrees of freedom compliant arms and a sensor suite useful for mobile manipulation tasks, including a tilting laser scanner attached to the head,

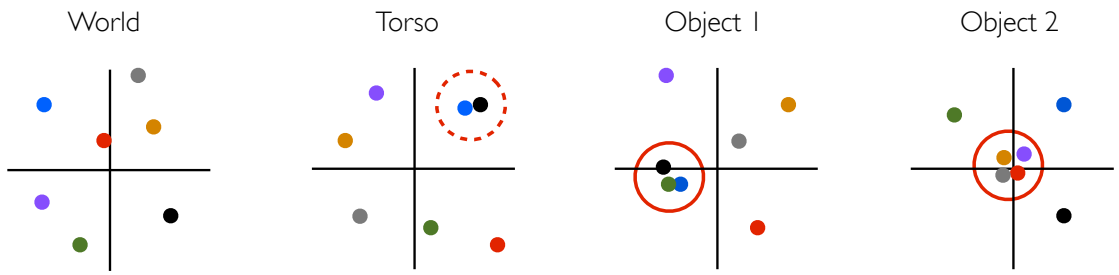


Figure 3: An illustrative example showing the endpoints of seven trajectories plotted with respect to four different coordinate frames. The solid circles denote detected clusters, while the dotted circle indicates a candidate cluster that is rejected as noise, since all associated points can be assigned to a potentially larger cluster. In our experiments, this clustering happens in three dimensions, rather than two as illustrated here.

two stereo cameras, a Microsoft Kinect, a laser scanner mounted on the base, and a body mounted IMU. By design, the PR2 does not have to deal with the many degrees of freedom that come along with legs or complex hands, making it easier to design interfaces, create controllers, and interact with the robot.

We used hand-coded controllers to provide task demonstrations to the simulated robot. The robot is placed in a fixed position in front of a table, as shown in Figure 4. At the beginning of each demonstration, the robot looks downward and captures a stereo image of the table. It then removes the flat table top and obtains a point cloud for each of the objects on the table, recording their positions and dimensions. On the real robot, object positions are determined by a visual fiducial placed on each object of interest. Once the demonstration begins, data are collected by recording the 7 joint angles in the left arm and the gripper state (a scalar indicating its degree of closed-ness). Offline, the joint angles are converted to a series of 3D Cartesian positions and 4D quaternion orientations, which are subsampled down to 10 Hz and smoothed, along with the gripper positions.

3.3.1 Experiment 1: Pick and Place (Simulated)

The first experiment demonstrates the ability of a robot in our framework to learn and generalize a complex task by segmenting multiple task demonstrations, identifying repeated skills, and discovering appropriate segment reference frames. Each instance of the task begins with two blocks on the table—a smaller red block and a larger green block. The robot always starts in a "home" configuration, with its arms at its sides so that its field of view is unobstructed. We provide 5 task demonstrations for 5 different configurations of the blocks, as shown in the first row of Figure 4 (configurations 1 and 5 are identical, but the demonstration is performed at a higher speed in the latter configuration). In each demonstration, the robot first picks up the red block, returns to the home position, places

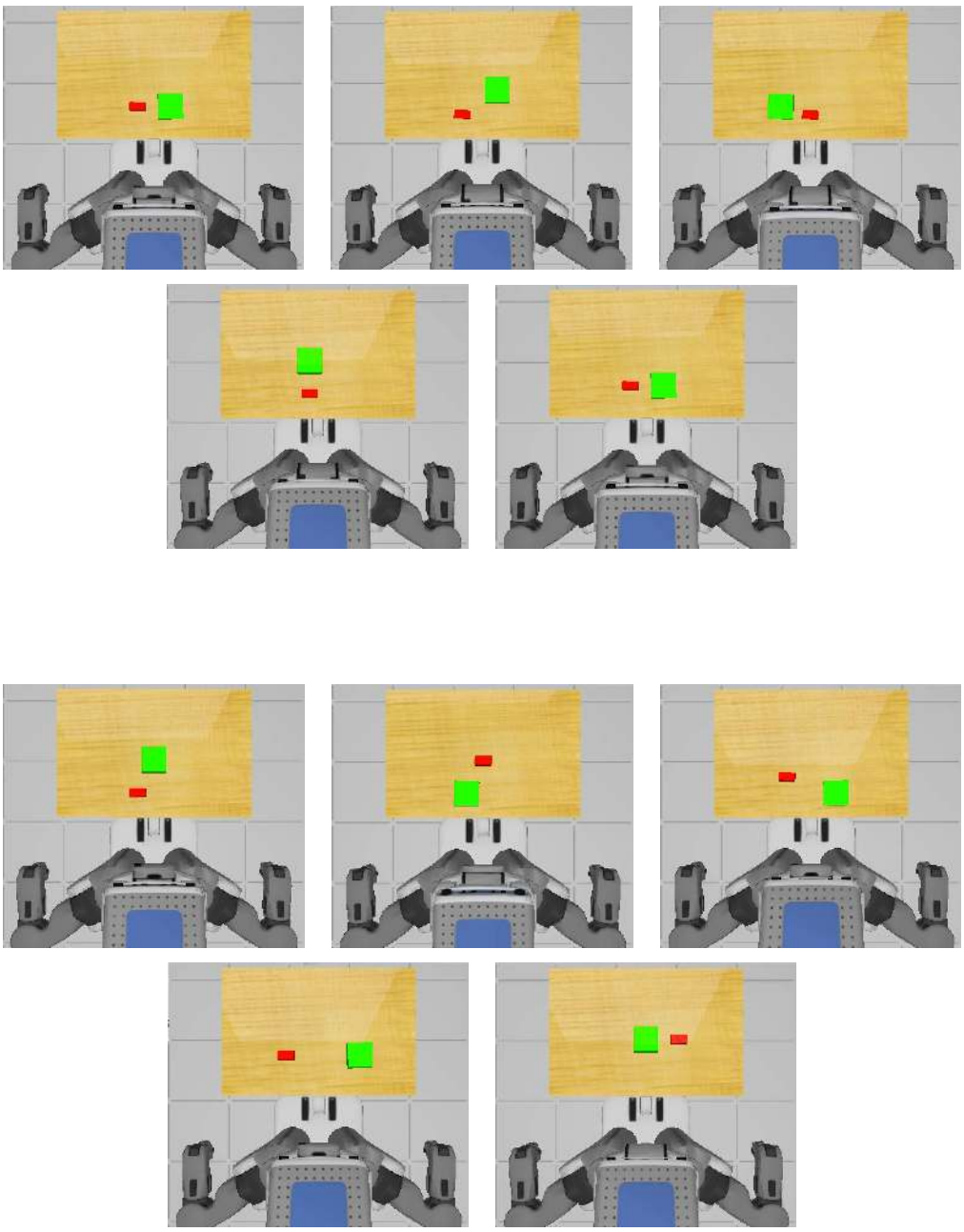


Figure 4: A top-down view of the robot and table layout for 5 task demonstration configurations (top) and 5 novel test configurations (bottom).

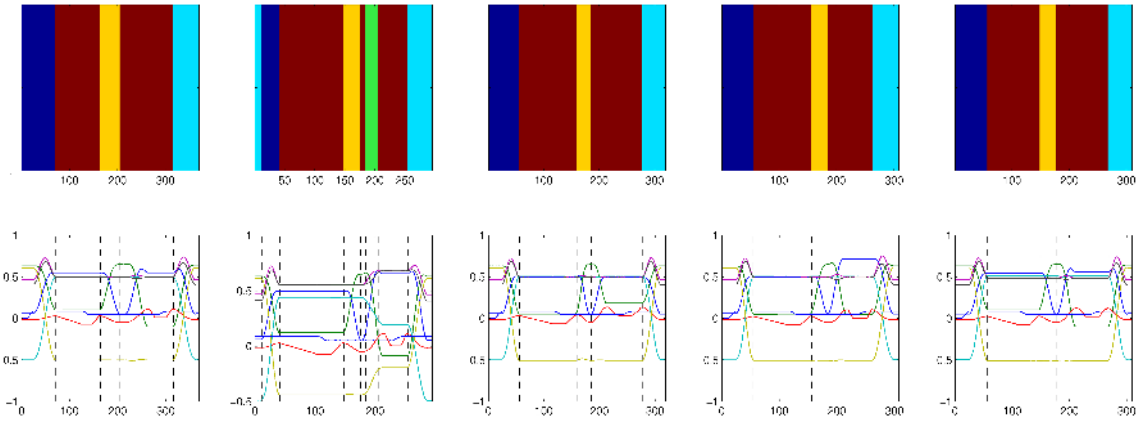


Figure 5: Top: BP-AR-HMM segmentations of the 5 demonstration trajectories for the pick and place task. Time (in tenths of a second) is shown on the x-axis. Skill labels at each time step are indicated by unique colors. Bottom: Segmentation points overlaid on the demonstrated 8D movement data.

the red block on the green block, and returns to the home position once more.[3]

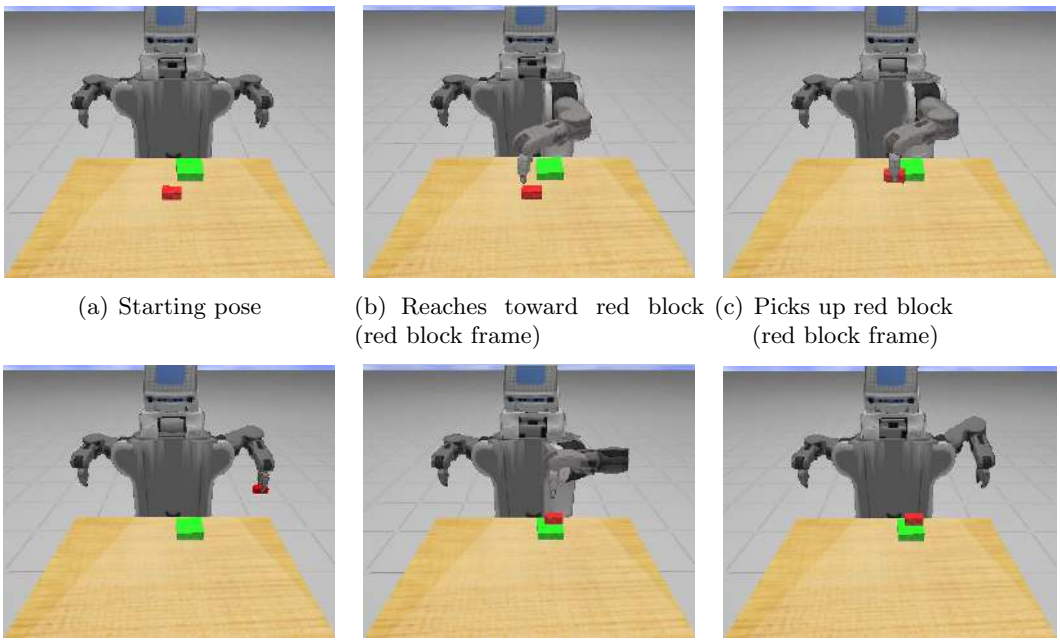
Figure 5 shows the results of segmentation. The top row shows one colored bar per skill, while the bottom row displays the skill divisions overlaid on a plot of each of the 8 dimensions of the demonstration data. The BP-AR-HMM consistently recognizes repeated skills across demonstrations, even though they occur at differing speeds and with different goals. The segmentations are highly similar, with the exception of the second demonstration, which identifies one additional skill that the others do not have. It is worth noting that despite the extra skill being inserted in the segmentation, the rest of the segmentation is essentially the same as the others. This is a direct benefit of the BP-AR-HMM allowing each trajectory to have its own switching dynamics, while sharing global features.

Next, we examine task generalization to 5 novel test configurations, shown in the bottom row of Figure 4, to determine whether our segmentation produced task-relevant results. Our method was able to successfully identify a coordinate frame for every segment except the extra segment in demonstration two (which is impossible to infer, since there is only one example of it). Using this information, the robot performed task replay as described in section 3.2.4. In all 5 novel configurations, the robot was able to successfully generalize and place the red block on the green block.[4]

Figure 6 shows the starting state of the robot and the resulting state after each DMP is executed in a novel test configuration. Here, it becomes clear that the results of both the segmentation and coordinate frame detection are intelligible. The first skill is a reaching skill to right above the red block. The second skill moves down, grasps the red block, and

[3]Due to the planning delay in the hand written controllers there are some pauses between segments which we remove to avoid biasing the segmentation algorithm.

[4]The green block in novel configuration 4 was partially out of the robot's visual range, causing part of it to be cut off. Thus, it placed the red block too close to the edge of the green block, causing it to tumble off. However, given the available information, it acted correctly.



(a) Starting pose

(b) Reaches toward red block (red block frame)

(c) Picks up red block (red block frame)

(d) Returns to home position (torso frame)

(e) Places red block on green block (green block frame)

(f) Returns to home position (torso frame)

Figure 6: Successful task replay on a novel test configuration for the pick and place task, demonstrating generalization. From left to right: the starting pose and the final point of each executed DMP. Automatically detected coordinate frames used for each segment are listed in parentheses.

moves back upward. The third skill goes back to the home position. The fourth skill reaches toward the green block, moves downward, releases the red block and moves back upward. Finally, the fifth skill goes back to the home position. Notice that the second and fourth segments are identified by the BP-AR-HMM as being the same skill, despite having different relevant coordinate frames. However, in both skills, the arm moves down toward an object, changes the gripper pose, and moves back upward; the reach from the home position toward the green block gets rolled into this skill, rather than getting its own, seemingly because it is a smoother, more integrated motion than the reach and grasp associated with the red block.

Given the commonality of pick and place tasks in robotics, success in this domain may seem trivial. However, it is important to keep in mind that the robot is given only demonstrations in joint space and absolutely no other *a priori* knowledge about the nature of the task. It does not know that it is being shown a pick and place task (or doing grasping at all). It is unaware of the number of subtasks that comprise the task and whether the subtasks will be object-related, gestural, or have other sorts of objectives—the only assumptions are that repeated instances of subtasks share motion statistics and take place relative to a coordinate frame defined by an object or the robot itself. Thereby, beginning with only motion data, the robot is able to automatically segment and generalize

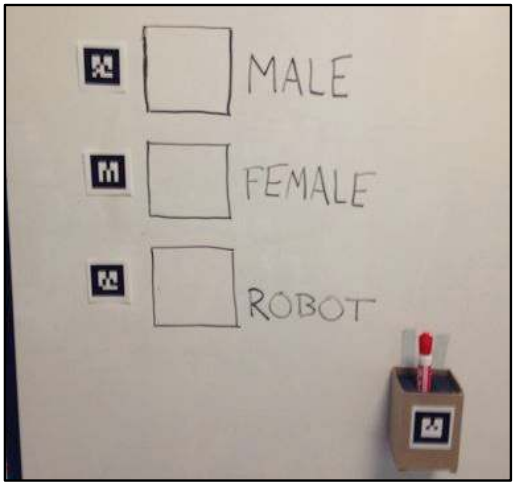


Figure 7: Example configuration of the whiteboard survey task.

a task with multiple parts, each having its own relevant coordinate frame.

3.3.2 Experiment 2: Using a Skill Library (Simulated)

The first experiment demonstrated that our method can learn and generalize a complex task when given a sufficient number of demonstrations. However, this type of learning will not scale up to more complex tasks easily unless the robot can incrementally build a library of skills over time that allows it to quickly recognize previously seen skill / coordinate frame combinations and reuse complex skill controllers that have been improved through practice. To demonstrate our system's capability to recognize skills in this manner, we simulate a previously existing library of skills by providing the robot with a pre-segmented demonstration of the previous experiment. We then give it a single demonstration of the task to see if it can segment it using the "library" of skills.

The BP-AR-HMM recognized each of the skills in the task as being a skill from the pre-existing library. Thus, assuming the robot already had learned about these skills from previous experiences, it would allow a user to provide only a single demonstration of this task and have the robot correctly segment and generalize the task to new configurations. This serves as a proof-of-concept that our proposed framework has the right basic properties to serve as a building block for future models that will scale up LfD to more complex tasks than have previously been possible. It also emphasizes that our method can learn tasks from unstructured demonstrations, as the majority of demonstrations were not even of the task in question, but of a sub-component, unbeknownst to the robot. This is a significant result, as it suggests that a similar technique could be used to incrementally build a growing library of skills over time that could be leveraged in new, but similar, tasks.

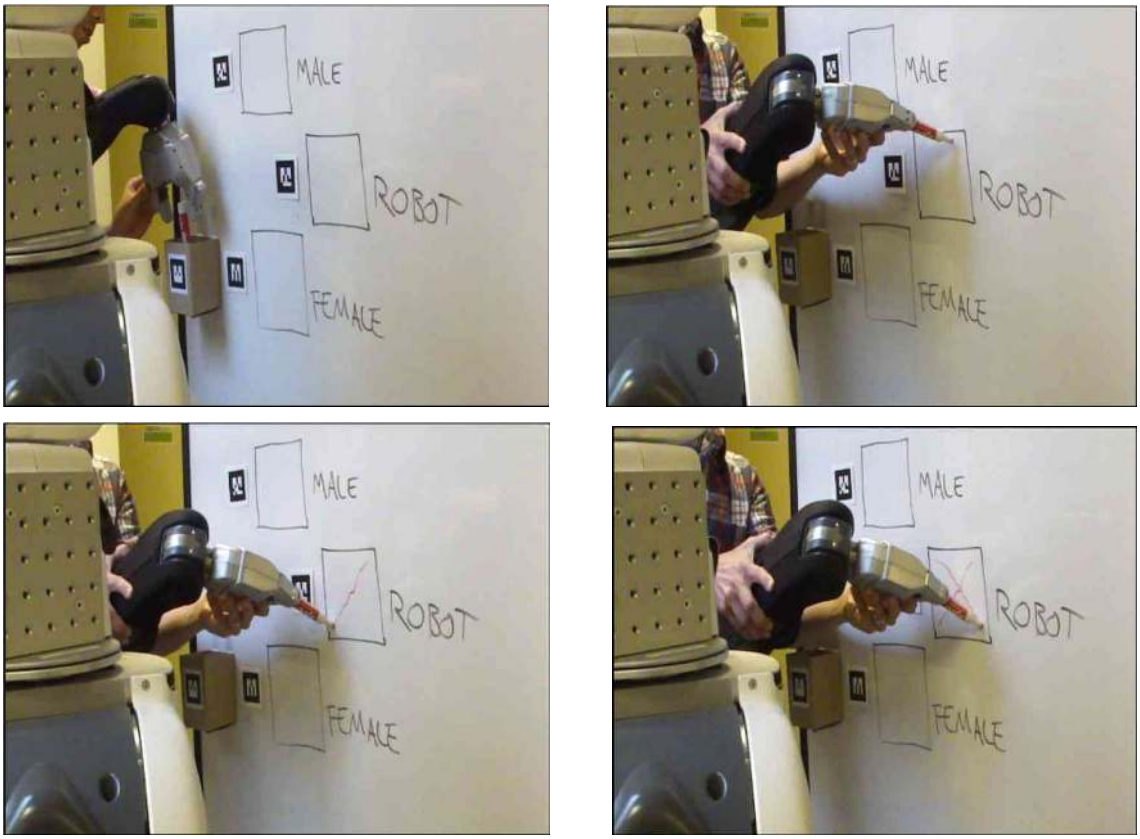


Figure 8: A kinesthetic demonstration of the whiteboard survey task.

3.3.3 Experiment 3: The Whiteboard Survey (Physical PR2)

Finally, we demonstrate that our method is scalable to a real robot system, using a physical PR2. Figure 7 shows one configuration of a task in which the PR2 must fill out a survey on a whiteboard by picking up a red marker and drawing an 'X' in the checkbox corresponding to "robot" while ignoring the checkboxes for "male" and "female". Each checkbox has its own unique fiducial placed one inch to the left of it, while the container that holds the marker has a fiducial directly on its front. The positions of the checkboxes and the marker container on the whiteboard, as well as the position of the whiteboard itself, change between task configurations. Two kinesthetic demonstrations in each of three task configurations (various arrangements of the checkboxes and marker box) were provided, along with one additional demonstration in which the marker is picked up and then lifted above the robot's head. *Kinesthetic* refers to demonstrations in which the robot is physically manipulated by the user to perform the task. An example demonstration is shown in Figure 8.

Figure 9 shows that the BP-AR-HMM generally parses the demonstrations into three main segments, corresponding to reaching for the marker, grasping and lifting the marker, and drawing an 'X' in the checkbox. However, the reaching and drawing segments are considered to be the same skill. This appears to happen because both motions are statistically similar, not in terms of absolute position, but in the way that the positions evolve over time

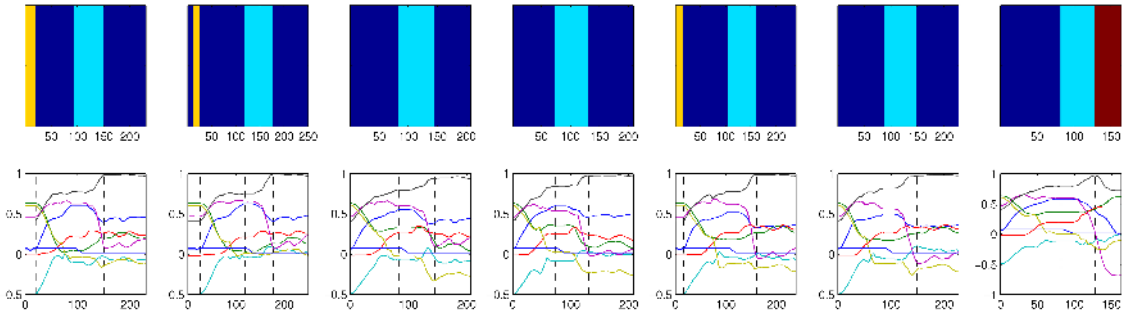


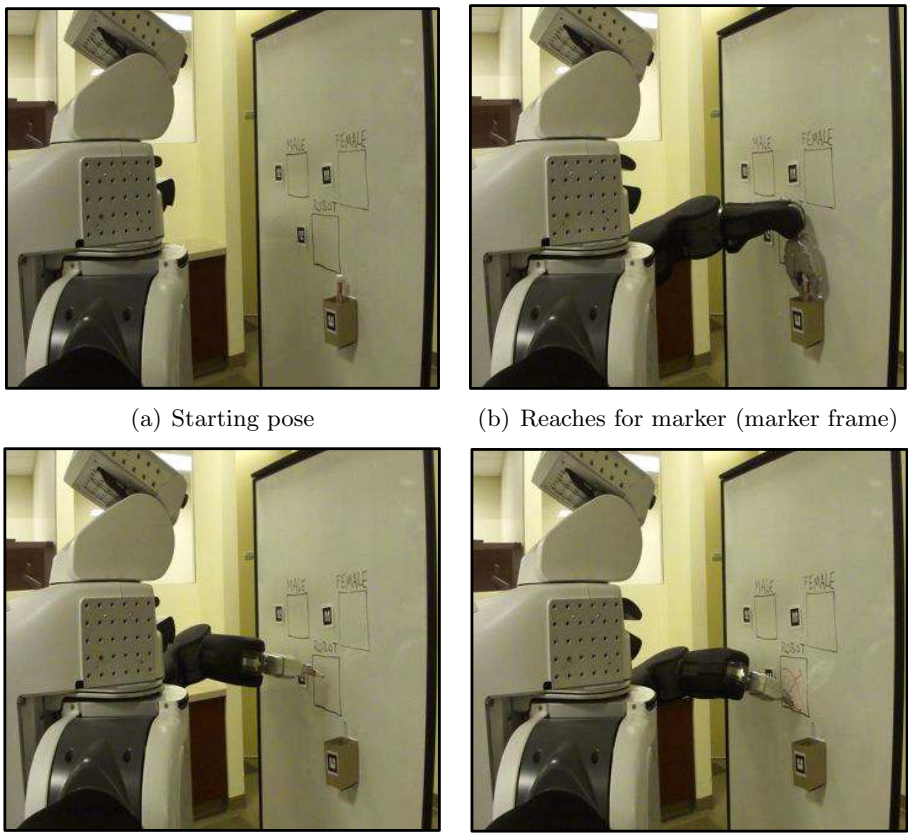
Figure 9: Top: BP-AR-HMM segmentations of the 7 demonstration trajectories for the whiteboard survey task. Time (in tenths of a second) is shown on the x-axis. Skill labels at each time step are indicated by unique colors. Bottom: Segmentation points overlaid on the demonstrated 8D movement data.

as a VAR system. Our coordinate frame detection successfully disambiguates these skills and splits them into two separate skill/coordinate frame combinations. Demonstrations 1, 2, and 5 contain a small additional skill near the beginning that corresponds to a significant twitch in the shoulder joint before any other movement starts, which appears to correspond to the teacher's first contact with the arm, prior to the demonstration. Finally, although the last demonstration is of a different task, the reaching and grasping/lifting skills are still successfully recognized, while the final motion of lifting the marker over the robot's head is given a unique skill of its own. Despite having only a single example of the over-head skill, the BP-AR-HMM robustly identified it as being unique in 50 out of 50 trial segmentations, while also recognizing other skills from the main task. After the learning phase, the robot was able to successfully replay the task in three novel configurations, an example of which is shown in Figure 10.

3.4 Discussion

We presented a novel method for segmenting demonstrations, recognizing repeated skills, and generalizing complex tasks from unstructured demonstrations. Though previous research has addressed many of these issues individually, our method aims to address them all in a single integrated and principled framework. By using the BP-AR-HMM and DMPs, we are able to experimentally learn and generalize a multiple step task on the PR2 mobile manipulator and to demonstrate the potential of our framework to learn a large library of skills over time.

Our framework demonstrates several of the critical components of an LfD system that can incrementally expand a robot's competencies and scale to more complex tasks over time. However, this work is only a first step toward such a system, and leaves a great number of directions open for future research. A more nuanced method must be developed for managing the growing library of skills over time, so that inference in our model does not become prohibitively expensive as the size of the library grows; currently inference in the BP-AR-HMM takes several minutes for only 7 demonstrations on a laptop with a 2.2



(a) Starting pose

(b) Reaches for marker (marker frame)

(c) Grasps marker and lifts toward checkbox (robot checkbox frame)

(d) Draws 'X' (robot checkbox frame)

Figure 10: Successful task replay on a novel test configuration for the whiteboard survey task, demonstrating generalization. From left to right: the starting pose and the final point of each executed DMP. Automatically detected coordinate frames used for each segment are listed in parentheses.

GHz Intel quad-core i7 processor and 8 GB of RAM. One possible way to mitigate this in the future is to "freeze" some skills in the library, such that they can be recognized in new demonstrations, but are no longer candidates for modification during inference.

While our model allows for DMP policy improvement through RL, we did not address such improvement experimentally. Future work may use techniques such as inverse RL [Abbeel and Ng, 2004] to derive an appropriate reward function for each skill so that policy improvement can be effectively applied. A learned reward (cost) function could also be used to in conjunction with a motion planner as a more adaptive alternate to DMP motion generation.

There are also many more opportunities to take advantage of abstractions and invariants in the data; searching for skill coordinate frames is a very simple example of a much richer class of generalization techniques. For example, we constructed DMPs from and ordering of single segments that came from the task configuration most similar to the current one that the robot faces. This is a rather inflexible way to sequencing skills that can lead to failure in

cases where errors might be made during execution or where adaptive behavior is required. In the next section, we examine more intelligent methods for skill sequencing that can be applied to make better use of the demonstration data that we have available. By leveraging additional structure, the robot can learn to classify and sequence skills adaptively based on observations and learn through interactive corrections how to deal with contingencies encountered during execution.

4 Incremental Grounded Skill Discovery

Automatic segmentation and recognition of repeated structure in a task makes learning more tractable, enables data reuse and transfer, and can reveal exploitable invariants in the data, as shown in the previous section. However, in those experiments, task replay consisted of essentially blind replay of a sequence of skills that had been observed previously, which can lead to brittle performance in many tasks. More complex tasks often have multiple valid execution paths, contingencies that may arise during execution, or exhibit partial observability and/or perceptual aliasing. Such tasks demand adaptive skill sequencing strategies that better utilize execution-time feedback from the robot.

Current LfD methods take a variety of approaches to sequencing, ranging from associative skill memories [Pastor et al., 2012], in which all primitives are made available for selection at each decision point, to other schemes in which primitives are executed in a fixed order [Akgun et al., 2012; Niekum et al., 2012]. We argue that a method between these two extremes is most effective, in which there is flexibility in choosing what primitive is scheduled next, but the choices are limited at each decision point to keep the discriminative task tractable as the number of primitives grow.

We present a novel method to sequence automatically discovered primitives that makes minimal assumptions about the structure of the task and can sequence primitives in previously unseen ways to create new, adaptive behaviors. As in the previous section, a Beta Process Autoregressive Hidden Markov Model is used to segment continuous demonstration data into motion categories with associated coordinate frames. Tests are then performed on the motion categories to further subdivide them into grounded movement primitives, in which exemplars of the motions are correlated with observations about actions, objects, and their relationships. These grounded primitives are then used to create a finite-state representation of the task, in which each state has an associated set of exemplars of the relevant movement primitive, plus a trained classifier used to determine state transitions. The resulting finite-state automaton (FSA) can then be used to adaptively replay a complex, multi-step task [Niekum et al., 2013].

However, initial demonstrations do not always cover all the possible contingencies that may arise during the execution of a task. When most LfD methods fail at task replay, the onus is on the user to design new demonstrations that can fill in the gaps in the robot's knowledge. Instead, a data-driven approach is introduced that provides additional data where they are most needed through interactive corrections. These corrections are provided by the user at the time of failure and are treated as additional demonstrations that can be segmented, used to improve the structure of the FSA, and provide additional examples of relevant primitives. Together, this allows for iterative, incremental learning and improvement of a complex task from unsegmented demonstrations. The utility of this

system is shown on a complex furniture assembly task using a PR2 mobile manipulator.

4.1 Constructing Finite-State Task Representations

When performing complex multi-step tasks, it is vital to not just have the right skills for the task, but also to choose and execute skills intelligently based on the current situation. Tasks cannot always be broken down into a simple linear chain of controllers; rather, many complex tasks require the ability to adaptively handle novel and changing situations in addition to unforeseen contingencies that may arise. Some recent work has focused on developing associative skill memories [Pastor et al., 2012] to adaptively match sensory input with predefined skills to perform a task. However, such skills cannot always be known ahead of time, and more structure than pure associativity may be required when there is *perceptual aliasing* [Grollman and Jenkins, 2010b] in the task. By automatically segmenting primitives from data and constructing a finite-state representation of a task, it is possible to combine skill-associativity with a powerful state framework that enables robots to automatically discover appropriate skills, reuse data efficiently, adapt to novel situations, and handle perceptual aliasing.

A finite-state automaton (FSA) is a natural representation for modeling transitions between discrete primitives. By using DMPs at a low-level to represent movement primitives and an FSA for high-level decision making, the advantages of both can be gained, allowing the representation of virtually any continuous motion, while also being able to flexibly make critical choices at a small number of discrete decision points. To define an FSA that represents a task, a notion of states and transitions is required. A state in the FSA ideally corresponds to a step or action in the task that implies that a particular primitive should be executed. Each state stores a list of exemplars of a particular primitive that have been observed from demonstration. Transitions are dictated by a mapping from observations to successor states, which can be implemented as a multi-class classifier.

However, segmentation of demonstrations via the BP-AR-HMM provides us with motion categories based on statistical properties of the movements, not task-relevant actions. For example, a small twist of the wrist required as part of a grasp can have a nearly identical VAR formulation as the continuous twisting required to screw a piece in during an assembly task; these movements have different task functions, but similar statistical properties. So how can task-grounded primitives be created from motion categories?

We make the assumption that exemplars of a grounded primitive will generally fall into the same motion category (i.e. have similar statistical properties), but that all examples of a particular motion category will not necessarily belong to the same primitive. Following this logic, exemplars of motion categories can be split into multiple groups that correspond to grounded primitives by using task-relevant splitting criteria. This can be achieved by performing splits based on the correlation of state visitation history with other types of information, such as the exemplar's coordinate frame, length, or successor state. The state visitation history of an exemplar (the states in the FSA that the previous segments in the demonstration trajectory belong to) is a critical piece of information for splitting, because it provides a notion of sequence—examples of a motion category may be repeated several times in a task, but may correspond to different task primitives that are appropriate during various phases of task progress. In this case, only parent states are examined, i.e. one-step

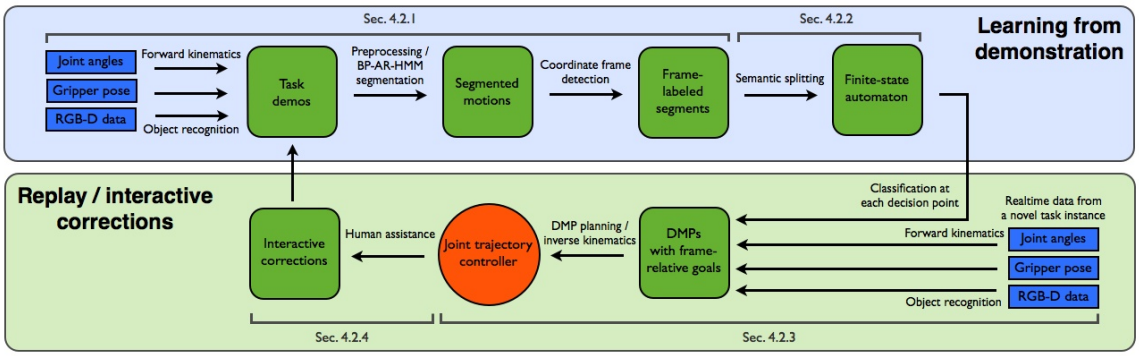


Figure 11: Overview of the iterative learning from demonstration framework.

histories, due to the relative sparsity of data in an LfD setting.

If a difference between exemplars at a single state (such as the coordinate frame of the exemplar) can be predicted by the parentage of the exemplar, then splitting the state has several benefits. First, it helps to determine which exemplars should be considered in different phases of the task. Such a split reduces the number of possible transitions from a single state, removing some of the burden from the transition classifiers and mitigating perceptual aliasing, where perceptual information alone is not enough to correctly determine the next action. This also has the effect of minimizing the number of inappropriate exemplars that can be chosen at a particular state, while maximizing data reuse by only splitting when it is warranted by correlative evidence—splitting at every node with multiple parents would induce a very strong notion of sequence, but would over-segment the data into many states, each of which would have very few associated exemplars and could only be executed in an order that had previously been observed.

It is not expected that this process will always work on the first try, due to factors like unforeseen contingencies and lack of sufficient data. Thus, a way to incrementally improve the FSA structure and task performance is required. For this, a data-driven method of providing interactive corrections is used that allows the user to halt task execution at any time and provide a corrective demonstration of the remainder of the task. This provides additional data where they are most needed—situations in which intervention is required for successful execution—through a natural, intuitive mechanism. Corrections can be used to account for unanticipated situations that were not covered in the original demonstrations, contingencies like missing a grasp or dropping an object, or incorrect movement sequencing. These corrections can then be treated as additional demonstrations and jointly segmented with the rest of the existing data, producing an improved FSA structure with additional exemplars of relevant primitives. This iterative process can be repeated as needed to address shortcomings in performance as errors occur. Figure 11 shows an overview of the whole system, which is described in greater detail in the following section.

4.2 Methodology

We now formally describe our methodology for implementing the system described in the previous section. Specifically, we discuss the technical details of collecting demonstrations,

performing segmentation, constructing an FSA, performing task replay, and collecting interactive corrections.

4.2.1 Demonstrations and segmentation

During the i^{th} demonstration, the pose information $X_i = (x_{i,1}, \ldots, x_{i,\tau_i})$ with $x_{i,t} \in SE(3) \times SE(3) \times \mathbb{R}^2$ is recorded for each time t at 10 Hz, consisting of the Cartesian pose of the end effector plus gripper pose (1-D measure of openness) for both arms. The active arm can be changed by pressing a button on a joystick; the previously active arm becomes stiff, the inactive arm becomes loose and ready for interaction, and the arm switch event is recorded for later use. Additionally, a filtered estimate of the last observed Cartesian pose of all n task objects $O_i = (o_{i,1,1}, \ldots, o_{i,n,\tau_i})$, with $o_{i,j,t} \in SE(3)$ recorded for each object j at each time step t. At the beginning of the task, if not all relevant task objects are visible, the robot will prohibit the demonstration from starting and look around until an initial pose is known for all n objects.

After a set of m demonstrations $(X_1, O_1), \ldots, (X_m, O_m)$ of the task have been collected in various configurations, the robot pose information $X_1, \ldots, X_m$ can be segmented and labeled by the BP-AR-HMM using the same procedure and parameters as in Section 3. However, this is done separately for data that comes from each arm, forcing segment breaks in locations where the active arm is switched and creates a separate library of motion categories for each arm.

The segmentation process provides a set of segment lists $S_1, \ldots, S_n$, such that $S_i = (s_{i,1}, \ldots, s_{i,q_i})$ and $concat(s_{i,1}, \ldots, s_{i,q_i}) = X_i$. Additionally, segmentation returns corresponding label vectors $L_1, \ldots, L_m$, such that $L_i = (l_{i,1}, \ldots, l_{i,q_i})$, where $l_{i,j} \in \mathbb{Z}$ is a label for the j^{th} segment in S_i, $s_{i,j}$. Each integer label corresponds to a unique motion category discovered by the BP-AR-HMM segmentation that is parameterized as a VAR process. Finally, the clustering method described in Section 3.2.3 is used to automatically discover coordinate frame assignment lists $A_1, \ldots, A_n$ with $A_i = (a_{i,1}, \ldots, a_{i,q_i})$ and $a_{i,j} \in \{\text{'object 1'}, \ldots, \text{'object n'}, \text{'torso'}, \text{'none'}\}$.

4.2.2 FSA construction

Defining the set L^* as the union of all the unique labels from the segment label vectors $L_1, \ldots, L_m$, a finite-state automaton that represents the task can begin to be constructed by creating nodes[5] $N_1, \ldots, N_u$, with corresponding labels $L^*_1, \ldots, L^*_u$, where $u = |L^*|$. For $k \in \{1, \ldots, u\}$, each node N_k is assigned a set E_k of all exemplars $s_{i,j}$ that have the same label as N_k, and the label of the previous and next segments is also recorded (or START or END if there is no previous or next segment), which we denote as $prev(s)$ and $next(s)$. A $u \times u$ transition matrix T can then be constructed, where each entry $T_{a,b}$ is set to 1 if there exists a directed transition from N_a to N_b, and 0 otherwise. This matrix is initialized using the sequence data, such that:

$$T_{a,b} = 1 \Leftrightarrow \exists i, j \mid (s_{i,j} = L^*_a) \wedge (s_{i,j+1} = L^*_b).$$

[5]The term 'node' is used rather than 'state' to avoid confusing it with the current observation or state of the robot.

Once this baseline FSA is constructed, nodes can be split by looking for differences amongst groupings of exemplars, based on the label of the segment that preceded the exemplar in the observed execution; in other words, separating exemplars based on groupings of the node's parents in the FSA. For each node N_k with more than one parent, each possible split of the parents into two disjoint sets, P_{in} and P_{out} is examined, with associated exemplar sets E_{in} and E_{out}, and descendant sets D_{in} and D_{out} such that:

$$\begin{aligned}
E_{in} &:= \{e : e \in E_k \mid prev(e) \in P_{in}\} \\
E_{out} &:= \{e : e \in E_k \mid prev(e) \in P_{out}\} \\
D_{in} &:= \{next(e) : e \in E_{in}\} \\
D_{out} &:= \{next(e) : e \in E_{out}\}
\end{aligned}$$

Three criteria are then used to determine if the node should be split into two: (a) if the unique set of coordinate frame assignments corresponding to the exemplars in E_{in} is disjoint from that of E_{out}, (b) if the unique set of next segment labels (the $next(e)$) corresponding to the exemplars in E_{in} is disjoint from that of E_{out}, and (c) if the segment lengths of the exemplars in E_{in} come from a significantly different distribution than those in E_{out}. The difference in the length distributions is tested using the Kolmogorov-Smirnov test [Massey, 1951] as follows. Let F_{in} and F_{out} be the empirical cumulative distribution functions of the lengths of the exemplars in E_{in} and E_{out}. The test statistic z is then calculated based on the maximum distance between the empirical CDFs, without making any assumptions about the distributions involved:

$$\begin{aligned}
G &= \sup_x |F_{in}(x) - F_{out}(x)| \\
z &= G\sqrt{\frac{|E_{in}|\,|E_{out}|}{|E_{in}| + |E_{out}|}}.
\end{aligned}$$

This suggests the node should be split if $z < K_\alpha$, the critical value for significance value α. Here, $\alpha = 0.05$ is used.

If any of the three tests indicate to split the node, the node N_k is deleted and two new nodes N_{k-A} and N_{k-B} are created with exemplars E_{k-A} and E_{k-B}, such that:

$$\begin{aligned}
T_{p,N_{k-A}} &= 1 \Leftrightarrow p \in P_{in} \\
T_{p,N_{k-A}} &= 1 \Leftrightarrow p \in P_{out} \\
T_{N_{k-B},d} &= 1 \Leftrightarrow d \in D_{in} \\
T_{N_{k-B},d} &= 1 \Leftrightarrow d \in D_{out} \\
E_{k-A} &:= E_{in} \\
E_{k-B} &:= E_{out}
\end{aligned}$$

This process, which we call *node splitting*, is then repeated, iterating over all the nodes until no more splits can be made.

Once the structure of the FSA is finalized, a classifier is trained for each node that has multiple descendants. This is used as a transition classifier to determine which node to transition to next, once execution of a primitive at that node has taken place. For this

task, a nearest neighbor classifier is used, but could be replaced by any multi-class classifier. For each classifier C_k corresponding to node N_k, training examples $Y_k = (y_{k,1}, \ldots, y_{k,r_k})$ are provided, corresponding to the final observation of the robot pose and object poses during each exemplar, plus a class label corresponding to the label of the node the given exemplar transitioned to next. Thus, training examples are recorded as $y \in SE(3)_1 \times \cdots \times SE(3)_{n+2} \times \mathbb{R}^2$, where n is the number of task objects. One exception to the above rule is the START node, which has no data associated with it; for this, the first observation of the first segment of the demonstration is used as training data.

Each classifier C_k is then trained using the training examples Y_k, along with their appropriate class labels. Once the classifier has been trained, future observations y can be classified so that appropriate transitions can be made at each decision point.

4.2.3 Task replay

Given a novel situation, the FSA can be used to replay the task by beginning at the START node that has no exemplars associated with it. The current observation is classified using C_{START} to determine which node to transition to first. This is a standard node N_i that contains exemplars E_i. To execute an appropriate movement, a DMP must be constructed from the exemplars; a single exemplar is chosen by examining the first observations associated with each exemplar and choosing the nearest candidate to the current observation y. The chosen exemplar is used to learn the weights for a DMP as described in Section 2.3.

To execute the DMP, the goal is shifted based on the coordinate frame that the segment was assigned to, so that the relative 6-DOF offset of the goal from the current origin of the coordinate frame is the same as it was in the exemplar. Then, the current robot pose is given as a starting state and the DMP is used to generate a movement plan to the goal with a resolution of 0.1 seconds. The plan is then executed, followed by a transition dictated by the node's transition classifier, using the current observation as input. This process is repeated until the END node is reached, a timeout occurs, or the robot is manually stopped by the user.

4.2.4 Interactive corrections

At any time during execution, the user can push a button on the joystick to stop the robot so that an interactive correction can be made. The robot immediately stops execution of the current movement and switches modes to accept a kinesthetic demonstration from the user. From the beginning of execution, the robot has been recording pose data in case of an interactive correction, and it continues to record as the user provides a demonstration of the remainder of the task.

After any number of replays and interactive corrections have taken place, the corrections can be integrated with the existing data for improved performance. All the old data plus the recordings from the interactive corrections are segmented from scratch; we do not begin with the old burned-in BP-AR-HMM, because the addition of new data may require a significantly different segmentation, and we wish to avoid systematic biasing of the sampling process. Following segmentation, the FSA is reconstructed as described above with the new data included, providing additional exemplars and refining the structure of the FSA. Algorithm 2 provides pseudocode for this iterative process.

Algorithm 2 Incremental Semantically Grounded Learning from Demonstration

Input: Demonstration trajectories $\mathbf{T} = (T_1, \ldots, T_N)$ and corresponding object pose observations $\mathbf{O} = (O_1, \ldots, O_N)$

Loop as desired:

1. *Segment the demonstrations with the BP-AR-HMM:*
   (segments, labels) = $Segmentation(\mathbf{T})$
2. *Find coordinate frame assignments:*
   coord_frames = $CoordinateFrameDetection$(segments, labels, $\mathbf{O}$)
3. *Learn params of a DMP for each segment:*
   dmps = $LearnDmpParams$(segments, coord_frames)
4. *Construct FSM:*
   fsm = $BuildFSM$(segments)
   sfsm = $SemanticSplit$(fsm)
   classifiers = $TrainClassifiers$(sfsm, segments)
5. *Replay task based on current observations:*
   state = START
   while state != END **and** $IsInterrupted()$ == False **do**
     curr_obs = $GetCurrentObservation()$
     action = $Classify$(classifiers, state, curr_obs)
     state = $GetNextState$(sfsm, state, action)
     $ExecuteDmp$(action)
   end while
6. *Collect interactive correction if necessary:*
   if $IsInterrupted()$ = True **then**
     $(T_{corr}, O_{corr}) = GetCorrectionTrajectory()$
     $\mathbf{T} = (T_1, \ldots, T_N, T_{corr})$
     $\mathbf{O} = (O_1, \ldots, O_N, O_{corr})$
   end if

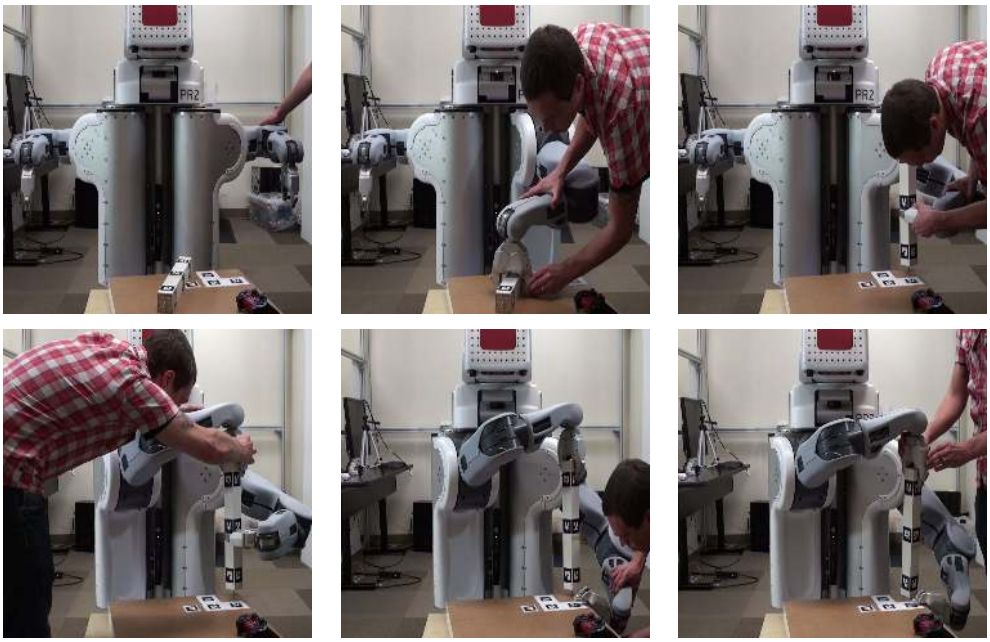


Figure 12: A kinesthetic demonstration of the table assembly task.

4.3 Experiments

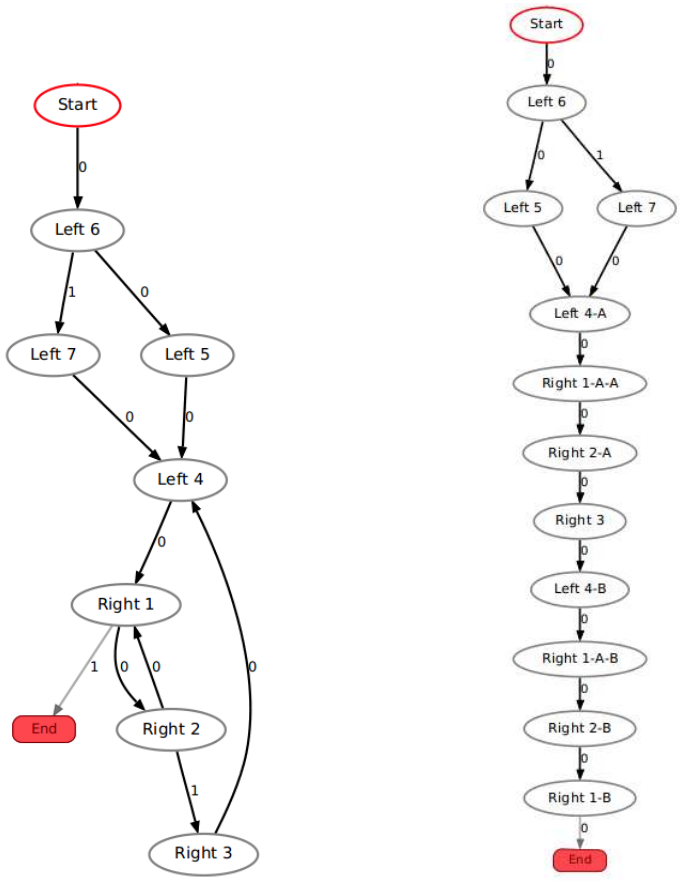
For all experiments, a PR2 mobile manipulator is used as the robotic platform. Task examples are provided to the robot via kinesthetic demonstrations, in which the teacher physically moves the arms of the robot to perform the task and uses a joystick to set the degree of closure of the grippers. Visual fiducials are used to track relevant pre-determined *task objects* using combined visual and depth data from a head-mounted RGB-D camera on the PR2.

4.3.1 Experiment 1: Demonstrations, corrections, and replay

We evaluated our system on a furniture assembly task, using the PR2 to partially assemble a small off-the-shelf table.[6] The table consists of a tabletop with four pre-drilled holes and four legs that each have a screw protruding from one end. Eight kinesthetic demonstrations of the assembly task were provided, in which the tabletop and one leg were placed in front of the robot in various positions. In each demonstration, the robot was made to pick up the leg, insert the screw-end into the hole in the tabletop, switch arms to grasp the top of the leg, hold the tabletop in place, and screw in the leg until it is tight. An example of this progression is shown in Figure 12. To make the insertion more robust, a bit of human insight is applied by sliding the screw around the area of the hole until is slides in. This allows the insertion to be successful even when perception is slightly inaccurate.

The demonstrations were then segmented and an FSA was built as described in Section 4.2. Figures 13(a)–(b) show the structure of the FSA before and after node splitting occurred; it can be seen that the node splits yield a much simpler, linear structure that better

[6]Ikea LACK table: `http://www.ikea.com/us/en/catalog/products/20011413/`



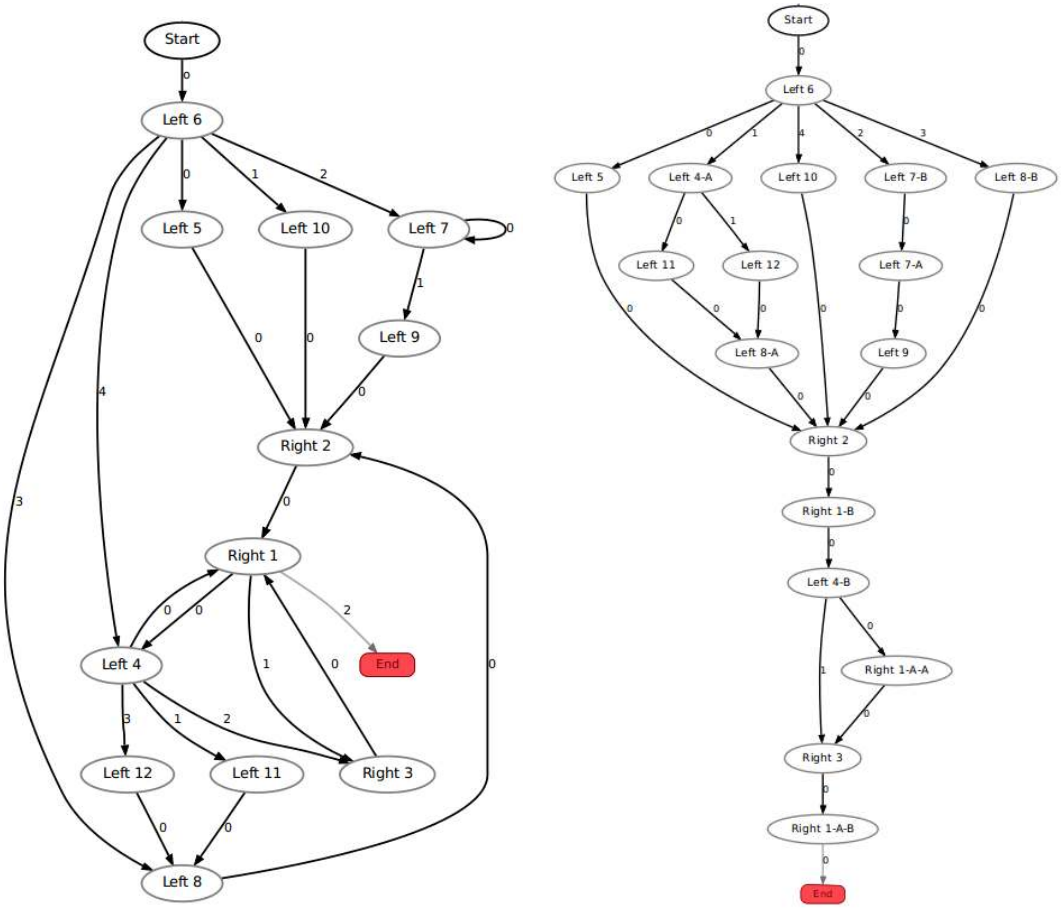
(a) Demonstrations only, before node splitting

(b) Demonstrations only, after node splitting

Figure 13: FSA structure from demos before and after node splitting without interactive corrections. Nodes are labeled with the relevant arm (left/right), ID numbers, and A's and B's to denote splits.

characterizes the flow of task progress, while leaving room for branching when necessary. At this stage, task replay was sometimes successful, but several types of errors occurred intermittently. Two particular types of errors that occurred were (a) when the table leg was at certain angles, the robot was prone to missing the grasp, and (b) when the leg was too far from the robot, it could not reach far enough to grasp the leg at the desired point near the center of mass. In both cases interactive corrections were provided to recover from these contingencies. In the first case, a re-grasp was demonstrated, and then the task was continued as usual. In the second case, the robot was shown how to grasp the leg at a closer point, pull it towards itself, and then re-grasp it at the desired location.

After the interactive corrections were collected, the old data was re-segmented with the two new corrections and used to re-build the FSA. Figures 14(a)–(b) show the structure of the FSA before and after node splitting occurred. After splitting, this FSA is similar to that of Figure 13(b), but with several branching pathways near the beginning that then



(a) Demonstrations + interactive corrections, before node splitting

(b) Demonstrations + interactive corrections, after node splitting

Figure 14: FSA structures before and after node splitting with interactive corrections added in. Nodes are labeled with the relevant arm (left/right), ID numbers, and A's and B's to denote splits.

bottleneck into a linear flow. This is exactly the sort of structure that would be expected, given the nature of the corrections—the robot attempts a grasp, chooses a path based on the outcome of the grasp, and then once the leg is successfully picked up and ready to be inserted, all paths converge on a common (nearly) linear sequence for the rest of the task.

Using this new FSA, the robot was able to recover from two types of errors in novel situations. Figure 15 shows a successful recovery from a missed grasp, while Figure 16 shows the robot bringing the table leg closer for a re-grasp when it is too far away. Finally, Figure 17 shows a full successful execution of the task without human intervention, demonstrating that these error recovery capabilities did not interfere with smooth execution in cases where no contingencies were encountered.

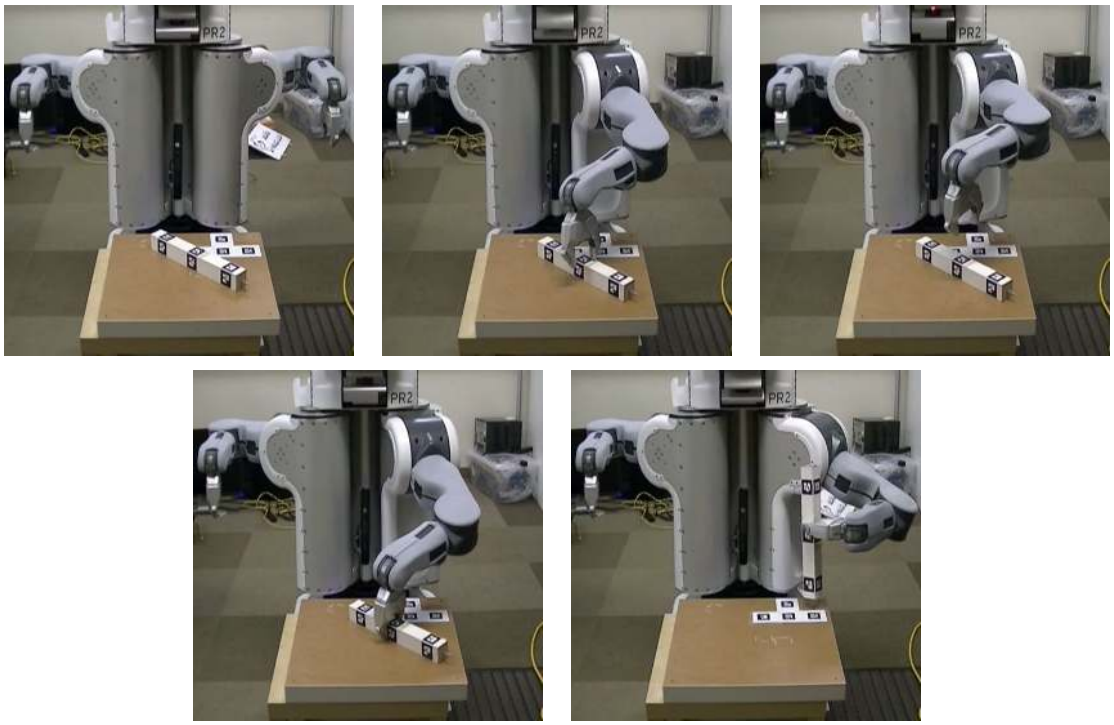


Figure 15: A recovery behavior when the robot misses the original grasp.

4.3.2 Experiment 2: Method comparisons

The table in Figure 1 shows a quantitative comparison that demonstrates the benefits of FSA-based sequencing and interactive corrections. Ten trials of the table assembly task were attempted using four different sequencing methods with the segmented demonstration data from Experiment 1. The first method ('ASM') used a framework similar to that of Associative Skill Memories by Pastor et al. [2012], in which all primitives were available to be chosen at every decision point; classification was performed via a nearest neighbor classifier over the first observations of the exemplars associated with each movement category. The second ('FSA-basic') and third ('FSA-split') methods used an FSA before and after node splitting, respectively. Finally, the fourth method('FSA-IC') used an FSA after splitting with interactive corrections (also from Experiment 1) integrated as well.

Each set of ten trials was split up into three groups: four trials in which the table leg was straight and close to the PR2 ('Straight'), three trials in which the leg was too far away to grasp at the center of balance ('Far away'), and three trials in which the leg was at a difficult angle that could cause a missed grasp ('Difficult angle'). These were all novel configurations that had not been seen before, but the latter two were designed to produce situations similar to the interactive corrections collected earlier. During each trial, the operator was allowed to provide small assists to help the robot by moving an object or the robot's end effector by a maximum of 5 cm to compensate for minor perceptual or generalization errors. The entries in the table in Table 1 denote the number of assists that were required during each trial, or 'Fail' if the robot failed to complete the task successfully. Here, we defined success

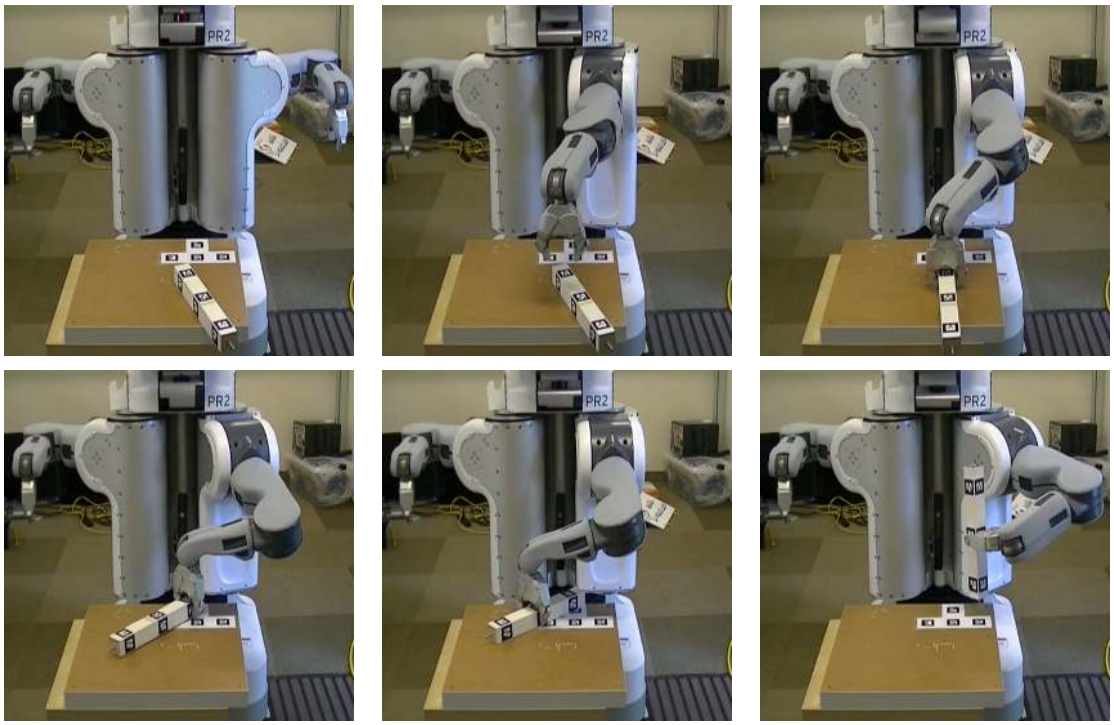


Figure 16: A recovery behavior when the table leg is too far away to grasp at the desired location.

as screwing the leg in far enough so that it was freestanding when the robot released it.

All ten trials with the 'ASM' and 'FSA-basic' methods resulted in failure, but for different reasons. While the ASM provided maximum sequencing flexibility, it also inherently made the classification problem difficult, since all choices were available at every step. Indeed, most failures were caused by misclassifications that caused the robot to choose inappropriate primitives or get stuck in an infinite loop, repeating the same primitive indefinitely. The FSA avoided infinite loops by reducing the number of choices at each decision point and providing ordering constraints between the nodes. However, it still frequently chose poor exemplars or inappropriate primitives. Without node splitting, several different primitives often got combined into a single node, corrupting the structure of the FSA, and making classification and exemplar selection more difficult.

Using node splitting, we observed seven successes and a modest number of required assists with the 'FSA-split' method, failing only in the 'Far away' case. In these cases, the robot reached for the leg until its arm was fully extended, but stopped far short of the intended grasp point; the difference was far too large to be fixed with a small assist. Once interactive corrections were added in 'FSA-IC', the number of successes increased to nine and the number of required assists dropped by more than 25%—a significant improvement. The remaining assists were largely due to small perceptual errors during the screw insertion, which requires a high level of precision. These errors appeared to be random, lacking structure that could be leveraged; additional interactive corrections were provided, but did not further improve performance. This reveals an important limitation of interactive

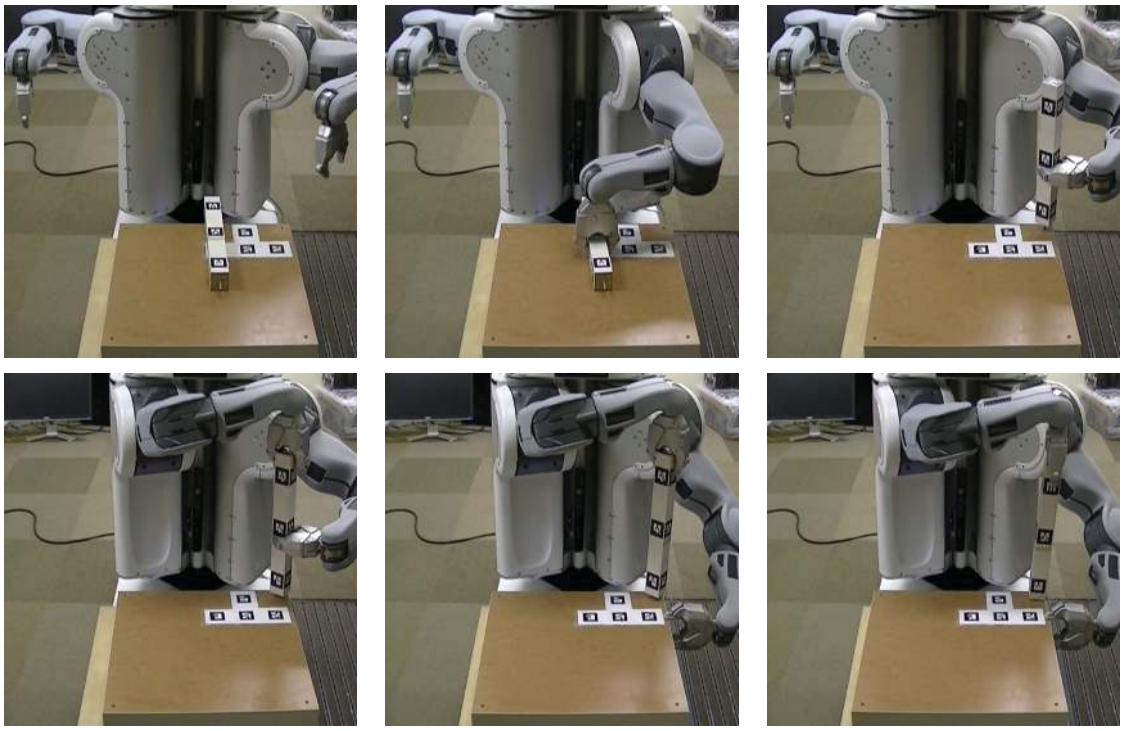


Figure 17: A full successful execution of the table assembly task without any human intervention. The task FSM is constructed using both the original demonstrations and the interactive corrections.

	ASM	FSA-basic	FSA-split	FSA-IC
Straight	Fail	Fail	1	0
	Fail	Fail	1	2
	Fail	Fail	2	2
	Fail	Fail	1	2
Far away	Fail	Fail	Fail	1
	Fail	Fail	Fail	1
	Fail	Fail	Fail	Fail
Difficult angle	Fail	Fail	2	1
	Fail	Fail	3	1
	Fail	Fail	3	2
Successes / Avg assists	0 / –	0 / –	7 / 1.857	9 / 1.333

Table 1: Ten trials of the task with corresponding performance data for four different types of sequencing: an associative skill memory (ASM), an FSA without node splitting (FSA-basic), an FSA with node splitting (FSA-split), and an FSA with splitting and interactive corrections (FSA-IC). 'Fail' indicates failure and integer values correspond to the number of human assists required during successful executions.

corrections—in general, they can only be used to recover from observable, structured errors.

4.3.3 Experiment 3: Looping behaviors

For this final experiment we designed a task that contains a behavioral loop, with the goal of capturing this loopy structure in the FSA. Figure 18 shows a demonstration of a peg-in-hole task in which the robot must pick up a rectangular peg and insert it in the hole in the center of the table. In three of the demonstrations of this task, the robot was shown how to complete the task normally by picking up the peg and inserting it in the hole immediately. However, in eight other demonstrations, trajectories were provided that slightly miss the hole several times before successfully getting the peg in. This is a looping "retry" behavior, in which the peg is lifted in the air and attempted to be inserted again until successful.

The goal of this experiment was for the robot to learn a loopy FSA, such that it would automatically retry the peg insertion step until it was classified as successful. For such an FSA to be learned, the segmentation of the demonstrations would have to identify each insertion attempt as a distinct segment. Unfortunately, this was not the case; Figure 19 shows a segmentation of the demonstrations in which all insertion attempts are clumped together into one long segment at the end of each demo (the green segment). This final segment is longer or shorter in each demo depending on the number of repetitions, but is never split into more than one segment.

This failure reveals an interesting feature and limitation of our algorithm. Our method makes no distinction between seeing the same motion category several times in succession and seeing that motion category once for a longer period of time—each segment is just a contiguous string of observations where the same motion category is active, so there

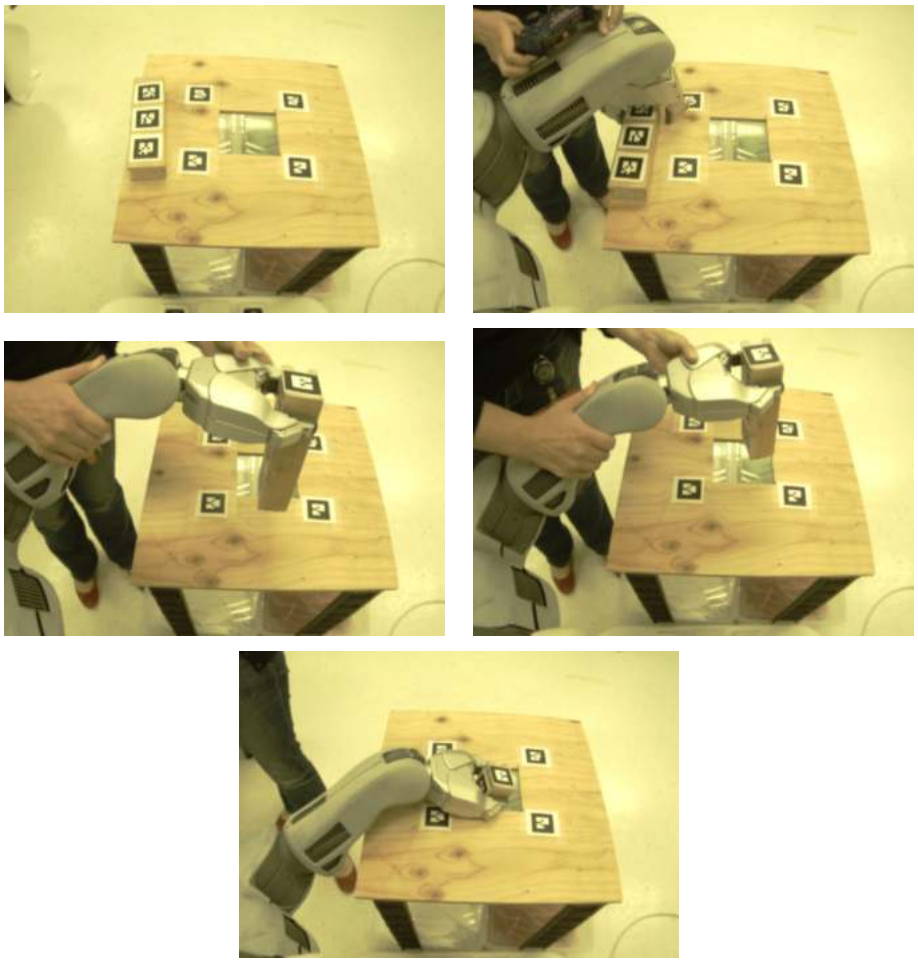


Figure 18: A demonstration of the peg-in-hole task. Panels 3 and 4 show failed insertions attempts and and retries, while panel 5 shows the final successful insertion.

cannot be multiple adjacent segments of identical motion categories. Thus, since the system identified the lifting and insertion of the peg all as a single motion category, it was not possible to segment out each repetition of that motion. Interestingly, if lifting the peg and inserting the peg were identified as separate motion categories, the repeating pattern would have been able to be discovered as a two-segment cycle.

Thus, we conclude that in principle, the proposed system can find looping structure in tasks, but in practice it may often fail, due to the granularity at which segmentation occurs. At a deeper level, this may be a fundamental problem with a segmentation system that infers statistical motion categories rather than subgoals that the demonstrations imply. Future work may examine ways to overcome this difficulty.

4.4 Related Work

The most closely related work to ours is that of Grollman et al. [2010b], which addresses *perceptual aliasing* by using a nonparametric Bayesian model to infer a mixture of experts

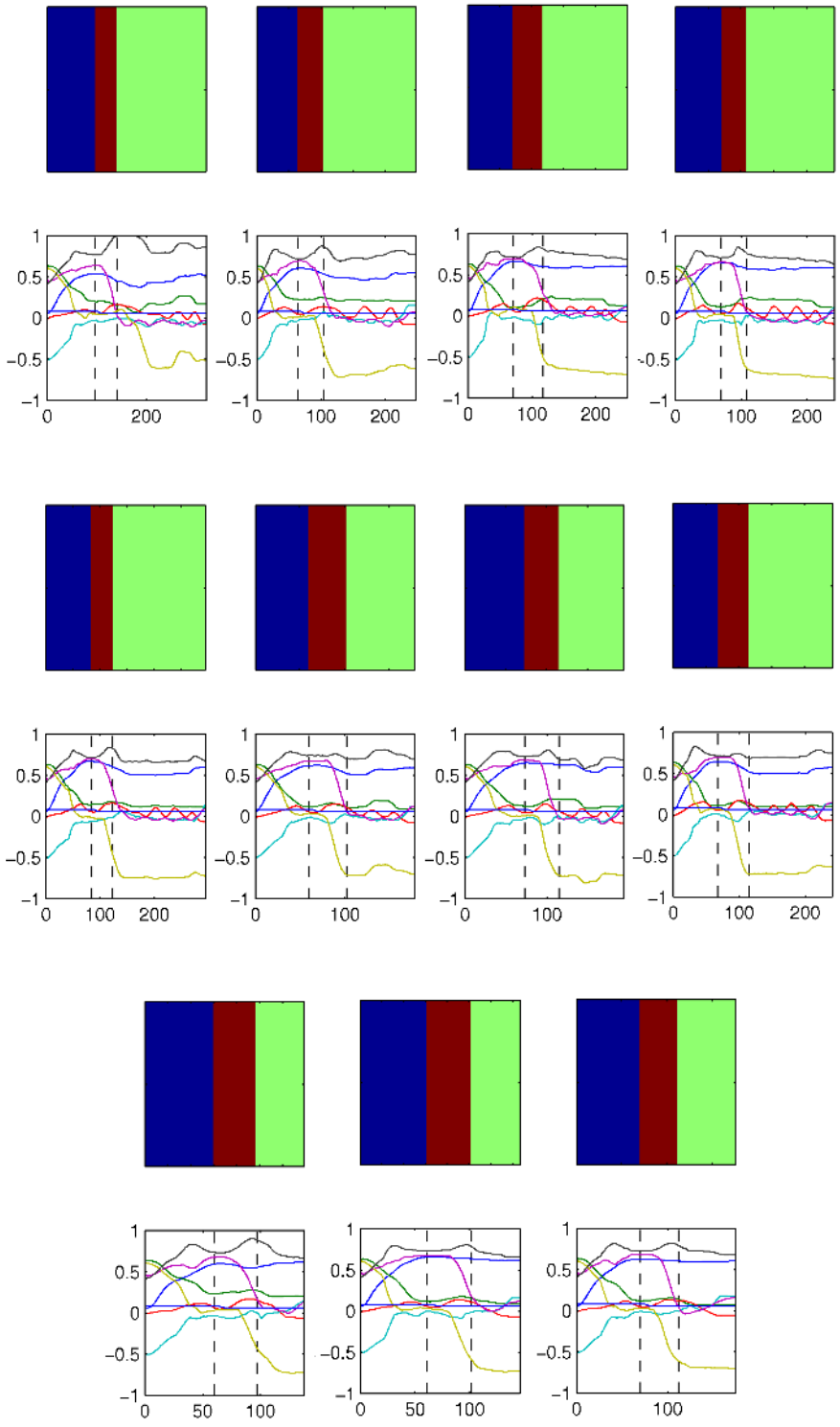


Figure 19: Segmentation of 8 demonstrations of the peg-in-hole task with repeated retries of insertions (top 2 rows) and 3 demonstrations of successful first-try insertions (bottom row).

from unsegmented demonstration data and then using multi-map regression to assign observed states to experts. Butterfield et al. [2010] extend this work by getting rid of the assumption that data is independent and identically distributed, leading to policies that better model the time-series nature of tasks. Although we use a Bayesian mechanism to parse demonstration trajectories, rather than inferring policies, we discover repeated dynamical systems which are considerably simpler to model than more generalized policies. Additionally, by leveraging coordinate frames and node splitting, we are able to identify critical task goals and ground our primitives with respect to the task at hand.

Other work aims to intelligently sequence predefined primitives, rather than discover them from data. Toussaint et al. [2010] use heuristics to translate perceptions into predefined symbols, which are then used by a rule-based planner. In the control basis framework [Huber and Grupen, 1997], a graph of admissible behaviors is automatically built based on the predicates and constraints of multiple hand-crafted controllers, allowing safe composite policies to be learned. Given a set of predefined skills, Riano et al. [2012] evolve the structure of a finite-state automaton that defines transition behavior between the skills. Pastor et al. [2012] demonstrate one skill at a time and then use nearest neighbor classification to associate skills with observations, while also monitoring skill executions for proprioceptive anomalies so that recovery behaviors can be initiated. Wilcox et al. [2012] use a dynamic scheduling algorithm to adapt execution of a predefined set of events to a user's preferences and temporal disruptions, while maintaining critical synchronization invariants. Ekvall and Kragic [2006] search for skill ordering constraints in tasks with many valid execution orderings.

Several other approaches also allow users to interactively provide additional data and corrections to a robot. Thomaz and Breazeal [2006] provide an interface that allows users to bias the robot's exploration and provide positive and negative feedback during task execution. Another approach uses user-defined keyframes that identify critical points in a task that must be reproduced [Akgun et al., 2012]. Upon replay, the teacher can use an interface to view and edit keyframes to fine-tune task performance. Muhlig et al. [2011] rely entirely on real-time human-robot interaction to ensure proper sequencing of skills and recovery from errors, using hand signals and speech to control skill flow, halt execution, and identify important task objects.

4.5 Discussion

Flexible discovery and sequencing of primitives is essential for tractable learning of complex robotic tasks from demonstration. We introduced a novel method to split automatically discovered motion categories into task-grounded primitives, sequence them intelligently, and provide interactive corrections for incremental performance improvement. Sequencing primitives with a finite-state automaton allows exemplars of movement primitives to be grouped together in a task-relevant way that attempts to maximize data reuse, while minimizing the number of options that the agent must choose amongst at each step. This approach makes the sequencing classification task easier, while also providing a mechanism for grounding each primitive based on state visitation history and observed characteristics like coordinate frame, length, and successor state.

This approach was validated on a furniture assembly task using a PR2 mobile manip-

ulator. It was shown that the robot could learn the basic structure of the task from a small number of demonstrations, which were supplemented with interactive corrections as the robot encountered contingencies that would have led to failure. The corrections were then used to refine the structure of the FSA, leading to new recovery behaviors when these contingencies were encountered again, without disrupting performance in the nominal case. A quantitative measure was also provided to show that using an FSA with node splitting provides advantages that lead to improved classification and task performance compared to more naive methods.

While performance was able to be improved through interactive corrections, future work could include a mechanism to improve task performance and individual primitives automatically though self-practice. Additionally, we only took advantage of the multiple exemplars of each primitive by selecting amongst them; in the future, it would be beneficial to integrate the exemplars to better model the user's intentions. Only vision and pose data were used as part of the discriminative state space, but several other types of input such as force data could be valuable for decision making, or even for modulating our DMPs, as in [Pastor et al., 2011b]. Finally, the useful looped structure of repetitive behavior proved difficult to capture, so future work may look at ways to capture contiguous repetitions of the same movement. With improvements such as these, the presented method might function as one element of a deployable system that allows researchers and end users alike to efficiently program robots via unsegmented, natural demonstrations.

5 Summary and Conclusion

We have demonstrated two primary techniques that extend the generalization capabilities of learning from demonstration algorithms in complex robotic tasks. First, we introduced an algorithm that discovers repeated structure in demonstration data through nonparametric Bayesian segmentation and coordinate frame detection to create flexible skill controllers. Then, building on these learned skills, we incrementally learn and improve a finite-state representation of a task that allows for intelligent, adaptive sequencing of skills and recovery from errors. Skills in the FSA were correlated with external observations, imbuing the skills with task-relevant meaning and greater reusability, while improvement was facilitated through interactive corrections.

Several experiments were performed to validate these algorithms, both in simulation and on a PR2 mobile manipulator. We showed how our methods can be used to teach robots complex, multi-step tasks from a relatively small number of demonstrations. In the final set of experiments, the PR2 was able to learn to partially assemble a small IKEA table from only 8 demonstrations. Performance was then improved by providing interactive corrections to the robot as mistakes were made during execution. Finally, it was shown that this combination of techniques allowed the robot to incrementally learn to improve at the task, recover from errors, and adapt to novel situations and unforeseen contingencies.

5.1 Future Work

The novel techniques we have presented reveal new possibilities for more generally applicable robot learning from demonstration algorithms. However, both the algorithms and

experiments are necessarily limited in scope, leaving many opportunities for improvement in future work.

A key piece of our system, the Beta Process Autoregressive Hidden Markov Model, may be able to be improved in several ways that could lead to improved segmentation and performance. We used the BP-AR-HMM to find repeated dynamical systems in the data and then performed coordinate frame detection as a separate step afterwards. Ideally, this would be done all as a single step, in a principled, fully Bayesian way during inference on the model. Aside from being more principled, performing coordinate frame detection within the model might allow us to find repeated dynamical systems that would have previously eluded the BP-AR-HMM; some motions may look very different in the robot frame, but may look very similar in some other object's frame of reference. Additionally, the development of more efficient inference mechanisms would allow our method to work on larger data sets. Finally, it may be worthwhile to explore other types of autoregressive dynamics than linear dynamical systems.

Our system only considered spatial data for the purposes of evaluating the state of the world and creating DMP plans. However, other sensing modalities such as force/torque could be used to enhance the capabilities of the robot. Force sensing, for example, could allow the robot to feel whether a screw was in place, rather than having to guess based on noisy and/or occluded spatial observations. Some work has already been done by Pastor et al. [2011b] on adapting DMP plans on-line based on force data so that executions "feel" the same to the robot as they did during demonstrations, resulting in improved performance and robustness to small errors.

However, more complex concepts like "screw is in hole" may require an additional level of semantic understanding. Our approach gave task-related meaning to states in a finite-state representation of a task by correlating each state with observations of the world or associated actions executed when in that state. However, more abstract, complex concepts may require *functionality understanding*—for example, realizing that both a fork and a toothpick can be used to pick up a piece of cheese, since they both have pointed ends to skewer it with.

Task replay can also be improved through further innovations. Rather than integrate all examples of a particular skill that we have identified, our system uses a simple nearest-neighbor strategy to choose the single most relevant example to use for DMP training. However, all the available data could be used more efficiently if a principled way to integrate this data was designed. However, it is well known that the average of solutions to a problem is often not itself a solution, making this problem non-trivial, though some approaches have been explored [Bitzer and Vijayakumar, 2009; Coates et al., 2008]. Furthermore, by using DMPs, we are performing somewhat "blind" replay of skills. If preconditions and postconditions for each skill could be discovered, more traditional planning algorithms could be used to perform each skill more adaptively. Finally, for more dynamic tasks than we considered, each skill could also be improved through reinforcement learning if necessary, perhaps by inferring a reward function through inverse reinforcement learning.

The experiments in this work were limited to a single task at a time with relatively few demonstrations. However, in the future, algorithms such as these must scale to demonstrations of many tasks, producing a library of skills over time that can be reused in many situations. This requires more efficient inference algorithms, as well as strategies to manage

such a library of skills and to choose appropriate skills in various situations. Furthermore, collecting all this data of many different tasks may take more time than any one user is willing to put in. Further research into web robotics and remote laboratories [Osentoski et al., 2012] may provide mechanisms to crowdsource this type of data collection.

Finally, natural language integration may be useful for providing a simpler interface for the user to command tasks and give feedback to the robot. This could allow the user to teach the robot the names of objects, actions, and tasks during demonstrations. The robot may be able to infer associations between language and actions, allowing it to perform novel tasks from natural language commands, as long as it has seen similar components of the task before. In fact, some existing related work has already demonstrated a principled system for inferring user intentions from natural language [Tellex et al., 2011]. Natural language commands could also be used for feedback on a task, such as "good" and "bad", or to modify the robot's behavior, like "slower" or "harder".

5.2 Conclusion

A perennial goal of robotics has been the creation of robots that can exhibit a wide range of intelligent behaviors in diverse environments. While advances in machine learning techniques have improved the quality of such learned behaviors, both researchers and end-users alike need tools that can help them deal with the vast number of behaviors and environments that must be mastered by a robot deployed in the real world. Learning from demonstration has shown to be a promising paradigm to quickly program robots in a natural way, but has often been limited by poor generalization.

We have shown that learning from demonstration algorithms can partially overcome their traditionally limited generalization capabilities by acquiring grounded skills through the discovery of repeated statistical structure in demonstrations. The discovery of repeated structure provided critical insights into task invariants, features of importance, high-level task structure, and appropriate skills for the task. This allowed the robot to reproduce complex, multi-step tasks in novel situations by learning from a small number of unstructured, whole-task demonstrations and interactive corrections.

To achieve this goal, we drew from recent advances in Bayesian nonparametric statistics and control theory to develop novel learning from demonstration algorithms and an integrated system capable of learning tasks from start to finish; this system combined perception, kinesthetic input, control, segmentation, clustering, and classification algorithms to enable experimentation on a PR2 mobile manipulator. A series of experiments were conducted, culminating in a complex table assembly task that the PR2 was able to reproduce in a number of novel situations, even when contingencies and error conditions were encountered.

While this work advanced the state of the art in learning from demonstration algorithms, we also believe that the integrated system itself is an equally important contribution. As robotics attempts to move forward into the home and workplace, it is our belief that many of the most difficult technical problems will be integrative in nature, rather than limited to a particular subfield such as perception or control. The scope of our work was necessarily limited and leaves much room for future improvement, but we have provided the basis for a principled way identify and leverage structure in demonstration data. Going forward,

we hope this can serve as a step toward a deployable system that allows researchers and end users alike to efficiently program robots, paving the way for robots in the home and workplace.

6 Acknowledgements

Scott Niekum, Sarah Osentoski, and Andrew G. Barto were funded in part by the NSF under grant IIS-1208497.

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty First International Conference on Machine Learning*, pages 1–8, 2004. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015430.

B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398, 2012.

B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 12–20, 1997.

B. Bauml, T. Wimbock, and G. Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2592–2599, 2010.

A. Billard, S. Calinon, R. Dillmann, and S. Schaal. *Handbook of Robotics*, chapter Robot programming by demonstration. Springer, Secaucus, NJ, USA, 2008.

S. Bitzer and S. Vijayakumar. Latent spaces for dynamic movement primitives. In *9th IEEE-RAS International Conference on Humanoids*, pages 574–581, 2009.

M. Bollini, J. Barry, and D. Rus. Bakebot: Baking cookies with the pr2. In *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011.

A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, pages 20–27, 2011.

J. Butterfield, S. Osentoski, G. Jay, and O. Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *Proceedings of the Tenth IEEE-RAS International Conference on Humanoid Robots*, 2010.

S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the Second Conference on Human-Robot Interaction*, 2007.

T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. In *IEEE/RSJ International Conference onIntelligent Robots and Systems*, pages 267–274, 2010.

S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 2007.

S. Chiappa and J. Peters. Movement extraction by detecting dynamics switches and repetitions. *Advances in Neural Information Processing Systems*, 23:388–396, 2010.

A. Coates, P. Abbeel, and A. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine Learning*, pages 144–151. ACM, 2008.

K. Dixon and P. Khosla. Trajectory representation using sequenced linear dynamical systems. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3925–3930. IEEE, 2004.

S. Dong and B. Williams. Motion learning in variable environments using probabilistic flow tubes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1976–1981. IEEE, 2011.

S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363, 2006.

S. Ekvall and D. Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):223–234, 2008.

E. Fox, E. Sudderth, M. Jordan, and A. Willsky. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th International Conference on Machine Learning*, pages 312–319. ACM, 2008.

E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Sharing features among dynamical systems with beta processes. *Advances in Neural Information Processing Systems 22*, pages 549–557, 2009.

E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Joint modeling of multiple related time series via the beta process. *arXiv:1111.4226*, 2011.

M. Gienger, M. Muhlig, and J. Steil. Imitating object movement skills with robots: A task-level approach exploiting generalization and invariance. In *International Conference on Intelligent Robots and Systems*, pages 1262–1269. IEEE, 2010.

D. Grollman and O. Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 261–266. IEEE, 2010a.

D. H. Grollman and O. C. Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *Proceedings of the International Conference on Robotics and Automation*, 2008.

D. H. Grollman and O. C. Jenkins. Incremental learning of subtasks from unsegmented demonstration. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 261–266, 2010b.

M. Huber and R. A. Grupen. Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1366–1371, 1997.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. *Advances in Neural Information Processing Systems 16*, pages 1547–1554, 2003.

O. C. Jenkins and M. Matarić. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 441–448, Jul 2004a.

O. C. Jenkins and M. J. Matarić. Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion. *International Journal of Humanoid Robotics*, 1(2): 237–288, Jun 2004b.

H. Kjellström, J. Romero, and D. Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 115 (1):81–90, 2011.

G. Konidaris, S. Osentoski, and P. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

G. D. Konidaris, S. R. Kuindersma, A. G. Barto, and R. A. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, 2010.

D. Kulic, W. Takano, and Y. Nakamura. Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics*, 25(5):1158–1166, 2009.

F. J. J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

B. Michini and J. P. How. Bayesian nonparametric inverse reinforcement learning. In *European Conference on Machine Learning*, 2012.

S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31 (2):249–267, 2012.

M. Mühlig, M. Gienger, and J. J. Steil. Interactive imitation learning of object movement skills. *Autonomous Robots*, 32(2):97–114, 2011.

J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2):79–91, 2004.

G. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 295–302, 2007.

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 663–670, 2000.

M. Nicolescu and M. J. Matarić. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 241–248, 2003.

S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246, 2012.

S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, 2013.

S. Osentoski, B. Pitzer, C. Crick, G. Jay, S. Dong, D. Grollman, H. B. Suay, and O. C. Jenkins. Remote robotic laboratories for learning from demonstration. *International Journal of Social Robotics*, 4(4):449–461, 2012.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal. Skill learning and task outcome prediction for manipulation. In *Proceedings of the 2011 IEEE International Conference on Robotics & Automation*, 2011a.

P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 365–371, 2011b.

P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards associative skill memories. *IEEE-RAS International Conference on Humanoid Robots*, 2012.

J. Peters, S. Vijayakumar, and S. Schaal. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learning*, pages 280–291, 2005.

A. Poritz. Linear predictive hidden markov models and the speech signal. In *Proceedings of the Seventh IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1291–1294. IEEE, 1982.

D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.

L. Riano and T. McGinnity. Automatically composing and parameterizing skills by evolving finite state automata. *Robotics and Autonomous Systems*, 60(4):639–650, 2012.

L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. Learning collaborative impedance-based robot behaviors. In *AAAI Conference on Artificial Intelligence*, Bellevue, Washington, USA, 2013.

S. Schaal. Dynamic movement primitives–a framework for motor control in humans and humanoid robotics. *The International Symposium on Adaptive Motion of Animals and Machines*, 2003.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. *International Symposium on Robotics Research*, pages 561–572, 2004.

W. D. Smart and L. P. Kaelbling. Effective reinforcement learning for mobile robots. In *2002 IEEE International Conference on Robotics and Automation*, pages 3404–3410, 2002.

J. Tang, A. Singh, N. Goehausen, and P. Abbeel. Parameterized maneuver learning for autonomous helicopter flight. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1142–1148. IEEE, 2010.

Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st national conference on Artificial intelligence*, pages 1000–1005, 2006.

M. Toussaint, N. Plath, T. Lang, and N. Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. *IEEE International Conference on Robotics and Automation*, pages 385–391, 2010.

R. Wilcox, S. Nikolaidis, and J. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proceedings of Robotics: Science and Systems*, 2012.

B. D. Ziebart, A. Maas, J. D. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.