Robotics: Science and Systems 2013
Berlin, Germany, June 24-28, 2013

Incremental Semantically Grounded Learning from Demonstration

Scott Niekum*, Sachin Chitta†, Bhaskara Marthi‡, Sarah Osentoski§ and Andrew G. Barto*
*University of Massachusetts Amherst, Amherst, Massachusetts 01003
Email: sniekum@cs.umass.edu
†Willow Garage, Menlo Park, CA 94025
‡Vicarious Systems, Inc, Union City, CA 94587
§Robert Bosch LLC Research and Technology Center, Palo Alto, CA 94304

Abstract—**Much recent work in robot learning from demonstration has focused on automatically segmenting continuous task demonstrations into simpler, reusable primitives. However, strong assumptions are often made about how these primitives can be sequenced, limiting the potential for data reuse. We introduce a novel method for discovering semantically grounded primitives and incrementally building and improving a finite-state representation of a task in which various contingencies can arise. Specifically, a Beta Process Autoregressive Hidden Markov Model is used to automatically segment demonstrations into motion categories, which are then further subdivided into semantically grounded states in a finite-state automaton. During replay of the task, a data-driven approach is used to collect additional data where they are most needed through interactive corrections, which are then used to improve the finite-state automaton. Together, this allows for intelligent sequencing of primitives to create novel, adaptive behavior that can be incrementally improved as needed. We demonstrate the utility of this technique on a furniture assembly task using the PR2 mobile manipulator.**

I. Introduction

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. For this reason, robot learning from demonstration (LfD) [2] has become a popular way to program robots. LfD allows users to teach a robot by example, often eliminating the need for specialized knowledge of the robotic system and taking much less time than it would take an expert to design a controller by hand.

While much LfD research has focused on tasks that can be represented by monolithic policies, some recent work has focused on automatically segmenting demonstrations into simpler primitives that can be sequenced to perform complex, multi-step tasks [7, 10, 13]. Such segmentations can be performed by humans, but this may require specialized knowledge, such as the robot's internal representations and kinematic properties. Furthermore, manually managing, memorizing, and reusing a library of primitives becomes intractable for a human user as the library grows in size. Thus, it is advantageous for primitives to be automatically segmented and managed.

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. However, autonomous segmentation and management of primitives also implies the need for the automated sequencing of these primitives to perform a task. Current LfD methods take a variety of approaches to sequencing, ranging from associative skill memories [16], 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 [1, 13]. 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.

A novel method is presented 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. Specifically, 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 semantically grounded movement primitives that are used to create a finite-state representation of the task. In this representation, 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 replay a complex, multi-step task.

However, initial demonstrations do not always cover all the possible contingencies that may come up 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.

II. Background

A. Beta Process Autoregressive Hidden Markov Model

The Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM) [5] fixes two major problems with the standard HMM for segmenting continuous demonstration data. First, rather than depending on a fixed number of hidden modes, it uses a beta process prior that leverages an infinite feature-based representation, in which each demonstration 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 demonstrations, and an appropriate number of modes is inferred directly from the data without the need for model selection. Second, rather than modeling observations as independent given the mode, the BP-AR-HMM is *autoregressive* and can describe temporal dependencies between continuous observations as a vector autoregressive (VAR) process, a special case of a linear dynamical system. The generative model for the BP-AR-HMM can be summarized as follows [6]:

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

First, a draw B from a Beta Process (BP) provides a set of global weights for the potentially infinite number of modes. Then, for each demonstration, 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 modes, are present in the i^{th} time series. Thus, B encourages sharing of features amongst multiple demonstrations, while the X_i leave room for variability. Next, given the features that are present in each demonstration, for all modes j, the transition probability vector $\pi_j^{(i)}$ is drawn from a Dirichlet distribution that is symmetric, with the exception of the dimension corresponding to self-transition bias κ. A mode $z_t^{(i)}$ is then drawn for each time step t from the transition distribution of the mode at the previous time step. Finally, given the mode at each time step and the *order* of the model, r, the observation is computed as a sum of mode-dependent linear transformations of the previous r observations, plus mode-dependent noise.

B. Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) [9] are a formulation that can describe the evolution of dynamical systems over time using a system of nonlinear differential equations. DMPs have many desirable properties for reproducing and generalizing movements in LfD: they are provably stable and convergent, scale naturally in time and space, and afford simple LfD and reinforcement learning algorithms.

One formulation of DMPs [14] for discrete movements can be described as:

$$
\begin{aligned}
\tau\,\dot{v} &= K(g-x) - Dv - K(g-x_0)s + Kf(s) && (1)\\
\tau\,\dot{x} &= v && (2)\\
\tau\,\dot{s} &= -\alpha s, && (3)
\end{aligned}
$$

for spring constant K, damping constant D, position x, velocity v, goal g, phase s (a 0-1 surrogate for time), temporal scaling factor τ, and constant α. The nonlinear function f is approximated by a weighted set of basis functions $\psi_i(s)$, scaled by the phase variable, s: $f(s) = \sum_{i=1}^{N} w_i \psi_i(s)s$.

Given a demonstration trajectory $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ of length T, we can use LfD to learn the weights for the basis functions [18]. Rearranging Eq. 1, integrating Eq. 3, and substituting in the demonstration for the appropriate variables results in:

$$
f_{\mathrm{target}}(s) = \frac{-K(g-x(s)) + D\dot{x}(s) + \tau\,\ddot{x}(s)}{g - x_0}. \quad (4)
$$

A simple linear regression problem to find the weights w_i can then be obtained by choosing an α, setting K and D for critical damping, and setting the goal to $g = x(T)$.

III. Constructing Finite-State Task Representations

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 semantically meaningful 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 semantically meaningful 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 semantic meanings, but similar statistical properties. So how can semantically grounded primitives be created from motion categories?

We make the assumption that exemplars of a semantically grounded primitive will generally fall into the same motion category (i.e. have similar statistical properties), but that all

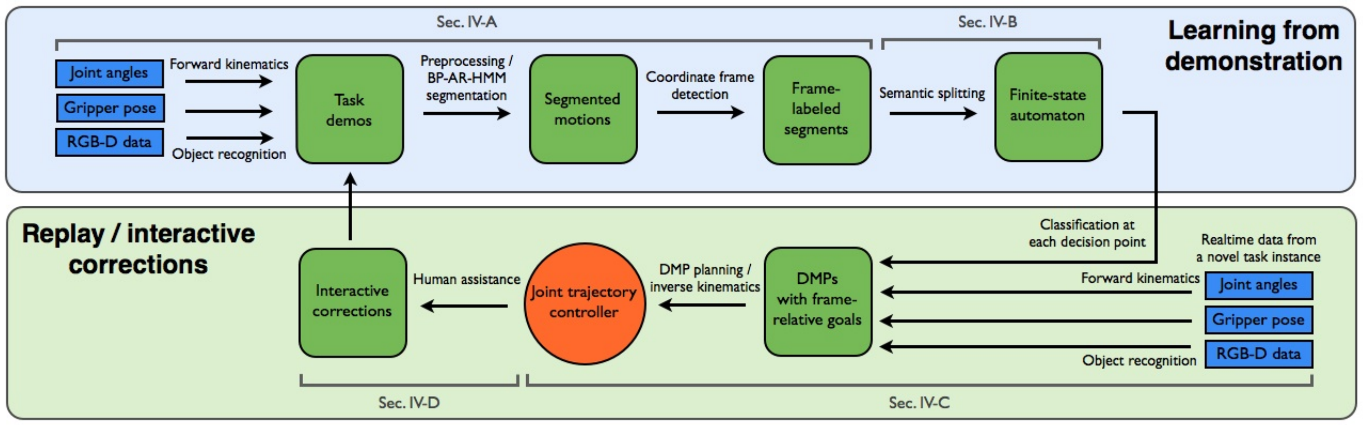


Fig. 1. Overview of the iterative learning from demonstration framework.

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 semantically meaningful splitting criteria. This can be achieved by performing splits based on the correlation of state visitation history with other types of semantic 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 semantically different primitives that are appropriate during various phases of task progress. In this case, only parent states are examined, i.e. one-step histories, due to the relative sparsity of data in an LfD setting.

If a semantic 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 semantically incorrect 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 1 shows an overview of the whole system, which is described in greater detail in the following section.

IV. Methodology

A. Demonstrations and segmentation

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. AR tags, a type of visual fiducial, 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.

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

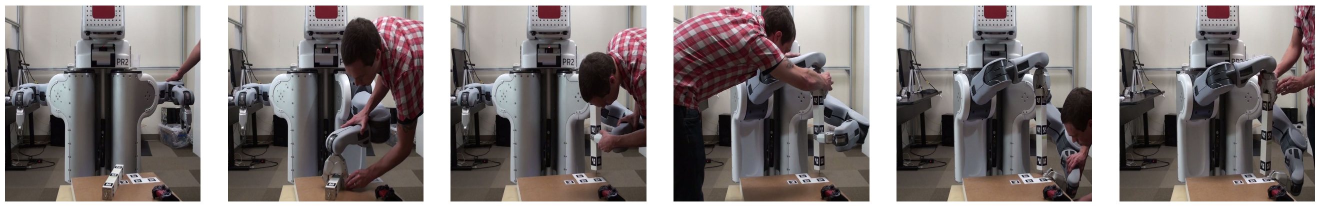


Fig. 2. A kinesthetic demonstration of the table assembly task.

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 Niekum et al. [13]. 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 Niekum et al. [13] 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 $a_{i,j} \in$ {'object 1', ..., 'object n', 'torso', 'none'}.

B. 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[1] $N_1, \ldots, N_u$, 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:

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

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 [11] 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:

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

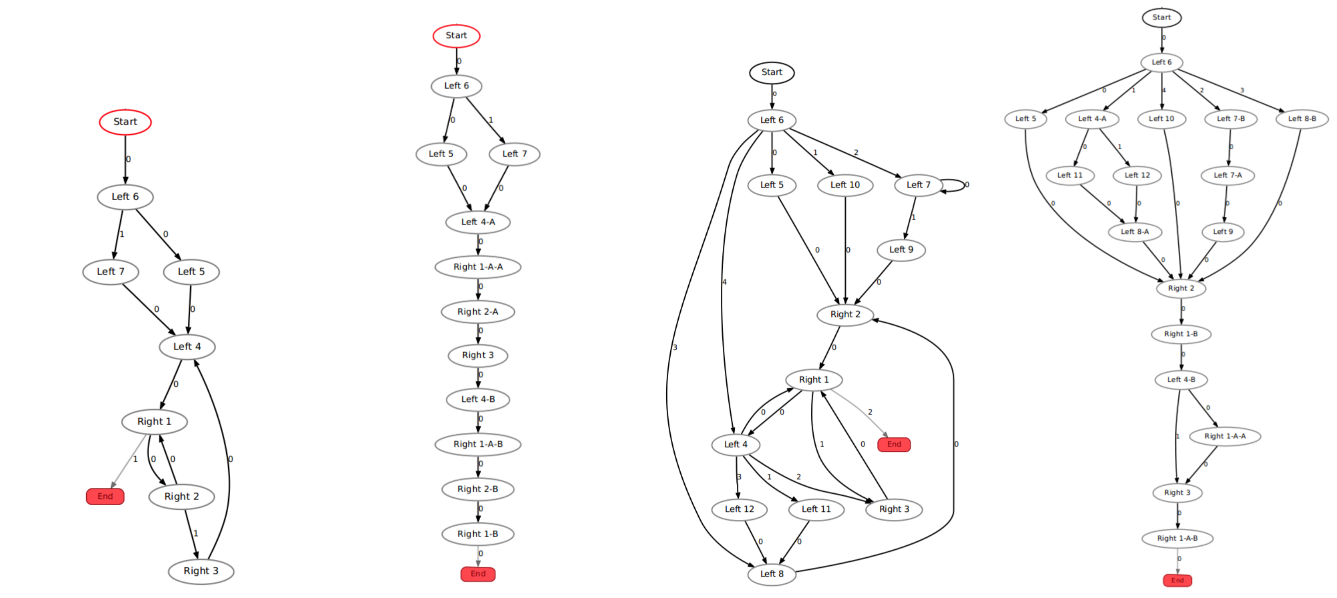
This suggests the node should be split if $z < K_\alpha$, the critical value for significance value α. Here, $\alpha = 0.05$ is used.

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 of *semantic splitting* is then repeated, iterating over all the nodes until no more splits can be made.

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



(a) Demonstrations only, before semantic splitting

(b) Demonstrations only, after semantic splitting

(c) Demonstrations + interactive corrections, before semantic splitting

(d) Demonstrations + interactive corrections, after semantic splitting

Fig. 3. FSA structures before and after semantic splitting. Nodes are labeled with the relevant arm (left/right), ID numbers, and A's and B's to denote splits.

Once the structure of the FSA is finalized, then 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.

C. 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 II-B.

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.

D. 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 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 biasing the sampling process. Finally, the FSA is reconstructed with the new segmentations, providing additional exemplars and refining the structure of the FSA.

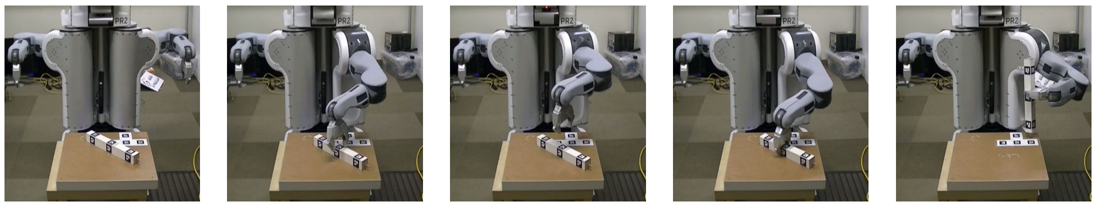


Fig. 4. A recovery behavior when the robot misses the original grasp.

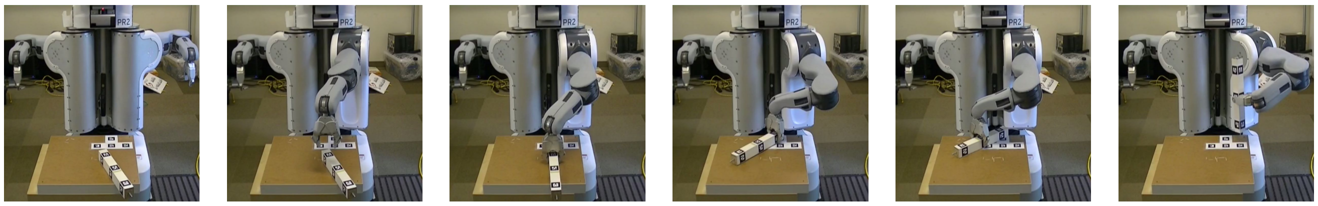


Fig. 5. A recovery behavior when the table leg is too far away to grasp at the desired location.

V. Experiments

A. Experiment 1: demonstrations, corrections, and replay

We evaluated our system on a furniture assembly task, using a PR2 mobile manipulator to partially assemble a small off-the-shelf table[2]. 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 2.

The demonstrations were then segmented and an FSA was built as described in Section IV. Figures 3(a)–(b) show the structure of the FSA before and after semantic splitting occurred; it can be seen that the node splits yield a much simpler, linear structure that better 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 3(c)–(d) show the structure of the FSA before and after semantic splitting occurred. After splitting, this FSA is similar to that of Figure 3(b), but with several branching pathways near the beginning that then 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 4 shows a successful recovery from a missed grasp, while Figure 5 shows the robot bringing the table leg closer for a re-grasp when it is too far away. Finally, Figure 6 shows a full successful execution of the task without human intervention, demonstrating that these error recovery capabilities did not interfere in cases where no contingencies were encountered.

B. Experiment 2: method comparisons

The table in Figure 7 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. [16], 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 semantic 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

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

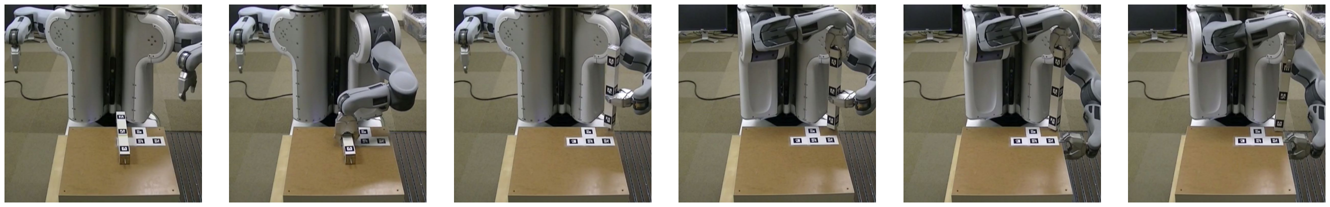


Fig. 6. A full successful execution of the task without any human intervention.

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 Figure 7 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 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 semantic splitting, several semantically different primitives often got combined into a single node, corrupting the structure of the FSA, and making classification and exemplar selection more difficult.

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

	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

Fig. 7. Ten trials of the task with corresponding performance data for four different types of sequencing. 'Fail' indicates failure and integer values correspond to the number of human assists required during successful executions.

VI. Related Work

The most closely related work to ours is that of Grollman et al. [7], which addresses *perceptual aliasing* by using a nonparametric Bayesian model to infer a mixture of experts from unsegmented demonstration data and then using multimap regression to assign observed states to experts. Butterfield et al. [3] 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.

Other work aims to intelligently sequence predefined primitives, rather than discover them from data. Toussaint et al. [20] use heuristics to translate perceptions into predefined symbols, which are then used by a rule-based planner. In the control basis framework [8], 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. [17] evolve the structure of a finite-state automaton that defines transition behavior between the skills. Pastor et al. [16] 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. [21] 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 [4] 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 [19] 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 [1]. Upon replay, the teacher can use an interface to view and edit keyframes to fine-tune task performance. Muhlig et al. [12] 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 task objects.

VII. Conclusion

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 semantically 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 semantically meaningful 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 semantically 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 robot. 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 lead 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, showing that an FSA with semantic 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. Finally, 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 [15]. With such improvements, 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.

VIII. Acknowledgements

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

References

[1] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching : A human-robot interaction perspective. *International Conference on Human-Robot Interaction*, 2012.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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, 2010.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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, 2011.

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

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

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

[19] 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.

[20] 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.

[21] 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.