Auton Robot (2016) 40:211–227
DOI 10.1007/s10514-015-9448-x

Simultaneously learning actions and goals from demonstration

Baris Akgun[1] · **Andrea Thomaz**[1]

Received: 27 June 2014 / Accepted: 19 June 2015 / Published online: 9 July 2015
© Springer Science+Business Media New York 2015

Abstract Our research aim is to develop interactions and algorithms for learning from naïve human teachers through demonstration. We introduce a novel approach to leverage the goal-oriented nature of human teachers by learning an action model and a goal model simultaneously from the same set of demonstrations. We use robot motion data to learn an action model for executing the skill. We use a generic set of perceptual features to learn a goal model and use it to monitor the executed action model. We evaluate our approach with data from 8 naïve teachers demonstrating two skills to the robot. We show that the goal models in the perceptual feature space are consistent across users and correctly recognize demonstrations in cross-validation tests. We additionally observe that a subset of users were not able to teach a successful action model whereas all of them were able to teach a mostly successful goal model. When the learned action models are executed on the robot, the success was on average 66.25 %. Whereas the goal models were on average 90 % correct at deciding on success/failure of the executed action, which we call monitoring.

Keywords Learning from demonstration · Goal learning · Human–robot interaction

✉ Baris Akgun
bakgun@gatech.edu

Andrea Thomaz
athomaz@cc.gatech.edu

[1] College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

1 Introduction

There is a growing interest in deploying robots in dynamic and unstructured humans environments. Endowing robots with all of the skills needed in these environments is a large task, and our work is aimed at empowering end-users to teach these skills through Learning from Demonstration (LfD) interactions. Our work is focused on LfD techniques for personal robots that will have a non-robotics expert as their end-user, addressing both the interactions and the algorithms that are best suited for LfD from these naïve human teachers. This scenario presents several interesting design challenges:

- We need the demonstrations to be easy for a teacher to provide.
- The system needs to learn efficiently from very few interactions with the teacher.
- We want generic LfD systems that can learn a variety of skills without tuning parameters or features for each skill to be learned.
- We cannot assume that all end-user will give accurate demonstrations of complex motor skills (e.g., in our case skills for a 7-DOF manipulator).

In this work we present an approach that addresses all of these challenges. We make use of keyframe demonstrations since our prior studies with end-users have shown that these result in more consistent and noise-free demonstrations compared to providing full trajectories (Akgun et al. 2012a, b, c). Keyframes are a sparse (in time) set of sequential points that the teacher demonstrates to the robot. In addition to ease of use, our focus is on efficiency. In the experiments presented here, the robot receives less then ten demonstrations of a skill for its modeling task.

The approach we take to skill learning in this work is motivated by an observation we have made several times in experiments with end-users teaching object-directed motor skills: they tend to concentrate on achieving the goal of the demonstrated skill rather than on consistent demonstrations of how to achieve it (Akgun et al. 2012b,c; Akgun and Thomaz 2013). In retrospect, this is quite logical and agrees with a vast literature in developmental psychology pointing to the fact that humans are goal-oriented in their perception and imitation of motor skills from a very early age (Csibra 2003; Meltzoff and Decety 2003). Additionally, we believe that asking people to provide keyframe demonstrations is particularly aligned with goal demonstrations, since they are sparsely highlighting the salient parts of the skill.

Given this observation, the approach we take in this work learns an action model and goal model for a skill simultaneously, but we maintain them as separate learning problems. This is a novel approach to skill learning in LfD where typically the problem is defined as learning in a motion space (e.g. joint space, end-effector space) or in a combined sensorimotor space that is known in advance to be good for representing a particular skill. In this work, we use the action model to execute the learned skill and the goal model to monitor the execution:

1. *Learning a goal model* The feature space of the goal model is a generic and high-dimensional visual representation of how the object of attention changes in the workspace over the course of the action. We use the same feature space to learn two different skills (pouring from a cup and closing a box), i.e., we do not change our representation based on a skill. This constitutes learning the perceptual objective of the skill, and in this work we show that goal models taught by users with just 9 demonstrations are able to accurately monitor the learned action model. Even demonstrations that resulted in unsuccessful action models were able to successfully model the perceptual goal of the skill in order to achieve accuracy in the monitoring task. The average success rate of skill execution monitoring was 90 %. In future work this will be used as an objective function in policy search to improve the action model itself.
2. *Learning an action model* The feature space of the action model is the end-effector pose with respect to an object of attention. Prior work suggests end-users vary in their ability to give good action model demonstrations, and our experimental findings presented here confirms this. The average success rate of execution success was 66 %.

In the next section we provide an overview of related work. We then detail our approach in Sect. 3 and our implementation in Sect. 4. We explain our data collection procedure in Sect. 5 and evaluate the results in Sect. 6. We discuss our approach, system requirements, results, and future work in Sect. 7. We make concluding remarks and summarize our work in Sect. 8.

2 Related work

General surveys of the field of LfD can be found in Argall et al. (2009) and Chernova and Thomaz (2014). In LfD, existing works tend to fall into two categories, what we call skill learning and task learning. One of the aims of our work is to bridge this dichotomy, by learning task goals for a skill learning method.

2.1 Task learning

The main idea of *task learning* methods is to map motor and sensor level information to pre-defined symbols and learn the relationship between them (Levas and Selfridge 1984; Nicolescu and Matarić 2003; Ekvall and Kragic 2008; Dantam et al. 2012). Typically these approaches assume a pre-defined mapping of sensor data to objects/symbols, and assume the task uses a given set of primitive actions. In contrast, we aim to learn the perceptual goals of a skill without assuming predefined symbols for actions or objects.[1]

A similar approach is described in Jäkel et al. (2006), which presents a method that learns kinematic task constraints through low-level demonstrations and generalizes them with semi-supervision. The aim is to learn these constraints for planning. The demonstrator, environment and objects are fully modeled. We focus on simple interactions with naïve teachers rather than heavily instrumented expert demonstrators. Another similar work is presented in Chao et al. (2011). Pre- and post-conditions for pick and place skills are learned from continuous perceptual features from human demonstrations. The aim is to use the learned models to bootstrap learning new skills (in the form of pre- and post-conditions). Our insight is that the salient keyframes provided by the teacher are highly suitable for learning perceptual representations of goals (similar to Chao et al. 2011) but taking intermediate steps into account), learning the accompanying action at the same time and using the goal models to monitor executed action models.

2.2 Skill learning

There is a large body of literature on learning motor control level models, or *skill learning*. Dynamical system approaches such as Stable Estimator of Dynamical Systems (SEDS) (Khansari-Zadeh and Billard 2011) and Dynamic Movement Primitives (DMP) (Pastor et al. 2009) as well as mixture

[1] We only assume that a segmentation for objects is available.

models, such as Gaussian Mixture Models (Calinon et al. 2007), are skill learning methods. These methods all involve estimating parameters of a dynamical system, a distribution and/or a function approximator.

Classification based methods, such as Hovland et al. (1996) and Jenkins et al. (2000), can be seen as skill learning methods in which input demonstrations are mapped to action primitives or robot states. The transition structure between the primitive actions or states represent the skill.

Most current work does not incorporate the perceptual state into the learning problem other than object pose estimation.

A similar idea related to monitoring and presented along skill learning is given in Pastor et al. (2011). The robot executes its learned skill, collects sensory data and the successful executions are labelled by hand. The robot then builds a Gaussian model for the trajectories for each sensory state dimension, which requires a high number of skill executions. These models are used in hypothesis testing during future executions to monitor the skill. In contrast, we use the sensory data obtained during the skill demonstrations to learn a goal model without further manual labelling and skill repetition.

2.3 Inverse reinforcement learning

Another approach to skill learning, is Inverse reinforcement learning (IRL) (Abbeel and Ng 2004) or similarly inverse optimal control (IOC) (Ratliff et al. 2009). In IRL, a reward or cost function is estimated from demonstrations and then used to extract a policy. The main idea behind the IRL approaches is that the reward function is a better representation of the demonstrated skill than the policy. Our goal learning idea is similar to the main idea of inverse reinforcement learning; extracting a reward function from demonstrations.

2.4 Keyframes and trajectory segmentation

Keyframe related ideas exist in other fields as well. In industrial robot programming, these are referred to as *via-points*. A robot programmer records important points by positioning the robot and specifies how to move between them. Keyframes are used extensively in computer animation (Parent 2002). The animator creates frames and specifies the animation in-between. However, there is no learning component in these approaches. In Miyamoto et al. (1996), keyframes are extracted automatically from continuous demonstrations and updated to achieve the demonstrated skill. Another approach is to only record keyframes and use them to learn a constraint manifold for the state space in a reinforcement learning setting (Bitzer et al. 2010). Whole body grasps for a simulated humanoid are learned in Hsiao and Lozano-Perez (2006) by forming template grasp demonstrations via keyframes, which are the start/end points of a demonstration and the points of contact and points of lost contact with the objects.

A related topic to keyframes in LfD is trajectory segmentation. Keyframes are similar to the segmentation points. In our prior work (Akgun et al. 2012a), we introduced *hybrid demonstrations*, in which teachers can demonstrate both keyframes and trajectories in any sequence and number in the context of a single demonstration. This gives the teacher the ability to segment trajectories as well as provide keyframes where the trajectory information does not matter.

Some LfD methods employ automatic trajectory segmentation to break down the demonstrations to facilitate learning. Kulić et al. (2012) describes an incremental LfD system in which the demonstrations are first segmented and the resulting trajectory segments are treated as small actions which are then learned using Hidden Markov Models (HMM). Then these models are grouped together with the similar models to form a tree structure. In addition, a graph is built between action groups to form a higher level representation. A similar approach is presented in Niekum et al. (2013), which uses DMPs to learn small actions. These methods also aim to bridge the gap between *skill learning* of particular actions and *task learning* of higher level plans or graph structure. An example LfD system towards this direction is presented in Niekum et al. (2015), which builds a finite state representation of the task through demonstrations and leverages this for adaptive skill sequencing and error-recovery. Their work differs to ours since they are not building perceptual models for monitoring the execution.

Although similar, trajectory segmentation and keyframes are not directly comparable. Some of the trajectory segmentation methods aim to break down trajectory demonstrations to learnable chunks to alleviate model selection, and as such the segmentation points may not be of specific importance. Other trajectory segmentation methods try to find important points (e.g. when the relevant reference frame changes during demonstration) to gain higher level information but this is a hard task and it depends on a predetermined definition of these important points. In contrast, the aim of keyframes is to let the human teacher highlight the important parts of the skill from the human teacher's perspective.

2.5 Human–robot interaction

One of the primary distinctions of our work is our focus on human–robot interaction in the context of LfD. For example, most methods implicitly assume the demonstrators are good at using the robot. In general, the demonstrators used in evaluations are the developers of the methods themselves.

The related work on LfD evaluations with non-experts is sparse. Suay et al. (2012) tested three LfD methods, with naïve users. One of their findings was that these users were

not able to teach successful policies whereas experts (the authors themselves) were able to do so within minutes. Thomaz et al. have investigated how humans teach software agents and robots various tasks, and developed a reinforcement learning algorithm to leverage natural human behavior (Thomaz and Breazeal 2008a, b). In the context of skill learning, Cakmak and Thomaz investigated how naïve users can be guided to provide better demonstrations through teaching heuristics and active embodied queries by the robot in LfD interactions (Cakmak 2012).

In Akgun et al. (2012b, c), we evaluated trajectory demonstrations and keyframe demonstrations (temporally sparse set of ordered points) with end-users, finding that both have their merits. Trajectories are necessary for dynamic skills whereas skill with non-dynamic constraints were best taught with keyframes. Based on these results, we have introduced a keyframe-based learning from demonstration (KLfD) method in Akgun et al. (2012a) that learns from hybrid keyframe and trajectory demonstrations. Keyframes are extracted from the trajectory portions while retaining velocity and acceleration information. A dynamic time warping (DTW) approach is used to learn from the demonstrated and extracted keyframes. In our current work, we extend this approach to add goal models but remove the inclusion of trajectory demonstrations and use a Hidden Markov Model (HMM) approach to learning. This HMM approach can be used within the context of the original KLfD method as well. We think that the HMM approach is superior given that it is inherently stochastic and handles multiple demonstrations better than the DTW approach.

3 Approach

In the work presented here we concentrate on object-based manipulation skills in which dynamics is not a component of the goal. This class of skills encapsulates many day-to-day activities (e.g., fetching items, simple kitchen tasks, general cleanup, aspects of doing laundry or ironing, etc.). Our aim is to develop a generic LfD approach that does not require tuning per skill and that can handle a wide variety of such skills.

In our previous work, we investigated how naïve teachers demonstrate these types of goal-oriented object-based skills to robots (Akgun et al. 2012b, c). We observed that teachers were more concerned with achieving the skill than providing good demonstrations. People showed the robot *what-to-do* for a skill, rather than *how-to-do* the skill. For example, when teaching the robot how to close a box lid as pictured in Fig. 1, it was not uncommon to see a single demonstration that would include multiple attempts at swinging the hand to hit the lid until the box was successfully closed. This demonstrates the *goal* of the skill, but a single demonstration that includes several failed attempts is not a very good example of the target *action* itself.

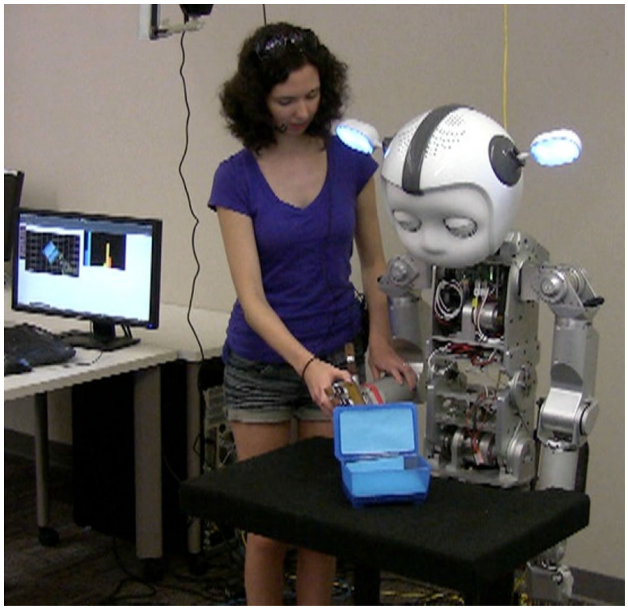


Fig. 1 A teacher providing a kinesthetic demonstration of a box closing skill to the robot

Our interpretation of this behavior is that teachers were goal-oriented. Their primary objective was to show the successful completion of the skill, and some people were also able to provide reasonable demonstrations for the exact motion the robot should take to achieve it. Based on this observation we propose a skill learning method that will learn goals in addition to actions and treat these modeling problems separately. This observed duality in human teachers has influenced our overall approach to LfD, illustrated in Fig. 2.

In addition to being goal-oriented, we found that some of the naïve users have difficulty in manipulating the robot to provide demonstrations (Akgun et al. 2012b, c). As a result, their trajectory demonstrations were inconsistent and noisy, which complicates learning. However, all the teachers were able to utilize keyframe demonstrations to provide data. The drawback of the keyframe demonstrations is that it loses any dynamics information. Since the demonstration pauses at each keyframe, the robot does not receive any information about target velocities for the keyframes. However, a large class of skills for personal robots do not feature complex dynamics, hence this is the class of skills we choose to focus on initially. Thus, keyframes help naïve teachers to provide better demonstrations to the robot and allow them to highlight important and salient aspects of the demonstrated skills. For these reasons, we utilize keyframes as our *input* in Fig. 2. The *input modality* in Fig. 2 refers to different ways teachers could demonstrate these keyframes to the robot such as teleoperation or kinesthetic teaching.

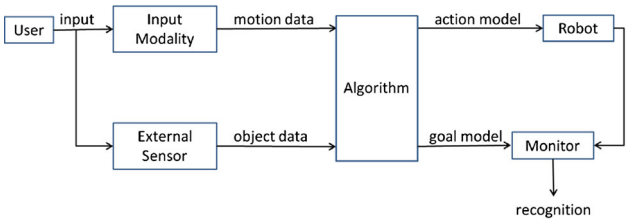


Fig. 2 The developed LfD system. The user demonstrates the skill by using keyframes. Two types of data is extracted at each keyframe; motion data (related to robot control) and object data (related to the object being manipulated for the skill). Then the same algorithm is used to learn two distinct models from the aforementioned data; an action model and a goal model. The learned action model is used to execute the skill and the learned goal model is used to monitor the execution

As seen in Fig. 2, the main idea of our approach is to use two different information streams, from the same set of demonstrations, to learn two different models. During demonstrations, we collect data from two sources: motion data and object data. The former is related to the motion of the robot, for example motor commands, joint positions, end effector pose etc. The action model is learned using this motion data. Our focus is on object related skills and the goals of these skills is to achieve certain object states during the skill. Hence, we use an *external sensor* to capture the state of the object. The goal model is learned using this object data. A single demonstration involves the teacher marking a series of keyframes that complete the skill. Each time the teacher marks a keyframe we record two different types: (1) a *goal keyframe* consisting of object data; and (2) an *action keyframe* consisting of motion data. Thus a single demonstration is treated as two simultaneous demonstrations: the *goal demonstration* is the set of goal keyframes, and the *action demonstration* is the set of action keyframes. In this work, we use the same learning algorithm on each type of data to learn the two different models but it is possible to use different algorithms for each. The exact skill modeling technique is not of key importance as long as some requirements for the models are satisfied (discussed further in Sect. 7).

There are multiple reasons for separating action and goal learning. For our purposes, data for both are from inherently different sources; the action data comes from the robot and the goal data comes from the object of interest. The separation allows for different levels of granularity between the models. The action might require multiple steps to change to state of the object. For example to lift a cup, the robot would need to approach the cup grasp it and then lift it. From the goal point of view, the object only changed its height. Learning the mapping between the action and goal spaces is an interesting problem. However, it would be highly complex and noisy and would require more data to learn a joint representation. We do not concentrate on this problem for this paper. Finally, the learned models have different purposes. The action model is used to execute the skill. The goal model is used to monitor the execution of the skill. The monitoring task involves using the goal model to classify the object data stream captured during skill execution.

We want to develop a generic LfD approach that can handle a variety of skills without tuning and that can represent the acceptable variance on executing the skill (as opposed to a single optimal way). Relatively generic feature spaces are needed to handle a variety of skills. This requires us to learn from multiple demonstrations so that there is enough information about the variance over how to execute the skill. Having a variance on the action model allows the robot to execute the skill with variety which is important in cluttered and/or new environments. Simply repeating a single demonstrations might fail or might not be possible at all. An example to the latter is when a transformed demonstration falls out of the robot's workspace due to new object location. The allowable variance over the execution is also useful for avoiding collisions in clutter but we are not addressing this problem in our current work. In addition, having multiple demonstrations allows us to estimate the variance on how the skill "looks", important for monitoring, especially given that sensors are noisy. A single demonstration is not enough to build a good goal model; even the same exact repetitions will not look the same.

Our approach and current implementation imposes several assumptions on the types of skills that we can handle. For this work we assume that skills have a single object of attention, they can be executed by a single end-effector, they are not cyclic and their goal involves a perceptual change in the object. Future work will look at removing some or all of these assumptions within the introduced framework.

4 Implementation

In this section, we describe our hardware platform, setup and how we implemented the individual components in Fig. 2.

4.1 Hardware platform

We are using a bi-manual upper-torso humanoid robot that has 7 degree-of-freedom (DoF) compliant arms with series-elastic actuation and gravity compensation. We employ an overhead ASUS Xtion Pro LIVE (RGBD camera) as our external sensor with an overhead view of the tabletop workspace seen in Fig. 1. We use the Point Cloud Library (PCL),[2] to process point cloud data. During the LfD interaction, there is a limited grammar of speech commands used to

[2] http://pointclouds.org/.

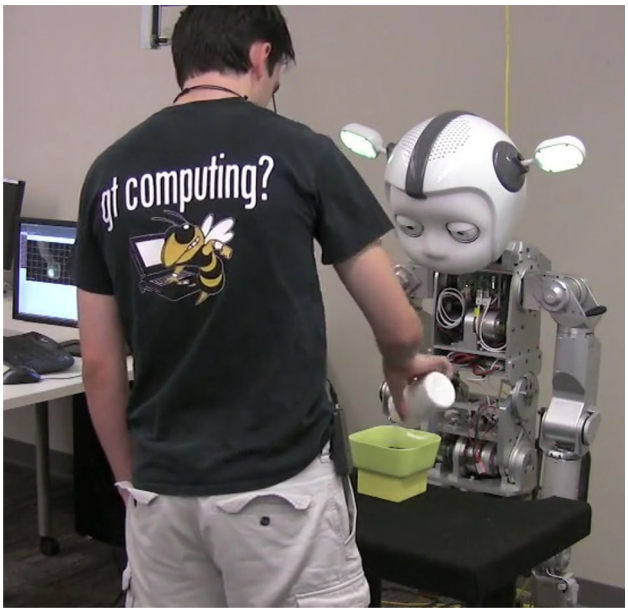


Fig. 3 A teacher providing a goal-only demonstration of the pour skill to the robot

mark keyframes and move the interaction forward. We use the Microsoft Windows 7 Speech API for speech recognition.

4.2 Demonstrations

In this work, the teachers use kinesthetic teaching to demonstrate skills to the robot. In kinesthetic teaching, the teacher guides the robot's arm physically to provide demonstrations, as shown in Fig. 1.

We also allow for the situation where the teacher provides a goal-only demonstration, by doing the skill themselves, as seen in Fig. 3. While this loses the action keyframes entirely, it does have the advantage that teachers can give demonstrations quickly and likely with less occlusions (due to only the human being in the view instead of human and the robot). In this case, when the teacher marks a keyframe, the robot only records a goal keyframe, and in the end the robot will have different size demonstration sets for learning the goal model and the action model. Our intention with this alternative is to provide a wider variety of ways that the teacher can provide goal demonstrations to the robot. In future work we will compare the utility of each interaction modality, but for the purpose of this paper we use both forms of goal demonstrations simply to show that a valid goal model can be learned from this.

The interaction flow and the demarcation of keyframes is controlled by speech commands. The list of speech commands and their function is shown in Table 1. The robot begins in an idle state, from which the teacher can put the robot in gravity compensation mode, open/close the hand, etc. prior to giving a demonstration. The teacher can choose whether to provide a kinesthetic demonstration by saying *I will guide you* (default) or a goal-only demonstration by saying *Watch me do it*. Then the demonstration begins when the teacher marks the first keyframe by saying *start here*. Each intermediate keyframe is marked by saying *go here*. And then the demonstration ends when the teacher says *end here*, taking the robot back to an idle state. As will be described in the data collection protocol (Sect. 5.2), this cycle is completed multiple times to provide a set of demonstrations for a skill.

Table 1 Speech commands for controlling the interaction and their function

Command	Function
Let's begin the experiment	Start the overall interaction
Release your arm	Put the robot in gravity compensation mode (needed for kinesthetic teaching)
Hold your arm	Put the robot in joint position mode
Let's learn a new skill	Start learning the first skill or move to the next skill
Watch me do it	Transition to a goal-only demonstration
I will guide you	Transition to a kinesthetic demonstration
Start Here	Indicate the starting of a skill demonstration and mark the first keyframe
Go Here	Mark a keyframe. Effective only between start here and end here commands
End Here	Indicate the end of a skill demonstration and mark the last keyframe
Close your hand	Close the robot's fingers and mark a keyframe if used between start here and end here commands
Open your hand	Open the robot's fingers and mark a keyframe if used between start here and end here commands

4.3 Feature spaces

4.3.1 Object data

In selecting a feature space for the object data, our aim is to have a feature space that is going to allow the robot to build a visual model of how the object is changes over the course of the action. Thus we select a set of features commonly utilized in the literature for object tracking and perception tasks; a combination of location, color, shape and surface features. As advances in object perception are made, this object feature space can be updated to reflect the state of the art.

We use an overhead RGBD camera as our external sensor. As a result, the raw sensor information for the object data is the colored point cloud data. The goal keyframe consists of features extracted from this RGBD data. We first segment the object(s) in the point cloud data, fit a bounding box and then extract features. This section details each of these steps.

To segment the objects, we use the approach in Trevor et al. (2013) to find spatial clusters of similar color. This procedure often results in over segmentation, especially if the object is occluded. The clusters are inspected and, if needed, are merged manually by one of the authors.[3]

As stated in Sect. 3, we assume that there is a single object of attention for each manipulation action that is to be learned. In the work presented here selecting this object is simplified by using a clean workspace in which only the object of attention is visible, in future work this could be selected automatically based on which objects the hand moves closest to, or which objects changed the most over the action, or by interacting with the teacher etc.

After segmentation, we fit a rotated bounding box to the object and use the pose of this box as the object pose. An example of the segmentation and the bounding box results can be seen in Fig. 5. We extract color and generic shape related features from the object using the point cloud and bounding box data. These include average RGB (3), number of points in the cloud, centroid of the bounding box (3), rotation of the bounding box with respect to the table normal, bounding box volume, bounding box area, bounding box side lengths (3), aspect ratio parallel to the table plane, bounding box area to volume ratio (scaled down) and bounding box volume to number of cloud points ratio, resulting in 16 features. Some skills can change the object location (e.g. pick and place) and color (e.g. pouring a different colored liquid in to a cup). Remaining features are extensions of commonly used 2D features. They represent the generic silhouette of the object, which can be changed by certain skills. Overall these are more global features for the object.

In addition to these, we use Fast Point Feature Histogram (FPFH) descriptors presented in Rusu et al. (2010) to capture surface features, to go beyond the object silhouette. We chose these features since they have been shown work for object recognition. The first step to calculate these features is to estimate the surface normals at each point followed by finding the centroid of the points. Then, the angular deviations between the axes of the surface normals at each point to the centroid of the object is calculated and binned to form a histogram. In our work, we use 9 bins per angle and utilize the reference implementation in the PCL. In the end, we get a 43 $(16 + 9 \times 3)$ dimensional goal space, treated as $\mathbb{R}^{43}$, for the goal keyframe.

[3] Integrating more robust object segmentation into the perception pipeline is left for future work. Since our experiments in this article are performed on a batch of demonstrations offline, robust online tracking is not our current focus. This will become important in our future work when we want learning to be an incremental online process, and we believe solutions exist for obtaining a robust segmentation for this purpose.

4.3.2 Motion data

We use the end-effector with respect to the target object as our motion data, which constitutes the action keyframe. After a demonstration, we transform the end-effector poses to the object reference frame (as calculated in Sect. 4.3.1).[4] We represent the end effector pose as the concatenation of a 3D vector as the translational component and a unit quaternion (4D) as the rotational component, resulting in a 7D vector, hence the action model lives in a 7 dimensional action space, treated as $\mathbb{R}^7$. A point in this action space is projected onto the space of rigid body transformations, SE(3), by normalizing the quaternion part wherever necessary (e.g. before execution).

Some existing work uses a state space that is known in advance to be appropriate for representing a particular skill. For example, learning a pool stroke with the specialized state space of cue rotations, tip offset and elbow posture (Pastor et al. 2011). Other seminal examples with a skill specific feature space include the pendulum swing up and balance task (Atkeson and Schaal 1997), playing table tennis (Mülling et al. 2013), flipping pancakes (Kormushev et al. 2010) and flying a model helicopter (Abbeel et al. 2010). In all of these prior works automatic feature selection problem is acknowledged as an important problem for future work. In other LfD work, the problem is defined as learning in the joint space, end-effector space (with respect to robot or landmark(s)) or both. We argue that the end-effector space is more general in most situations and hence more suitable to our aim of allowing end-users to teach a wide range object directed skills without changing the feature space. Although it is not addressed by our current work, we think a feature selection mechanism, applied on a per skill basis, on top of this generic space would be highly suitable.

4.4 Notation

In this section we define a list of symbols for several of the constructs introduced so far, to be used in the rest of the text.

- a_i^j: The ith action keyframe for the jth demonstration, $a_i^j \in \mathbb{R}^7$.
- g_i^j: The ith goal keyframe for the jth demonstration, $g_i^j \in \mathbb{R}^{43}$.

[4] This object based representation is fairly common in robotics.

A^j: The jth action demonstration, a set of action keyframes where $m(j)$ is the number of keyframes (the number of keyframes can be different for each demonstration), $A^j = \left\{a_1^j, a_2^j, \ldots, a_{m(j)}^j\right\}$.

G^j: The j^{th} goal demonstration, a set of goal keyframes, $G^j = \left\{g_1^j, g_2^j, \ldots, g_{m(j)}^j\right\}$.

D_A: The set of n action demonstrations, $\left\{A^1, \ldots A^n\right\}$.

D_G: The set of k goal demonstrations, $\left\{G^1, \ldots G^k\right\}$. k and n can be different, but in general $k \geq n$, since we get both action and goal demonstrations during kinesthetic teaching and also allow for goal-only demonstrations.

q_r: The rth observed keyframe during a skill execution, $q_r \in \mathbb{R}^{43}$.

Q: The set of observed keyframes during a skill execution, $\left\{q^1, \ldots q^p\right\}$, where p is the number of keyframes used in execution.

M_A: The action model.

M_G: The goal model.

4.5 Algorithm

In this work we use a Hidden Markov Model (HMM) to represent both the action model and the goal model of the skills. HMMs are useful tools for modeling sequential data where observations are noisy and sample independence assumption is too constrained. Keyframe demonstrations lend themselves naturally to such a model since they can be treated as sequential observations that are not independent. We interpret the hidden states as canonical steps (or true keyframes) of the skill. In addition, HMMs are generative which enables us to use them in skill execution.

We model the emissions as multivariate Gaussian distributions on the corresponding state space (either the action space or the goal space) and use full covariance. The HMM notation we use consists of the following:

N : The number of states

s_j : The jth state ($j = 1 \ldots N$). The states are not directly observable.

S : The set of all states, $S = \{s_1, \ldots, s_N\}$

μ_j: The emission mean for the jth state

Σ_j: The covariance matrix for the jth state

T: The $N \times N$ state transition matrix, $T(k, j) = P\left(s(t) = s_k | s(t-1) = s_j\right)$ is the transition probability from state j to state k

y: An emission vector

$P(y|s_j)$: The probability for the emission y in state s_j, $P(y|s_j) \sim \mathcal{N}(\mu_j, \Sigma_j)$

π: The N dimensional prior probability vector

ζ: The N dimensional terminal probability vector

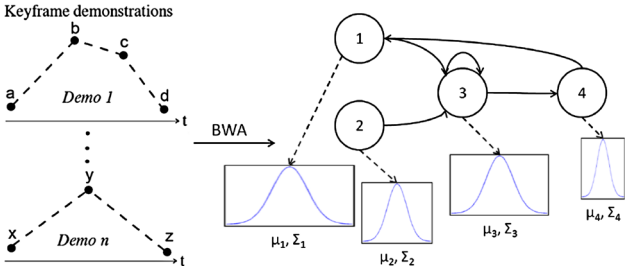


Fig. 4 A depiction of the learning process. The resulting model is a representative 4-state HMM with emission distributions. In addition to these, we learn transition probabilities, prior probabilities and terminal values

The set of action demonstrations D_A is used to learn the action model, M_A. Similarly the set of goal demonstrations, D_G is used to learn the goal model, M_G. We learn these models from multiple demonstrations and both M_A and M_G are individual HMMs.

We train these HMMs with the Baum–Welch algorithm (BWA) (Baum et al. 1970), which is an Expectation–Maximization (EM) type algorithm, initialized with k-means clustering, which itself is initialized uniformly randomly in the state space and restarted 10 times. We use the Bayesian Information Criterion (BIC) for model selection, i.e. , to select the number of states of the HMMs . We start by setting the number of states as the minimum number of keyframes seen during the demonstrations. The number of states is increased until we hit a minimum BIC score and select the corresponding one. Since the learning is initialized randomly, we run BWA 10 times given a number of states and select the model with the highest likelihood to calculate BIC.[5] Note that the action model and the goal model can have different number of states after training.

The standard BWA calculates T, π, $\mu_{1\ldots N}, \Sigma_{1\ldots N}$. In addition, we calculate the terminal probabilities. The terminal probability of a state represents the likelihood of that state being the last state for the HMM, and is calculated similar to the prior probability. We denote terminal probabilities with ζ. A representative learning process and the resulting HMM is shown in Fig. 4. We are going to use superscript to denote model membership for the parameters, for example π^A is the prior probabilities for the action model, T^G is the transition matrix for the goal model etc.

Another advantage of HMMs is that a single EM-step run of the system, with the Gaussian emission models, is polynomial, i.e. tractable, as opposed to learning an arbitrary dynamic Bayesian network (DBN).[6] This tractability is due to the Markov assumption of state transition, emissions only

[5] These details are given for completeness but another model selection procedure can be used as well.

[6] Although tractable approximate methods exist for DBNs.

depending on current state and the tractability of Gaussian model parameter estimation.

4.6 Action execution

To execute the learned skill, we generate a state path from M_A, extracting end-effector poses using the emission distribution and splining between the poses to get a trajectory for the robot to follow. This process is detailed in Algorithm 1.

We start by finding the most likely path between each prior state in π^A to each terminal state in ζ^A and store them (lines 1–11). The function *FindStatePath* (p, z, T^A) does the path finding between the state p and the state z, given T^A. It applies Dijkstra's algorithm on the negative logarithm of the entries of T^A as the edge weights. The shortest path we get by using the addition of the negative log-likelihoods is equivalent to the one we would get by maximizing the multiplication of the probabilities. We then select the most likely path among these paths, given the transition probabilities T^A, prior probabilities π^A and terminal probabilities ζ^A (line 12). If the initial position of the robot is important, we can select the prior state that is closest to this and then only consider the corresponding paths.

Algorithm 1 Tra = GenerateTrajectory(M_A)

1: $\Phi = \emptyset$
2: **for all** $p \in S^A$ **do**
3: **if** $\pi^A(p) \neq 0$ **then**
4: **for all** $z \in S^A$ **do**
5: **if** $\zeta^A(z) \neq 0$ **then**
6: ϕ = FindStatePath(p, z, T^A)
7: $\Phi = \Phi \cup \phi$
8: **end if**
9: **end for**
10: **end if**
11: **end for**
12: $\rho = \text{argmax}_{\phi \in \Phi}\ (loglik(\phi))$
13: $R \leftarrow \emptyset$
14: **for all** $s \in \rho$ **do**
15: $R \leftarrow \mu_s$
16: **end for**
17: Tra = Spline(R,v_{avg}, Δt)
18: **return** Tra

It might be the case that there is a cycle in the resulting path, for example when the user starts and ends the demonstrations at close enough robot poses. For this paper, we assume that the skills are not cyclic thus when we detect a cycle in the generated path, we stop i.e. allow for only a single cycle. Another way to resolve this is to interact with the user and ask if the skill has a cyclic component and how many times (or until what condition) the cycle should be executed. This is not something we address in our current approach.

Once we have selected a state path, we extract the resulting emission means of the generated path in the same sequence (lines 13–16), transform them to the robot coordinate frame (skipped in the algorithm for clarity) and fit a quintic spline between them (line 17). The robot follows the resulting trajectory (in the end effector space) to execute the skill. The transformation is done based on the current object pose, as estimated by the perception system. We assume a constant average speed (v_{avg}) between two points to decide on the timing of the spline, choose the initial and target velocities and accelerations as zero between the points and use a given time step (Δt) to decide on the density of the trajectory in time. This results in a straight path in the end-effector space between two points.

4.7 Goal monitoring

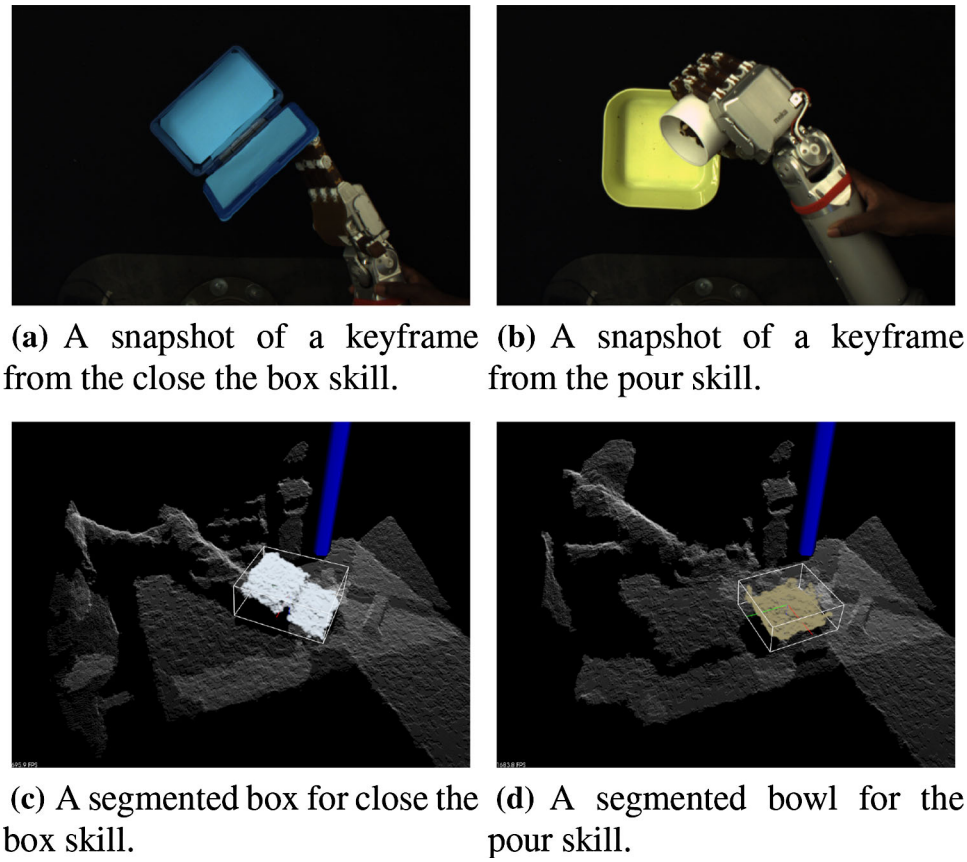
The goal model, M_G, is used to monitor the execution of the action model, M_A. This information could then be used by the robot for error recovery or to ask the human teacher for help in case of a failure or to move onto its next task in case of success.

To perform monitoring during execution, the robot extracts an observation frame (q) from object data in the goal space at each action keyframe it passes through. The action keyframes are at the emission means obtained as described in Algorithm 1. This results in an observation sequence, $Q = \{q_1, \ldots, q_p\}$, where p is the length of the state path ρ that is calculated in Algorithm 1.

A sequence of a short length can have a high likelihood score but it might not be enough to complete the skill. For example, observing an incomplete execution of a skill would yield a high likelihood but in reality it should be a failure since the skill is not completed. This is the reason we are estimating the terminal probabilities from demonstration data. On the other hand, it is not enough to just check whether the end-state is a terminal state for all the skills. Sub-goals of the skill might be important to achieve it but not be visible at the end state. We use the forward algorithm to calculate the likelihood of the the observed sequence with the inclusion of the terminal probabilities, $p_s = P(Q; M_G)$, given the goal model. We then decide failure if $p_s < \tau_s$ holds true where τ_s is a selected threshold.

In the current implementation, the monitoring decision is made at the end of execution. However, there is no technical limitation for it to be done as the skill is being executed. The likelihood of the current observation sequence can be calculated online and evaluated with a threshold. The only difference would be that the terminal state check would be done at the end of the execution. This could be used to determine early failure and show when the action failed. However, these are left for future work.

Fig. 5 Image snapshots as seen by the overhead camera



(a) A snapshot of a keyframe from the close the box skill.

(b) A snapshot of a keyframe from the pour skill.

(c) A segmented box for close the box skill.

(d) A segmented bowl for the pour skill.

5 Data collection

To test our approach, we collected demonstration data from 8 users (4 male and 4 female) with ages between 18 and 26 (median 21.5). They were recruited from the campus community, and none had prior experience interacting with a humanoid robot in an LfD setting. Each person spent an average of 30 min with the robot, first practicing the interaction and then providing 9 demonstrations for each of our 2 skills. In this section we detail the skills and the data collection procedure.

5.1 Skills

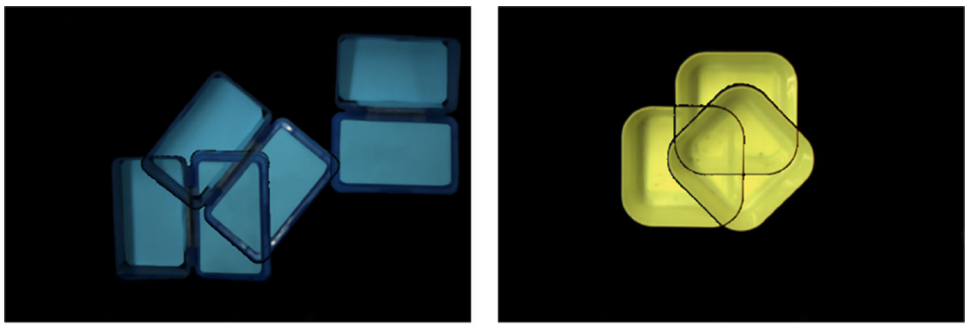
We have one practice skill and two evaluation skills. *Touch* is the practice skill, with the goal of touching two objects in a given order. This skill is used to get the participants familiar with the two types of demonstration (kinesthetic and goal-only) and the keyframe interaction dialog in general. The two evaluation skills are as follows.

In the *close the box* skill, the aim is to close a particular box, as seen in Fig. 5a. The reference point of this skill is the box. The pose of the end-effector is encoded with respect to the box reference frame. The centroid and angle of the bounding box features for the keyframe sequence is also encoded with respect to the reference frame of the box in the first keyframe. The success metric for this skill is whether or not the box lid is closed.

The aim of the *pour* skill is to pour coffee beans from the cup to the square bowl, as seen in Fig. 5b. We assume the object of interest is the target bowl, since the cup can be considered as the part of the end-effector. The end-effector pose and the relevant features are encoded with respect to the bowl. The amount of coffee beans in the cup is measured to be the same across all demonstrations.

An interesting future work is to select the objects of interest and the reference points either automatically or through user interaction. However, for this study, we keep them fixed, and this is the only skill specific representation decision we make.

These two skills are very different from each other in terms of both object data and motion data. We chose these skills in order to show that we can learn different goal models without engineering the feature space for a particular task. These are two different examples of the class of object directed motion tasks that the approach is designed for (see Sect. 3). While we would have liked to include even more skills, this decision is a trade-off with collecting a greater set of demonstrations from a single user. We opted for asking each participant to do six complete demonstrations and three goal-only demonstra-



(a) Close the box (b) Pour

Fig. 6 The three poses of the objects for demonstrations for both skills overlaid

tions of each skill, which took around 30 min to complete. This was our target length for teaching sessions since longer sessions risk losing the participant's interest and could affect the quality of data.

5.2 Protocol

The data collection session begins by first giving the participant a brief verbal description of kinesthetic teaching, keyframes, speech commands and the interaction flow. Then the participant is asked to pose the arm in certain configurations in order to get them acquainted with kinesthetic teaching and moving the robot. Once the participant is familiar with kinesthetic teaching, we move on to the practice keyframe demonstration of the touch skill, followed by doing the same skill as a goal-only demonstration. During the practice skill, we help the user by showing how to achieve a certain pose or by pointing out the right speech command (see Table 1). This assistance is not given during the evaluation skills. The introduction and practice takes 5–8 min. The participant stands to the right of the robot during the kinesthetic demonstrations (see Fig. 1), and is positioned on the opposite side of the table during goal-only demonstrations (see Fig. 3).

After the practice session, we collect data for the two evaluation skills, counterbalancing to control for order effects. Before each skill, we explain the skill with similar words as they are described in Sect. 5.1, without explicitly showing it with the robot, so as not to bias their motion input. For each skill, there are three initial poses of the reference objects. These poses can be seen in Fig. 6. For each pose of the object, the participant is first asked to show a goal-only demonstration and then to provide two kinesthetic demonstrations. The objects are placed such that the same point of view for the user is maintained for both of the demonstration modes.[7] This is repeated for the second skill, then the data collection session ends. The overall experiment takes results in 18 demonstrations ($2 \times (6 + 3)$) from each participant.

[7] They are mirrored since the participant is standing across the table in one and standing next to the robot in the other.

The reason we collect multiple demonstrations from different poses of the objects is to build a more general model of the action, as pointed out in Sect. 3. As a result, a direct playback of the user demonstrations is not always feasible to execute the skill. For example, the arm motion required for completing the close the box skill for the horizontal box position in Fig. 6 would be out of the robot's workspace if transformed for the vertical box position. Similarly for the pour skill, the demonstrations for the rotated bowl would not be applicable to the non-rotated bowl, unless we use our prior knowledge of rotation independence of the skill.

6 Evaluation

We first evaluate the models via cross-validation with the demonstration data. We do this analysis for both the action and the goal models in order to show the level of similarity of the demonstrations in the two different feature spaces. However, cross-validation is just a first step to assess model performance. The real aim of the goal models is to provide information about the success of the skill execution, and the aim of the action model is to produce successful executions of the skill. To this end, we evaluate both models in a series of robot trials. We evaluate the action models by running the generated trajectories on the robot and we evaluate the goal models on recognition accuracy of the success/failure of these executed actions.

Throughout the analysis, we fix $\log(\tau_s) = -500$ for the goal models and $\log(\tau_s) = -1000$ for the action models. The threshold for the goal model is chosen such that there is correct classification of both successful and unsuccessful trials, based on cross-validation results. The threshold for the action model is chosen based on the distinct cutoff of likelihood estimations; anything below the selected threshold was too low (e.g. at the smallest floating-point value). We threw out 1 demonstration of participant 4 and 1 of participant 8 for the pour skill due to the object being fully occluded in one of their keyframes.

6.1 Cross-validation on demonstration data

6.1.1 Aggregate models

Our first analysis is designed to show the between-participant similarity in goal versus action demonstrations. We use a modified k-fold cross-validation. Instead of randomly dividing the data, we leave one participant's demonstration set out as test data and train a single aggregate model with all of the other participants' demonstration data together. Since there is more goal data than action data, due to goal-only demonstrations, we ran the same analysis twice for the goal models; once for all the demonstrations and once with only kinesthetic demonstrations (removing the goal-only demonstrations).

- *Goal model recognition accuracy* The average results for all the users is 100 % correct recognition for both the close the box skill and the pour skill. This shows that the users' demonstration were overall similar to each others'. We get 100 % correct recognition for both of the skills when we remove the goal-only demonstrations as well.
- *Action model recognition accuracy* The average recognition accuracy of the action models across all users is also high, 89.6 % for the close the box and 97.5 % for the pour skill. These results suggest that the demonstrations are consistent overall with a relatively large set of data.

Table 2 The cross-validation results for the goal model

	Close the box		Pour	
	All	Reduced	All	Reduced
(a) 1 vs 7: trained with 1 user, tested against 7				
1	0	0	76.2	30.1
2	88.9	71.4	60.3	45.2
3	58.7	38.1	71.4	83.3
4	98.4	95.2	95.2	47.6
5	88.9	85.7	81.0	73.8
6	98.4	92.9	85.7	71.4
7	96.8	100.0	71.4	61.9
8	63.5	42.9	60.3	71.4
Avg.	74.2	65.8	75.2	60.7
(b) Single user: cross-validation with the 8–9 demonstrations for a single user				
1	100.0	100.0	88.9	66.7
2	88.9	83.3	100.0	66.7
3	100.0	100.0	100.0	100.0
4	88.9	100.0	75.0	40.0
5	100.0	100.0	100.0	100.0
6	77.8	100.0	100.0	100.0
7	88.9	100.0	88.9	100.0
8	100.0	66.7	75.0	40.0
Avg.	93.6	93.8	91.0	76.7

Avg. refers to the average results. The columns under "All" refers to data including goal-only demonstrations and "Reduced" refers to data from only kinesthetic demonstrations

6.1.2 Between participants

Next we evaluate the generality of each individual participant's model, training with a single participant's data and using the other seven participants' demonstrations as test data. This analyzes the ability of the model built from one participant's data to generalize to other participants' data. The action models performed very poorly in this task, even though they performed well with the aggregate data. As a result we do not include them in the analysis.

These results are shown in Table 2a. For the close the box skill, apart from participant 1, all the other participants had better than chance goal recognition performance and participants 2, 4, 5, 6 and 7 had very good performance.[8] We get somewhat lower recognition performance for the pour skill but all participants did better than chance with participants 5, 6 and 7 had very good performance. These results imply that our participants provided perceptually similar demonstrations. As seen in the table, some participants (4, 5, 6) provided quite general demonstrations (i.e., good variance) across both skills (higher than 80 % accuracy); and the average recognition accuracy for all the participants was 74.2 % for close the box, and 75.2 % for pour. We believe these results are quite good considering the low number of training data in this analysis (1 participant = 9 demonstrations) and the high dimensionality of the feature space of the goal model. The results without the goal-only demonstrations have slightly lower success rates apart from participants 1 and 4 for the pour skill.This is because the kinesthetic demonstrations tend to have have more occlusions due to having both the user's hand and the robot's end-effector over the target bowl in the frame of the sensor, resulting in worse data for the pour skill.

6.1.3 Within participants

Lastly, we evaluate within-participant recognition performance by applying leave-one-out cross-validation for each participant's demonstration set. These results are seen in Table 2b. The recognition results for the goal models are similar across both skills and as expected are better than the generalized 1 versus 7 task, with an average 93.6 % accuracy for close the box and 91.0 % for pour. The results without the goal-only demonstrations are the very similar for the close

[8] Participant 1 has only provided between 2 or 3 keyframes per demonstration whereas other participants provided 4–6. As a result, participant 1s goal model was not able to recognize the demonstrations of other users.

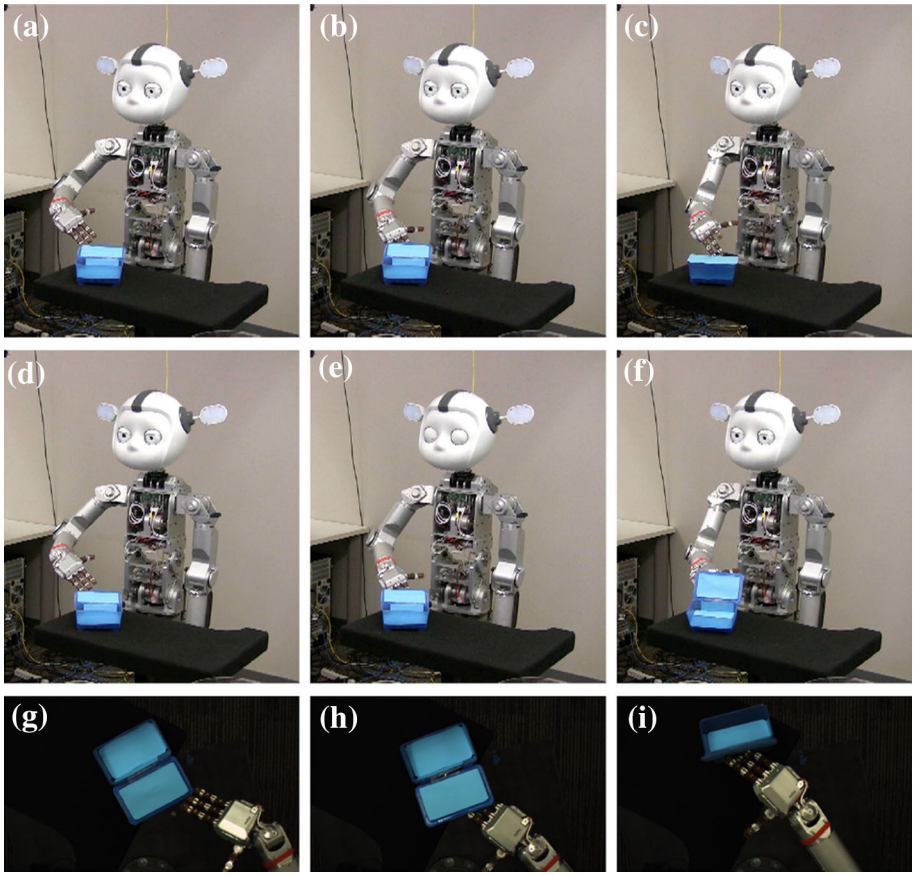


Fig. 7 Image snapshots for a close the box execution. The *first row* shows a successful execution and the *second row* shows a failed one. In the failure case, the robot's fingers got stuck to the body of the box during, as shown in the *last row*. As a result it tilted the box, instead of closing it

the box skill but differ for some participants in the pour skill. The reason is same as before, more occlusions when both the users' hands and robot's end-effector occluding the object.

The action models were not successful in this cross-validation recognition task as well. This was somewhat expected in the between-participant case, due to a wide range of possibilities to demonstrate the skill and user differences. However, the within-participant results show that the number of demonstrations we obtained is not enough to model the variance and/or the different ways to execute the skill. This does not imply that we cannot learn good action models but imply that the demonstrations are diverse.

6.1.4 Discussion

These three sets of results show that demonstrations are similar and consistent both between and within the users in the goal space, and even 9 demonstrations per user is enough to learn accurate goal models for these skills. We also show high recognition rates without the goal-only demonstrations (i.e. 6 demonstrations). However, more action data was need to span the state space with varied examples. These results are expected given that (1) naïve users are goal oriented and (2) there are many ways to accomplish the same skill in the action space but all of these will look similar in the goal space.

6.2 Robot trials: skill execution

One aim of learning both models is to be able to execute the skill on the robot and know whether the execution succeeded or not. This is arguably the main purpose of LfD, making use of the learned models in practice. The individual learned action models had very low recognition performance in the cross-validation tests (both within- and between-participants), but low cross-validation scores do not imply that the resulting action models are useless, a more fair analysis of the action models is evaluating their success at generating motion.[9] We perform this analysis by executing the skills with each of the learned action models, using Algorithm 1. An example of executing a learned action model for the close the box skill is depicted in Fig. 7.

We tested the learned action model for each skill for the 8 individual participants. We used each model to generate robot motion, executing it 5 times and noting the success

[9] The cross-validation tests how similar the demonstrations are but not how the action itself is modelled.

Table 3 Skill execution and monitoring results

	Close the box			Pour		
	Execution success (%)	Monitoring results		Execution success (%)	Monitoring results	
		True Pos:Neg	False Pos:Neg		True Pos:Neg	False Pos:Neg
1	100	5:0	0:0	100	5:0	0:0
2	0	0:5	0:0	100	5:0	0:0
3	60	2:2	0:1	0	0:3	2:0
4	40	2:3	0:0	100	4:0	0:1
5	40	2:1	2:0	100	5:0	0:0
6	100	5:0	0:0	100	5:0	0:0
7	40	2:2	1:0	0	0:4	1:0
8	80	4:1	0:0	100	5:0	0:0
Average	57.5	22:14	3:1	75	27:9	3:1

or failure as the ground truth. We regarded the fully closed box as successful for the box skill and for the pour skill, we regarded it successful if the robot was able to pour most of the coffee beans to the bowl (i.e., a bean or two bouncing out was still called success). These results are seen in Table 3, under the *Execution Success* columns. There is a wide range of success rates across participants. Six of the eight participants achieved a 100 % success rate (5/5) with at least one of their skills, and two people did so for both. Whereas three people had one of their action models with a 0 % success rate. In general the pour skill was more successful, with a 75 % success rate across all participants, compared to the close the box skill with a 57.5 % aggregate success rate.

There were a few common modes of failure. For the close the box skill, the fingers sometimes touched too lightly to the lid and lost contact. This is exacerbated by the highly complaint fingers of the robot as they bend slightly with touch. Another case was when the fingers got stuck on the body of the box and tilt it instead of closing it, as shown in the bottom row of Fig. 7. This happens due to user demonstrations; not enough clearance is demonstrated when going from under the lid to over the box.[10] For the pour case, the common mode of failure was having not enough downward rotation to pour the entire cup.

6.3 Robot trials: skill monitoring

While executing the action models, the robot extracts an observation frame in the goal space at each action keyframe of the skill, as described in Sect. 4.7. We then provide each of these observation sequences as input to the goal model for the corresponding participant, as described in Sect. 4.7. These results are seen in Table 3, under the *Monitoring Results* columns. This table shows the correctly recognized execution outcomes (true positives and true negatives) and the mistakes (false positives and false negatives) for each participant's goal models across both skills. The recognition accuracy at monitoring is good for both of the skills. Looking at the overall rates, only 4 of the the 40 trials was incorrectly classified across both skills. Thus, a 90 % success rate in the goal monitoring task for each skill. The exciting result is that even for participants with low execution success, their goal model is good. This is a result we expected based on the successful cross-validation results, but is reassuring to see that the goal models perform well on new data observed when the robot is executing the learned action models.

[10] In an interactive scenario, the teacher might realize this and fix it with their follow-up demonstrations.

7 Discussion

7.1 System requirements

In this work, we propose the LfD system in Fig. 2 and provide a single implementation of the components. There could be other implementations. In this section, we discuss the requirements of the components and talk about other potential implementations in relation to existing work.

The information to learn action models and goal models need to be available during a demonstration. In our current implementation, we use kinesthetic teaching to get motion data. Teleoperation is an option to get direct access to this data. Another option is to observe the teacher doing the skill and then use a re-targeting function to map it onto the robot, which is an example of indirect access to motion data. For the object data, we use a RGBD camera. Other sensors can be used instead or in addition, such as 2D cameras (e.g. Akgun and Thomaz 2013) and force/torque sensors. Even though we are not using any such method, feature selection methods can

be employed on top of the resulting data given that the goal space has high number of dimensions. Currently, the method assumes that there is a single object of attention which is known beforehand.

Keyframes play an important role in our implementation. Keyframes help users highlight important parts of a skill, but they are not the reason to decouple action and goal learning. Trajectories can also be used with the proposed method, as a tractable way that can overcome difficulties associated with naïve teacher input is used. In the current work, it seems that keyframes help the goal models (the salient points argument) more so than the action models.

As mentioned in Sect. 3, we use the same learning algorithm to learn two different models but it is possible to use different algorithms for each. The exact skill modeling and learning method is not of key importance as long as some requirements for the models are satisfied.

The requirement on the action model is that it needs to be generative, in the sense that it should be able to generate motion related information to execute the skill. The methods mentioned in Sect. 2.2 are all candidates for action learning. After learning, the skill is executed by either using the action model as a policy or using it to generate a trajectory and following it with some other means.

The requirement on the resulting goal model is that it needs to be suitable for classification to be used in monitoring the skill execution. Given our LfD scenario, we have access to only successful demonstrations. With only positive examples, the monitoring task becomes a one-class classification problem; the classifier is trained only on positive labels. Any method that can model the probability distribution of the object data can be used with likelihood calculation given observations and thresholding or hypothesis testing. In this paper, we have used HMMs. Conditional Random Fields is another option. Other alternatives include nearest neighbors and one-class SVMs, but applying these to sequential data of arbitrary length may not be straightforward.

7.2 Future work

The results set up interesting future work. The current use of goal models is to monitor the executed action and try again if failed. A next step is to use the prior state information to decide if the skill is actually executable, treating this information to judge whether the pre-conditions are met.

A future use of the learned goal models in this work is to utilize them as an objective in a policy search to improve the learned action models. In Sect. 2.3, we mentioned that IRL is similar in spirit to our approach; learning goals is similar to learning rewards. However, there are some problems to overcome. Keyframes provide sparse rewards which is not typical for IRL. The dimensionality of our goal space is prohibitive for some of the existing IRL methods to be used in interaction time.[11] The IRL idea was developed with expert demonstrators[12] in mind. The naive user input to IRL methods has been considered but, suitable methods for a generic interaction have not been explored. Future work would look at how to overcome these in an IRL sense or utilize the goal models in other ways to do policy search. An example to the latter would be to execute sampled trajectories from the action model and use them to update the action model if deemed successful by the goal models.

Another alternative is to do reward shaping to alleviate some of the problems with sparsity and delay in the context of RL. This would require the teacher to be in the loop during skill improvement to provide feedback for reward shaping, since this would need to be skill specific. Having the teacher in the loop, at least initially, would be useful to both refine the action models and the goal models.

Policy search and reinforcement learning in general can be costly and unsafe for robots. Safe exploration is an important area of future research. Goal models can be utilized online to detect early failures in exploration to mitigate some of these problems.

The user preference on the types of demonstrations (kinesthetic vs goal-only) is an important point to consider to develop future LfD interactions. We did not have a post experiment interview or a survey in this work but consider this to be a useful future addition.

Our aim is to build a generic LfD approach. Towards this end, several parts of the current approach needs to be improved/augmented. These include generalizing to skills that have multiple objects of interest, selecting reference frames for motion data, allowing action models for bi-manual manipulation and adding mechanisms to handle cyclic behaviors.[13]

8 Conclusion

Our aim is to build LfD techniques for personal robots that will have an everyday person as their end-user. In this paper we address the goal-oriented nature of human teachers, with the expectation that people will be more focused on successfully demonstrating task goals than particular motor trajectories.

We developed a novel framework for LfD with naïve users, learning task level goals and motor level actions simultaneously, but maintaining them as separate learning problems.

[11] Fast enough to have a fluid interaction with the user.

[12] Expert in the sense of demonstrations, not necessarily the underlying algorithms.

[13] We can represent cyclic behaviors with the current action model but currently have no means to decide on when to stop the cycle, see Sect. 4.6.

Explicitly separating the learning problem into these two spaces leverages the fact that human demonstrators are going to be goal-directed, and good at showing *what to do*, while only a subset of those teachers may also focus on showing the robot good demonstrations of *how to do it*. We use a motion space to learn action models and a generic perceptual feature space to learn the goal model. Our framework is outlined in Fig. 2.

We collected data from eight naïve users for 2 skills to evaluate our approach. We have seen that the skill demonstrations are more consistent in the goal space, both across users and within users. This confirms our observation about the goal-oriented nature of naïve users. Some users were not able to teach successful action models with the average success rates being 57.5 % for the close the box skill and 75 % for the pour skill. Successful goal models can be learned from all the users, even for users with less successful/failed action models. The average execution monitoring success rate was 90 %.

This sets up interesting future work. We see the potential in using the goal models to facilitate self-learning. We also envision several scenarios in which the goal model can be used to decide when to prompt the teacher for particular input to further refine the learned models.

Acknowledgments This work has been supported by US National Science Foundation CAREER award #0953181, and by US Office of Naval Research grant #N000141410120.

References

Abbeel, P., & Ng, A. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)* (pp. 1–8).

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*(13), 1608–1639.

Akgun, B., & Thomaz, A. (2013). Learning constraints with keyframes. In *Robotics: Science and Systems: Workshop on Robot Manipulation*.

Akgun, B., Cakmak, M., Jiang, K., & Thomaz, L. A. (2012a). Keyframe-based learning from demonstration. *International Journal of Social Robotics*, *4*(4), 343–355. doi:10.1007s12369-012-0160-0

Akgun, B., Cakmak, M., Wook Yoo, J., & Thomaz, LA. (2012b). Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (pp. 391–398).

Akgun, B., Subramanian, K., & Thomaz, A. (2012c). Novel interaction strategies for learning from teleoperation. In *AAAI Fall Symposia 2012, Robots Learning Interactively from Human Teachers*.

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

Atkeson, CG., & Schaal, S. (1997). Robot learning from demonstration. In *Proceedings of 14th International Conference on Machine Learning, Morgan Kaufmann* (pp. 12–20).

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*, 164–171.

Bitzer, S., Howard, M., & Vijayakumar, S. (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3219–3225).

Cakmak, M. (2012). Guided teaching interactions with robots: Embodied queries and teaching heuristics. PhD thesis, Georgia Institute of Technology.

Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B Special Issue on Robot Learning by Observation, Demonstration and Imitation*, *37*(2), 286–298.

Chao, C., Cakmak, M., & Thomaz, A. (2011). Towards grounding concepts for transfer in goal learning from demonstration. In *Proceedings of the Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob), IEEE*, vol. 2 (pp. 1–6).

Chernova, S., & Thomaz, A. L. (2014). *Robot learning from human teachers*. San Rafael, CA: Morgan & Claypool Publishers.

Csibra, G. (2003). Teleological and referential understanding of action in infancy. *Philosophical Transactions of the Royal Society of London*, *358*, 447–458.

Dantam, N., Essa, I., & Stilman, M. (2012). Linguistic transfer of human assembly tasks to robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

Hovland, G., Sikka P., & McCarragher, B. (1996). Skill acquisition from human demonstration using a hidden Markov model. In *1996 IEEE International Conference on Robotics and Automation*, vol 3, (pp. 2706–2711). IEEE

Hsiao K., & Lozano-Perez, T. (2006). Imitation learning of whole-body grasps. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5657–5662). IEEE

Jäkel, R., Schmidt-Rohr, S. R., Rühl, S. W., Kasper, A., Xue, Z., & Dillmann, R. (2012). Learning of planning models for dexterous manipulation based on human demonstrations. *International Journal of Social Robotics, Special Issue on Robot Learning from Demonstration*, *4*, 437–448.

Jenkins, O., Mataric, M., Weber, S., et al. (2000). Primitive-based movement classification for humanoid imitation. In *Proceedings of 1st IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*.

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable non-linear dynamical systems with Gaussian mixture models. *IEEE Transaction on Robotics*, *27*, 943–957.

Kormushev, P., Calinon, S., & Caldwell, D.G. (2010). Robot motor skill coordination with em-based reinforcement learning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Kulić, D., Ott, C., Lee, D., Ishikawa, J., & Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, *31*(3), 330–345.

Levas, A., & Selfridge, M. (1984). A user-friendly high-level robot teaching system. In *Proceedings of the IEEE International Conference on Robotics, Atlanta, Georgia* (pp. 413–416).

Meltzoff, A. N., & Decety, J. (2003). What imitation tells us about social cognition: A rapprochement between developmental psychology and cognitive neuroscience. *Philosophical Transactions of the Royal Society of London*, *358*, 491–500. doi:10.1098/rstb.2002.1261.

Miyamoto, H., Schaal, S., Gandolfo, F., Gomi, H., Koike, Y., Osu, R., et al. (1996). A kendama learning robot based on bi-directional theory. *Neural Networks*, *9*, 1281–1302.

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, *32*(3), 263–279.

Nicolescu, M.N., & Matarić, M.J. (2003). Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the 2nd International Conference on AAMAS*. Melbourne, Australia.

Niekum, S., Chitta, S., Marthi, B., Osentoski, S., & Barto, A.G. (2013). Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems, 9*.

Niekum, S., Osentoski, S., Konidaris, G.D., Chitta, S., Marthi, B., & Barto, A. G. (2015). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, *34*(2), 131–157.

Parent, R. (2002). *Computer animation: Algorithms and techniques. Morgan Kaufmann series in computer graphics and geometric modeling*. San Francisco: Morgan Kaufmann.

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

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

Ratliff, N., Ziebart, B., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., & Srinivasa, S. (2009). Inverse optimal heuristic control for imitation learning. In *Proceedings of AISTATS* (pp. 424–431).

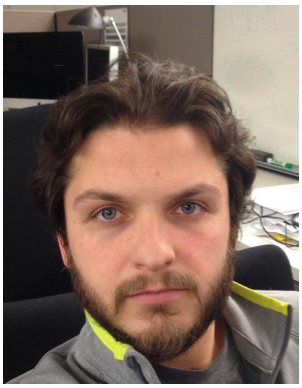
Rusu, RB., Bradski, G., Thibaux, R., & Hsu, J. (2010). Fast 3D recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan.

Suay, H. B., Toris, R., & Chernova, S. (2012). A practical comparison of three robot learning from demonstration algorithms. *Journal of Social Robotics, Special Issue on Robot Learning from Demonstration*, *4*, 319–330.

Thomaz, A. L., & Breazeal, C. (2008a). Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers. *Connection Science, Special Issue on Social Learning in Embodied Agents*, *20*, 91–110.

Thomaz, A. L., & Breazeal, C. (2008b). Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence Journal*, *172*, 716–737.

Trevor, A.J.B., Gedikli, S., Rusu, R.B., & Christensen, H.I. (2013). Efficient organized point cloud segmentation with connected components. In *3rd Workshop on Semantic Perception, Mapping and Exploration (SPME)*. Karlsruhe, Germany.



Baris Akgun is a Ph.D. candidate in Robotics at the Georgia Institute of Technology. He received his B.Sc. degree in Mechanical Engineering and an M.Sc. degree in Computer Engineering from the Middle East Technical University, Turkey in 2007 and 2010 respectively. His research interests include human–robot interaction, learning from demonstration and machine-learning for robotics.

Andrea Thomaz received a B.Sc. degree from the University of Texas, Austin, in 1999, and M.Sc. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 2002 and 2006, respectively. She is an Associate Professor in the School of Interactive Computing, Georgia Institute of Technology, where she directs the Socially Intelligent Machines Lab. Her work focuses on social robots and socially guided machine learning. She is published in robotics, human–robot interaction, and artificial intelligence.