Autonomous Robotics manuscript No.
(will be inserted by the editor)

Simultaneous Learning of Hierarchy and Primitives (SLHAP) for Complex Robot Tasks

Anahita Mohseni-Kabir · Changshuo Li · Victoria Wu · Daniel Miller · Benjamin Hylak · Sonia Chernova · Dmitry Berenson · Candace Sidner · Charles Rich

Received: date / Accepted: date

Abstract We have developed a new interaction paradigm for robot learning from demonstration, called simultaneous learning of hierarchy and primitives (SLHAP), in which information about hierarchy and primitives is naturally interleaved in a single, coherent demonstration session. A key innovation in the new paradigm is the human demonstrator's narration of primitives as he executes them. Hierarchy is represented using hierarchical task networks and motion planning constraints on the primitives are represented using task space regions. We implemented a proof-of-concept system and produced a video illustrating a typical interaction. The underlying algorithms which make SLHAP possible are described and evaluated.

Keywords learning from demonstration · hierarchical task network · task space region · motion planning · tire rotation

1 Introduction

This work introduces a new interaction paradigm for robot learning from demonstration (LfD), called simultaneous learning of hierarchy and primitives (SLHAP), which addresses an important gap in current LfD methods. Generally speaking, LfD research has been divided into two subproblems (Chernova and Thomaz 2014): learning low-level motion trajectories and learning high-level tasks. With few exceptions,[1] researchers have focused on one or the other subproblem, with the resulting lack of natural interaction scenarios for when *both* low-level and the high-level task knowledge needs to be learned.

Also, to position our work another way within LfD research, we are focused on interactive learning from human teachers, which means that only one or a few demonstrations are practical. Many LfD methods rely on large numbers of demonstrations.

1.1 Example Task Domain

As a concrete example, consider Fig. 1, which depicts the high-level task of tire rotation as a hierarchical composition of primitives, such as picking up a nut, screwing it on, etc. All of the technical discussions in the following sections, as well as the accompanying video, share this example task.

Based on current LfD research, teaching a task such as this to a robot with no prior knowledge of either the primitives or the hierarchy would require separately demonstrating each of the primitives, and then demonstrating how they are composed. This is not, however, how you would expect to teach tire rotation to another human. Instead, as in the accompanying video, you would demonstrate the entire high-level task once from beginning to end, interleaving learning the task hierarchy with learning the primitives.

Notice in Fig. 1 that, for our purposes, tire rotation involves four primitives manipulating nuts: PickUpNut (PUN), PutDownNut (PDN), Screw (S) and Unscrew (US); and four primitives manipulating tires: PickUpTire (PUT), PutDownTire (PDT), Hang (H) and Unhang (UH). Each of these primitives occurs multiple times in tire rotation. A tire goes on a

This work is supported in part by the Office of Naval Research under grant N00014-13-1-0735.

Charles Rich
Worcester Polytechnic Institute
100 Institute Rd, Worcester, MA 01609
Tel.: +1-508-831-5945
E-mail: rich@wpi.edu

[1] Niekum et al (2015) learn both low-level motion trajectories and high-level tasks (as state machines) from demonstration, but the high-level tasks are not hierarchical.

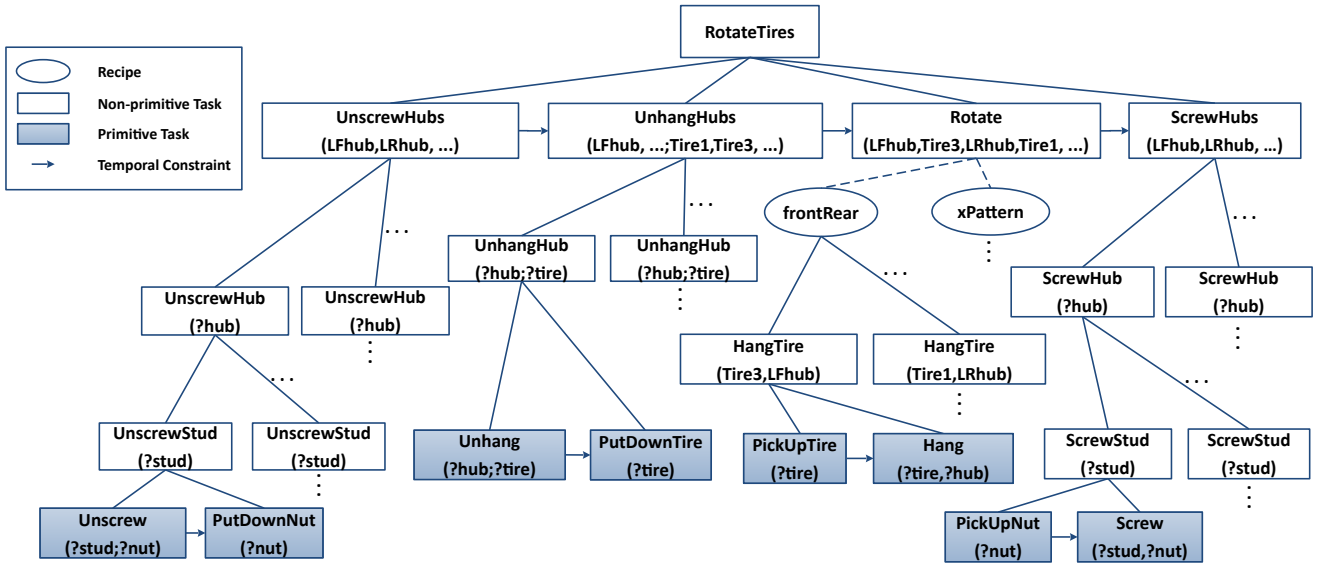


Fig. 1 Hierarchical task network for tire rotation.

hub which has three *studs*, each of which receives a nut. In the simulated environment evaluation described in Section 3, there are four hubs and four tires. The proof-of-concept system described in Section 2 uses a physical mockup with two hubs and one tire, in which the studs are simple unthreaded posts onto which a nut slides.

The formalism in Fig. 1 is called a hierarchical task network (HTN), and is widely used in artificial intelligence (Erol et al 1994). An HTN is a tree in which the fringe nodes denote *primitive* tasks, i.e., actions that can be directly executed by the robot, and the other nodes denote *non-primitive* tasks, which must be decomposed in order to be executed. Both primitive and non-primitives may also have parameters, corresponding to inputs and outputs.

Hierarchy is important because humans deal with complex tasks by breaking them down into subtasks. Furthermore, the intermediate-level non-primitives are often reusable in other situations. For example, unscrewing all the nuts on a hub is one of the steps in changing a flat tire. The non-primitives also provide a vocabulary for communication between the human and robot if they are performing the task collaboratively, for example, to discuss delegation or the completion status of a subtask.

Section 2 presents our proof-of-concept system implementing SLHAP, which produced the accompanying video. Sections 3, 4 and 5 describe the underlying algorithms which makes SLHAP feasible. Each of these sections contains a review of related work and an evaluation. Section 6 concludes and briefly discusses future work.

2 Proof-of-Concept System

The left half of Fig. 2 shows the architecture of our SLHAP proof-of-concept system, including section numbers indicating where each component is described in detail. The photograph in the upper-left corner represents the human-robot interaction, which takes place inside a Vicon motion capture cell in which the realtime poses of the human's head[2] and right hand and the mock-up tire are recorded. The full geometry of the environment (location of hubs and studs) is also known and used in the task learning process. The robot employs text-to-speech to communicate with the human, and the human replies via speech recognition. The simultaneous nature of the learning process is reflected in the two large blue arrows indicating information flowing between the human-robot interaction and the components for learning hierarchy and learning primitives.

The right half of Fig. 2 is a partial transcript of a typical SLHAP human-robot interaction. Basically, the human performs the primitive steps of the overall task in order starting at the beginning, with the robot observing his motions. The key innovation in our new paradigm is for the human to *narrate* primitives as he performs them, as illustrated in turns ① and ⑥. At various points, for example ②, ⑤ and ⑦, the robot interrupts the human to ask for information that will help it learn a hierarchical task model (this kind of behavior is called active learning (Cakmak et al 2010)). The rightmost column in Fig. 2 highlights what the underlying algorithms are learning at various points in the interaction, but from

[2] The human's head pose is only used to control where the robot "looks"; it is not part of the task learning process.

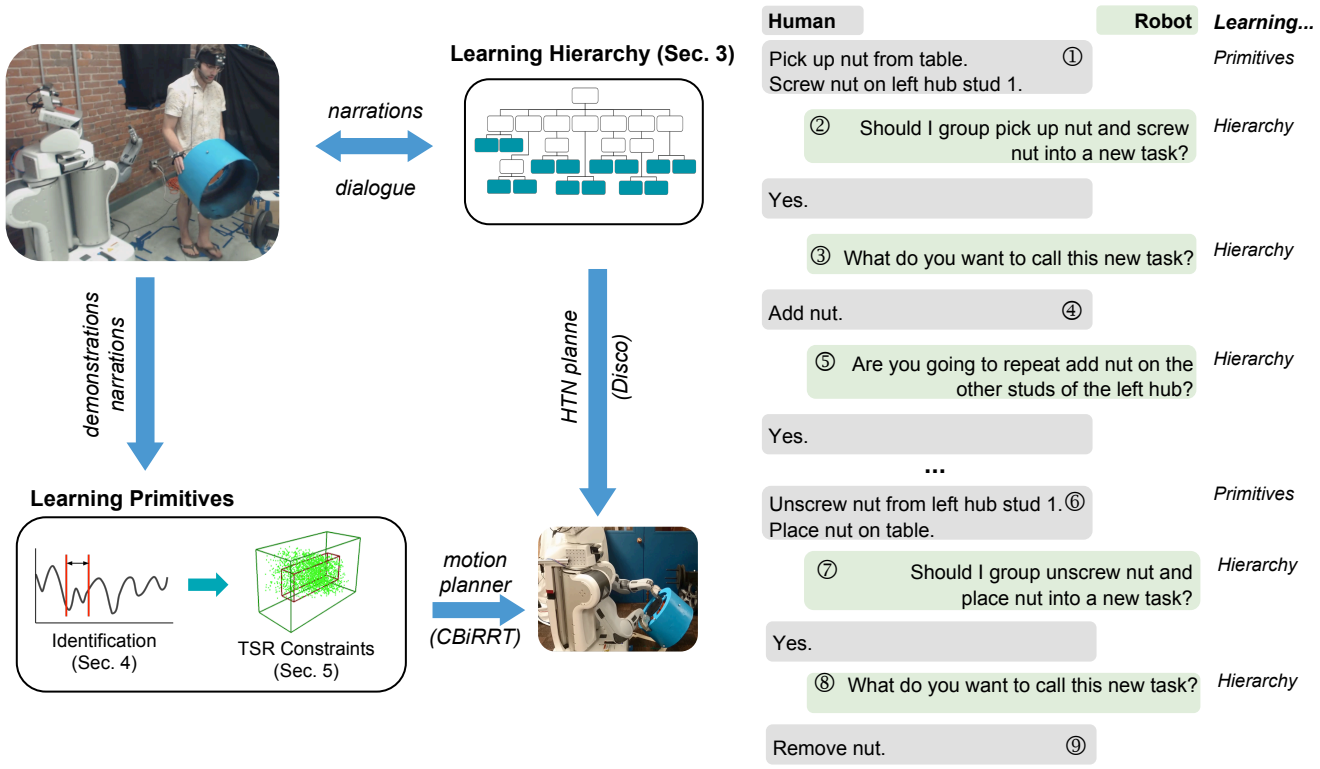


Fig. 2 Architecture of SLHAP proof-of-concept system and partial transcript of interaction in accompanying video.

the human's point of view, the whole interaction flows as a natural dialogue with an active learner.

Not shown here, but in the video, the human can also command the robot to execute an already learned non-primitive, as long as the motion learning for all of the constituent primitives has been completed. This can happen even before the entire task has been learned, which can be useful for collaboration. For example, in the video, rather than adding three nuts for the second time, the human says to the robot, "Please add three nuts to the right hub." As the robot executes this task, it narrates its own actions, both the primitives and non-primitives, to exhibit that it has learned not only the sequence of constituent primitives, but also the hierarchy of this task.

Returning to the architecture diagram in Fig. 2, the learning hierarchy component uses the symbolic information in the narrations (the name of the primitive and its parameters) to start building the fringe of an HTN. The non-primitive nodes of the HTN result from human-robot dialogue, wherein the system asks grouping questions, e.g., ② and ⑦, and the human provides the names of non-primitives, e.g., ④ and ⑨. Questions such as ⑤ result in non-primitive nodes grouping repeated subtasks.

The learning primitives component uses the timing of the narrations synchronized with the motion trajectory data of the demonstrations to, first, identify the section of the motion trajectory that is most representative of each primitive, and then to learn from this trajectory the reusable motion planning constraints for the primitive, expressed in the task space region (TSR) formalism. The TSR learning step solves the problem of retargeting the motion from the human to the robot.

The photograph in the lower right corner of the architecture represents the robot autonomously performing the task it has learned, which entails interleaving HTN planning, implemented using Disco (Rich and Sidner 2012), to sequence through the primitives and motion planning, implemented using CBiRRT (Berenson et al 2011), to apply the TSR constraints for each primitive.

2.1 Limitations of Current Implementation

The system implemented in the accompanying video is a proof of concept, with the main limitations enumerated below. None of these invalidate the basic feasibility of the SLHAP paradigm.

- Speech recognition and understanding is not general-purpose; we use a push-to-talk button operated offscreen and a predefined grammar for the human utterances.
- Movement of the robot base is not autonomous; base is controlled by offscreen joystick.

– Learning primitives is not real-time; the primitives that the robot executed in the video were previously learned offline from similar motion capture data.
– The TSR's for four of the eight primitives were not learned; however, these primitives (picking up and putting down a tire or a nut) do not need TSR learning, because they are only constrained at their endpoints.

In Section 6, we discuss the challenges of overcoming these limitations in future work.

3 Learning Hierarchy

The inputs to the learning hierarchy component in Fig. 2 are the human's narration of primitives and answers to the system's questions. The outputs of the component are the system's questions and an HTN representing the learned task hierarchy.

3.1 Related Work

There is extensive research on learning from demonstration (Argall et al 2009), hierarchical task learning (Garland et al 2001), interactive task learning (Cakmak and Thomaz 2012; Hayes and Scassellati 2014), and learning from a single task iteration via instructions (Huffman and Laird 1995; Mohan and Laird 2011). The closest work to ours is by Rybski et al (2007), who developed an algorithm that combines spoken language understanding, dialogue, and physical demonstration to learn complex tasks. Their task representation allows non-primitives to be constructed from simpler tasks in order to create hierarchical structures. Their robot also verifies the HTN with the human by asking questions and allowing the human to add additional conditionals. However, their approach differs from ours in that it does not leverage ordering or data flow constraints (see below). Furthermore, the robot's questions are limited to filling in missing conditionals, while our questions are suggestions are aimed at improving the task quality and teaching efficiency (see evaluation below).

3.2 Heuristics for Grouping

The key algorithmic content in this component is two domain-independent heuristics for creating new non-primitives in an HTN by grouping (primitive and/or non-primitive) tasks. Because these are heuristics, they are not applied automatically, but instead used to generate questions/suggestions to the human, such as ② and ⑦ in Fig. 2. If the human answers yes (accepts the suggestion), then the grouping is done and the human is asked to provide a name for the new non-primitive (see ③ and ⑧).

The first heuristic, called the *data flow* heuristic, suggests grouping two consecutive tasks whenever an output of the first task is an input of the second task and the robot is *holding* the object in question between the two tasks. Question ② is generated by this heuristic, because the nut that is the output[3] of "pick up" is the same as the input to "screw," and the robot is holding the nut between the two actions.

The second heuristic, called the *parts* heuristic, relies on knowing something about the structure of the environment. Specifically, this heuristic suggests that, when an object has multiple parts of the same type, such as the three studs of a hub (or the four hubs of a car), when an action is applied to one part, it is likely to be repeated on the other parts. It is obviously easy to come up with counterexamples, but the cost of asking is relatively low. Question ⑤ is generated by this heuristic, because the human has just performed "add nut" on one of the studs of a hub.

3.3 Evaluation

In order to investigate the value of the two heuristics above, we conducted a between-participants study with 32 participants in two conditions: *No-Suggestions* (15), in which the robot provided no suggestions; and *Suggestions* (17), in which suggestions based on either heuristic were made when appropriate. The study was conducted in a simulated environment in which the participants were asked to teach a robot how to rotate the tires on car with four hubs and tires. All interaction was via a graphical user interface (GUI), identical in both conditions.

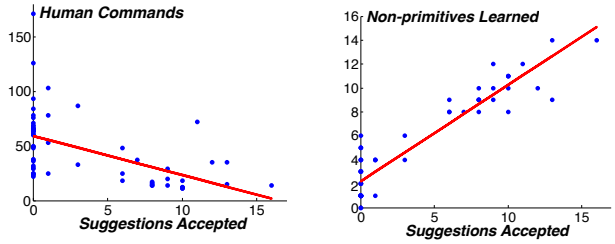
In order to develop familiarity with the GUI and the overall interaction style, each participant started with a training activity involving teaching a robot how to build a tower of blocks. The training process did not include any suggestions from the system, to avoid biasing participants in the No-Suggestions condition.

After training, participants performed the main study activity in which they were asked to teach the robot how to rotate tires. They were given a picture of the desired final locations of the tires and detailed written descriptions of the eight primitives described in Section 1.1, but no step-by-step instructions on how to perform task. The only difference between the two conditions was whether or not grouping and/or repetition suggestions were provided.

3.3.1 Measures and Analysis

We used the following three objective measures to evaluate participant performance:

[3] The output of a task is any object whose properties, such as location, are changed by the task.



(a) Teaching effort (b) Task quality

Fig. 3 Accepting more suggestions decreases teaching effort and increases task quality.

- *Teaching Effort:* The effort expended by the teacher, measured as the number of commands (for executing both primitives and non-primitives) made by the human during the interaction.
- *Task Quality:* The quality of the learned tire rotation HTN, measured as the number of non-primitives in the HTN (considering only HTNs which correctly achieve the specified final tire configuration). We chose this measure because hierarchy is a valuable property in complex tasks for both communication and reuse.
- *Teaching Efficiency:* The efficiency of the interaction, measured by dividing task quality by the teaching effort (using the definitions above).

3.3.2 Hypotheses and Results

We formed the following hypotheses about the effect of suggestions on the learning interaction.

- **Hypothesis 1:** Teaching effort will be less in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 2:** Task quality will be greater in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 3:** Teaching efficiency will be greater in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 4:** In the Suggestions condition, participants who accepted more suggestions will have (a) lower teaching effort and (b) higher task quality.

All four hypothesis were supported by our study.

Analysis of the number of commands showed that teaching effort was significantly higher in the No-Suggestions than in the Suggestions condition, supporting Hypothesis 1. To further study the impact of suggestions on teaching effort, Fig. 3(a) shows via linear regression how the number of commands decreases with the number of robot suggestions accepted by the human, supporting Hypothesis 4(a).

Analysis of the number of non-primitives learned shows that the task quality was significantly higher in the Suggestions than in the No-Suggestions condition, supporting Hypothesis 2. Additionally, Fig. 3(b) shows via linear regression how the number of non-primitives learned increases with the number of robot suggestions accepted by the human, supporting Hypothesis 4(b).

Analysis of the teaching efficiency measure provides insight into how teaching effort and task quality vary in combination between conditions. Teaching efficiency was is significantly higher in the Suggestions than in the No-Suggestions condition, supporting Hypothesis 3.

Further details about this component can be found in Mohseni-Kabir et al (2015).

4 Learning Primitives: Identification

This section and the next describe the two subcomponents in learning primitives (see Fig. 2), which deal only with primitives and sequences of primitives–no hierarchy or non-primitives.

The input to the first subcomponent, called *identification*, is the synchronized time series of demonstration motion data and narrations for the primitives, as shown in Fig. 4 (not including the S1 and S2 markings). The output of identification is, for each primitive, a single motion trajectory extracted from the input, which is provided to the second subcomponent, TSR constraints, described in Section 5.

The fundamental insight underlying the algorithms in this section is that the reusable[4] primitive trajectories in the overall motion data are separated by *transition* motions that depend on the context, in particular the preceding and following primitives. To give a concrete example, suppose that in the overall task, one occurrence of screwing a nut onto a stud is preceded by picking the nut up from the table, while another occurence is preceded by unscrewing the nut from another stud. The reusable trajectory for the unscrew

[4] Reusable by the human; retargeting the primitive for the robot is addressed by the TSR constraint learning subcomponent.

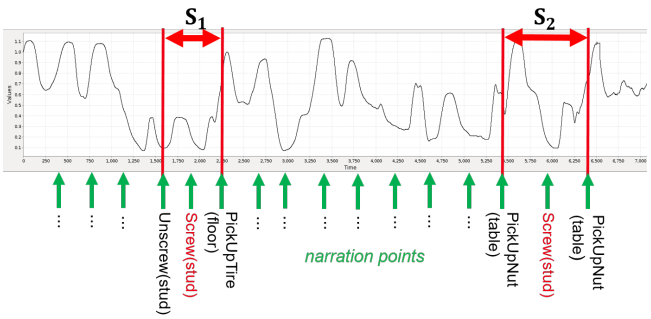


Fig. 4 Narration points superimposed on motion data for distance between right hand and stud with enclosing sections indicated for two occurrences of Screw primitive.

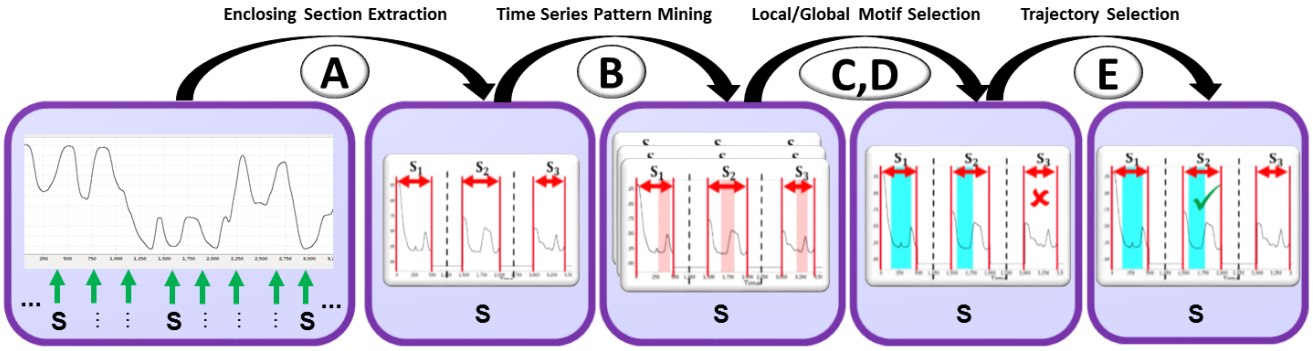


Fig. 5 Steps in the algorithm for identifying a single primitive.

primitive should contain only the motions that are consistent between these two occurrences.

Thus comparing demonstrations of the same primitive in different preceding and following contexts is fundamentally what enables the algorithms below to determine the boundaries of each primitive. Unfortunately, this also means that if a given primitive is always preceded/followed by the same primitive, it is impossible to identify the boundary between them. In this situation, there are two choices: the system can 1) consider the two co-occurring adjacent primitives as a single primitive until more demonstrations are provided, or 2) compute which context variations are missing and request one or more specific short supplementary demonstration sequences to provide the needed variations (see Section 4.3).

Trimming transition motions from the identified primitive trajectories is important not only because it removes useless motions, but also because it increases the reusability of the primitive in environments with different obstacles. When the learned primitives are executed in sequence by the robot, general-purpose motion planning is used to generate new transition motions dynamically, based on the context, including avoid obstacles that may not have been present during the demonstration.

4.1 Related Work

Other researchers, such as Kulic et al (2008) and Chiappa and Peters (2010), have explored time series segmentation using assumptions about the motion primitives. However, our approach is unique in leveraging the human's semantic knowledge of the task via narration to aid in the primitive identification process.

Identifying approximately recurrent unknown patterns in time series is known as the *motif* discovery problem in the data mining field, and has received a great deal of attention. Many systems (Oates 2002; Minnen et al 2007; Mohammad and Nishida 2015; Senin et al 2014) discover motifs of variable length. In this work, we use the GrammarViz motif discovery tool (Senin et al 2014) due to its efficiency, easy of use, and ability to find motifs of unknown length.

4.2 Steps in Identification Algorithm

Fig. 5 provides an overview of the identification algorithm, which is applied for each primitive used in the overall task.

(A) Enclosing Section ExtractioniňĄGG

The first step of our algorithm leverages narration to approximately identify the section of the motion data corresponding to each primitive. We do not expect the human to start talking exactly when he starts a primitive and stop talking exactly when the primitive is over. Therefore, for each primitive, we take the time between the end of the *previous* narration utterance and the beginning of the *next* narration utterance as a conservative first estimate of the section of the time series enclosing the narrated primitive. The sections labeled S1 and S2 in Fig. 4 are examples of this first extraction step for the Screw primitive in the given time series. These initial enclosing sections obviously contain transition motions—trimming them is the major goal of the remaining steps in the algorithm.

To obtain the output of step A, we then concatenate the enclosing sections for every occurence of the given primitive into a new synthetic time series with a "spacer" (constant value near zero) between each, as shown in Fig. 5.

(B) Time Series Pattern Mining

The next step of the algorithm uses GrammarViz to discover recurrent patterns (motifs). Since GrammarViz processes only one-dimensional data, we cannot give it the full 3D trajectory of the manipulator (human's right hand) and the manipulated objects. Thus far, we have found it adequate to use the distance between the manipulator and the *reference object* as our single dimension. The reference object

is explicitly mentioned in the narration, e.g., "Screw nut on *left hub stud 1*," and is extracted through some trivial natural language processing.

GrammarViz converts the continuous motion data into a sequence of symbols (essentially a text string) using two discretization parameters. Since GrammarViz's motif discovery is quite sensitive to the settings of these two parameters, we run GrammarViz on a range of values for these parameters. Each successful run identifies a different motif that recurs in each occurrence of the primitive. For example, the third box in Fig. 5 shows a motif, shaded in red, that has been identified in three occurrences of the S primitive. The unshaded parts of the data are either part of the preceding and following primitives, or transition motions.

(C) Local Motif Ranking

The goal of the next two steps is to select the motif that best abstracts the given primitive. We start by eliminating some clearly invalid motifs that either 1) include part of the artificially added spacer, or 2) contain more shaded trajectories than there are primitive occurrences. We explored two utility metrics to rank the remaining valid motifs:

- *Length metric*: This reflects the simple intuition that longer motif occurrences are better because they cover more of the data.
- *Density metric*: This metric looks in more detail at how consistently a given motif explains the data. A density histogram is generated for each data point by counting, across all occurrences of a motif, the number of times that data point is included in an occurrence. For each motif occurrence, the area under the density function in the occurrence's data interval is the utility of that occurrence.

For each of these metrics, we compute the net utility of a motif by averaging the utility of all its occurrences.

(D) Global Motif Selection

Up to this point, each primitive was analyzed independently (locally). However, this does not guarantee that the motifs for adjacent primitives in the overall task sequence do not overlap with each other (which would clearly be wrong). We therefore consider all possible permutations of motifs for each primitive in the overall task sequence and eliminate those that have any overlapping motif occurrences. Finally, among the remaining valid permutations, we select the permutation with the greatest sum over the sequence of primitives of net motif utilities.

(E) Trajectory Selection

The preceding step selects a single motif for each primitive, but each occurrence of this motif corresponds to a different occurrence of the primitive with slightly different motion trajectory data. As the last step in primitive identification we need to select a single motion trajectory to pass on to the next stage of processing, described in Section 5, which is to learn a TSR from a single demonstration. In order to find the most representative trajectory for this purpose, we calculate the distance between each pair of occurrences using dynamic time warping and choose the occurrence with the least average distance from other occurrences. If there are only two occurrences, we choose the shorter.

4.3 Evaluation

In the same mock-up environment used in the accompanying video, we recorded motion data and synchronized video for nine participants executing four task sequences (two main tasks and two supplementary sequences) containing a total of 24 occurrences of the eight tire rotation primitives. The participants were given detailed written descriptions of the eight primitives and each sequence. The first main task involved mounting a tire on a hub with a single nut, unmounting it, and then remounting it on another hub, and then finally unmounting it again: PUT, H, PUN, S, US, PDN, UH, H, PUN, S, US, PDN, UH, PDT. A table was provided on which to put down the nut. The second main task involved mounting and unmounting a tire on a hub, but with the second hub used as a storage location for the nut instead of the table: PUT, H, US, S, US, S, UH, PDT.

We designed the two main tasks to provide at least two different contexts (preceding and following primitives) for the four most constrained primitives (see Section 5): Unscrew, Screw, Unhang and Hang. Two short supplementary sequences provide the needed additional contexts to identify the remaining primitives: PUN, PDN and PUT, PDT.

We used the data gathered here to evaluate our identification algorithms in three different ways. The results below are aggregated across all participants.

4.3.1 Viable Trajectories

For our first evaluation, we had two expert judges (the first and third authors) make a binary judgement about the trajectory selected for each primitive by the identification subcomponent (using the density heuristic for step C). The judges viewed a video clip of each participant performing the selected trajectory for each primitive and were told to reject a trajectory only if it did not accomplish the primitive (according to the written instructions) or contained more than one primitive. The judges rejected only 14% of the selected

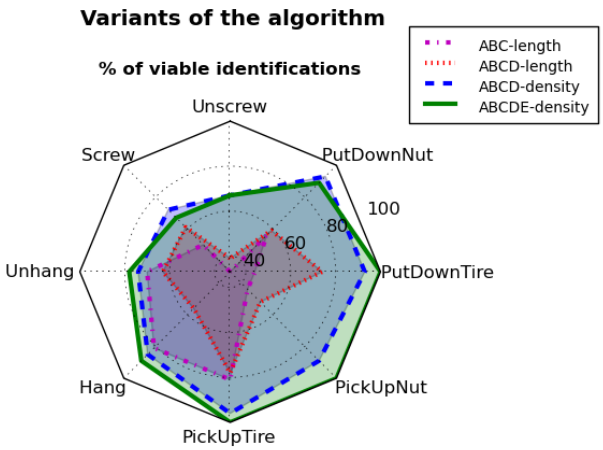


Fig. 6 Radar chart showing the percentage of viable trajectories identified for each primitive using different variants of the algorithm.

trajectories, which means that our system was 86% accurate in identifying viable trajectories.

Primitives that involve large motions, such as PickUpTire and PutDownTire, were identified with 100% accuracy. Primitives with smaller motions, such as Screw and Unscrew, were more challenging, resulting in 66% accuracy. One of the reasons these primitive are more challenging is that small variabilities in their execution can have a greater impact on the overall trajectory. For example, our data contains Screw and Unscrew occurrences both for an empty stud and for a stud with a mounted tire. The resulting difference in stud length led to variability within even a single person's demonstrations of the same primitive, making it difficult for the algorithm to accurately determine the boundaries between primitives.

4.3.2 Trajectory Trimming Errors

In the second stage of our evaluation, we looked more closely at the trimming of trajectories. Our expert judges viewed the original videos of the complete sequences and hand coded what they considered to be the correct start and end times for each primitive. The Cronbach's alpha for the agreement between the two experts was 0.94, which is strong. The normalized mean error between the expert start/end values and the system's values was 0.12 with a standard deviation of 0.09. The errors were normalized by dividing, in each case, by the length (time) of the enclosing section (as determined by step A above). Thus an error of 0.12 could arise, for example, by the start time being early by approximately 1/16 of the width of the section and the end time being late by 1/16. There was little variance in the error between different primitives.

4.3.3 Comparing Algorithm Variants

Finally, we compared the performance of four variants of our algorithm (letters refer to steps in Fig. 5) with respect to the percentage of viable trajectories: ABC with the length heuristic, ABCD with the length heuristic, ABCD with the density heuristic, and ABCDE with the density heuristic. The results for each variant for each primitive is shown in the radar chart in Fig. 6.

The chart shows that adding step D had no consistent impact on algorithm performance. However, the density heuristic is overall significantly better than the length heuristic. Adding step E (using dynamic time warping to compare occurrences) improved performance for most primitives. Overall, the ABCDE-density variant of the algorithm outperformed all the other variants.

Further details about this subcomponent can be found in Mohseni-Kabir et al (2016).

5 Learning Primitives: TSR Constraints

The second subcomponent in learning primitives (see Fig. 2) is learning TSR constraints. The algorithm in this subcomponent is applied independently to each primitive. For each primitive, the input is a single demonstrated motion trajectory and the output is the constraints that define the primitive, represented as a task space region (TSR) sequence. The algorithm also relies on a geometric model of the demonstration environment. A TSR (Berenson et al 2011) specifies a volume in configuration space ($SE(3)$) using a reference transform, an offset transform, and a matrix of dimensional bounds.

Many manipulation tasks include primitives where the robot must navigate an object through a narrow passage (e.g., sliding a nut onto a post) or obey a continuous pose constraint (e.g., not tilting a cup of water while you are carrying it). While motion planning algorithms capable of performing such tasks exist, they tend to be very slow, because their search cannot easily be biased to concentrate on the parts of the configuration space that are most relevant to the narrow passages.

Our approach is to focus the motion planner on relevant parts of the configuration space, not by attempting to replay the human demonstration, but by inferring constraints from it. Basically, we explore regions around the demonstration by sampling, analyzing the geometric properties of the samples, and then extracting relevant constraints, represented as TSRs. The resulting TSRs allow a motion planner such as CBiRRT (Berenson et al 2011) to generate feasible plans much more quickly than without them and in a greater variety of new environments.

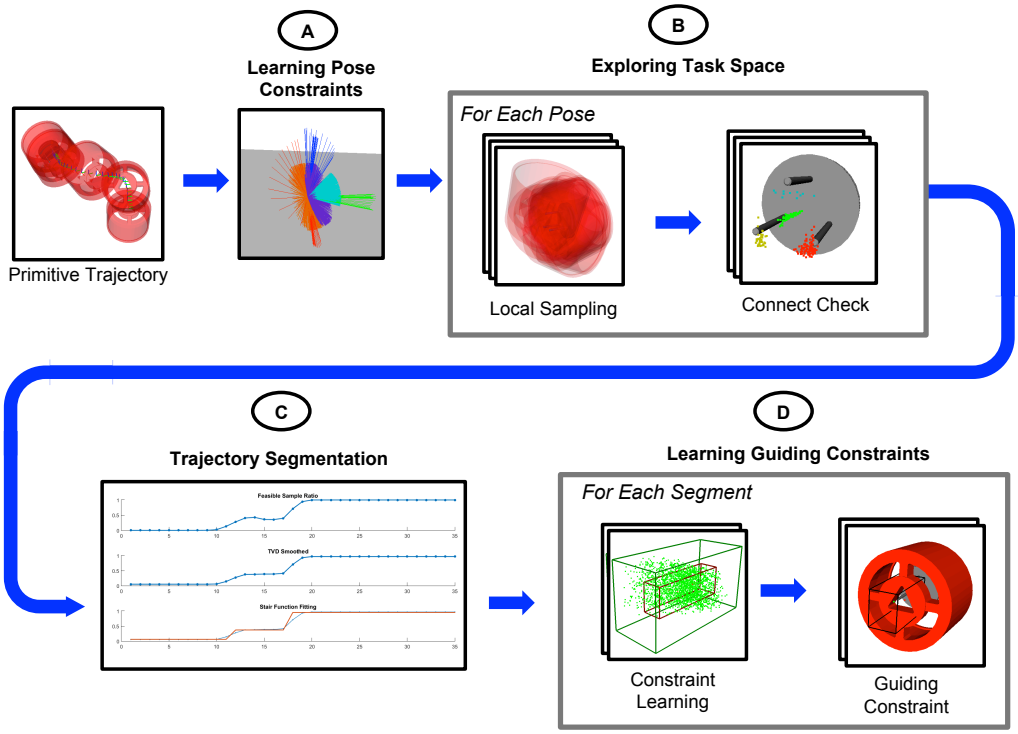


Fig. 7 Steps in the TSR constraint learning algorithm for each primitive.

5.1 Related Work

Previous methods for learning constraints from demonstration can be divided into two classes according to the type of input: 1) kinesthetic demonstrations (Akgun et al 2012; Phillips et al 2016), which bypass the retargeting problem, but require the demonstrator to have a good understanding of the robot's kinematics to prevent noisy demonstrations; and 2) observed human demonstrations (Baisero et al 2015; Mollard et al 2015), perhaps with verbal comments (Pardowitz et al 2007), where the demonstrator can act naturally, but retargeting the demonstrated motion to the robot is a challenge.

Most methods that learn constraints from demonstration require multiple demonstrations, which are often used to compute the variance along the trajectory. Some additional preprocessing, such as data alignment (Akgun et al 2012; Ye and Alterovitz 2011), is also needed. Given the aligned data, the multiple demonstration trajectories (Calinon et al 2007) or key frames (Akgun et al 2012) are represented as Gaussian mixture models and a solution is found using by Gaussian mixture regression. The drawback of these methods is that they do not generalize to new environments where the task is similar but new obstacles are present and/or the start/goal are moved. To overcome this limitation, Ye and Alterovitz (2011) learn a cost function from multiple demonstrations, and use a sampling-based planner to find a feasible path with low cost.

What distinguishes our work is that: 1) we learn from only a single demonstration; 2) our method does not require transferring a human motion to the robot, i.e., solving the retargeting problem; 3) the constraints we learn can transfer to similar tasks, i.e. a different start/goal pose or additional obstacles; and 4) our method scales to tasks in Euclidean group $SE(3)$.

5.2 Steps in TSR Constraint Learning Algorithm

Fig. 7 provides an overview of the TSR constraint learning algorithm. For this algorithm, a primitive is specified by a moving object (e.g., a nut), a reference object (e.g., a stud), and the start/goal poses of the moving object with respect to the reference object. The information we extract from the human demonstration of a primitive is a trajectory (series of poses) of the moving object in task space, which is $SE(3)$. This trajectory is assumed to be feasible (no collisions) and connect the start and goal poses.

(A) Learning Pose Constraints

For learning pose constraints, we use Euler angles (roll, pitch and yaw) as the parametrization of the orientations. Similar to principle component analysis (PCA), we wish to find a reference frame in which the components with the largest weight are aligned with the axes. However, since Euler angle

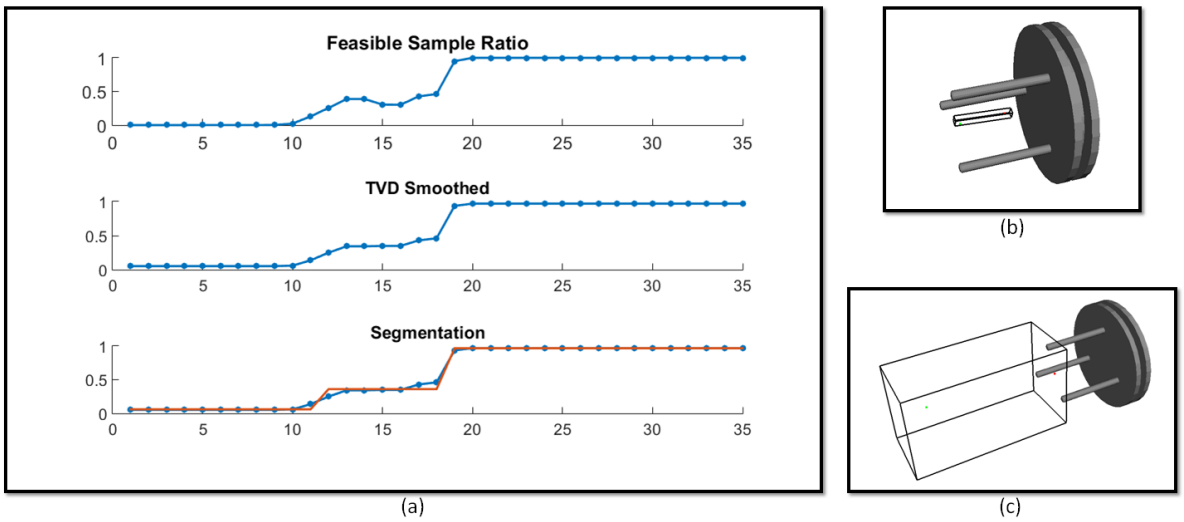


Fig. 8 (a) Results of smoothing and staircase fitting for Unhang. The horizontal axis is the time index in the demonstrated trajectory. The optimal staircase function is in red in the segmentation graph. (b) Learned TSR of first segment. (c) Learned TSR of second segment.

space is not linear, we use a random volume descent (RDV) method to find the reference frame in which the volume of the axis-aligned bounding box of the demonstrated orientations is smallest.

In the resulting reference frame, we calculate the range of the demonstrated orientations in each dimension (roll, pitch and yaw). Then we say a dimension is unconstrained if the range of this dimension is greater than a predetermined threshold; otherwise this dimension is constrained to the range in the demonstration. These orientation constraints are then added to a TSR with unbounded position constraints.

(B) Exploring Task Space

To explore the task space near the demonstration trajectory, we first sample poses of the moving object around each of the poses in the trajectory. Samples are generated using small random rotations and translations of the moving object. Samples are discarded if they are outside the learned pose constraints computed in step A. Samples are also discarded if they collide with the environment. This gives us an initial set of feasible samples.

Next, the feasible samples are evaluated based on connectivity. We use a local planner to try to connect each random sample by a straight line in $SE(3)$ to the demonstrated pose it is based on. If this is not possible, the sample is discarded.

Finally, we calculate the *feasible sample ratio*, which is the ratio of the final number of feasible connected samples versus the total original number of samples. This ratio is computed for every demonstrated pose, thus obtaining a series of ratios for the entire trajectory.

(C) Trajectory Segmentation

We segment the trajectory using the feasible sample ratio, because different segments of a primitive have significantly different constraints, and thus should be represented with different TSRs. For example, when removing a nut from a stud, the nut is highly constrained when it is on the stud, and is free when it is off the stud. The key to segmenting the trajectory is to identify points where there are significant changes in the feasible sample ratio. First, we smooth the series of ratios using total variation denoising (TDV) (Rudin et al 1992). TDV has the advantage that it smooths noise in relatively flat stages while not shifting step edges. Each step change in the smoothed signal implies a significant change in the feasible sample ratio and represents the start of a new segment. We then fit a series of step functions (a staircase function) to the smoothed signal (Levy-leduc and Harchaoui 2008). Each flat region of the staircase is a segment.

Fig. 8(a) shows the output of trajectory segmentation for the Unhang primitive. The feasible sample ratio curve clearly has a 3-step shape, as during the demonstration, the tire goes from a highly constrained region (sliding off the studs), to a region with some constraints (handling the tire close to the hub), and finally to a region with almost no constraints (free space motion).

(D) Learning Guiding Constraints

For each segment determined in step C, we learn a guiding constraint TSR from the samples in this segment. Similar to step A for pose constrains, our goal is to find the range bounds in each dimension in task space: if the bounds are too large, the power of TSRs to the narrow the search space is lost,

but if the bounds are too small, the TSR will not perform well in new environments. In order to make sure the TSR includes at least one solution, the bounds must include the demonstrated poses. Since the output of this process depends on the reference frame, we again use RDV to choose the reference frame, since $SE(3)$ is not linear.

If the feasible sample ratio of the entire segment is close to 1, we assume there is no constraint on this segment, and the guiding TSR is unbounded. Otherwise, we use an iterative shrinking process to calculate the TSR bounds. We first calculate the demonstrated TSR, which corresponds to the smallest axis-aligned bounding box that includes only the demonstrated poses. Then we iteratively shrink the feasible TSR, which starts with the smallest TSR bounds that include all of the connected feasible samples calculated in step B. On each iteration, we remove the sample pose that is farthest from the demonstration TSR, until either a predetermined ratio between feasible samples and all samples been reached or the feasible TSR has become the identical to the demonstration TSR.

Figures 8(b) and (c) show the guiding TSRs learned for the first two segments of Unhang. Only the position boundaries are shown. The third segment has no translation constraints and therefore is not shown.

5.3 Evaluation

In the same mock-up environment used in the accompanying video, we recorded motion data for a single, isolated demonstration of each of the four tire rotation primitives that have a narrow passage: Unhang, Hang, Unscrew and Screw. We manually trimmed the resulting trajectories.[5]

For each primitive, we compared four planning variants using the CBiRRT sampling-based motion planner: 1) ordinary sampling with no TSR constraints; 2) bridge sampling (Hsu et al 2003), a competing method for sampling narrow passages, with no TSR constraints; 3) ordinary sampling with learned TSR guiding constraints; and 4) ordinary sampling with learned TSR pose and guiding constraints. The planner was run in the OpenRAVE simulation environment with the same geometry as the mock-up, 50 trials for each primitive for each variant. A trial that could not find a solution within three minutes was considered a failure.

Table 1 shows the resulting success rates. The first two planning variants did not succeed in planning any of primitives in any of the trials. Thus TSR constraints of some kind are necessary to successfully plan any primitive in under 3 minutes. Comparing the last two rows, we can see that using both pose and guiding constraints has the best performance.

[5] We also recorded motion data for a cup retrieval task to specifically evaluate the pose constraint learning—see Li and Berenson (2016).

Planning Variant	Unhang	Hang	Unscrew	Screw
CBiRRT	0/50	0/50	0/50	0/50
CBiRRT with bridge sampling	0/50	0/50	0/50	0/50
CBiRRT using guiding constraints	41/50	43/50	50/50	50/50
CBiRRT using pose & guiding constraints	50/50	50/50	50/50	50/50

Table 1 Success rates for planning variants.

This is because pose constraints consider how the moving object is manipulated by the demonstrator, and rule out many poses that the robot's end effector can not reach, even though they are collision-free.

A disadvantage of our approach is that it cannot guarantee completeness, for two reasons: First, because we learn from only a single demonstration, we may learn spurious pose constraints. For example, for unhanging a tire, we may unnecessarily constrain the tire to be held in a particular orientation while it is being manipulated in free space. Enforcing this pose constraint significantly reduces planning time, but we lose some reusability, e.g., to avoid new obstacles. Second, because in building the TSR, we only explore a limited region around the demonstration trajectory, planning solutions that are outside of this region will not be considered. This limitation can be overcome by running an unconstrained planner in parallel with the planner using TSRs.

Further details about this subcomponent can be found in Li and Berenson (2016).

6 Conclusion

We have developed a new LfD interaction paradigm, called simultaneous learning of hierarchy and primitives (SLHAP), in which the information needed for a robot to learn the structure of an HTN and the motion planning constraints of its constituent primitives is provided by a human teacher in a single coherent, natural interactive teaching session. A key innovation in the new paradigm is the human demonstrator's narration of primitives as he executes them.

The accompanying video illustrating SLHAP was generated using the proof-of-concept system we have implemented. This system is made possible by algorithms, which we also developed, for learning hierarchy, identifying primitives, and learning TSR constraints. Each of these algorithms has been separately described and evaluated.

A direction for immediate future work is to address the current limitations of the proof-of-concept system listed in Section 2.1:

– Solutions for open-microphone (no push-to-talk) speech recognition already exist in home-assistant consumer

products. The bigger challenge is removing the need for a predefined grammar for human utterances. For speech recognition, cloud services already exist that will transcribe unrestricted natural language with high accuracy. However, the bigger challenge is to somehow restrict the human demonstrator to only say things that are within the realm of what the system can actually do, without requiring an onerous training process.
- Automated motion planning for the robot base is well within the current state of art. It was not included in the proof-of-concept system due to lack of time and resources.
- A significant amount of additional algorithmic study and development is needed in order to make all of the algorithms described above run in real time.
- Our existing TSR learning algorithms should easily work for PickUpTire, PutDownTire, PickUpNut and PutDownNut.

Addressing the system limitations as above opens the door to a number of interesting user studies to answer questions such as: Is it enough to simply tell demonstrators to "narrate what you are doing"? How consistent will be the names chosen for primitives and non-primitives? How natural will demonstrators find the overall interaction?

In a more algorithmic vein, the choice of the distance between the demonstrator's hand and the reference object as the signal dimension for time series pattern matching (step B in Section 4.2) needs further investigation. It is clear that this will not work for all possible manipulation primitives. In learning theory, this relates to the issue of automatic feature selection.

References

Akgun B, Cakmak M, Jiang K, Thomaz AL (2012) Keyframe-based learning from demonstration. International Journal of Social Robotics 4(4):343–355

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

Baisero A, Mollard Y, Lopes M, Toussaint M, Lutkebohle I (2015) Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In: IROS

Berenson D, Srinivasa SS, Kuffner J (2011) Task space regions: A framework for pose-constrained manipulation planning. The International Journal of Robotics Research

Cakmak M, Thomaz AL (2012) Designing robot learners that ask good questions. In: ACM/IEEE International Conference on Human-Robot Interaction, ACM, pp 17–24

Cakmak M, Chao C, Thomaz A (2010) Designing interactions for robot active learners. Autonomous Mental Development, IEEE Transactions on 2(2):108–118

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 (Cybernetics) 37(2):286–298

Chernova S, Thomaz A (2014) Robot learning from human teachers. Synthesis Lectures on Artificial Intelligence and Machine Learning 8(3):1–121

Chiappa S, Peters JR (2010) Movement extraction by detecting dynamics switches and repetitions. In: Advances in neural information processing systems, pp 388–396

Erol K, Hendler J, Nau D (1994) HTN planning: Complexity and expressivity. In: Proc. 12th National Conf. on Artificial Intelligence, Seattle, WA

Garland A, Ryall K, Rich C (2001) Learning hierarchical task models by defining and refining examples. In: International Conference on Knowledge Capture, pp 44–51

Hayes B, Scassellati B (2014) Discovering task constraints through observation and active learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems

Hsu D, Jiang T, Reif J, Sun Z (2003) The bridge test for sampling narrow passages with probabilistic roadmap planners. In: ICRA

Huffman SB, Laird JE (1995) Flexibly instructable agents. Journal of Artificial Intelligence Research 3:271–324

Kulic D, Lee D, Ott C, Nakamura Y (2008) Incremental learning of full body motion primitives for humanoid robots. In: Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on, IEEE, pp 326–332

Levy-leduc C, Harchaoui Z (2008) Catching change-points with lasso. In: Advances in Neural Information Processing Systems, pp 617–624

Li C, Berenson D (2016) Learning object orientation constraints and guiding constraints for narrow passages from one demonstration. In: International Symposium on Experimental Robotics

Minnen D, Starner T, Essa IA, Isbell Jr CL (2007) Improving activity discovery with automatic neighborhood estimation. In: IJCAI, vol 7, pp 2814–2819

Mohammad Y, Nishida T (2015) Exact multi-length scale and mean invariant motif discovery. Applied Intelligence pp 1–18

Mohan S, Laird JE (2011) Towards situated, interactive, instructable agents in a cognitive architecture. In: AAAI Fall Symposium Series

Mohseni-Kabir A, Rich C, Chernova S, Sidner CL, Miller D (2015) Interactive hierarchical task learning from a single demonstration. In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, ACM, pp 205–212

Mohseni-Kabir A, Wu V, Chernova S, Rich C (2016) What's in a primitive? identifying reusable motion trajectories in narrated demonstrations. In: IEEE International Symposium on Robot and Human Interactive Comm. (ROMAN)

Mollard Y, Munzer T, Baisero A, Toussaint M, Lopes M (2015) Robot programming from demonstration, feedback and transfer. In: IROS

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

Oates T (2002) Peruse: An unsupervised algorithm for finding recurring patterns in time series. In: Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, IEEE, pp 330–337

Pardowitz M, Knoop S, Dillmann R, Zollner R (2007) Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37(2):322–332

Phillips M, Hwang V, Chitta S, Likhachev M (2016) Learning to plan for constrained manipulation from demonstrations. Autonomous Robots 40(1):109–124

Rich C, Sidner CL (2012) Using collaborative discourse theory to partially automate dialogue tree authoring. In: Proc. Int. Conf. on Intelligent Virtual Agents, Santa Cruz, CA, pp 327–340

Rudin LI, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena

60(1):259–268

Rybski PE, Yoon K, Stolarz J, Veloso MM (2007) Interactive robot task training through dialog and demonstration. In: ACM/IEEE Int. Conf. on Human-Robot Interaction, pp 49–56

Senin P, Lin J, Wang X, Oates T, Gandhi S, Boedihardjo AP, Chen C, Frankenstein S, Lerner M (2014) Grammarviz 2.0: a tool for grammar-based pattern discovery in time series. In: Machine Learning and Knowledge Discovery in Databases, Springer, pp 468–472

Ye G, Alterovitz R (2011) Demonstration-guided motion planning. In: ISRR