Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice

Monica N. Nicolescu * and Maja J. Matarić
monica|mataric@cs.usc.edu
Computer Science Department, University of Southern California
941 West 37th Place, Mailcode 0781
Los Angeles, CA 90089-0781

ABSTRACT

Among humans, teaching various tasks is a complex process which relies on multiple means for interaction and learning, both on the part of the teacher and of the learner. Used together, these modalities lead to effective teaching and learning approaches, respectively. In the robotics domain, task teaching has been mostly addressed by using only one or very few of these interactions. In this paper we present an approach for teaching robots that relies on the key features and the general approach people use when teaching each other: first give a demonstration, then allow the learner to refine the acquired capabilities by practicing under the teacher's supervision, involving a small number of trials. Depending on the quality of the learned task, the teacher may either demonstrate it again or provide specific feedback during the learner's practice trial for further refinement. Also, as people do during demonstrations, the teacher can provide simple instructions and informative cues, increasing the performance of learning. Thus, *instructive demonstrations*, *generalization* over multiple demonstrations and *practice trials* are essential features for a successful human-robot teaching approach. We implemented a system that enables all these capabilities and validated these concepts with a Pioneer 2DX mobile robot learning tasks from multiple demonstrations and teacher feedback.

1. INTRODUCTION

Robots that can successfully and efficiently interact with humans require adaptation and learning capabilities for most non-trivial interactions. This enables robots not only to adapt and improve their performance, but also to be more accessible to a larger range of users, from the lay to the skilled.

Designing controllers for robotic tasks is usually done by people specialized in programming robots. Even for them, most often this is a complicated process, and it essentially requires creating by hand a new and different controller for each particular task. If robots are to be effective in human-robot domains, even users without programming skills should be able to interact with them and "re-program" them.

Thus, *automating the robot controller design process* becomes of particular interest. A natural approach to this problem, and one that has most widely been used, is *teaching by demonstration*, but other methods have been developed as well. Gestures [17], natural language [11], and animal "clicker training" [9] are also natural techniques that have been successfully applied for teaching robots various tasks. However, the majority of robot teaching approaches to date has been focused on learning policies [6, 16], or temporally extended sensory-motor skills [3]. Techniques for learning complex task structures have also been presented [10], but they are highly sensitive to the structure of the environment and of the demonstration, and do not allow for further improvement or adaptation if the task is not learned correctly in the first trial.

Our goal is to develop a flexible mechanism that allows a robot to learn and refine representations of *high level tasks*, *from interaction with a human teacher*, *based on a set of underlying capabilities (behaviors) already available to the robot*.

Since people are very good at learning from a teacher's training, we are interested in the key features that make this process efficient, and seek to develop a similar robot teaching strategy. Human teachers rely on concurrent use of multiple instructive modalities, including primarily demonstration, verbal instruction, attentional cues, or gestures. On the part of the learner, the process is also more complex than a one-shot teaching experience. "Students" are typically given one demonstration of the task and then they perform a set of practice trials under the supervision of the teacher, in order to show what was learned. If needed, during these runs the teacher provides *feedback cues* to indicate corrections (irrelevant actions or missing parts of the task). Alternatively, the teacher may also provide additional demonstrations that the learner could use for *generalization*. Most of these aspects are generally overlooked in the majority of robot teaching approaches, which focus mostly on only one or very few of these instructive and learning modalities. We believe that considering these issues makes significantly easier and improves the learning process by conveying more information about the task, while in the same time allowing for a very flexible robot teaching approach.

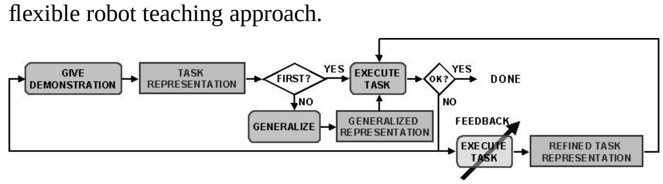


Figure 1: Learning and refining tasks through demonstrations, generalization and teacher feedback

We propose a method for learning representations of high level tasks, similar to the one people use when teaching each other. Our overall strategy for learning and refining task representations is presented in Figure 1. The flexibility of this strategy consists in allowing the teacher to choose the methods considered most appropriate at any given time: after a first demonstration, either provide additional training examples or give feedback on what has been learned

*The first author is a PhD student at the University of Southern California during a practice trial. Our experiments show that similar effects can be achieved by following different teaching approaches (i.e. various combinations of demonstrations and feedback), allowing the teacher to adapt his teaching techniques to each particular case.

In the following sections we describe the specifics of our teaching process, followed by the methods for refining learned task representations through generalization and feedback. Next, we present the robot experiments, and discussions of the obtained results and of related work. We end with conclusions on the presented approach.

2. TEACHING PROCESS: EXPERIENCED DEMONSTRATION AND INSTRUCTION

For our work, we assume that the robot is equipped with a set of skills, in the form of behaviors [1], and we focus on a strategy that would help a robot build a high-level task representation of a more complex, sequentially structured task, built from the existing behavior set. We do not attempt to reproduce exact trajectories or actions of the teacher, but rather learn the task in terms of its high-level goals.

The most common approach for robot teaching is through the use of demonstrations, the same strategy we are also going to use as the main modality for instruction. Two different methods for learning from demonstration exist: *learning by observation*, in which the learner passively observes the teacher performing a task, and *learning by experience*, in which the robot performs the task along with the teacher during the demonstration.

In our particular approach to learning, we use *learning by experienced demonstrations*. This implies that the robot actively participates in the demonstration provided by the teacher, and experiencing the task through its own sensors. This is an essential characteristic of our approach, and is what provides the robot the data necessary for learning. In the mobile robot domain the demonstrations are achieved by following and interacting with the teacher. We assume that the teacher knows what skills the robot has, and also by what means (sensors) they can be detected. The ability to learn from the observations gathered during the demonstration is based on the robot's ability to relate the observed states of the environment to the known effects of its own skills (see Section 3). The advantage of *putting the robot through* the task during the demonstration is that the robot is able to adjust its behaviors (through their parameters) using the information gathered through its own sensors. In addition to experiencing parameter values directly, executing the task during the demonstration provides observations that contain temporal information for proper behavior sequencing, which would be tedious to design by hand for tasks with long temporal sequences.

Irrespective of the demonstration strategy being used, an important challenge for these learning methods is to distinguish between the relevant and irrelevant information being perceived. Putting the entire responsibility on the learner to decide between relevant and irrelevant observations, such as when learning solely by observation, increases the complexity of the problem and leads to more complicated, sometimes ineffective solutions. During demonstrations humans almost always make use of additional simple cues and instructions that facilitate the learning process and bias the learner's attention to the important aspects of the demonstration (e.g. *"watch this"*, *"lift that"*, etc.). Although simple, these cues have a large impact on the robot's learning performance: by relating them with the state of the environment at the moment when they are received, the learner is provided with information that may otherwise be impossible or extremely hard to obtain only from the observed data. For example, while teaching a robot to go and pick up the mail, the robot can detect numerous other aspects along its path (e.g., passing by a chair, meeting another robot, etc.). These observations are irrelevant for getting the mail, and simple cues from the teacher could easily indicate that.

Therefore, in order for a robot to learn a task effectively, the teacher also needs to provide it with additional information beyond the perceived demonstration experience. To achieve this, we add *verbal instruction* to the existing demonstration capabilities of our system. With this, the teacher can provide the following types of information:

- **"HERE"** - indicates moments in time during the demonstration when the environment presents aspects that are relevant for the task. These indications are general (simple hints meaning *"pay attention now"*) and by no means spell out for the robot the representation of the presented task. While such indications allow the robot to distinguish some of the irrelevant observations, they may still not help it to perfectly learn the task. For this, generalization techniques (Section 4) and feedback-practice runs (Section 5) will be applied.
- **"TAKE", "DROP"** - instructions that induce the robot into performing certain actions during the demonstration (in this case **Pick Up** and **Drop** small objects), actions that would be otherwise impossible to trigger in a teacher-following-only learning approach. In our case, we instruct the robot to open and close its gripper, when the task to be learned involves moving certain objects around.
- **"START", "DONE"** - instructions that signal the beginning and the end of a demonstration, respectively.

The next section presents the algorithm for learning task representations, based on the observations and cues gathered during a single demonstration.

3. LEARNING TASK REPRESENTATIONS

The ability of the robot to learn from experienced demonstrations is enabled by the particular structure of our control architecture. We use an extension of the standard Behavior-Based System we developed, which provides a simple and natural way of representing complex tasks and sequences of behaviors in the form of networks of *abstract behaviors* (Figure 2). In a behavior network, the links between behaviors represent precondition-postcondition dependencies, which can have three different types: *permanent*, *enabling* and *ordering*. Thus the activation of a behavior is dependent not only on its own preconditions (particular environmental states) but also on the postconditions of its relevant predecessors (*sequential preconditions*). More details on this architecture can be found in [14].

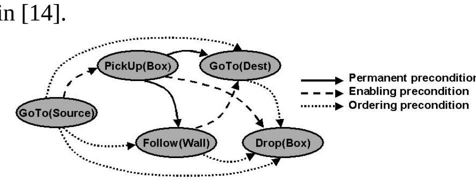


Figure 2: Example of a behavior network

Within this architecture, behaviors are build from two components: one related to perception (*Abstract behavior*), the other to action (*Primitive behavior*). The *abstract behavior* is an explicit specification of the behavior's activation conditions (i.e., preconditions), and its effects (i.e., postconditions). The behaviors that do the work that achieves the specified effects under the given conditions are called *primitive behaviors*. An *abstract behavior* takes sensory information from the environment and, when its preconditions are met, activates the corresponding *primitive behavior(s)*, which achieve the effects specified in its postconditions.

The *abstract behaviors* embed representations of a behavior's goals in the form of abstracted environmental states. This is a key feature of our architecture, and a critical aspect for learning from experience. *In order to learn a task the robot has to create a link between perception (observations) and the robot's behaviors that would achieve the same observed effects.*

During the demonstration, while the robot follows the human teacher, all its available behaviors continuously monitor the status of their postconditions (without executing any of their actions). Whenever the observations match a primitive's goals, this represents an example of the robot having seen something it is also able to do, and the corresponding *abstract behaviors* fires, allowing the robot to identify during its experience the behaviors that are relevant for the task being learned. The feedback cues received from the teacher are used in conjunction with these observations, to eliminate any irrelevant observations.

The general idea of the algorithm is to add to the network task representation an instance of all behaviors whose postconditions have been detected as true during the demonstration, and during which there have been relevance signals from the teacher, in the order of their occurrence (on-line stage). At the end of the teaching experience, the intervals of time when the effects of each of the behaviors have been true are known, and are used to determine if these effects have been active in overlapping intervals or in sequence. Based on the above information, the algorithm generates proper dependency links between behaviors (i.e., *permanent*, *enabling* or *ordering*) (off-line stage). This one-shot learning process is described in more detailed in [13]. The only differences to the work presented here are that in that case teaching was performed without any cues and verbal instruction from the teacher, and the experiments were performed in "clean" environments, so that all robot's observations would be considered relevant for the task. Also, the construction of the task representations was done off-line.

The next section describes our approach for generalizing over several task representations learned with the method described so far.

4. GENERALIZATION FROM MULTIPLE DEMONSTRATIONS

Another capability that allows humans to learn effectively is the ability to generalize over multiple given demonstrations. For a teaching by demonstration approach to be efficient, it is essential that the robot learn from as few demonstrations as possible. A robot house keeper is of little use if the owner must show it hundreds of times how to bring in the mail. Therefore, statistical learning techniques, which rely on a large number of training examples, are not appropriate for our desired approach.

Given the directed acyclic graph (DAG)-like structure of the behavior network representation of the robot tasks, we consider the *topological* representation of such a network to be a linked list of behaviors, obtained by applying a *topological sort* on the behavior network graph. By using the topological form of the networks as training examples for our domain, the problem of generalization from multiple demonstrations is equivalent to inferring a regular expression (Finite State Automaton (FSA) representation) from a set of given sample *words* (Figure 3(a)). In this analogy, each symbol in a given *word* corresponds to a behavior in a topological representation.

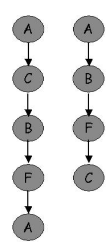
Unfortunately, applying standard methods for regular expression inference, such as the K-TSI Inference Algorithm [5], or Morphic Generator Grammatical Inference (MGGI) [15], to this generalization problem yields results that are too complex (in terms of the obtained FSA representations) even for very simple examples. This is due to the fact that these methods assume that all the training examples are correct and they try to fit them as well as possible. For our robot domain, in which the inaccuracies in the training examples are exactly the problem we need to solve, these methods are therefore not well suited.

In robotics, existing methods for generalization from demonstrated examples are largely based on function approximation [8]. Since our tasks are encoded in graph-like representations, we need a different method for generalizing across them.

4.1 Computing the common sequence

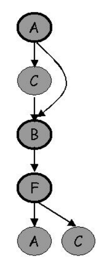
The reason we are interesting in giving a robot the ability to generalize over multiple teaching experiences is that its limited sensing capabilities, the quality of the teacher's demonstration, and particularities of the environment generally prevent the robot from correctly learning a task from only one trial. The two main inaccuracies that occur in the learning process are *learning irrelevant steps* (false positives) and *omission of steps that are relevant* (false negatives).

Our approach for generalization is to build a task representation that encodes the specifics of each input example, but most importantly that points out the parts that are common to each of them. As a measure of similarity we consider the longest list of common nodes between the *topological* forms of the sample tasks. Based on this information we further construct a *generalized topology* in which nodes that are common to both tasks will be merged, while the others will appear as alternate paths in the graph. For example, for the examples presented in Figure 3(a), behaviors A, B and F constitute the longest subsequence of common nodes. The representation resulted after "merging" the initial graphs at their common nodes is shown in Figure 3(c).



Longest common sequence

X \ Y	A	B	F	C
A	↖1	←1	←1	←1
C	↑1	↑1	↑1	↖2
B	↑1	↖2	←2	↑2
F	↑1	↑2	↖3	←3
A	2	↑2	↑3	↑3

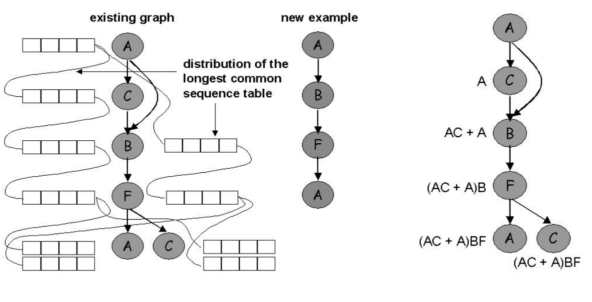


(a) (b) (c)

Figure 3: Generalization across multiple demonstrations. (a) Training examples; (b) Longest common sequence table; (c) Generalized topology

In order to find the similarity between the two inputs we rely on a standard dynamic programming approach for computing the *longest common subsequence (LCS)* [2] between two sequences of symbols. If $X = < x_1, x_2, ..., x_m >$ and $Y = < y_1, y_2, ..., y_n >$ are two sequences of symbols, and the **prefix** of a sequence is defined as $X_i = < x_1, x_2, ..., x_i >$, the algorithm computes a longest common subsequence table (Figure 3(b)) that encodes in each element $tbl[i, j]$: i) the length of the longest common subsequence of the sequences X_i and Y_j, and ii) a pointer to the table entry corresponding to the optimal subproblem solution chosen when computing $tbl[i, j]$. The right bottom element of the table contains the length of the LCS for the entire sequences X and Y. The running time of the algorithm is $O(mn)$, with m and n being the lengths of the two sequences X and Y, typically small for our domain.

We obtain the generalized topology by traversing the LCS table starting in the right bottom corner and following the arrows: an "↖" arrow indicates nodes that are common to both training examples and that are merged, while "←" and "↑" arrows indicate nodes that belong to only one sequence. These latter cases are added as alternate paths of execution (Figure 3(c)).



(a) (b)

Figure 4: Incorporating new demonstrations: (a) Efficient computation of a generalized topology; (b) Behavior preconditions in a topological representation

The generalization process is incremental, meaning that each newly acquired experience is *incorporated* into the existing topological task representation. If this topology is already the result of a previous generalization, and has the form of a DAG with alternate paths of execution (Figure 3(c)), in a simplistic approach, incorporating a new demonstration into the existing structure would amount to running the same algorithm described above between the new example and all the possible paths of that graph. Since a LCS table encodes the common subsequences for all possible subproblems ($LCS(X_i, Y_j)$, with $i = [1, m]$ and $j = [1, n]$), we can efficiently apply this algorithm without having to compute a different table for each path of the graph. For this, we construct a structure that contains the LCS table in the form of a linked list of rows computed as the ones above. Within this structure, each node has associated a row for each different possible path from the root(s) to that node (Figure 4(a)). Each of these rows points to the row associated to the parent node on the corresponding path. As a result, each path in the graph has associated a linked list of rows that encodes the measure of similarity between that path and the new given example. To compute the generalized topology from this structure, we traverse the list that embeds the longest of the possible subsequences, similarly with traversing the LCS table above. This process is efficient in terms of both computational time and space, as different paths "share" the parts of the table that are common to each other. For our example, the obtained generalized topology is not changed by incorporating the new example, as shown in Figure 4(b).

The generalization between multiple learned tasks (encoded as behavior networks) is performed at the level of their topological representations and provides a *generalized topology*. The underlying temporal dependencies between behaviors which are encoded in the behavior networks have to be further used to construct the *generalized behavior network* associated with it, as described in the next section.

4.2 Updating the network dependencies

In order to ensure proper behavior sequencing we need to transfer the temporal dependencies between behaviors to the generalized behavior network.

For any behaviors A and B belonging to the *generalized topology*, we compute the dependencies between them as follows:

- if A and B do not belong to the LCS, but are both part of the same task, take the dependency they had between them in that task,
- if both A and B are part of the LCS, (i.e., there are dependencies between them in both tasks), take the value that is the least restrictive as shown in Figure 5,
- if A and B are each part of a different underlying task, and if A is a predecessor of B in the topological representation, add an **ordering** constraint from A to B.

Net2 \ Net1	Ordering	Enabling	Permanent
Ordering	Ordering	Ordering	Ordering
Enabling	Ordering	Enabling	Enabling
Permanent	Ordering	Enabling	Permanent

Figure 5: Updating temporal dependencies between behaviors

4.3 Computing behavior preconditions in generalized topologies

In a simple behavior network (whose topology is only a sequence of behaviors and not a DAG), the task-dependent preconditions for a given behavior (the ones that depend on the execution of its predecessors) have the form of a conjunction between the status of all its predecessor behaviors.

In a generalized topology, since multiple alternate paths can exist to a particular behavior, the preconditions are encoded as combinations of conjunctions and disjunctions of the different paths. Thus, computing the preconditions for each behavior becomes equivalent to computing the regular expression from a FSA representation (Figure 4(b)).

For example, evaluating the preconditions for behavior F means checking that either the goals of A and C and B or those of just A and B are or have been true in accordance with the types of dependencies between them and behavior F, as given by the generalized behavior network computed above.

To summarize, the generalization process between two *behavior networks* is performed at the level of their *topological* representations, resulting in a *generalized topological structure*. The temporal dependencies between behaviors in the generalized task representation, which are encoded in a corresponding *behavior network*, are computed from the information contained in the underlying *behavior networks* involved in the generalization.

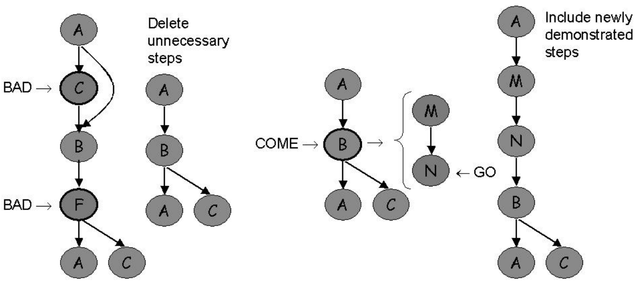
5. PRACTICE AND TEACHER FEEDBACK

Generalization over several training examples helps in identifying the steps that were observed most often and that most probably are a part of the task. However, repeated observations of irrelevant steps may inadvertently bias the learner toward including them in the representation. Also, limitations in the sensing capabilities of robots and particular structures in the environment may prevent the robot from observing steps that are relevant.

The *practice* trials allow the teacher to observe the execution of the robot and to point more accurately to where problems occurred. The following feedback cues can be given, with the associated effects:

- **"BAD"** - indicates that the behavior that the robot is currently executing, or the one that has just been finished (assuming a response time of 10sec) is irrelevant for the task. This behavior is then labeled as irrelevant and is removed from the task representation (Figure 6(a)).
- **"COME"** → **"GO"** - the robot has missed relevant steps of the task during its previous learning experiences. At a "COME" command the robot enters into the learning mode previously described, and starts following the teacher who demonstrates again

the missing part of the task. When these parts have been demonstrated, the teacher ends the short demonstration with "GO", after which the robot continues executing the remaining part of the task. The newly learned steps are incorporated into the task representation as presented in Figure 6(b). The arrow next to behavior B means that the "NEW" message was received while the behavior was active, or shortly after the behavior finished its execution. By intervening with feedback at this particular time, the teacher implies that the steps to be added should have happened before B's execution, as represented in the final task structure. This assumption is natural, since the teacher could not detect the problem until after it occurred. Alternatively, if the robot's actions carry enough intentional information to show that the robot is going to do something wrong, the teacher can also give feedback before allowing the robot to complete the current step.



(a) "BAD" feedback (b) "COME", "GO" feedback

Figure 6: Using feedback for task refinement

Both types of instructions can be applied at any time during the practice runs and and for as many times as the teacher considers it needed. This will be shown by our experimental results on learning from practice and teacher feedback (Section 6.2).

6. EXPERIMENTAL RESULTS

We implemented and tested our concepts on a Pioneer 2-DX mobile robot, equipped with two rings of sonars (8 front and 8 rear), a SICK laser range-finder, a pan-tilt-zoom color camera, a gripper, and on-board computation on a PC104 stack. We performed the experiments in a 5.4m x 6.6m arena. The robot was programmed using AYLLU [19], an extension of C for development of distributed control systems for mobile robots. For the voice commands and feedback we used an off-the-shelf Logitech cordless headset, and the IBM ViaVoice software recognition engine.

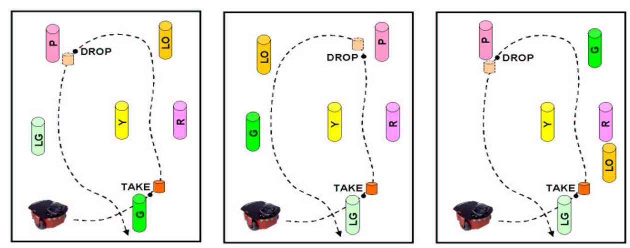
The robot has a behavior set that allows it to track cylindrical colored targets, to pick up, and drop small colored objects:

- **PickUp(ColorOfObject)** - the robot picks up an object of the color *ColorOfObject*. The goal state is achieved when the robot senses and has the object in the closed gripper.
- **Drop** - the robot drops what it has between the grippers. The goal state is achieved when there is nothing breaking the IR beams of the gripper.
- **Track(ColorOfTarget, GoalAngle, GoalDistance)** - the robot tracks a cylindrical target of the color *ColorOfTarget*. The goal state is achieved when the robot gets at *GoalDistance* and *GoalAngle* to the target. The robot has the ability to track such targets within a [0, 180] degrees field of view, by combining the information from the camera and the laser rangefinder. This enables the robot to track targets around it with the laser, even after they disappear from its visual field.

We performed two sets of robot teaching experiments to validate the key features of our proposed approach. First, we show how a robot can learn an object transport task through multiple demonstrations of the same task, in different environments, followed by a practice run during which the teacher gave very simple feedback. Second, we demonstrate how a similar transport task can be learned in only two steps: an initial demonstration and a single practice run, which in this case involves more complex teacher feedback. Section 6.1 presents the results obtained after the generalization steps, and Section 6.2 discusses the two different cases of task refinement through practice and feedback. Videos of all the experiments presented in the paper are available on the web at:
http://robotics.usc.edu/~monica/Research/generalization.html

6.1 Learning by generalization from several examples

We demonstrate the *generalization* abilities of the robot by teaching it an object transport task in three consecutive demonstrations, performed in different environmental setups (Figure 7), and purposely designed to contain incorrect steps and inconsistencies. The next section shows how already learned/generalized tasks can be further refined through *practice* and *feedback*. As discussed above, giving **"HERE"** cues during the demonstrations, does not help the robot in perfectly detecting the relevant parts of the task. In these three training experiments, solely for the purpose of demonstrating the *generalization* technique, we include all of the robot's observations into the learned task representations, to simulate that the robot was not able to discern the relevant aspects despite the teacher's instructions. The importance of such messages, however, will be shown in the *practice-feedback* experiments.



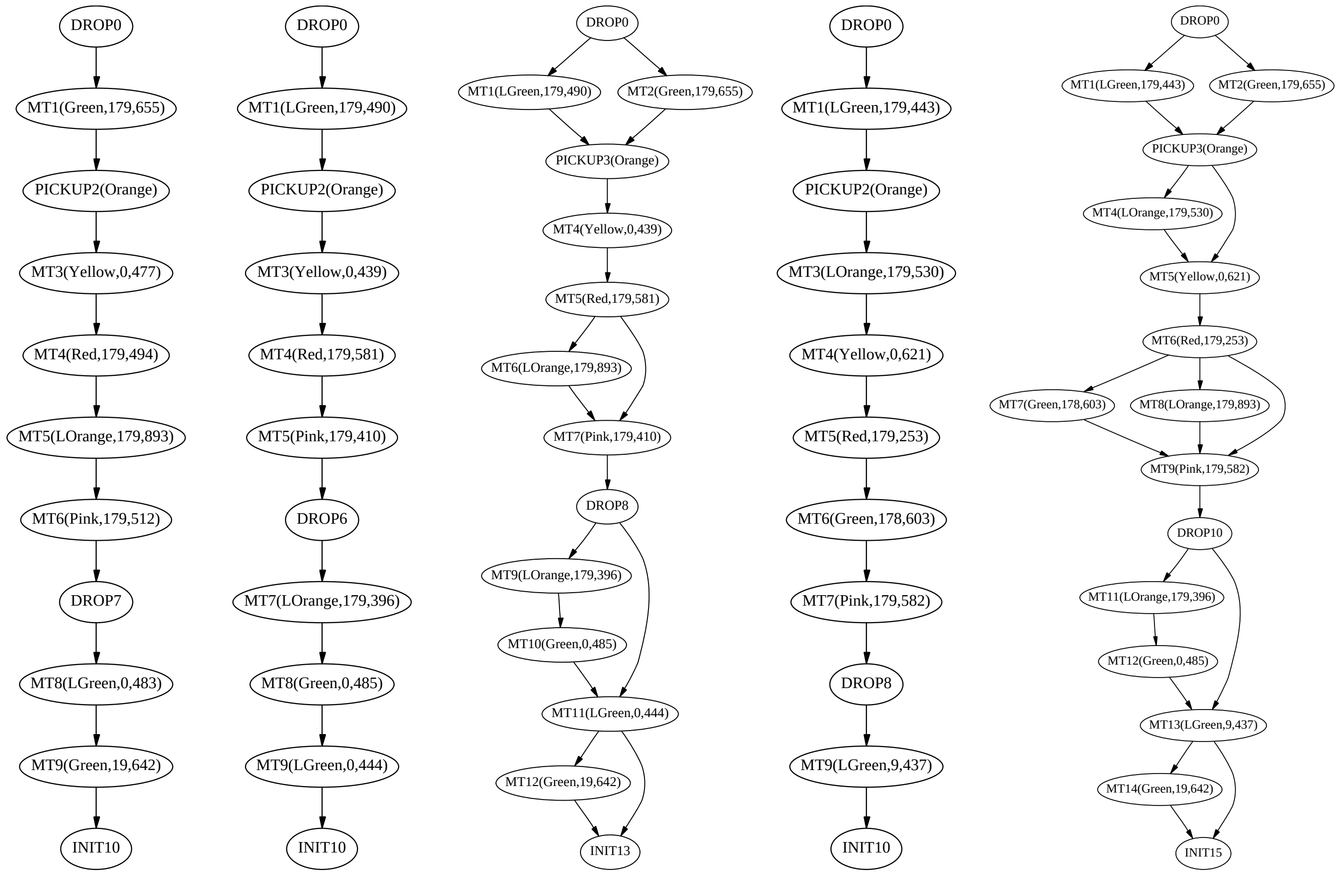
(a) First demonstration (b) Second demonstration (c) Third demonstration

Figure 7: Structure of the environment and course of demonstration

The environment consists of a set of cylindrical targets, in colors that the robot is able to perceive. The teacher leads the robot around these targets, while also instructing it when it has to pick up or drop a small orange box. The task to be learned is as follows: go to either the **Green (G)** or the **Light Green (LG)** targets, then pick up an **Orange (O)** box, go between the **Yellow (Y)** and **Red (R)** targets, go to the **Pink (P)** target, drop the box there, then go to the **Light Orange (LO)** target and come back to the target **Light Green**.

The sketched courses of the three demonstrations show that none of them corresponds exactly to the target task. Besides containing unnecessary steps (such as a final visit to a **Green** target in the first trial), these training runs also contain inconsistencies, such as the visits to the **Light Orange** target which happened at various stages during the demonstrations. Figure 8 presents the task representations obtained after each "learning → generalization" process. For all these experiments, in order to validate the correctness of the learned/generalized representations, after each teaching experience we had the robot execute them in the same environment in which they had been demonstrated. In all cases the robot performed the task correctly for the particular stage of the generalization process.

Each new demonstration is compared with the existing task structure, while computing their similarity in the form of their longest common sequence. Common nodes are then merged, while the others appear as alternate execution paths. Due to the generalization,



(a) First demonstration (b) Second demonstration (c) Generalized task (d) Third demonstration (e) Generalized task

Figure 8: Evolution of the task representation over three successive demonstrations

the following types of alternative paths can be obtained:

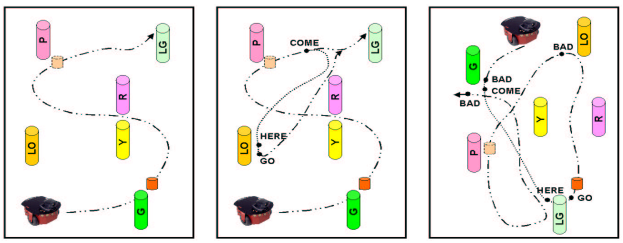
- both paths contain actual behavior(s). For example, Figure 8(c) encodes the fact that both going to the **Green** or to the **Light Green** targets is acceptable for the task. For such alternate paths, the robot will choose opportunistically between them, as induced by the state of the environment (e.g., go to the target seen first).
- one path is a direct link to the end of the other alternate sequence. In Figure 8(c), there is a direct link from **MT5(Red,...)** to **MT7(Pink,...)**, bypassing the behavior **MT6(LOrange,...)**. For such paths, the robot will automatically chose the direct path, shortcutting the alternate sequence. These unattainable paths could be removed from the graph, but we are keeping them for informative purposes. Also, we can envision extensions in which teacher feedback could eliminate such direct links ("marking" as wrong certain transitions from one step to another).

The generalization method has the advantage that it compactly encodes (in the form of an acyclic graph) the actual "rules" that can be drawn from the multiple demonstrations. The generalized task captures the main structure of the task while at the same time dealing with the irrelevant and inconsistent parts of the demonstrations: both of these situations are captured as becoming a part of a bypassed alternate paths which will never be executed. While it is good that the irrelevant actions are thus pruned, the steps demonstrated inconsistently but which are still necessary will have to be included by different means. These results are to be expected: *generalization* alone, when provided with inconsistent examples, is not enough for learning a correct representation. The next section shows how *practice* and *teacher feedback* can be used for solving this problem.

6.2 Learning from practice and teacher feedback

In order to demonstrate the robustness of our architecture to changing environments and the advantages of learning high-level representations of tasks, we had the robot execute the last generalized network (Figure 8(c)) in a different environment than any of the three presented before (Figure 9(a)).

The robot correctly executed that task in the new setup. However, as mentioned before, the generalized network does not yet represent the target task desired by the user. The missing part is a visit to the **Light Orange** target, which should happen right after dropping the box and before going to the **Light Green** target. Since the generalization process already built the remaining of the task structure, simple feedback during a robot practice run is enough for refining it to the desired structure. We performed the practice run in the newly changed environment: Figure 9(b) shows the robot's sketched trajectory and (dotted) the teacher's intervention. After dropping the box at the destination **Pink** target, the robot started servoing toward the **Light Green** target. Observing this tendency, the teacher intervened (**"COME"**): the robot switched to learning mode, and followed the teacher who lead it to the missed **Light Orange** target. The use of the informative feedback cues (**"HERE"**) during this learning stage was essential, as the robot also passed by and detected other targets (**Pink** and **Yellow**) while following the teacher, and which thus have been ignored. After demonstrating the missing step, the teacher signaled the end of the "learning" intervention (**"GO"**) and the robot continued and finished the task by going to the remaining **Light Green** target. Figure 10(a) shows the structure of the task after this practice run. The newly added steps are marked on the graph: they also include a **Drop** behavior, as the robot had nothing in the gripper at the point of the demonstration, and therefore the goals of this behavior have also been detected as true. At the time of the execution, the existence of this behavior will have no influence, since the robot would have already dropped the object at this point.



(a) Generalized task: new environment (b) Feedback after third demonstration (c) Feedback after first demonstration

Figure 9: Structure of the environment and course of task execution during practice and feedback

We now consider an alternate approach for instruction, starting from the first demonstration in the previous section. Let us assume that for a second object transport task the teacher considers as wrong the initial visit to a **Green** target, when a **Light Green** target should be visited instead. Also, the visit to the **Light Orange** target is wrong, and not a part of the task as well. Figure 9(c) shows the trajectory of the robot and (dotted) the intervention and messages of the teacher during the robot's practice run. The effects of this feedback are that: the visit to the **Green** target was replaced by a visit to the **Light Green** target, and the visits to the **Light Orange** and **Green** have been removed. Figure 10(b) presents the structure of the task after this practice run.

To validate the correctness of the learned representations, we had the robot execute the tasks learned after both of the practice runs described above: in each case the execution proved that the robot correctly adapted its task representations according with the teacher's feedback, which thus matched the target tasks desired by the user.

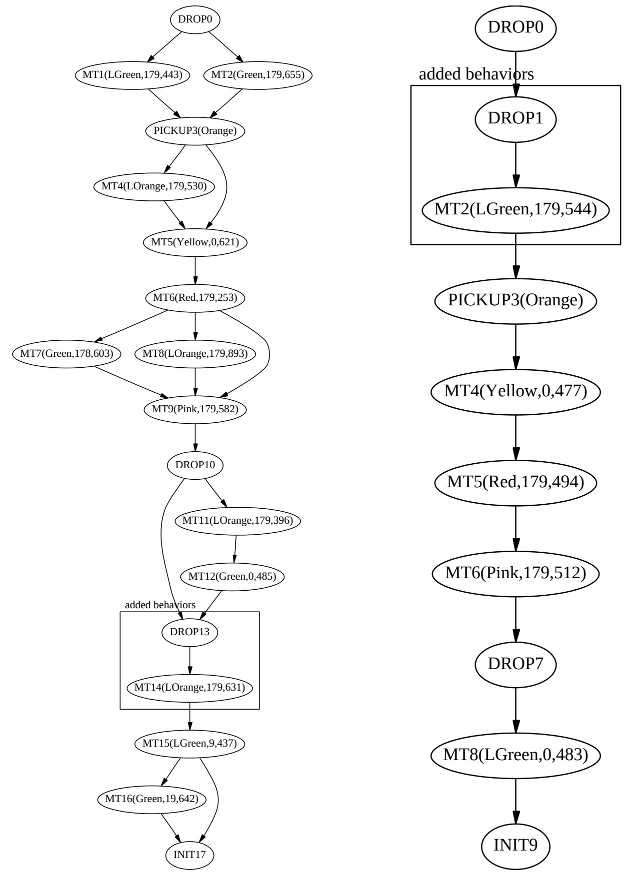
We observe that the *practice-feedback* runs are a much faster and precise method for refining previously learned tasks. Since feedback can be given at any step during the robot's practice, wrong steps can be marked immediately upon observing them being executed, and missing steps can be added as soon as the teacher detects that the robot had skipped to a future step in the task.

7. DISCUSSION

The experimental results validated the ability of our approach to incorporate multiple means for instruction and learning in order to teach robots long, complex, and sequentially structured tasks.

Also, we showed that *generalization* and *feedback* can be used interchangeably in various combinations, providing the teacher the flexibility to instruct the robot in the manner considered most suited for each case.

An important feature of the practice-feedback approach that we need to stress is the natural characteristic of this process. In order to give the robot appropriate feedback, the teacher doesn't need to know the structure of the task being learned, and thus is shielded from having to know any details about the robot's control architecture. Instead, he simply relies on observing the actions performed by the robot: if they comply with the desired representation, no feedback is given, and if they do not, the corresponding situations are treated with appropriate feedback as described in our experiments.



(a) After the 3rd demonstration (b) After the first demonstration

Figure 10: Topologies obtained after practice and feedback

8. RELATED WORK

Successful approaches for acquiring high-level task information from demonstration have been developed for robotic manipulators learning assembly problems [10], [7]. Since they rely solely on passive observations of a teacher demonstration, these methods have to make use of complex computer vision techniques, in carefully structured environments, in order to infer all the information necessary for the task.

In the mobile robot domain, the majority of the robot teaching by demonstration approaches have mostly been focused on learning reactive policies, collections of reactive rules that map environmental states to robot actions. [6] demonstrates learning to navigate a maze (i.e., learning forward, left, and right turning behaviors). [12] presents a behavior-based approach for learning reactive motor behaviors (door passage, wall following) and outlines a strategy for learning history-dependent behaviors. A very interesting aspect of this work is that "teaching" can be performed by an already existing behavior running on the robot: this enables *behavior cloning*, in which the same functionality can be obtained by using different sensors for input.

Similarly with our approach, [18] assumes the existence of a set of primitive capabilities, from which a more complex controller can be built through demonstration. Using PCA (Principal Component Analysis), primitives such as guarded moves and edge-to-surface alignment for a robotic arm can be learned and subsequently recognized during further demonstrations.

The above techniques, however, use demonstration as the only means for teaching, and do not benefit from the advantages that could be gained from using additional instruction abilities. Furthermore, the complexity of the tasks that are learned is limited to reactive policies or short sequences of sensory-motor primitives. Our approach allows for learning of high-level tasks that involve arbitrarily long sequences of behaviors.

Methods for robot task teaching that consider additional instructive modalities in addition to demonstration have also been proposed. [8] presents an approach in which *good/not good* feedback was given at the end of a run in which the robot performed the demonstrated skill. This approach also considers the refinement of learned skills by practice, by using an exploration element which alters the skill during execution. The *good/not good* feedback was used to assess the quality of the exploration. However, giving such delayed reward generates problems of credit assignment. In contrast, by giving feedback during or right after a wrong task step occurred, our approach enables the robot to precisely identify the irrelevant actions.

[4] considers fusing user intention with demonstration information as additional means for instruction. The approach enables the robot to successfully learn the correct task, but may become burdensome for the teacher as he needs to provide (at each step) information on what goals he had in mind, and what actions/used objects were relevant. In contrast, our approach relies solely on the teacher's observation of the robot's execution during practice.

9. CONCLUSIONS

Learning capabilities are essential for successful integration of robots in human-robot domains, so robots can learn from human demonstrations and allow for natural interaction with people. Due to inherent challenges of the learning process, it is also important that robots be able to improve their capabilities by receiving additional training and feedback. Toward this end, we presented an approach for teaching by demonstration inspired from the one humans use with each other, to enable a robot to learn and refine representations of complex tasks. By using simple relevant *cues* we enable the robot to distinguish between relevant and irrelevant information during the learning process. Concise *instructions* allow for a richer demonstration, by actively involving the robot in the process. Through *generalization*, the robot can incorporate several demonstrations of the same task into a single graph-like representation. Natural *feedback cues* provided by the teacher through speech allow the robot to further refine this representation. We demonstrated these concepts on a Pioneer 2DX mobile robot, learning various tasks from demonstration, generalization, and practice.

10. REFERENCES

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, CA, 1998.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[3] Y. Demiris and G. Hayes. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In K. Dautenhahn and C. Nehaniv, editors, *Imitation in Animals and Artifacts*. MIT Press, 2002.

[4] H. Friedrich and R. Dillmann. Robot programming based on a single demonstration and user intentions. In *Proc., 3rd European Workshop on Learning Robots*, 1995.

[5] P. Garcia and E. Vidal. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):920–925, 1990.

[6] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proc. of the Intl. Symp. on Intelligent Robotic Systems*, pages 198–204, Grenoble, France, 1994.

[7] K. Ikeuchi and T. Suehiro. Towards an assembly plan from observation. In *Proc. of ICRA*, pages 2171–2177, 1992.

[8] M. Kaiser. Transfer of elementary skills via human-robot interaction. *Adaptive Behavior*, 5(3-4):249–280, 1997.

[9] F. Kaplan, P.-Y. Oudeyer, E. Kubinyi, and A. Miklûsi. Robotic clicker training. *Robotics and Autonomous Systems*, 38:197–206, March 2002.

[10] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *T-RA*, 10:799–822, 1994.

[11] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein. Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 38:171–181, March 2002.

[12] K. S. Michael Kasper, Gernot Fricke and E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, 34:153–164, February 2001.

[13] M. N. Nicolescu and M. J. Matarić. Learning and interacting in human-robot domains. In C. C. White and K. Dautenhahn, editors, *IEEE Transactions on Systems, Man, and Cybernetics,Part A: Systems and Humans, Special Issue on Socially Intelligent Agents - The Human in the Loop*, volume 31, pages 419–430. IEEE, September 2001.

[14] M. N. Nicolescu and M. J. Matarić. A hierarchical architecture for behavior-based robots. In *Proc., First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna, ITALY, July 2002.

[15] E. V. P. Garcia and F. Casacuberta. Local languages, the successor method and a step towards a general methodology for the inference of regular grammars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):841–845, 1987.

[16] S. Schaal. Learning from demonstration. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 1040–1046. MIT Press, Cambridge, 1997.

[17] R. Voyles and P. Khosla. A multi-agent system for programming robotic agents by human demonstration. In *Proc., AI and Manufacturing Research Planning Workshop*, August 1998.

[18] R. M. Voyles, J. D. Morrow, and P. K. Khosla. Towards gesture-based programming. *Robotics and Autonomous Systems*, 22:361–375, November 1997.

[19] B. B. Werger. Ayllu: Distributed port-arbitrated behavior-based control. In *Proc., The 5th Intl. Symp. on Distributed Autonomous Robotic Systems*, pages 25–34, Knoxville, TN, 2000. Springer.