Learning task representations from experienced demonstrations

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

Abstract

We present an approach for building high-level task representations from a robot's own experiences of interacting with a teacher. A learner robot follows a human teacher and maps its own observations of the environment to the known effects of its available skills, building at run-time a representation of the experienced task in the form of a behavior network. The learner can act next as a teacher, and transfer the acquired knowledge to other robots. To enable this we introduce an architecture that extends the capabilities of behavior-based systems by allowing the representation and execution of complex and flexible sequences of behaviors. We demonstrate our approach in a set of experiments in which robots learn representations for multiple tasks from both human and robot teachers.

Introduction

There are various strategies for representing a robot's observations, actions and environmental states. For robots with an already given set of capabilities, the problem of particular interest is that of acquiring and executing representations of complex tasks based on the available skill repertoire.

One of the most common methods for transferring task knowledge is teaching by demonstration. The majority of the approaches to this problem to date has been limited to learning policies, collections of reactive rules that map environmental states to actions. We are interested in developing a mechanism that would allow robots to learn representations of high level tasks, based on the underlying capabilities already available to the robot. More specifically, instead of having to write, by hand, a system that achieves a particular task, we want to allow a robot to automatically build it from the experience it had while interacting with a human or a robot teacher. We propose to achieve this through a process of anchoring the robot's observations of the environment to the robot's own skills that can achieve the same observed environmental states. The mappings created during a human-teacher demonstration experience are then combined into a high-level representation of a task.

It is particularly apt to address this problem in behavior-based systems (BBS), where representation has not been studied extensively (Matarić 1992), yet whose robust and adaptive properties are suitable to the human-robot interaction domain. Toward this goal, we have developed a behavior representation that extends the capabilities of BBS and addresses some of their limitations.

The first step we focus on is to use the flexibility of the representation to enable a robot to learn high-level complex tasks from the experience of interacting with a teacher. We demonstrate our approach on an object delivery task, which, if designed by hand, would require complex sequencing and complex logic activation conditions, typically hard to represent (and even harder to learn) within a behavior-based framework. In our earlier work (Nicolescu & Matarić 2001) we have described examples of robots learning a variety of different tasks, in a clean environment, in which only the information relevant to the task was present. In this paper we extend this approach to environments in which various distractors are present. To solve this problem we allow the teacher to indicate salient times when the robot should focus its attention on the environment. These indications of the teacher are general (simple hints like *"pay attention now"*) and by no means spell out for the robot the representation of the presented task.

As a next step, we are also interested in analyzing the ability of a robot to learn from a former learner robot, in order to facilitate the transfer of acquired knowledge from human to robots, then further to other robots, and so on. A learner robot uses the same strategy for both human and robot teachers. As expected, the experiments showed that the performance of learning from a human is superior to the one of learning from another robot, due to differences in demonstration accuracy.

The remainder of the paper is organized as follows: first we describe the specifics of the anchoring problem in the teaching by demonstration domain and present our behavior representation and the behavior network construct that uses them to represent general strategies and plans. Next we explain the algorithm for learning

task representations from experienced interaction with human or robot teachers, and demonstrate our experimental results. We discuss the relevant previous work in this area and conclude with a summary and directions for future research.

Anchoring task representations in observations

The process of learning high-level task representations is based on a teacher demonstration stage, during which the robot experiences the task through its own sensors.

The general idea for constructing these representations is to create a mapping between the robot's observations of the environment and the known effects of its own actions (in our case, behaviors). In this particular case of anchoring the connection is made between environmental states at the physical (sensory input) level and high-level robot skills (through their corresponding postconditions, abstracted environmental states) at the abstract level. The robot identifies activities that it is able to perform by detecting in the environment states that match the effects its own behaviors. In the next section we describe how this can be achieved within a behavior-based framework.

The observations-skills mappings are combined to construct high-level representations of robot tasks; the algorithm is presented in the Section **Learning from demonstrations**. These representations are not *symbolic*, but rather they are built from robot skills and take the form of *behavior networks*, described in the next section. The networks encode a robot-experienced task at a high level of abstraction.

The observations also provide information about objects in the environment relevant to particular skills from the robot's behavior set. Through the same anchoring process these objects are linked to the behaviors that operate on them. The objects do not have any *symbols* assigned, but they are linked to behaviors through their properties: observed physical features, such as *color*, *size*, etc.

To summarize, the anchoring process in our teaching by demonstration domain is characterized by:

- grounding high-level task representations to observations in the form of behavior networks;
- linking objects to robot behaviors through the objects' observed physical features.

Next we describe our behavior architecture and the network representation of robot tasks.

Behavior representation

We are using a behavior-based architecture that allows the construction of the robot task in the form of a behavior network (Nicolescu & Matarić 2000), and provides a simple and natural way of representing complex sequences of behaviors. In a behavior network, the links between behaviors represent precondition-postcondition dependencies; 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*). By separating these different types of preconditions and testing the task relevant (sequential) preconditions via the network links, we give more generality to the behaviors and allow them to be reused without redesign for any different task. Each behavior has a representation of its goals (abstracted environmental states) and continuously computes and updates the *met/not met* status of these goals (on the *Effects* output) in order to be available for the successor behaviors (at the *Precondition* input) (see Figure 1). Embedding goal representations in the behavior architecture is a key feature of our behavior networks and, as we will see, for learning of task representations.

We distinguish between three types of *sequential preconditions* which determine the activation of behaviors during the behavior network execution:

- **Permanent preconditons**: preconditions that must be met during the entire execution of the behavior. Any change from met to not met in the state of these preconditions will automatically deactivate the behavior. They enable the representation of sequences of the type: *the effects of some actions must be permanently true during the execution of a behavior.*
- **Enabling preconditions**: preconditions that must be met only once, immediately before the activation of a behavior. Their state can change during the behavior execution, without influencing the activation of the behavior. They enable the representation of sequences of the type: *the achievement of some effects is sufficient to trigger the execution of a behavior.*
- **Ordering preconditions**: preconditions that must have been met at some point before the behavior is activated. They enable the representation of sequences of the type: *some actions must have been executed before this behavior could be executed.*

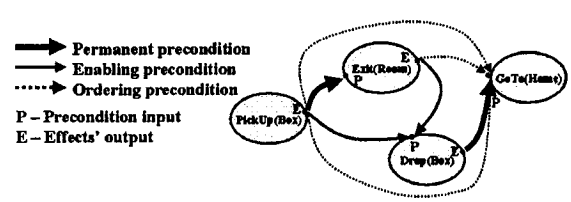


Figure 1: Example of a behavior network

In a network, a behavior can have any combination of the above preconditions. Figure 1 represents a generic behavior network and the three types of precondition-postcondition links presented above.

In the behavior networks we present later in the paper, we always use a default behavior **Init** whose role it is to initiate the network links and detect the completion of the task. Therefore, **Init** has as *predecessors* all the behaviors in the network.

All the behaviors are continuously *running* (i.e. performing the computation described below), but only

one behavior is *active* (sending commands to the actuators) at a given time.

Similar to (Maes 1990), we employ a continuous mechanism of spreading of activation from the behaviors that achieve the final goal to their predecessors (and so on), as follows: each behavior has an *Activation level* that represents the number of successor behaviors in the network that require the achievement of its postconditions. Any behavior with activation level greater than zero will send activation messages to all the predecessor behaviors that do not have (or have not yet had) their postconditions met. This activation level is set to zero after each execution step, so that at the next step it could be properly re-evaluated, in order to respond to any environmental changes that might have occurred.

The activation spreading mechanism works together with precondition checking in order to determine whether a behavior is active (and thus able to execute its actions), as follows. Activate the behavior if:

(*Activation level* != 0) AND
(All **ordering preconditions** = $TRUE$) AND
(All **permanent preconditions** = $TRUE$) AND
((All **enabling preconditions** = $TRUE$) OR
(the behavior was active in the previous step)).

The behavior network representation has the advantage that it can adapt to environmental changes, whether they be favorable (achieving the goals of some of the behaviors, without them being actually executed) or unfavorable (undoing some of the already achieved goals). Since the conditions are continuously monitored, the system continues with execution of the behavior that should be active according to the environmental state (either jumps forward or goes back to a behavior that should be re-executed).

Learning from demonstrations

The demonstration process

In a demonstration, the robot follows a human/robot teacher and gathers observations from which it constructs a task representation. The ability to learn from observation is based on the robot's ability to relate the observed states of the environment to the known effects of its own behaviors.

In this *learning* mode, the robot follows a human teacher, while all its available behaviors are continuously monitoring the status of their postconditions (without executing any of their actions). Whenever a behavior signals the achievement of its effects, this represents an example of the robot having seen something it is also able to do. The fact that the behavior postconditions are typically abstracted environmental states allows the robot to interpret high-level effects (such as approaching a target, a wall, or being given an object).

Thus, embedding in behaviors representations of their goals provides the information needed for the learning process: without it, using only executable behaviors, the robot would not be aware of their effects (as is usually the case in behavior-based systems), and thus could not directly recognize the experiences that it can also perform. However, if the robot is shown actions for which it does not have any representation, it will not be able to observe or learn from those experiences. For the purposes of our research, it is reasonable to accept this constraint; we are not aiming at teaching a robot new behaviors, but at showing the robot how to use its existing capabilities in order to perform more complicated tasks.

To enable a robot to distinguish between irrelevant and relevant observations, the teacher is allowed to signal points in time when the environment presents aspects relevant to the task. The robot will consider any behaviors whose observed effects are achieved at that time to pertain to the task and include them in the task representation. The human teacher points out saliencies by showing a bright color marker that can be detected by the robot's vision system. A robot teacher simply broadcasts a simple one bit message when it has just accomplished execution of one of the behaviors in the task being demonstrated. The marker and the binary message carry the same information, that of considering the observations of the environment as relevant to the demonstrated task.

Next, we present the algorithm that constructs the task representation from the observations the robot has gathered during the demonstration.

Building the task representation from observations

During the demonstration, the robot acquires the status of the postconditions for all of its behaviors, as well as the values of the relevant behavior parameters. For example, for the parameterizable **Track** behavior, which takes as parameters a desired angle and distance to a target, the robot will continuously record the observed angle and distance whenever the target is visible (i.e. behavior's postconditions are true). The last observed values are the ones that the robot keeps as learned parameters for that behavior. We take this approach since we consider that the last moment when a behavior's postconditions are true is the most representative for the values that its parameters should take. While this may not be true in some cases, it gives proper results for our behavior set.

Before describing the algorithm, we present a few notational considerations. Suppose a behavior A, whose postconditions are true within the interval $[t1_A, t2_A]$ and a behavior B, that is active within the interval $[t1_B, t2_B]$ (see Figure 2).

- If $t1_B \geq t1_A$ and $t1_B \leq t2_A$, behavior A is a predecessor of behavior B. Moreover, if $t2_B \leq t2_A$, the postconditions of A are permanent preconditions for B (case 1). Else, the postconditions of A are enabling preconditions for B (case 2).
- If $t1_B > t2_A$, behavior A is a predecessor of behavior B and the postconditions of A are ordering preconditions for B (case 3).

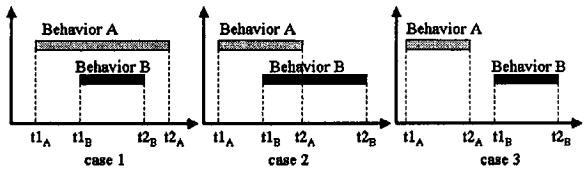


Figure 2: Precondition types description

Each node in a learned behavior network representation maintains the following information: behavior type, a unique ID (to differentiate between possible multiple **instances** of the same behavior), observed values of the behavior's parameters, interval of time I during which the behavior's postconditions have been true, and a flag that shows whether the teacher has indicated any saliencies within I.

The general idea of the algorithm is to add to the network task representation an instance of all behaviors whose postconditions have been true during the demonstration, in the order of their occurrence. 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 find if these effects have been active in overlapping intervals or in sequence. Based on the above information and according to the notational considerations presented above, the algorithm generates the proper network links (i.e., precondition-postcondition dependencies).

Behavior network construction

/* Online processing */

1. At each time step, for each behavior:

- *If the behavior's postconditions have just become true:*

  ⇒ Add to the behavior network an instance of of the behavior it corresponds to. (Along with it, save the time step as the start of behavior activation.)

- *Else, if the behavior's postconditions are true and have previously been true:*

  ⇒ Update the corresponding behavior in the network with its current parameter values, computed from observations, and any teacher indicated saliency.

- *Else, if the behavior's postconditions have just become false:*

  ⇒ If in the network there is any previous behavior of the same type with an ending time within some ϵ from the starting time of the current behavior, merge the two behaviors (updating the information carried with the network nodes accordingly).

/* Off-line processing (at the end of the demonstration)*/

2. Filter the network in order to eliminate false indications of some behavior's effects. (These nodes can be detected for having very small durations or unreasonable values of behavior parameters.)

3. For each node, representing a behavior instance J:
   For each node, representing a behavior instance K added to the network at a later time:
   Compare the end-points of the interval I_j (corresponding to behavior J) with those of interval I_k (corresponding to behavior K):

- If $t2_j \geq t2_k$, then postconditions of J are **permanent preconditions** for K (case 1). Add this permanent link to behavior K in the network.
- If $t2_j < t2_k$ and $t1_k < t2_j$, then postconditions of J are **enabling preconditions** for K (case 2). Add this enabling link to behavior K in the network.
- If $t2_j < t1_k$, then postconditions of J are **ordering preconditions** for K (case 3). Add this ordering link to behavior K in the network.

Experimental results

To validate the capabilities of the system we have described, we performed several evaluation experiments that demonstrate the ability of a robot to learn high-level task representations from both human and robot teachers.

We implemented and tested our concepts on two Pioneer 2-DX mobile robots, equipped with two rings of sonars (8 front and 8 rear), a SICK laser range-finder, a pan-tilt-zoom color camera, a gripper (only for one of the robots), and on-board computation on a PC104 stack. We performed the experiments in a 5.4m x 6.6m arena[1]. The robots were programmed using AYLLU (Werger 2000), an extension of C for development of distributed control systems for mobile robots.

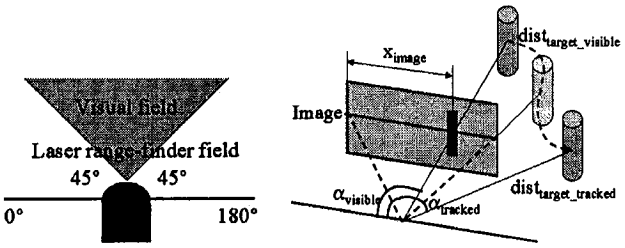
Learning from demonstration

We have designed two different experiments which rely on navigation and object manipulation capabilities of the robots. First, we focus on the performance of learning from human teachers and second we address the issue of knowledge transfer between robots, in robot(teacher)-robot(learner) demonstration experiments.

Initially, the robot was given a behavior set that allowed it to track colored targets, open doors, pick up, drop, and push objects. The **Track** behavior allows the robot to follow colored targets at any distance in the [30, 200] cm range and any angle in the [0, 180] degree range. The information from the camera is merged with data from the laser range-finder in order to allow the robot to track targets that are outside of its visual field (see Figure 3). The robot uses the camera to detect the target in the first place and then it continues to track the same target after it goes out of the visual field. As long as the target is visible to the camera, the robot uses its position in the visual field (x_{image}) to infer an

[1]Videos of the experiments are available at http://robotics.usc.edu/~monica/Research/learning.html

approximate angle to the target $\alpha_{visible}$ (the "approximation" comes from the fact that we are not using precise calibrated camera data and we compute it without taking into consideration the distance to the target). We get the real distance to the target $dist_{target\_visible}$ from the laser reading in a small neighborhood of the $\alpha_{visible}$ angle. When the target disappears from the visual field, we continue to track it by looking in the neighborhood of the previous position in terms of angle and distance which are now computed as $\alpha_{tracked}$ and $dist_{target\_tracked}$. Thus, the **Track** behavior gives the robot the ability to keep track of positions of objects around it, even if they are not currently visible.



(a) Sensor coverage (b) Principle for target tracking

Figure 3: Merging laser and visual information for tracking

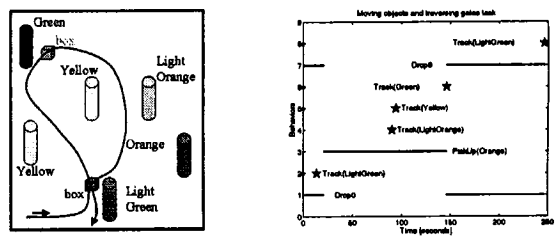
Learning from humans The goal of the experiments presented in this section is to demonstrate the ability of the robots to learn representations of complex tasks that involve navigation and object manipulation capabilities (i.e., picking up and dropping objects). The experiments also show that the learning algorithm allows for learning higher-level behaviors such as going through a door/gate based on simpler behaviors already available to the robot.

The setup for this experiment is presented in Figure 4(a). The task to be learned by the robot is the following: pick up the orange box placed near the light green target (the source), go through the "gate" formed by the yellow and light orange target, drop the box at the dark green target (the destination) and then come back at the source target. The orange and the yellow target at the left are distractors that should not be considered as belonging to the task. In order to teach the robot that it has to pick up the box, the human led the robot to it and then, when sufficiently near it, placed it between the robot's grippers. At the destination target, the teacher took the box from the robot's grippers. Moments in time signaled by the teacher as being relevant to the task are: giving the robot the box while close to the *light green* target, teacher reaching the *yellow* and *light orange* target, taking the box from the robot while at the *green* target, and teacher reaching the *light green* target in the end. Thus, although the robot observed that it had passed the *orange* and distant *yellow* targets during the demonstration, it did not include them in its task representation, since the teacher did not signal any relevance while being at them.

We performed 10 human-robot demonstration experiments to validate the performance of the **behavior network construction** algorithm. We evaluated each learned representation both by inspecting it structurally and by having the robot perform it, to get physical validation that the robot learned the correct task. In 9 of the 10 experiments the robots learned a structurally correct representation (sequencing of the relevant behaviors) and also performed it correctly. In one case, although the structure of the behavior network was correct, the learned values of one of the behavior's parameters caused the robot to perform an incorrect task (instead of going between two of the targets the robot went to them and then around). The learned behavior network representation of this task is presented in Figure 5. The labels of the nodes represent a combination between a behavior's name, parameters (if any) and an ordering ID. For example, *MTLGreen1* refers to behavior **MixedTracker** for a *light green* target.

As a base-case scenario, to demonstrate the reliability of the learned representation and that learned performance does not degrade over time, we performed 10 trials, in which a robot repeatedly executed one of the learned representations of the task presented above. In 9 of the 10 cases the robot correctly completed the execution of the task. The only failure was due to a timeout in executing the **TrackLGreen** behavior; the robot failed to find that target within a pre-determined amount of time.

In Figure 4(b) we show the robot's progress during the execution of a successful task, more specifically the instants of time or the intervals during which the postconditions of the behaviors in the network were true.



(a) (b)

Figure 4: a) Environment for the **Object manipulation** task; b)Achievement of behavior postconditions

During this experiment, all three types of behavior preconditions were detected: during the demonstration the robot is carrying an object for the entire time while going through the "gate" and tracking the destination target, and thus the links between **PickUp** and the behavior corresponding to the actions above are **permanent** preconditions (case 1 of the learning algorithm). **Enabling** precondition links appear between behaviors

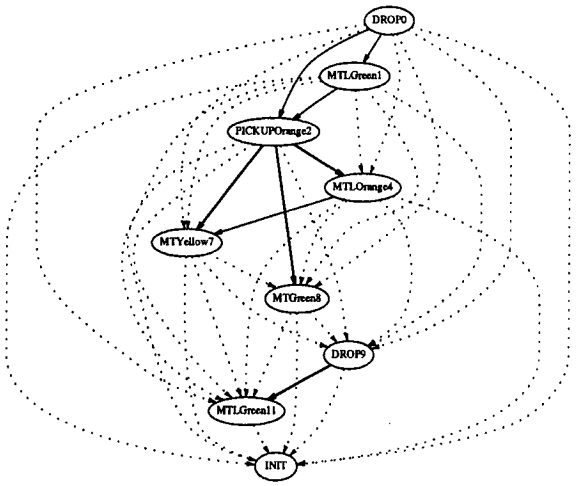


Figure 5: Task representation learned from human demonstration for the **Object manipulation** task

for which the postconditions are met during intervals that only temporarily overlap, and the **ordering** preconditions enforce a topological order between behaviors, as it results from the demonstration.

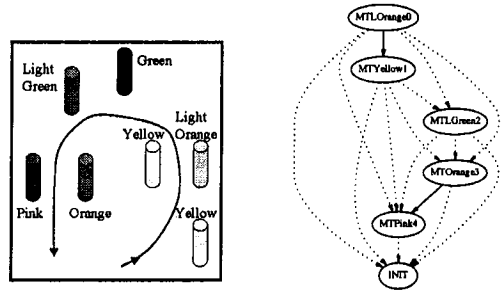
The ability to track targets within a [0, 180] degree range allows the robot to learn to naturally execute the part of the task relative to going through a "gate". The way this experience is mapped into the robot's representation can be described as follows: "track the light orange target until it is at 180 degrees (and 40cm) with respect to you, then track the yellow target until it is at 0 degrees (and 50cm)" . At execution time, since the robot is able to track both targets even after they disappeared from its visual field, the goals of the above **Track** behaviors were achieved with a smooth, natural trajectory of the robot passing through the "gate".

Learning from robots In this section we extend the problem of learning task representations to the case of learning from robot teachers. We are interested in determining the reliability of the information that is passed around from robots to robots by means of teaching by demonstration.

We ran experiments to determine the task transfer rate (TTR), the number of successful transmissions of the same task from a teacher to a learner. A TTR of k means that the task was transmitted from the original demonstrator (usually a human) $k-1$ times, until the failure point. This variable follows a geometric distribution, for which we determine the expected mean value and the confidence interval (Clemans 1959).

The task selected for the experiments is to go through a "gate" formed by the yellow and light-orange targets (Figure 6(a)), visit the light-green target, and come back through the pink and orange targets. Two distractor targets (green at the top and yellow at the right bottom corner) were present in the environment, which the robots had to ignore during the learning and the execution process.

We performed three human-led demonstrations, from which a learner robot correctly built the task representation each time. As a base case, to show that the performance of the robot does not degrade over time for the same task representation, we performed 10 trials in which a robot repeatedly executed the above task. In all 10 trials the robot correctly executed the task.



(a) (b)

Figure 6: a) Environment for the robot-robot teaching task; b) Task representation learned from robot demonstration for the **Visiting targets** task

Next, we performed 10 trials in which two robots, starting from a correctly learned task, switched roles in teaching each other the same task, each time using the information acquired at the previous step. Figure 6(b) presents the correct learned behavior network for this task. For each of the above trials we recorded the number of successful teaching experiences until the learning broke down. The maximum and minimum number of teaching experiments before learning failed were 6 and 2 respectively. The observed mean for the TTR obtained from the experiments is 2.5, with a 98% confidence interval of [1.4 8]. As the statistical evaluation shows, any information learned from a human can be further transfered to other robots at least one more time in the worst case, despite the naive approach we have employed for the robot teacher.

The major factor that determined the difference between the performance obtained in the case of a human versus a robot teacher is the quality of the demonstration: a human facilitates a learner's observations, whereas the robot teacher has to wonder around searching, due to its own limited sensing capabilities. As a result, the learner's observations are skewed by the teacher's "hesitant" demonstration.

Discussion The presented approach allows a robot to automatically build reliable task representations from only one trial. Furthermore, the tasks the robot is able to learn can embed arbitrarily long sequences of behaviors, which become encoded within the behavior network representation. Any behavior can be used with-

out customization, with different activation conditions, within the same or across different tasks, thus increasing behaviors reusability and decreasing controller design efforts.

The robots are able to perform in more complex environments and to infer and eliminate the irrelevant observations using only very simple cues from the teacher. While this approach does not completely eliminate irrelevant environment state from being observed, it biases the robot to notice and (if capable) capture the key elements. We are currently investigating techniques for generalization across multiple demonstrations of the same task, performed in changed environments and in the absence of cues provided by a teacher.

Related work

The ability to represent and execute sequences is necessary for learning the types of tasks we are interested in teaching robots. This is particularly relevant in the behavior-based framework, where sequential behavior is usually trigerred through the world, rather than through internal sequences (Arkin 1998).

By augmenting the behaviors with representations of their goals, we take advantage of both the ability of the deliberative, STRIPS-like architectures to operate at high-level of abstractions, and the robustness of BBS. The common approach to bridging the gap between these architectures is the use of the hybrid (or *three-layer systems*) systems (Gat 1998), which need a middle layer to solve the difference in representation and time-scales between the physical and the abstract levels. Our architecture achieves this goal using behaviors with the same representation and time scale.

An early example of embedding representation into BBS was done by (Matarić 1992). The representation was also constructed from behaviors, and was used exclusively for mapping and path planning. While the approach successfully integrates deliberative capabilities into a BBS, it is limited to the navigation task, while our representations are meant to be task-independent and could embed any general behaviors representing the robot's capabilities: in our case, both navigation and object manipulation.

In the context of behavior-based robot learning, most approaches have been at the level of learning policies, situation-behavior mappings, at least in physical robot domain. The method, in various forms, has been successfully applied to single-robot learning of various tasks, including hexapod walking (Maes & Brooks 1990), box-pushing (Mahadevan & Connell 1991), most commonly navigation (Dorigo & Colombetti 1997), and also to multi-robot learning (Matarić 1997b).

In the context of teaching by demonstration, (Hayes & Demiris 1994) demonstrated simplified maze learning (i.e., learning forward, left, and right turning behaviors). (Schaal 1997) used model-based reinforcement learning to speed-up learning for a system in which a 7 DOF robot arm learned the task of balancing a pole from a brief human demonstration. These approaches focus on the **action** imitation level (resulted in reactive policies), while we are concerned with representing and repeating high-level tasks with sequential and/or concurrently executing **behaviors** which embed history (the ordering of behaviors' execution).

A connectionist approach to learning from human or robot demonstrations which also addresses the problem of sequence learning is presented in (Billard & Dautenhahn 1998). The architecture allows the robots to learn a vocabulary of "words" representing properties of objects in the environment or actions shared between the teacher and the learner and also to learn sequences of such "words". Key differences from our work are that the representations encoded in our architecture are built from behaviors rather than low-level actions, and also that due to their structure they are at a higher and more intuitive level.

Conclusions

We have presented an approach that allows a robot to build task representations from its own experiences of interacting both with a human and a robot teacher. We described an architecture that extends the capabilities of behavior-based systems by allowing the representation and execution of complex behavioral sequences while reducing the complexity of the mechanism required to build them. The behavior networks are also flexible, allowing for dynamical reconfiguration and avoiding the customized behavior redesign usually required to capture the specifics of different tasks.

We showed how the use of our behavior representation enables a robot to relate the observed changes in the environment with its own internal behaviors. We presented an algorithm that uses the benefits of this behavior representation in order to allow the robot to learn high-level task representations, even from a single demonstration. The experimental results demonstrate the flexibility and robustness of the algorithm and validate the reliable extensions our architecture brings to typical BBS.

References

Arkin, R. C. 1998. *Behavior-Based Robotics.* CA: MIT Press.

Billard, A., and Dautenhahn, K. 1998. Grounding communication in autonomous robots: an experimental study. *Robotics and Autonomous Systems, Special Issue on Scientific methods in mobile robotics* 24:1-2:71–79.

Clemans, K. G. 1959. Confidence limits in the case of the geometric distribution. *Biometrika* 46(1/2):260–264.

Dorigo, M., and Colombetti, M. 1997. *Robot Shaping: An Experiment in Behavior Engineering.* MIT Press, Cambridge.

Gat, E. 1998. On three-layer architectures. In Kortenkamp, D.; Bonnasso, R. P.; and Murphy, R.,

eds., *Artificial Intelligence and Mobile Robotics.* AAAI Press.

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

Maes, P., and Brooks, R. A. 1990. Learning to coordinate behaviors. In *Proc., AAAI*, 796–802.

Maes, P. 1990. Situated agents can have goals. *Journal for Robotics and Autonomous Systems* 6(3):49–70.

Mahadevan, S., and Connell, J. 1991. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Eighth Intl. Workshop on Machine Learning*, 328–337.

Matarić, M. J. 1992. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation* 8(3):304–312.

Matarić, M. J. 1997a. Behavior-based control: Examples from navigaton, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2–3):323–336.

Matarić, M. J. 1997b. Reinforcement learning in the multi-robot domain. *Autonomous Robots* 4(1):73–83.

Nicolescu, M. N., and Matarić, M. J. 2000. Extending behavior-based systems capabilities using an abstract behavior representation. Tech Report IRIS-00-389, IRIS, USC, Los Angeles, California.

Nicolescu, M. N., and Matarić, M. J. 2001. Learning and interacting in human-robot domains. *To appear in IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Socially Intelligent Agents - The Human in the Loop.* C. White and K. Dautenhahn, editors.

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

Werger, B. B. 2000. Ayllu: Distributed port-arbitrated behavior-based control. In *Proc., The 5th Intl. Symp. on DARS*, 25–34.