In *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation* pages 463-468, Banff, Alberta, CANADA, July 29 - August 1, 2001.

Experience-based learning of task representations from human-robot interaction

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 that allows a robot to learn task representations from its own experiences of interacting with a human. The robot follows a human teacher and maps its own observations of the environment into a representation of what has constituted the human's demonstration. The robot then builds a representation of the experienced task in the form of a behavior network. 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 this architecture in a set of experiments in which a mobile robot learns representations for multiple tasks and is able to execute the tasks, even in changing environments.

1 Introduction

Teaching robots to perform various tasks has become a topic of growing interest for many researchers. 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 controller that achieves a particular task, we want to enable a robot to automatically construct it from the experience it had while interacting with a human. It is especially apt to address this problem in the context of behavior-based systems (BBS), where representation has not been studied extensively [13], yet whose robust and adaptive properties are suitable to the human-robot interaction domain. Towards this goal, we have developed a behavior representation that extends the capabilities of BBS and addresses some of their limitations.

In behavior-based systems [2, 14], behaviors are typically invoked by build-in reactive conditions, and as a consequence, they are unnatural for, and thus rarely applied to complex problems that contain temporal sequences. Since we seek a method that allows learning representations of general tasks that would require the sequential activation of the robot's behaviors, we need a mechanism that would allow first the representation and then the execution of such sequences. Attempts to address the issue of representing and executing complex sequential tasks have resulted in two distinct approaches: 1) hybrid control architectures, and 2) behavior-based architectures that only partly address the above problem. The representation that we describe in this paper is a behavior-based solution, one that does not alter the nature of the underlying systems or change its representation or time-scale.

Another limitation of BBS that we address is that behaviors are typically redesigned for each task, as new task specifics require different activation conditions for each of the behaviors in the system. Therefore, behaviors have to be continuously updated and customized for each task, even if the underlying processing remains unchanged. Since we are interested in automating the process of BBS generation, we have developed a behavior representation that allows for flexible activation conditions, without the need for behavior redesign or recompilation when switching to another task.

In the remainder of the paper we describe our behavior representation and the behavior network construct that uses them to represent general strategies and plans. Next we present the concept of learning task representations from experienced interaction with humans, and demonstrate our experimental results. We discuss the relevant previous work in this area and conclude with a summary and directions for future research.

2 Behavior representation

We are using a behavior-based architecture that allows the construction of the robot task in the form of behavior networks [16]. This architecture provides a simple and natural way of representing complex sequences of behaviors and the flexibility required to learn high-level task representations.

In our 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*). We introduce a representation of goals into each behavior, in the form of abstracted environmental states. The *met/not met* status of those goals is continuously updated, and communicated to successor behaviors through the network connections, in a general process of activation spreading. Embedding goal representations in the behavior architecture is a key feature of our behavior networks and, as we will see, critical aspect of learning task rep-

*This work is supported by DARPA Grant DABT63-99-1-0015 under the Mobile Autonomous Robot Software (MARS) program and by the ONR Defense University Research Instrumentation Program Grant.

resentations.

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 automatically deactivates the behavior. These preconditions enable the representation of sequences of the type: *the effects of some actions must be permanently true during the execution of this behavior.*
- **Enabling preconditions**: preconditions that must be met immediately before the activation of a behavior. Their state can change during the behavior execution, without influencing the activation of the behavior. These preconditions enable the representation of sequences of the type: *the achievement of some effects is sufficient to trigger the execution of this behavior.*
- **Ordering constraints**: 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 can be executed.*

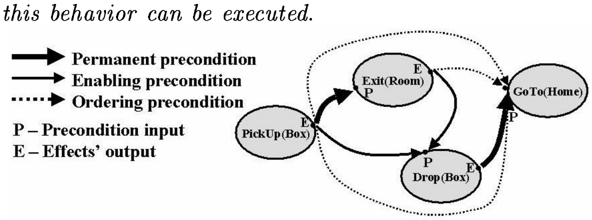


Figure 1: Example of a behavior network

From the perspective of a behavior whose goals are **Permanent preconditions** or **Enabling preconditions** for other behaviors, these goals are what the planning literature [17] calls goals of *maintenance* and of *achievement*, respectively. In a network, a behavior can have any combination of the above preconditions. The goals of a given behavior can be of maintenance for some successor behaviors and of achievement for others. Thus, since in our architecture there is no unique and consistent way of describing the conditions representing a behavior's goals, we distinguish them by the role they play as preconditions for the successor behaviors. Figure 1 shows a generic behavior network and the three types of precondition-postcondition links.

A default **Init** behavior initiates the network links and detects the completion of the task; it has as predecessors all the behaviors in the network. All behaviors in the network are continuously *running* (i.e., performing the computation described below), but only one behavior is *active* (i.e., sending commands to the actuators) at a given time.

Similar to [10], we employ a continuous mechanism of activation spreading 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 sends 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 to determine whether a behavior should be active, and thus able to execute its actions. A behavior is activated iff:

(*Activation level* != 0) AND
(All **ordering constraints** = $TRUE$) 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 executes the behavior that should be active according to the current environmental state.

3 Learning from human demonstrations

3.1 The demonstration process

In a demonstration, the robot follows a human 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 observed 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 the goals of each behavior into its own representation enables to robot to perform a mapping between what it observes and what it can perform. This provides the information needed for learning by observation. This also stands in contract with traditional behavior-based systems, which do not involve explicit goal representation and thus any computational reflection.

Of course, 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.

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

3.2 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 a parameterizable **Track** behavior, which takes as parameters a desired angle and distance to a target, the robot continuously records the observed angle and distance whenever the target is visible (i.e., the

Track behavior's postconditions are true). The last observed values are kept as learned parameters for that behavior.

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).

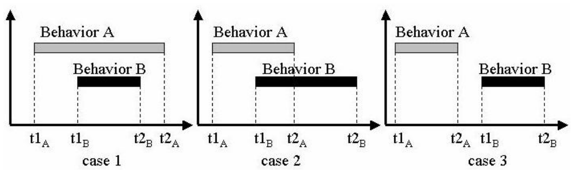


Figure 2: The three precondition types

The general idea of the algorithm is to find the intervals when the effects of all the behaviors have been true and then to find, for all behaviors, if their effects have been active in overlapping intervals or in sequence. The list of intervals is ordered temporally, so one-directional comparisons are all that are needed; no reverse precondition-postcondition dependencies could exist.

Behavior network construction

1. *Filter the data in order to eliminate the false indications of a behavior's effects.* These cases can be detected as having very small durations or unreasonable values of the behavior parameters.
2. *Build a list of intervals for which the effects of any behavior have been true,* **ordered** *by the time these events happened.* These intervals contain information about the behavior they belong to and the values of the parameters (if any) at the end of the interval. Multiple intervals related to the same behavior will generate different **instances** of that behavior.
3. *Initialize the behavior network as empty.*
4. *For each interval in the list, add to the behavior network an instance of the behavior it corresponds to.* Each behavior is identified by a unique ID to differentiate between possible multiple instances of the same behavior.
5. *For each interval I_j in the list:*
   For each interval I_k at its right in the list:
   Compare the end-points of the interval I_j with those of all other intervals I_k on its right in the list: (we denote the behavior represented by I_j as J and the behaviors represented in turn by I_k with K)

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

4 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.

Before describing the experiments, we define the evaluations criteria we used in order to analyze the experimental results, specifically the notions of **success** and **failure**. We define a **successful** experiment to be one for which all of the below properties hold true:

- the robot learns the correct task representation for the demonstration,
- the robot correctly reproduces the demonstration,
- the task performance finishes within a certain period of time (in same and also in changed environments),
- the robot's reports on its reproduced demonstration (ordering and characteristics of demonstrated actions) and user observation of robot's performance match and represent the actual task demonstrating by the human.

We characterize an experimental as having **failed** if any one of the properties below holds true:

- the robot learns an incorrect representation of the demonstrations and thus performs incorrectly,
- the time limit allocated for the task is exceeded,
- the robot learns a correct representation but performs it incorrectly.

4.1 Learning from demonstration

We have designed three different experiments which rely on navigation and object manipulation capabilities of the robot. Initially, the robot was given a behavior set that allowed it to track colored targets, open doors, pick up, drop, and push objects.

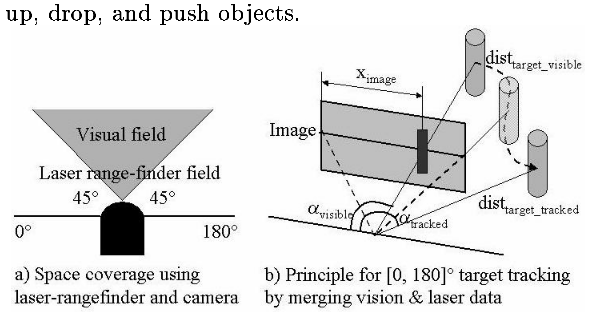


Figure 3: Merging laser and visual information for tracking

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 behavior merges the sensory data from the color camera and the laser range-finder in order to enable the robot to track targets that are anywhere in the camera or the laser field of view, thus increasing the combined field of view of the robot (see Figure 3). The robot uses the camera to initially detect the target and then continues to track it with the laser 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 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, the robot continues to track it with the laser 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, by merging the information from two types of sensors, camera and laser, the **Track** behavior gives the robot the ability to keep track of positions of objects around it, even if they are not currently in the camera's field of view.

The behaviors were implemented using AYLLU [19], an extension of C for development of distributed control systems for mobile robots.

We performed the three experiment sets in a 4m x 6m arena. They consisted of a demonstration phase (the human presented a task to the robot and the robot built a task representation) and the execution phase, when the robot performed the learned task.

Learning to visit a number of targets in a certain order

The goal of this experiment set was to test the model's ability to teach the robot to reach a set of targets in the order indicated by the arrows in Figure 4. The robot's behavior set contains a **Track** behavior, parameterizable in terms of the colors of targets that are known to the robot. Therefore, during the demonstration phase, different instances of the same behavior produced output according to their settings.

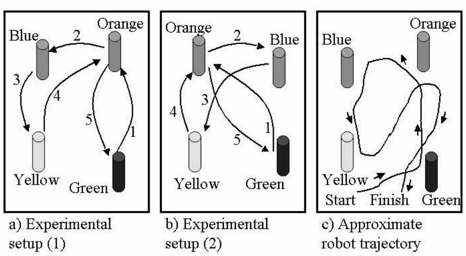


Figure 4: Experimental setup for the **Visit targets** task

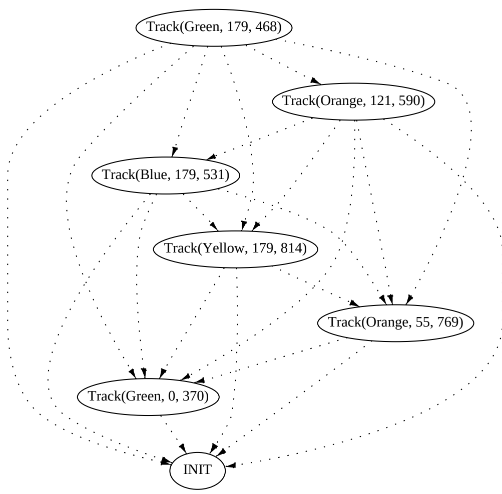


Figure 5: Task representation learned from the **Visit targets** demonstration

During the five demonstration trials we performed, the robot learned the correct representation for this task (Figure 5). The parameters of the **Track** behavior (in the order they appear in the graph) are: the observed color of a target, the observed angle to the target (in degrees) and the observed distance to the target (in mm). All the precondition-postcondition dependencies between behaviors in the network are **ordering** type preconditions; this is evident in the robot's observation data presented in Figure 6. In all experiments the robot met the allotted 5-minute time constraint for this task. We also experimented with a changed environment (Figure 4(b)); in all five trials the robot correctly performed the task it had learned in the previous setup.

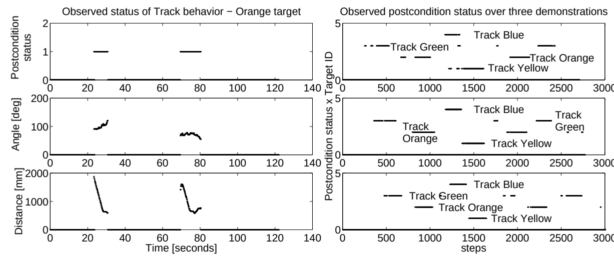


Figure 6: Observation data gathered during the demonstration of the **Visit targets** task

Learning to slalom

In this experiment set, the goal was to test the model's ability to teach a robot to slalom through four targets placed in a line, as shown in Figure 7(a). We changed the size of the arena to 2m x 6m for this task.

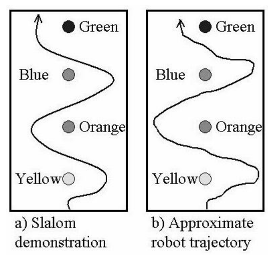


Figure 7: The **Slalom** task

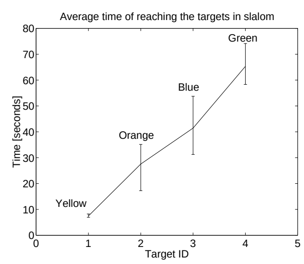


Figure 8: Average time of reaching the targets during the **Slalom** task

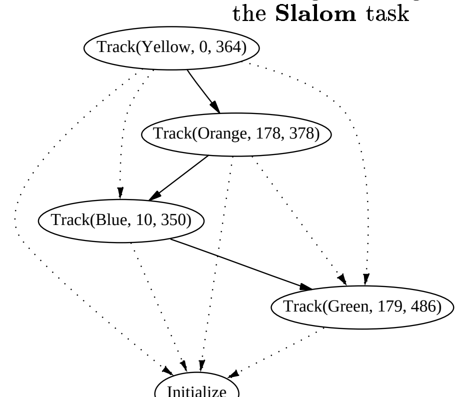


Figure 9: Task representation learned from the **Slalom** demonstration

During 5 different trials the robot learned the correct task representation as shown in the behavior network from Figure 9. For this case, the relation between behaviors that track consecutive targets is of **enabling** precondition type, since for this environmental setup, the robot began to detect (and therefore to track) a new target while still being near the previous one.

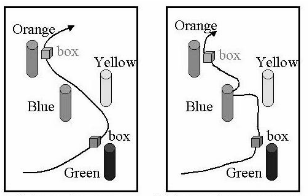
We performed 20 experiments, in which the robot correctly executed the slalom task in 85% of the cases.

The failures consisted of two types: 1) the robot, after passing one "gate", could not find the next one due to the limitations of its vision system; and 2) the robot, while searching for a gate, turned back towards the already visited "gates".

Figure 7(b) shows the approximate trajectory of the robot executing the slalom task on its own and Figure 8 shows the time of passing each gate averaged over five of the successful experiments.

Learning to traverse "gates" and move objects from one place to another

This experiment set tests the model's ability to handle tasks of increased complexity, as it involves the robot's object manipulation capabilities and also learning higher-level behaviors such as going through a door/gate.



a) Traversing gates and moving objects
b) Approximate trajectory of the robot

Figure 10: The **Object manipulation** task

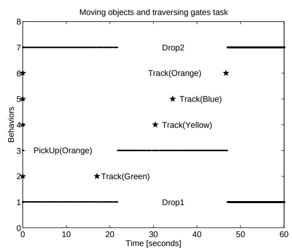


Figure 11: Progress (achievement of behavior postconditions)

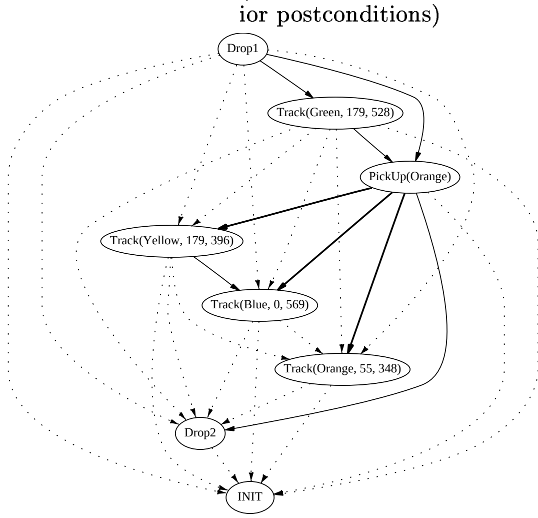


Figure 12: Task representation learned from the **Object manipulation** demonstration

The setup for this experiment is presented in Figure 10. Close to the green target there is a small orange box. In order to teach the robot that the task is to pick up the orange box placed near the green target (the source), the human led the robot to the box, then when sufficiently near it, placed the box between the robot's grippers. After leading the robot through the "gate" formed by the blue and yellow target, when reaching the orange target (the destination), the human took the box from the robot's gripper. During this experiment, all three types of behavior preconditions were detected, as can be seen in the learned behavior network representation in Figure 12. 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", although the robot did not have a pre-existing behavior for this capability.

The robot built a correct task representation in 90% of the cases (out of 10 trials). The "errors", in fact, represented specialized versions of the correct representation. The robot correctly executed the task in 90% of the cases (out of 10 trials in which the robot was given a correct task representation). The failure involved exceeding the allocated amount of time for the task. This happened when the robot failed to pick up the box because it was too close to it and inadvertently pushed it along (and thus kept it invisible) while it was searching for it. Figure 11 shows the robot's progress during the successful execution of the task, more specifically the instants of time or the intervals during which the postconditions of the behaviors in the network were true.

Discussion

The approach we presented allows a robot to automatically build reliable task representations from only one human demonstration. Furthermore, the tasks the robot is able to learn can embed arbitrarily long sequences of behaviors and simultaneous behaviors, which become encoded within the behavior network representation. Also, as is seen in the third experiment set, in the absence of a *GoThroughGate* behavior, the robot is able to represent that part of the task in a more concise manner than if the controller were to be designed by hand.

As any one-shot learning system, after seeing only a single demonstration of the task to be learned, our system learned a potentially overly specialized representation of the task. Additional demonstrations of the same task would allow it to generalize at the level of the constructed behavior network. Standard methods for generalization can be directly applied within our framework and we will explore them in our future work.

Another approach which we are already investigating is to allow the human to signal the saliency of particular events or objects. While this does not eliminate irrelevant environment state from being observed, it biases the robot to notice and (if capable) capture its key features.

5 Discussion and related work

The work presented in this paper combines the behavior-based systems' (BBS) capability to operate in uncertain, unstructured, dynamically changing environments, with robot learning techniques, to enable robots to build flexible representations of complex tasks.

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 we work with, where sequential behavior is usually triggered through the world, rather than through internal sequences [2, 14]. By augmenting the behaviors with representations of their goals (abstractions of environmental states), we take advantage of both the ability of the deliberative, STRIPS-like architectures to operate at high-level of abstraction, 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 (e.g., [5], [1], [3]), which need a middle layer to solve the difference in representation and time-scales between the physical and the abstract levels.

An early example of embedding representation into BBS was done by [13]. 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 of the robot's capabilities: in our case, both navigation and object manipulation.

In the context of behavior-based robot learning, almost all approaches have been at the level of learning policies, situation-behavior mappings, at least in physical robot domains. The method, in various forms, has been successfully applied to single-robot learning of various tasks, including hexapod waling [11], box-pushing [12], most commonly navigation [4], and also to multi-robot learning [15].

Another relevant approach has been in teaching robots by demonstration. [7] demonstrated simplified maze learning, while [18] presents a system in which a 7 DOF robot arm learns the task of balancing a pole from a brief human demonstration. While these approaches are focused on the level of **action** imitation, we are concerned with representing and repeating high-level tasks with sequential and/or concurrently executing **behaviors**.

Approaches to learning high-level, sequential task representations have been presented in the domains of navigation [6] and assembly [9, 8]. Our work differs in that the task representations that we build are general, not restricted to a particular domain, and also, that in our case the environment is generally less constrained (e.g., involves searching for targets), and the robots do not use complex "teacher tracking" mechanisms that aid building of task representations.

6 Conclusions

We presented an approach that allows a robot to learn task representations from its own experiences of interacting with a human. 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 for capturing 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

[1] Philip A. Agre and David Chapman. What are plans for? *Journal for Robotics and Autonomous Systems*, 6:17–34, 1990.

[2] Ronald C. Arkin. *Behavior-Based Robotics.* MIT Press, CA, 1998.

[3] Ronald C. Arkin and Tucker Balch. Aura: Principles and practice in review. *Journal of Experimental and Theoretical AI*, 2-3:175–189, Apr-Sep 1997.

[4] Marco Dorigo and Marco Colombetti. *Robot Shaping: An Experiment in Behavior Engineering.* MIT Press, Cambridge, 1997.

[5] Erann Gat. On three-layer architectures. In David Kortenkamp, R. Peter Bonnasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robotics.* AAAI Press, 1998.

[6] P. Gaussier, S. Moga, J. Banquet, and M. Quoy. From perceptionaction loops to imitation processes: A bottom-up approach of learning by imitation. Tech Report FS-97-02, AAAI Fall Symp., 1997.

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

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

[9] Hirochika Inoue Yasuo Kuniyoshi, Masayuki Inaba. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *T-RA*, 10:799–822, 1994.

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

[11] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *Proc., AAAI*, pages 796–802, Boston, MA, 1990.

[12] Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Eighth Intl. Workshop on Machine Learning*, pages 328–337, 1991.

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

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

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

[16] Monica N. Nicolescu and Maja J. Matarić. 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*, 2001. C. White and K. Dautenhahn, editors.

[17] Stuart Russell and Peter Norvig. *AI: A Modern Approach.* Prentice Hall, NJ, 1995.

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

[19] Barry Brian Werger. Ayllu: Distributed port-arbitrated behavior-based control. In *Proc., The 5th Intl. Symp. on DARS*, pages 25–34, 2000.