# Robot Learning by Demonstration using Forward Models of Schema-Based Behaviors

**Adam Olenderski, Monica Nicolescu, Sushil Louis**
*University of Nevada, Reno*
*1664 N. Virginia St., MS 171, Reno, NV, 89523*
{*olenders, monica, sushil*}*@cse.unr.edu*

Keywords: Learning by demonstration, human-robot interaction, mobile robots

Abstract: A significant challenge in designing robot systems that learn from a teacher's demonstration is the ability to map the perceived behavior of the trainer to an existing set of primitive behaviors. A main difficulty is that the observed actions may constitute a combination of individual behaviors' outcomes, which would require a decomposition of the observation onto multiple primitive behaviors. This paper presents an approach to robot learning by demonstration that uses a potential-field behavioral representation to learn tasks composed by superposition of behaviors. The method allows a robot to infer essential aspects of the demonstrated tasks, which could not be captured if combinations of behaviors would not have been considered. We validate our approach in a simulated environment with a Pioneer 3DX mobile robot.

## 1 INTRODUCTION

Learning from demonstration is a natural method for augmenting a robot's skill repertoire. The main difficulty of this approach is the interpretation of observations gathered from the instruction, as the robot has to process the data coming from its sensors, then translate it into appropriate *skills* or *tasks*. If the robot already has a basic set of capabilities, interpreting the demonstration becomes a problem of creating a mapping between the perceived action of the teacher to a set of existing primitives (Schaal, 1999).

The behavior-based approach is particularly well suited for autonomous robot control due to its modularity and real-time response properties. While behavior-based control emphasizes the concurrent use of behaviors, existing learning by demonstration strategies have mostly focused on learning unique mappings from observations to a single robot behavior. This approach relies on the assumption that there exists a unique corresponding behavior underlying any one of the demonstrator's actions. However, this is a strong assumption which does not hold in a significant number of situations. Biological evidence, such as the schema theory (Arbib, 1992), suggests that motor behavior is typically expressed in terms of concurrent control of multiple different activities. This view is also similar to the concept of *basis behaviors* (Matarić, 1997), in which a complete set of underlying primitives is capable of generating the entire motor repertoire of a robot.

Based on these considerations, it becomes of significant interest to address the problem of learning a decomposition of a demonstrator's actions onto possibly multiple primitives from a robot's repertoire. Our hypothesis is that such a strategy will enable robots to learn aspects of the task that could not otherwise be captured by a sequential learning approach. In this paper, we assume that the robot is equipped with a set of basic capabilities, which will constitute the foundation for learning. The learning method we propose relies on a schema-based representation of behaviors (Arkin, 1987), which provides a uniform output in the form of vectors generated using a potential fields approach. This representation allows for cooperative behavior coordination, i.e., fusion of commands from multiple behaviors, which in our case is performed through vector addition.

In this paper we present a method that uses forward models to learn tasks which constitute combinations of multiple (in our case two) concurrent primitive behaviors. While, in general, more than two behaviors can contribute to the overall task, the results we present in this paper demonstrate that significant aspects of the task can be learned even with only two concurrent behaviors. The results we present demonstrate the potential of learning approaches for cooperative behavior coordination and provide a strong support for future exploration of these methodologies. The learning algorithm we present in this paper is based on the idea of finding the contribution, expressed as a weight, of each of the primitive behaviors to the demonstrated task. These weights incorporate essential information and can capture subtle differences between tasks, which, even if achieving the same goals, can lead to very different ways of task execution.

The reminder of the paper is structured as follows: Section 2 describes previous approaches to robot task learning and the use of forward models for mapping observed actions onto a robot's skill repertoire. Sec-

tion 3 presents our schema-based behavior representation and Section 4 describes our learning approach. We present our experimental results in Section 5 and conclude with a discussion (Section 6) and a summary of the proposed approach (Section 7).

## 2 RELATED WORK

The work we present in this paper falls in the category of *learning by experienced demonstrations*. This implies that the robot actively participates in the demonstration provided by the teacher, and experiences 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. The advantage of using such techniques is that the robot is freed from interpreting and relating the actions of a different body to its own. The acquired sensory information is in terms of the robot's own structure and sensory-motor skills.

In the mobile robot domain, successful approaches that rely on this methodology have demonstrated learning of reactive policies (Hayes and Demiris, 1994), trajectories (Gaussier et al., 1998), or high-level representations of sequential tasks (Nicolescu and Matarić, 2003). These approaches employ a *teacher following* strategy, in which the robot learner follows a human or a robot teacher. Our work is similar to that of (Aleotti et al., 2004), who perform the demonstration in a simulated, virtual environment.

A significant challenge of all robotic systems that learn from a teacher's demonstration is the ability to map the perceived behavior of the trainer to their own behavior repertoire. A successful approach that has been previously employed for matching observations to robot behaviors is based on forward models (Schaal, 1997; Wolpert et al., 1998), in which multiple behavior models compete for prediction on the teacher's behavior (Wolpert and Kawato, 1998; Hayes and Demiris, 1994). The behavior with the most accurate prediction is the one said to match the observed action. In this paper we present a method that uses predictive models of the robot's behaviors to learn tasks which consist of combinations of concurrently running behaviors.

## 3 BEHAVIOR REPRESENTATION

Behavior-Based Control (BBC) (Matarić, 1997; Arkin, 1998) has become one of the most popular approaches to embedded system control both in research and in practical applications.

In this paper we use a schema-based representation of behaviors, similar to that described in (Arkin, 1987). This choice is essential for the purpose of our work, since it provides a continuous encoding of behavioral responses and a uniform output in the form of vectors generated using a potential fields approach.

In our system, a controller consists of a set of concurrently running behaviors. Thus, for a given task, each behavior brings its own contribution to the overall motor command. These contributions are weighted such that, for example, an *obstacle avoidance* behavior could have a higher impact than *reaching a target*, if the obstacles in the field are significantly dangerous to the robot. Alternatively, in a time constrained task, the robot could give a higher contribution to getting to the destination than to obstacles along the way. These weights affect the magnitude of the individual vectors coming from each behavior, thus generating different modalities of execution for the task.

## 4 LEARNING APPROACH

The main idea of the learning algorithm we propose is to infer, from a teacher provided demonstration, what is the contribution (expressed as a weight) of each of the robot's behaviors to the presented task. As mentioned earlier, these weights modulate the magnitude of vectors coming from the individual robot behaviors, thus influencing the resulting (fused) command and consequently the way the robot interacts with the world. However, choosing these weights is not a trivial problem. Numerous experiments might be needed to determine a set of weights that allow the robot to go through tight corridors and will keep it from colliding with obstacles as well. Therefore, in order to save time and other resources (such as robot power), it becomes of significant interest to derive the appropriate weights for the behaviors by demonstrating the desired navigation style to the robot.

For pairs of behaviors, this is accomplished through a learning algorithm that is based on the geometric properties of vectors. During the demonstration, a human uses a joystick to guide the robot through the task while the robot's behaviors continuously provide predictions on what their outputs would be (in the form of a vector) for the current sensory readings. However, instead of being translated into motor commands, these predictions are recorded along with the turning rate of the robot at that moment of time. Thus, for each time step we are provided with a pair of vectors $v_1$ and $v_2$ of known directions and magnitudes (from the two behaviors) and a vector representing the combination of the two vectors $v$, of known orientation, but whose magnitude is unknown (from the user commands). However, it is known that the resulting vector represents the combination of the two vectors $v_1$ and $v_2$, modified by some unknown weights $w_1$ and $w_2$ (Figure 1). The goal of the algorithm is to infer what are these weights, more precisely to learn the ratio $w = w_2/w_1$ between the two. Intuitively,

we are interested in computing the relative weight between the contribution of the two behaviors, which could produce the vector $v$, whose orientation is provided by the teacher's demonstration.

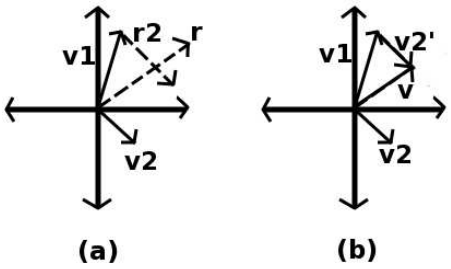


Figure 1: Visual representation of vectors $v_1$, $v_2$, (a) rays $r$ and $r_2$ and (b) vectors $v$ and $v_2^{'}$.

The learning algorithm analyzes each of the records as follows: of the two vectors $v_1$ and $v_2$ being analyzed, one vector's weight will be kept constant at 1 (say $v_1$) while the other (say $v_2$) will be assigned a weight relative to the first. For every record, the turn rate of the robot is interpreted as the angle of a ray $r$, which is the support of the resulting vector $v$, obtained from the teacher's demonstration. The origin of another ray $r_2$ whose orientation is the same as that of $v_2$ is placed at the terminal endpoint of $v_1$ (Figure 1(a)). The intersection point of rays $r$ and $r_2$ can be interpreted as the endpoint of two vectors $v_2^{'}$ and $v$, where the initial endpoint and angle of $v_2^{'}$ are the same as those of $r_2$, and the initial endpoint and angle of $v$ are the same as those of $r$ (Figure 1(b)). By the properties of vector addition, $v = v_1 + v_2^{'}$. Since $v$ is the vector corresponding to the command transmitted to the robot's motors via the operator's joystick, $v_1$ is the unaltered vector generated by one of the behaviors, and the angle of $v_2^{'}$ is the same as that of the other behavior's vector $v_2$, we can interpret $v_2^{'}$ as the product of the vector $v_2$ and the desired weight, which can be easily determined by dividing the magnitude of $v_2^{'}$ by the magnitude of $v_2$. That is, $w_2 = ||v_2^{'}||/||v_2||$, $w_1 = 1$, and the ratio between the two is the resulting $w$.

The process described above is performed on each of the recorded steps in turn, resulting in a weight ratio for every time step during the demonstration. While the computed ratios will vary throughout the task, a single ratio will emerge as being the most consistent with the entire task. We find this by removing obvious outliers (ratios that are in the thousands to hundreds of thousands range, generated when the difference between the turn rate and one of the component vectors is very close to zero, implying that for that timestep, one vector is infinitely stronger than the other), and taking the average of the derived weights over all the records.

# 5 EXPERIMENTAL RESULTS

To validate the proposed learning algorithm we performed experiments using the Player/Stage simulation environment (Gerkey et al., 2003) and a Pioneer 3DX mobile robot equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera.

## 5.1 Robot Behaviors

The robot is equipped with the following primitive behaviors: laser obstacle avoidance (*avoid*), attraction to a goal object (*attract*), attraction to unoccupied space (*wander*), sonar obstacle avoidance (*sonarAvoid*), distinct from the *avoid* behavior due to differences in sensor specifications, and random direction change (*random*), all implemented using a potential fields approach.

For the experiments described in this paper we only used the *avoid*, *attract* and *wander* primitives. All of these behaviors use information from the laser rangefinder, which returns the distance between the robot and any object the laser encounters, for each unit of angular resolution. We have determined that the maximum range at which obstacles should have an effect on the robot's behavior is 2 meters. If there are no obstacles within this range, the value of 2 meters is returned. Goal objects are represented as fiducials, which can be detected using combined information from the laser and PTZ camera. The laser's fiducial detector has a range of 8 meters.

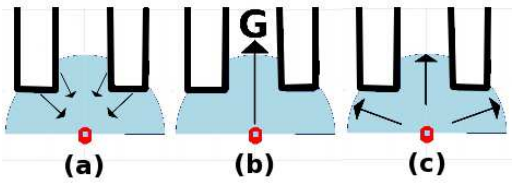


Figure 2: Potential field encoding of the following behaviors: (a) obstacle avoidance, (b) goal attract and (c) wander

Below we describe the implementation details of the three behaviors that we used:

- **Avoid.** The *avoid* behavior is activated whenever the laser rangefinder returns a value within a distance smaller than 2 meters. In this situation, for each laser reading that reaches an obstacle, the behavior generates a vector whose *angle* is the bearing to the obstacle plus 180 degrees and whose *magnitude* is equal to a

function of the distance between the robot and the obstacle. We add 180 degrees to the angle to reverse the direction of the vector, such that the robot is repulsed by the obstacle. The function used to determine the magnitude is $2/d - 1$ if $d$ is less than or equal to 2 and 0 if $d$ is greater than 2, where $d$ is the distance to the obstacle and 2 is the maximum range (in meters) at which obstacles can affect the robot's behavior. The resulting vectors are then combined through vector addition to form one vector representing the response of the *avoid* behavior.

- **Wander.** The *wander* behavior is activated whenever there are unoccupied spaces in the 180-degree front field of view of the robot. For each laser reading that does not reach an obstacle, a vector is generated whose *angle* is the bearing of that empty space and whose *magnitude* is 1. These vectors are then combined through vector addition to form one vector representing the response of the *wander* behavior.
- **Attract.** The *attract* behavior is activated whenever a goal object (fiducial) is in the robot's field of view. For each goal object that is detected, a vector is generated whose *angle* is the bearing to the object and whose *magnitude* is 1.

Within a particular controller, after these vectors are generated, each one is multiplied by a scalar that represents its relative importance compared to the other behavior(s) and then they are combined through vector addition. The resulting vector is then interpreted as a movement command and sent to the motors. This is done by translating the angle of the resulting vector into rotational velocity. If the angle is within the front 90 degrees of the robot, the robot will move forward while turning. If the angle is within either the left or right 90 degrees of the robot, the robot will stop forward movement and turn in place until the resulting vector is once again pointing in front of the robot. If the angle of the resulting vector is within the rear 90 degrees of the robot, the robot will reverse while turning in the direction indicated.

The experiments we describe below validate the learning algorithm's ability to derive the weights of these behaviors for various user-provided demonstrations: different task achieving strategies are represented by different weights, which will be captured by our proposed approach.

## 5.2 Experimental Setup

We tested the learning algorithm described above in two scenarios, each of which used a different behavior pair: one with *avoid/wander* and one with *avoid/attract*. For each scenario we performed two sets of experiments, comprising the teaching by demonstration and the validation stages respectively. Each set was undertaken in a different "world," defined by a map and the starting positions of both the robot and a goal object (Figure 3). In the first scenario, the robot has to learn the task of navigating safely in an environment with various sized corridors. In the second scenario, the robot has to learn how to reach a particular target, in an environment with tighter or larger open spaces.

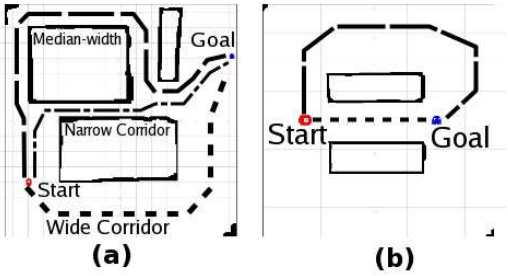


Figure 3: Experimental setup with possible paths: (a) Avoid/wander experiment, (b) Avoid/Attract experiment

In the demonstration step of both sets of experiments, a human operator uses a joystick to drive the robot to the goal object while a background process keeps track of the predictions of the component vectors, the robot's speed, and the robot's turn rate at each timestep, as described in Section 4. The user chooses one of several paths to the goal, each of which represents a different strategy of performing the same task. The different chosen paths have different effects on the component behaviors, more specifically on the weight that is placed on each behavior in turn. For example, in the experiment where *avoid* and *wander* are the two behaviors available, there are three different paths to the goal, each slightly narrower than the last. Ideally, if the user takes the path of medium width, then the robot should learn a set of behavior weights that allows it to traverse paths of that width and larger, while preventing it from entering narrower corridors. Likewise, for the setup in which *avoid* and *attract* are the behaviors the robot has available, there are two possible paths to the goal: one which goes through a narrow corridor and one which bypasses the corridor altogether. In this case, it is hoped that if the user drives the robot along the former path, the learning algorithm will derive a weight for the attract vector strong enough to compel the robot to traverse the corridor. Conversely, if the user bypasses the corridor, it is hoped that the algorithm will derive a weight for the attract vector that is too weak to overcome the contribution from the avoid vector when faced with such a tight space. The demonstration stops in both sets of experiments when the user reaches the goal object.

The learning algorithm then analyzes the information recorded in each of the demonstration runs. For each timestep recorded, the algorithm derives a

weight based on the component vectors, the turn rate and the speed of the robot for that timestep. The algorithm produces a set of behavior weights, as described in Section 4. While in our current implementation the learning is performed off-line, at the end of the demonstration, the processing could also be performed on-line, after each executed step.

To evaluate the performance of the learning algorithm we place the robot at various locations in the environment and equip it with an autonomous controller which uses the derived weights. If the robot performs the same strategy as demonstrated by the user (e.g., does not navigate corridors or tight spaces narrower than those through which the user drove it, but traverses any wider spaces), the experiment is considered a success.

## 5.3 Results

To test the learning algorithm on the first scenario that involves the *avoid/wander* behavior pair, we performed three separate experiments, one for each path indicated in Figure 3 (a). Furthermore, each of these experiments was repeated three times, resulting in the nine values in the first scenario portion of Table 1. First, the user navigated the robot through the narrowest path towards the goal. The result was a relatively large weight for the *wander* behavior, as compared with weights derived for *wander* in the next two demonstrations (see Table 1). We then activated the robot's controller using the learned weights in three different runs, starting from various initial positions: one inside the narrow corridor, one inside the median-width corridor, and one inside the wide corridor. The controller allowed the robot to easily traverse the narrow corridor, as well as wider areas, since the weight of *wander* was able to overcome the response of the *avoid* behavior.

In a second demonstration, the user guided the robot through the middle-width path, which resulted in a significantly smaller weight for the *wander* behavior (see Table 1). We tested the resulting controller in three different runs, starting the robot in each of the thee corridors (narrow, median-width and large). When placing the robot on the narrowest path we found that the weight of the *wander* behavior is no longer strong enough to counter the effect of obstacle avoidance, forcing the robot to reverse in an attempt to escape from the constricting space. However, the robot was easily able to traverse the median-width corridor, as well as the largest one.

Finally, the user took the widest path for the third demonstration, which resulted in a significantly lower weight for the *wander* behavior than in the previous two runs (see Table 1). In the three experiments we performed with the learned controller, the robot was not able to traverse the narrow and median-width corridors, even when placed there, but was able to traverse the widest corridor with ease.

Two subsequent repetitions of these experiments (both learning and validation) led to similar results (with slight differences due to variability in the user's demonstration), leading to the conclusion that the learning algorithm correctly derives the relative importance of the two component vectors in the *avoid/wander* scenario and that it accurately captured the strategy of the demonstrator.

Table 1: Behavior weights learned through demonstration (*Avoid* weights kept constant at 1)

| **First scenario** | *Wander* vs. *Avoid* weight | | |
|---|---|---|---|
| | Exp. 1 | Exp. 2 | Exp. 3 |
| Narrow corridor | 7.8 | 14.4 | 8.4 |
| Medium corridor | 3.8 | 3.2 | 3.3 |
| Wide corridor | 0.4 | 0.6 | 0.4 |

| **Second scenario** | *Attract* vs. *Avoid* weight | | |
|---|---|---|---|
| | Exp. 1 | Exp. 2 | Exp. 3 |
| Traverse corridor | 195.5 | 215.2 | 195.6 |
| Avoid corridor | 124.0 | 131.9 | 118.7 |

The experiments for the second scenario, for which the *avoid/attract* behavior pair was available to the robot, followed similar lines and achieved similar results. In a first demonstration, the user drove the robot through the narrow corridor and directly to the goal. This resulted in a weight for the *attract* behavior that allowed the robot, in the first validation run, to traverse the corridor, even in the presence of obstacle avoidance. In the second validation run, the robot was again able to reach the goal when placed at an initial position where it could see the goal, but was not separated from it by the narrow corridor.

In the second demonstration, the user turns the robot away upon approaching the entrance to the narrow corridor, opting instead to lead the robot around the obstacles from which it was constructed. As expected, this resulted in a much lower weight for the *attract* behavior, which, when used in the robot's controller, caused the robot to stop at the entrance of the corridor, where the magnitude of the *avoid* vector generated by the obstacle walls became equal to that of the *attract* vector generated by the goal at its other end. When placed initially at a location where it could see the goal without having to go through a corridor, the robot was easily able to approach the goal object. As with the *avoid/wander* behavior pair scenario, two subsequent experiments demonstrated the repeatability of these results, further confirming our learning algorithm's viability in deriving relative weights of pairs of behaviors using the potential fields method.

The trained controller is not restricted to a a particular path or execution sequence and is therefore

general enough to exhibit meaningful behavior in any evaluation environment. For example, if a robot is trained in corridors of 1 meter in width, the robot will not enter any corridor that is narrower than 1 meter, regardless of the world in which it is placed and without storing any explicit information about the width of the corridor. The relative weights of the *wander* and *avoid* behaviors will determine if the attractive force pulling the robot towards some corridor is strong enough to overcome the repulsive force generated by the *avoid* behavior.

# 6 DISCUSSION

The approach we presented demonstrates the importance of considering concurrently running behaviors as underlying mechanisms for achieving a task. The method we proposed allows for learning of both the goals involved in the task (e.g., reaching a target) and also of the particular ways in which the same goals can be achieved. The teacher's demonstration provides essential information about the task, such as what is more important: reaching the goal or staying away from obstacles. A purely sequential learning method would have identified the ultimate goal of the task, but would have failed to capture the different modalities in which the task could be achieved.

The proposed approach could also make a significant impact on modeling human behavior. Since the algorithm allows for learning the level of importance of various aspects of the task, it would enable understanding of the priorities different users have when performing that task, thus capturing the underlying strategies they employ. For example, the algorithm could distinguish between bold strategies, in which the user rushes through obstacles and toward the goal, and cautious strategies in which the user goes slowly around obstacles.

# 7 SUMMARY

We presented a method for robot task learning from demonstration that addresses the problem of mapping observations to robot behaviors from a novel perspective. Our claim, supported by biological inspiration, is that motor behavior is typically expressed in terms of concurrent control of multiple different activities. Toward this end, we developed a learning by demonstration approach that allows a robot to map the demonstrator's actions onto multiple behavior primitives from its repertoire. This method has been shown to capture not only the overall goals of the task, but also the specifics of the user's demonstration which indicate different ways of executing the same task.

**Acknowledgements.** This work has been supported by the Office of Naval Research under contract number N00014-03-1-0104 and by a UNR Junior Faculty Award to Monica Nicolescu.

# REFERENCES

Aleotti, J., Caselli, S., and Reggiani, M. (2004). Leveraging on a virtual environment for robot programming by demonstration. *Robotics and Autonomous Systems*, 47:153–161.

Arbib, M. (1992). Schema theory. In S.Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pages 1427–1443. Wiley-Interscience.

Arkin, R. C. (1987). Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *IEEE Conference on Robotics and Automation, 1987*, pages 264–271.

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

Gaussier, P., Moga, S., Banquet, J., and Quoy, M. (1998). From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. *Applied Artificial Intelligence Journal*, 12(78):701–729.

Gerkey, B., Vaughan, R. T., and Howard., A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proc., the 11th International Conference on Advanced Robotics*, pages 317–323.

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

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

Nicolescu, M. N. and Matari´c, M. J. (2003). Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Proc., Second Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia.

Schaal, S. (1997). Learning from demonstration. *Advances in Neural Information Processing Systems*, 9:1040–1046.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 6(3):323–242.

Wolpert, D. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329.

Wolpert, D., Miall, R., and Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Science*, 2:338–347.