Learning Behavior Fusion from Demonstration

Monica Nicolescu

University of Nevada, Reno

1664 N. Virginia St. MS 171

Reno, NV, 89523

monica@cse.unr.edu

Odest Chadwicke Jenkins

Brown University

115 Waterman St., 4th Floor

Providence, RI 02912-1910

cjenkins@cs.brown.edu

Adam Olenderski

University of Nevada, Reno

1664 N. Virginia St. MS 171

Reno, NV, 89523

olenders@cse.unr.edu

Eric Fritzinger

University of Nevada, Reno

1664 N. Virginia St. MS 171

Reno, NV, 89523

ericf@unr.nevada.edu

Abstract

A critical challenge in robot learning from demonstration is the ability to map the behavior of the trainer onto a robot's existing repertoire of basic/primitive capabilities. In part, this problem is due to the fact that the observed behavior of the teacher may consist of a combination (or superposition) of the robot's individual primitives. The problem becomes more complex when the task involves temporal sequences of goals. We introduce an autonomous control architecture that allows for learning of hierarchical task representations, in which: 1) every goal is achieved through a linear superposition (or fusion) of robot primitives and 2) sequencing across goals is achieved through arbitration. We treat learning of the appropriate superposition as a state estimation problem over the space of possible linear fusion weights, inferred through a particle filter. We validate our approach in both simulated and real world environments with a Pioneer 3DX mobile robot.

Keywords: human-robot interaction, learning from demonstration, particle filters, control architectures

I. INTRODUCTION

The problem of realizing autonomous robots could be phrased as the ability to transfer a user's intended control policy onto a robot. This basic problem exists regardless of whether this transfer occurs explicitly through computer programming or implicitly from demonstration or a reward criterion. However, the intended control policy may not have an obvious or scalable mapping onto the basic capabilities endowed to a robot. Our objective in the paper is to address this *policy transfer* problem in a manner that facilities task learning from demonstration and without the need to rely on special-purpose, task-dependent robot capabilities.

The difficulty of the policy transfer problem is increased due to a current divide between the complexity of robot control architectures and their ability to support automatic construction of controllers through learning. Elaborate learning algorithms focus on detailed learning of relatively simple tasks [1], [2], [3], [4]. Complex control architectures are highly tuned to specific domains and thus do not lend themselves naturally to learning [5], [6], [7]. In this paper, we propose an approach that bridges this gap and provides contributions both in the area of *robot control architectures* and in the area of *robot learning by demonstration*. To achieve our goal we address two major challenges in the above fields.

One significant challenge for designing robot systems that learn from demonstration is the interpretation of observations gathered from the instruction. The robot must process a continuous stream of data coming from its sensors and cast this information onto its knowledge and control repertoire. In most cases, this consists of segmenting the data stream into meaningful units, and then mapping them into appropriate *skills* or *tasks*. To match the observations of the demonstrator to actual robot behaviors, a successful approach that has been previously employed is based on forward models [8], [9], in which multiple behavior models compete for prediction on the teacher's behavior [10], [11], [12]. The behavior with the most accurate prediction is the one said to match the observed action. Biological evidence, such as the schema theory [13], suggests that motor behavior is typically expressed in terms of concurrent control of multiple different activities, which means that more than one behavior could be responsible for a demonstrator's actions. In addition, evidence from cognitive psychology [14] indicates the existence of a hierarchical model of motor control, in which a *willed behavior* control mechanism determines *sequencing* of activities through *concurrent activation of multiple motor primitives*. This model is mostly related to the class of *hybrid architectures* [15], which consist of a high-level decision making module and a set of low-level primitives. These architectures enable representation and execution of complex, hierarchical tasks, and thus are very well suited for a wide range of applications.

A second challenge addressed in this paper is to enable learning of tasks which consist of both sequencing and superposition during multiple concurrent activities. Most typically, control architectures that allow for encoding tasks that contain sequences of activities are based on the hierarchical model [7], [16], [5]. In these systems, also known as three-layer systems, reactive components are linked with a deliberative layer through an intermediary middle layer, which is specific to the architecture. Due to the complexity of designing the intermediate layer, these approaches do not lend themselves naturally to automatic learning of controllers from demonstration. In addition, existing learning by demonstration strategies have mostly focused on learning unique mappings from observations to a single robot behavior. These approaches rely on the assumption that there exists a unique corresponding behavior underlying any one of the demonstrator's actions, which is a strong assumption that only holds in a limited number of situations. In contrast, research on the correspondence problem in imitation [17], [18] is addressing this assumption, by considering the variety of imitation behaviors that can result from different correspondence mappings. For example, even in simple navigation or object manipulation tasks, a human's actions are typically the result of combining multiple concurrent activities, such as *seeking the goal*, *avoid*

static obstacles, *avoid dynamic obstacles*, *keep equilibrium*, etc. In terms of the coordination mechanism, a one-to-one correspondence is equivalent to using *arbitration* for behavior coordination. Behavior *fusion* subsumes arbitration, and by allowing multiple behaviors to be active at the same time it enables more general robot behavior. The above methods also rely on specialized, specifically tailored behaviors in order to perform a certain task. In contrast, our approach aims to use very simple behaviors, similar to reactive animal motor behaviors, and looks to achieve higher-level complexity through their combination.

This paper addresses the above challenges by introducing a novel control architecture and a new learning by demonstration technique. Together, these will allow for learning of tasks which consist of both sequencing and superposition of multiple concurrent activities.

A. Contributions

The contributions of the proposed *control architecture* are that it enables: **1)** the use of both *command arbitration and fusion* within a single control representation and **2)** *automated construction of such representations from demonstration*. Historically, these two main action selection mechanisms have been mostly employed separately in robot control [19], thus limiting the range of tasks that robots can execute. By recognizing the ability of *arbitration* to encode temporal sequences and the ability of *fusion* to combine concurrently running behaviors, we merge the strengths and features of both within a unique task representation.

The problem of estimating the state of the user lies at the core of human-robot interaction. Such state information can vary over several modalities, such as affective state or shared common belief [20], [21]. Human state in these problems is estimated and used to guide a robot's actions or a system design process to improve a user's experience or performance. In the case of learning from demonstration (LFD) [22], the objective is to estimate the human's decision state. That is, the single control policy out of all possible control policies that is utilized by the teacher during demonstration at a certain instance of time. Once estimated, the learned policy can be used as the robot's control policy, assuming an appropriate transform between embodiments. LFD strives to open the programming of robot control to broader populations in society by learning implicitly from human guidance, as opposed to explicit programming of a computer language. In addition to LFD, knowledge of a user's decision state can be used to inform on-line human-robot interaction, such as for adjustable

autonomy [23].

At a high level, estimating a user's decision state is finding the most likely control policy given sensory-motor observations from demonstration. In the context of a Markov Decision Process, a control policy $\mu(S) \rightarrow A$ is defined by a mapping of observed world state S to control outputs A. This control policy governed by a value function, $Q(S, A)$, that specifies the benefit for taking action A in world state S. This value function defines the decision state over the space of all state action (S, A) pairs. Atkeson and Schaal [24] define optimality (as a reward function) for this problem as minimizing the divergence in performance of the learned policy and observations from a demonstration. Specifically, given similar perceived environment states, the motor output predicted by the learned policy should, as closely as possible, result in similar outcomes in the environment. By phrasing optimality based on demonstration, policies can be learned for arbitrary tasks without (or with) bias towards specific tasks.

However, learning such policies is subject to issues of partial observability in world state and generalization to new situations. For instance, the approach of Atkeson and Schaal, and subsequent work [25], are geared for top-down planning over the space of control policies given fully observable and relevant aspects of world state. However, such methods are suited for limited or one-time generalization that varies or refines existing behavior (e.g., correcting a demonstrated locomotion gait or variations of a tennis swing).

To address these issues, we propose a behavior-based approach to learning from demonstration that uses behavior fusion to provide bottom-up generalization to new situations. Assuming a set of preexisting robot behaviors expressed as schemas [26] or potential fields, our aim is to learn a coordination policy that linearly fuses their combined output in a manner that matches the teacher's demonstration. We phrase the learning of this coordination as a fusion estimation problem, i.e., state estimation in the space of linear combinations of primitive behaviors. For domains such as mobile robotics, fusion estimation is often subject to ambiguous changes in world state that are attributable to a large space of solutions. To account for this ambiguity and dynamic changes to the user's fusion policy, a particle filter is used to infer fusion estimates from robot sensory observations and motor commands. We focus on the limited case in which fusion is assumed to be unimodal for each discrete combination of behavior preconditions. The contributions of our learning technique allow for: **1)** *learning of superposition* behavior fusion from existing innate robot primitives and **2)** *learning of sequential activities* from multiple superposition fusion primitives.

The paper is structured as follows: Section II discusses relevant related work, Section III describes our control architecture, and Section IV presents our learning algorithm. We describe our experimental results in Section V and Section VII presents our conclusion.

II. RELATED WORK

Our work falls into the category of *learning by experienced demonstrations*. This approach implies the robot actively participates in the demonstration provided by the teacher, and experiences the task through its own sensors. Towards this end, our goal is to bridge the latent mapping between the naturally demonstrated control policy and the robot's basic/innate capabilities. We discuss related approaches to achieving this objective.

Successful approaches to second person LFD have demonstrated learning of reactive policies [27], [12], trajectories [28], words [29], and sequential task descriptions [30]. Second person approaches, where the teacher and learner act from different perspectives, often employ a *teacher following* strategy, where the robot learner follows a human or a robot teacher. Such sequential (or arbitrated) representations for coordination can be considered a subset of our approach. In contrast, our aim is to avoid hard decisions in coordination of and fusing control commands from different behaviors, such as allowing a single behavior to be active at any single time. This results in a general-purpose policy, which would allow the robot to perform the demonstrated task in any new environments, from any initial positions. We do not attempt to reproduce exact trajectories (such as in [28]), but rather learn the underlying policy for executing the task. In making arbitration decisions, sequential methods represent all possible coordinations as discrete set of possibilities along each axis of fusion space. Platt et al. [31] proposed null-space composition as a fusion coordination mechanism limited to control states where behaviors do not affect each other. This coordination allows for off-axis (but discrete) combinations in fusion space. Although the work of Platt et al. is applied to dexterous manipulation, we focus only on the behavior coordination mechanism and not on platform specifics.

Many first person (e.g., direct teleoperation) and third person (e.g., omniscient teleoperation) approaches to LFD use various forms of supervised learning to estimate a mapping from input states to output actions. Atkeson and Schaal [24] use piecewise statistical regression bootstrapped by demonstration and planning. Smart and Kaelbling [32] use demonstration to bootstrap reinforcement learning techniques, such as Q-Learning. This approach is a viable option for fully observable

states, but are nontrivial to extend for partial observability. Neural networks have been used as a statistical regressor by Pomerleau for autonomous driving [33]. In terms of fusion estimation, neural networks have been previously employed in LFD [28], [29]. In [28] and in [29] the fusion is done over input data from multiple sensory modalities. In our work, fusion is performed over a set of "innate" robot primitives that are cooperatively coordinated from first or third person demonstration. Coordination in this sense is performed by estimating a linear combination of weights for fusing the output of each behavior. Our choice of the particle filter [34] is only one of several methods available to infer behavior fusion. The most straightforward choice is a linear least squares optimization. While least squares works well when minor ambiguity is present in the fusion likelihood, our prior testing showed that it did not sufficiently find functional fusion policies as the number of primitives increased and introduced greater ambiguity. Nonlinear methods, such as Levenberg-Marquardt, could yield better results. However, we chose the particle filter to account for ambiguity explicitly.

A significant challenge for 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. We focus on the specific problem of learning *behavior fusion from demonstration* that could be cast into more holistic approaches to human-robot interaction, such as work by Breazeal et al. [35]. One successful approach to this problem has been to match observations to robot behaviors based on *forward models* [8], [36], in which multiple behavior models compete for prediction of the teacher's behavior [10], [37], [38], and the behavior with the most accurate prediction is the one said to match the observed action.

In terms of demonstration segmentation, we use a model-based approach that extends the work of Nicolescu [39] towards greater generality. While in [39] each segment of the demonstration was mapped to a unique primitive behavior, in this work we propose to learn a fusion policy that achieves a goal as a combination of multiple primitives. Nicolescu [39] and Grollman et al. [40] discuss robot sensory time-series segmentation at greater depth.

An important limitation of BBS is that they are usually designed by hand for a single task: the standard behavior architecture prevents the automatic reusability of behaviors across different tasks and thus the automatic generation of BBS. Initial steps toward addressing this problem have been made with the architecture developed in [41], which allows for learning of task sequences from demonstration [30]. However, this architecture relies exclusively on *arbitration* for action selection, in that a single behavior in the system could be active at any given time. The architecture proposed in this paper extends the BBS approach by using both *arbitration* and *fusion* within the same representation.

III. BEHAVIOR REPRESENTATION

The architecture we propose in this paper is aimed at providing an appropriate infrastructure for learning complex, sequential tasks by demonstration. We base our approach on the Behavior-Based Control (BBC) paradigm, one of the most popular approaches to embedded and robotic system control. The main contribution of our architecture is that it combines command *arbitration* and command *fusion* within the same framework of BBC.

Our controllers (Figure 1) are built from two components: *behavior primitives* (BPs) and *fusion primitives* (FPs), which through the combination processes described below result in controllers in the form of *behavior networks* [41]. The *behavior primitives* perform a set of actions under given (relevant) environmental conditions. For BPs representation we use a schema-based approach, similar to work in [26]. This choice is essential for the purpose of our work because schemas with BBC provide a continuous encoding of behavioral responses and a uniform output in the form of vectors generated using a potential fields approach. These primitives are meant to express the basic, general capabilities of the robot and need not be oriented to accomplishing a broad range of tasks. A *fusion primitive* encapsulates a set of multiple concurrently running primitive behaviors through linear combination of the motor commands. Each primitive behavior component brings its own contribution to the overall motor command, as a vector expressed in the robot's coordinate system. The FPs combine these vectors by weighting them and fusing them through vector addition. 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. The details of the fusion primitives are described in more detail in Section IV.

With fusion primitives alone, a controller can only encode *flat representations* of tasks involving sequencing of fusion primitives. While such an architecture is expressive and flexible, it does not have the modularity needed when new, more complex tasks would have to be created from already existing ones. The best solution would be to specify a new task using abstractions of these existing modules, rather than combining their underlying behaviors into a larger, flat network. We enable this higher-level of representation by grouping fusion primitives into *behavior networks* [41], [42]. Behavior networks can be nested, allowing for the construction of hierarchical representations of robot tasks. In these networks, the

links between components represent task-specific precondition−postcondition dependencies. These links provide a simple and natural way of representing complex sequences of activities and also of hierarchically structured tasks (Figure 1).

We use the term *metabehavior* to describe both fusion primitives and nodes of a behavior network in that both have similar functions in the network. Each metabehavior encapsulates information about the behavior's preconditions and its goals (post-conditions). These conditions are continuously monitored whenever the behavior is active, in order to ensure the proper execution of the task. The postconditions of a behavior network node will be true when the execution of the subnetwork it represents is finished. The only difference between a behavior network node and a fusion primitive is that it activates underlying metabehaviors, while a fusion primitive activates only its component primitive behaviors. Thus, when a metabehavior is not active, all subordinate metabehaviors are disabled, and therefore can be regarded as non relevant to the task. When a behavior network node becomes active, all its underlying components are enabled, and the subnetwork becomes the current "network" that is being executed. When the execution of the subnetwork finishes, the behavior network node updates its goal status. Since the successor behaviors continuously check this status, they will detect the achievement of that goal and the execution continues with the new network node. To perform tasks encoded with this representation, the robot starts by activating the metabehavior at the topmost level in the task. The execution of the task's steps proceeds as described above. Figure 2 shows a sample behavior network.

In this architecture, using the links as task-specific activation conditions enables the reusability of behaviors and run-time reconfiguration of robot tasks. The behavior network representation has the advantage of being adaptive 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 pre and post-conditions of behaviors are continuously monitored, the system executes the behavior that should be active according to the current environmental state, thus providing the robot a sense of "awareness" about its progress in the task.

This architecture merges *fusion* (at the level of *fusion primitives*) and *arbitration* (at the level of *metabehaviors*). The output of FPs is the result of fusing BPs' vector outputs, while the metabehaviors output the result of the arbitration (in our case, sequencing) among their component behaviors.

IV. LEARNING BY DEMONSTRATION METHODOLOGY

Our methodology for learning by demonstration follows a three stage approach: (1) segmentation of observation traces, (2) behavior fusion estimation, and (3) task representation construction.

For this work, demonstrations consisted of guiding the robot through a navigation task, using a joystick, while the robot's behaviors continuously provide predictions on what their outputs would be (in the form of a 2D speed and heading vector in the robot's coordinate system) 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. This information is further processed, as described next.

A. Segmentation of Observation Traces

Research in child psychology [43] argues that for imitation learning, in early infancy, matching is based on goal states for the motor system, suggesting that observation of relevant goals carries significant meanings for the demonstrated task. Thus, the approach proposed in this work is to segment the observation trace whenever the goals of any robot behavior become met during the demonstration. Detecting these conditions is possible due to the representation of behaviors, which continuously monitor the status of their goals. The resulted segments represent significant stages in the task (through the goals they attain), and their sequence represents the order in which these goals have to be achieved.

During training we create empty-shell metabehaviors (fusion primitives), which contain only information about the goals we want the robot to detect, and whose only role during the demonstration is to monitor their goal status. Every time the goal of a behavior is met, this generates a new segment in the demonstration, which will result in a new step (fusion primitive) in the task. Thus, if the same goal is met multiple times during training, it will appear in multiple instances in the task representation. This gives us the ability to learn tasks such as "*do A* then *do B* then *do C* then *do B*, then *do A* again," in which the same goal can appear in the task in multiple instances. At this stage the network contains the fusion primitive shells and the precondition-postcondition links between them, as shown in Figure 2.

Within each segment multiple *situations* are possible, in which different subsets of behaviors could be active, depending on whether their preconditions are met or not. For a given set of n *primitive behaviors*, theoretically there could be 2^n combinations representing whether the n behaviors are active or not, based on their pre-conditions. Practically, this number

is much smaller, due to the fact that some behaviors are triggered by similar environmental conditions (such as the presence of an obstacle, for example), and thus some combinations are impossible to achieve. We call these situations *behavior applicability conditions* (BACs). These BACs encode in a n-bit binary array the values of 1 or 0, to represent whether the behavior is active or not, given its preconditions.

To identify the different BACs present in the demonstration, a further segmentation stage is performed in each segment, based on the binary decisions set by the preconditions of each behavior. This new segmentation of the demonstration trace is performed at the moments of time when the status of any of the behaviors' preconditions changes between met and not-met. The resulting BAC sub-segments represent different environmental situations, since different behaviors become "applicable" at the transition points, as described above. The weights of behaviors within each BAC sub-segment encode the mode of performing the current task given the situation and, thus within each BAC sub-segment, the weights of the applicable behaviors are constant. For example, for a *target reaching* task, the robot could behave under the influence of *corridor-follow, target-follow* and *avoid-obstacle* behaviors if in the presence of obstacle, but would behave only under the influence of *target-follow* if in an open space. The data gathered from each BAC sub-segment is processed as described in Section IV-B, resulting in a set of behavior weights for that situation. The sets of weights for the multiple possible BAC sub-segments are stored in a table, as shown in Figure 3. The index of each row in the table is the decimal equivalent of the n-bit BAC value.

B. Behavior Fusion Estimation

The primary function in behavior fusion estimation is to infer, from a teacher provided demonstration, the contribution (or weight) of each primitive in the robot's repertoire such that their combination matches the observed outcome. These weights modulate the magnitude of control vector output by the individual primitives, thus influencing the resulting command from fusion and consequently the way the robot interacts with the world. However, choosing these weights is a non-trivial problem. To save time and resources (such as robot power), we automatically estimate appropriate weights for fusing behaviors according to the desired navigation style as demonstrated.

For a set of N primitives, behavior fusion estimation is accomplished by estimating the conditional probability distribution of the fusion space (i.e., across weighting combinations) given robot observations over the demonstration

duration. As mentioned above, the information observed from the demonstration, for each timestep t, consists of a set of prediction vectors $V_p^t = v_1^t ... v_N^t$ from each primitive and a demonstration vector V_r^t expressing the realized control output of the robot. It is assumed that the resulting vector V_r^t is a linear combination of the prediction vectors $[v_1^t \cdots v_N^t]$ according to some unknown superposition weights $S^t = [s_1^t \cdots s_N^t]$:

$$V_r^t = \sum_{i=1}^{N} s_i^t v_i^t \tag{1}$$

We consider heading to be the most important consideration for behavior fusion in 2D navigation[1]. Consequently, we normalize command vectors to unit length. The goal of the algorithm is to infer the weights s over time or, more precisely the relative proportions among the weights that could produce the demonstration vector V_r.

Similar to Monte Carlo robot localization, a particle filter is used to recursively estimate the joint density in the parameter space of fusion weights S^t over time $t = 1 \cdots T$. Particle filters [44] have been used for state and parameter estimation in several different domains (such as robot localization [34], pose estimation [45], and insect tracking [46]). Restating these methods, mostly following [46], we use the standard form of the Bayes filter to estimate the *posterior* probability density $p(S^t | V_r^{1:t}, V_p^{1:t})$ in the space of fusion parameters given prediction and result vectors:

$$\begin{aligned} p(S^t | V_r^{1:t}, V_p^{1:t}) = & \\ & k p(V_r^t, V_p^t | S^t) \int p(S^t | S^{t-1}) p(S^{t-1} | V_r^{1:t-1}, V_p^{1:t-1}) \end{aligned} \tag{2}$$

where $p(V_r^{1:t}, V_p^{1:t} | S^t)$ is the *likelihood* of observing prediction and result vector given a vector of fusion parameters, $p(S^t | S^{t-1})$ is the *motion model* describing the expected displacement of parameter weights over a timestep, $p(S^{t-1} | V_r^{1:t-1}, V_p^{1:t-1})$ is the *prior* probability distribution from the previous timestep, and k is a normalization constant to enforce that the distribution sums to one. We simplify the likelihood using the chain rule of probability and domain knowledge (Eq. 1) that prediction vectors are not dependent on the fusion weights:

$$p(V_r^t, V_p^t | S^t) = p(V_r^t | V_p^t, S^t) p(V_p^{1:t} | S^t) = p(V_r^t | V_p^t, S^t) \tag{3}$$

[1]Speed could easily be incorporated into our formulation. However speed is marginalized over time by the slow drive of our robots

The resulting Bayes filter:

$$p(S^t|V_r^{1:t}, V_p^{1:t}) = \tag{4}$$
$$kp(V_r^t|V_p^t, S^t) \int p(S^t|S^{t-1}) p(S^{t-1}|V_r^{1:t-1}, V_p^{1:t-1})$$

has a Monte Carlo approximation that represents the posterior as particle distribution of M weighted samples $\{S_{(j)}^t, \pi_{(j)}^t\}_{j=1}^M$, where $S_{(j)}^t$ is a particle representing a specific hypothesis for fusion weights and $\pi_{(j)}^t$ is the weight of the particle proportional to its posterior probability:

$$p(S^t|V_r^{1:t}, V_p^{1:t}) \propto kp(V_r^t|V_p^t, S^t) \sum_j \pi_{(j)}^t p(S^t|S^{t-1}) \tag{5}$$

The estimation of the posterior at time t is performed by 1) importance sampling to draw new particle hypotheses $S_{(j)}^t$ from the posterior at time $t-1$ and 2) computing weights $\pi_{(j)}^t$ for each particle from the likelihood. Importance sampling is performed by randomly assigning particle $S_{(i)}^t$ to particles $S_{(j)}^{t-1}$ based on weights π^{t-1} and adding Gaussian noise. This process effectively samples the following proposal distribution:

$$S_{(i)}^t \sim q(S_{(i)}^t) \triangleq \sum_j \pi_{(j)}^t p(S^t|S_{(j)}^{t-1}) \tag{6}$$

and weights by the following likelihood as the distance between actual and predicted displacement direction:

$$\pi_{(i)}^t = p(V_r^t|V_p^t, S^t) = 2 - D(V_r^t, \hat{V}_{(i)}^t)/2 \tag{7}$$

where $D(a, b)$ is the Euclidean distance between a and b and:

$$\hat{V}_{(i)}^t = \frac{\sum_{k=1}^N S_{(i),k}^t v_k^t}{|\sum_{k=1}^N S_{(i),k}^t v_k^t|} \tag{8}$$

The process described above is performed on each of the recorded instances of time, resulting in a posterior distribution

of fusion weights for every time step during demonstration.

To enable crisp decision making and our hierarchical architecture, a single state is extracted from the fusion posterior distribution. Current particle filtering methods have three approaches to selecting appropriate parameters from the posterior distribution: the mean of the distribution, the maximum of the distribution, or the robust mean (the mean of particles in some neighborhood around the maximum). All three estimators have advantages and disadvantages. For simplicity, we chose the mean of the distribution, which assumes the posterior will be unimodal. While the computed fusion weights will vary throughout the task, a single fusion set will emerge as being the most consistent during the performance of each subtask. We find these fusion weights by removing obvious outliers and taking the average of the derived weights over all the records.

C. Construction of Task Representation

To construct the overall representation of the demonstrated task, the information from the previous two stages is processed and combined as follows:

- For each segment of the demonstration (corresponding to a goal) a new fusion primitive is created, whose goal is the one identified during the training procedure (and that was used for segmentation).
- Within each segment, for the corresponding fusion primitive, we include all sets of weights corresponding to different behavior applicability conditions (BACs) that were detecting during training.
- The fusion primitive resulting from the goal-achieving segments are linked in the order in which they occurred, and are encapsulated in a node in the behavior network, to form the complete representation of the learned task.

During the robot's execution of the task, at each time-step, based on the environmental conditions, the robot detects which behaviors are active at that time, and uses the weights from the corresponding BAC from the network. If the robot encounters a situation that has not been met during training, it will use a set of default weights, which keep it in safe wandering. Our experience has been that these situations happen only rarely, and that they last only for a few seconds. After this time, the robot re-enters a situation that it knows how to deal with.

The above method allows for learning of both the goals involved in the task (represented as the sequence of goal-achieving fusion primitives) and also of the particular ways in which the same goals can be achieved (captured in the

behavior weights). 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.

V. EXPERIMENTAL RESULTS

To validate the proposed learning algorithm we performed style-based navigation experiments with a Pioneer 3DX mobile robot. User demonstration consisted of navigation in simulated and physical environments according preferences, such as traversing a specific side of a corridor. The robot platform was equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera, both with the real robot and with the Player/Stage simulation environment [47]. The robot's complete/innate behavior set consists of: laser obstacle avoidance (*avoid*), attraction to a goal object (*attract*), attraction to unoccupied space (*wander*, *wander-left*, *wander-right*), attraction to walls (*wallAttract*, *wallAttract-left*, *wallAttract-right*), rear sonar obstacle avoidance (*sonarAvoid*), tangent wall follow (*wall-side-left*, *wall-side-right*) and circular avoid (*circular-avoid-left*, *circular-avoid-right*). The robot could not perform the demonstrated navigation and maintain user preferences through sequencing these behaviors alone. All the behaviors produce a motor command output in the form of a vector in the robot's coordinate system. Table I is a brief description of all the behaviors.

The *left* and *right* variants for the *wander*, *wallAttract*, behaviors produce the same output, only that they only consider the left and respectively right 90-degrees in front of the robot. In our experiments d_{max} was set to 2 meters.

We first report our results from the simulation environment and then experiments with a physical Pioneer 3DX robot. The experiments performed in the simulation domain are aimed at demonstrating the validity of our approach in tasks involving a single goal. They also show how choosing a lower-level behavior set allows for capturing more details of the demonstration. The physical world experiments show the ability of our system to learn sequenced, hierarchically structured tasks, composed of fusion primitives acquired from demonstration.

A. Results from the Simulated Environment

Using the Player/Stage simulation environment, we performed three different sets of experiments (scenarios), in each of which the robot was equipped with a different underlying set of primitives. In each scenario the robot was given a set of different demonstrations, from each of which it learned different behavior weights for controlling the robot. Thus, for

each specific scenario and demonstration, the robot had learned the task from a single trial. All demonstrations ended by taking the robot toward a goal represented by a colored target.

Simulated Scenario 1: Environment Navigation

During the first set of experiments, the robot was equipped with the following behaviors: *avoid*, *attract*, *wander*, *wallAttract*, *sonarAvoid*. We performed four different demonstrations, with the Pioneer 3DX robot, each consisting of making a tour of the UNR CS department (a sketch of which is represented in Figure 4, left). Every demonstration used a different navigation strategy as follows: 1) keep to the right, 2) keep to the center, 3) keep to the left and 4) slalom.

Results. From these experiments the robot had learned to safely navigate in the environment and to go toward the goal when seeing it. Since the robot was always close to walls on both sides during all demonstrations, and due to the fact that the single *wallAttract* behavior does not have a preference for left or right walls, the robot learns a strong preference for staying close to the walls, irrespective of which weights are used (from the right/left/center/slalom navigation). Figure 5 shows a typical path that the robot would take using the weights from these demonstrations. The image shows the robot pulling away from the wall when the goal is detected, demonstrating that the robot learns the proper weights for dealing with this situation. In this image, as for the following plots, the robot path is in red, the laser range readings are in blue and the detection of the goal is in green.

Simulated Scenario 2: Modified Wall Attraction

During the second set of experiments, the robot was equipped with the same set of behaviors as in Scenario 1, with the only difference that we replaced the general *wallAttract* behavior with *wallAttract-left* and *wallAttract-right*. The left and right wall attract behaviors respond only to the walls on their corresponding side, as opposed to all the combined approach of regular *wallAttract*. The demonstration experiments were performed in a simulated environment, simpler than our building (Figure 4, right). The four demonstrations consisted of taking the following paths through the environment: one through the upper corridor, one through the bottom (wide) corridor, in both cases keeping on the left, then on the right. With the weights learned in this environment we tested the behavior of the robot in a simulated map of our building.

Results. As opposed to the first set of experiments, due to the split of the wall-Attract behavior, the robot now learned to distinguish which side to favor when navigating a corridor. However, a more significant difference in behavior is apparent when the robot approaches a T-junction. When using the weights learned from the right-following demonstrations (for both

corridor demonstrations), the robot turns to the right when it approaches an intersection. That is, when it is given the choice of either “right or left” (T-junction) or “right or straight,” (rotated T-junction) it will go right, because of how strongly it is attracted to the right wall. When it is given the choice “straight or left,” it will go straight. With the weights learned from the left-following demonstrations, the robot exhibits a similar behavior for the left side. These results demonstrate the robustness of our approach, in that controllers learned in one environment translate to different environments as well.

Simulated Scenario 3: Modified Wandering

In the third set of experiments, the set of behaviors was the same as in Scenario 2, with the only difference that instead of the *wander* behavior, we used its split versions, *wander-left* and *wander-right*. These two behaviors seek left and respectively right open spaces, as opposed to the regular *wander*, which looks for open space in any direction. We performed three demonstrations, in the simulated environment from Figure 4 (left), each consisting of taking a tour of the building, as follows: 1) keep to the right, 2) keep to the center, and 3) keep to the left.

Results. As expected from these experiments, the robot has learned similar preferences as those in the second scenario. Figure 6 (top, left) shows the trajectory of the robot, using the controller learned from the left follow demonstration. The robot starts at the top of the left corridor, chooses a left turn at the T-junction, then stops at the goal. During the entire run the robot keeps closer to the wall on the left. Figure 6 (bottom) shows a similar preference, this time for following walls on the right and choosing a right turn at T-junctions. In this experiment, the robot starts at the top of the right corridor and moves down and left. While for the left and right preferences the robot makes a clear turn in a particular direction when reaching the T-junction, for the center experiment the robot shows that it does not have a preferred direction, as shown by how far it goes into the T-junction before deciding to turn, due to its wander behavior, as in Figure 6 (top, right). The robot navigates closer to the left due to the wandering behavior, which attracts the robot to the open spaces through the doors. We only show goal reaching capability for the left experiment: we stopped the center and right experiments before the robot made the full turn in the environment to reach the goal, located in a different area in the environment. If allowed to run longer, the robot was able to reach the goal in all situations.

In addition to learning the left and right preference, our results demonstrate that the additional decomposition of the underlying behavior set in simpler primitives for *wander-left* and *wander-right*, allowed the robot to capture additional aspects of the demonstration. In particular, whenever the robot found itself in the middle of a T-junction, with open space

on both sides, the robot would choose to go in the direction given by the preference expressed during the demonstration: right for the right weights and left for the left weights. This preference was demonstrated even in the case in which the robot had more open space in the opposite direction. Under equal weighting of left and right wandering, the robot would normally follow the larger open space. Figure 7 shows this preference through the robot's trajectory. In the left image, the robot is using the weights learned from the left-follow demonstration. While the robot starts oriented slightly toward the right in the middle of the T-junction, as shown by its laser profile, the higher weight of the *wander-left* behavior pulls the robot in the left direction. Similarly, in the right image, the robot uses the weights from the right-follow demonstration. Even if oriented slightly to the left, where there is more open space, the robot chooses to go right due to the higher weight of *wander-right*.

The approach we presented demonstrates the importance of considering concurrently running behaviors as underlying mechanisms for achieving a task. Our method 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 task can be performed. In addition, our results demonstrate the importance of choosing the primitive behavior set, an important and still open issue for behavior-based research. Our learned controllers are not restricted to a particular path or execution sequence and thus are general enough to exhibit meaningful behavior even in environments different from the one in which the demonstration took place.

B. Results from the Physical Robot Experiments

The role of these experiments is to validate our approach and the previous results in a real-world environment and to demonstrate the key feature of our method, which is the ability to learn sequential tasks that involve both sequencing and combination of multiple primitives. We performed two different sets of experiments (scenarios), in which the robot was equipped with the same set of underlying primitives: *avoid, attract, wander-left*, *wander-right*, *wallAttract-left, wallAttract-right, wall-side-left, wall-side-right*, and *circular-avoid-left, circular-avoid-right*. In each scenario the robot was given a different type of demonstration. For each scenario we report both qualitative and quantitative measures for the performance of the learning approach.

Physical Scenario 1: Free-space Navigation

In this scenario we performed a single type of demonstration, on the floor of our department building. The topology of the environment is represented in Figure 8. The task we demonstrated to the robot consists of achieving three goals, in a sequence, as follows:

- Starting from the *START* position, reach *Goal 1* by keeping close to the left wall.
- After reaching *Goal 1*, follow on to reach *Goal 2* by navigating in the center of the corridor.
- After reaching *Goal 2*, follow on to reach *Goal 3* by navigating on the right side of the corridor.

Each part of the above demonstration is represented by different color arrows in Figure 8. The goals were marked with different colored cylindrical targets, which can be detected by the camera. We use this simple method for target identification, since it is robust and since it is outside the scope of the paper to develop a computer vision approach to general object recognition.

We evaluate this task with respect to three criteria: correctness of the learned task description, consistency with user navigation style, and convergence of fusion weights. This navigation task was chosen to evaluation for two main reasons. First, it exhibits the multiple-goal, sequential characteristics we want to learn, and second, because for each goal the demonstrator uses a different navigation approach that the robot should learn. Both of these constitute the important features of our approach.

It is important to note that although the goals were placed in specific locations of our building, the robot is not aiming to learn a precise trajectory, but rather a high-level representation of the task: visit each of the three targets, in the given order, by navigating in the same manner as indicated by the demonstrator. With our approach, the robot achieves the same task, even if the location of the targets is changed.

Results. The demonstration data gathered during training was processed through the method described in Section IV. This resulted in a behavior network controller which contained three fusion primitives, one for each goal encountered during the demonstration. To validate the performance of our learning approach, we performed three types of validation experiments: one to evaluate the learning of the task sequence, one to evaluate the learning of different modes of navigation and one to evaluate the robustness and convergence of our learning algorithm.

1. Task sequence correctness. We validated that the robot had correctly learned the three-step task by first analyzing the behavior-network produced by our algorithm. Our analysis of the behavior network indicated that the robot captured

the correct sequence of goals. Next, we had the robot perform the learned task in 6 (six) different runs, in the same environment, and starting from an approximately similar starting position as during training (but not the same). The robot used performed the tasks encoded in the behavior networks, using the mechanism described in Section III.

The robot correctly performed the task to completion in 85% (5/6) of the trials. In the sixth trial, the robot visited *Goal 1* and *Goal 2* correctly, but it failed to visit *Goal 3*, due to a change in lighting conditions around the goal, which prevented the visual routine from detecting it. During all these trials we also monitored that the robot had captured the correct navigation style for each part of the task. The robot correctly navigated on the left while reaching for *Goal 1*, on the center for *Goal 2* and on the right for *Goal 3*.

It is important to note that the results obtained in simulation (Scenario 3, Figures 6, 7) have translated to the real-robot environment as well: 1) with the left-side navigation style, the robot turns left at a rotated T-junction (after the start), 2) with the center navigation style the robot turns right at both T-junctions and 3) with the right-side navigation style, the robot turns right at the rotated T-junction.

2. Consistency with user navigation style. In order to gather quantitative data that validates that the robot had learned the correct styles of navigation (keep left, center, or right) we performed a set of two different experiments, as follows. We equipped the robot with a scaled down version of the learned behavior network, which included only the fusion primitive responsible for reaching *Goal 1* (keep left) and respectively *Goal 3* (keep right).

For both navigation styles we placed the robot at the start position, facing to the left and then to the right. For each initial starting position (facing left/right), and navigation style we performed 3 (three trials) in which we left the robot navigate around the building until it reached again the start position. This resulted in 24 runs: 2 starting positions $*$ 2 navigation styles $*$ 3 trials each. There was no target placed in the environment, such that the robot would keep moving until coming back to the starting point. Figure 9 shows the trajectories that the robot took for left-side and right-side navigation. From each trial, we recorded the leftmost and rightmost value from the laser rangefinder and averaged it for each run. For validation of the center style navigation we did not perform any separate trial, but rather we we used the laser data gathered during the validation of scenario 1 experiments, from the interval in which the robot navigated in the center toward *Goal 2*. These results are shown in Figure 10.

We see that for left-style navigation the robot consistently stays closer to the left and for the right-style navigation it

stays consistently to the right. The variation in values for each run is due to several factors: first, the robot did not follow the exact same path for each run and second, the environment was changing from one run to another. With doors opening or closing on both sides of the corridor, the average distances to left and right walls were continuously changing. The standard deviation for this data is irrelevant in this scenario, due to the fact that the largest readings on both sides are given by openings created with opening of doors. As such, even if the robot consistently navigates on one side, it will occasionally pass by an open door on the same side, thus recording a very large value. For the center-style navigation, the averages for left and right distances are significantly closer to each other, indicating that the robot navigated consistently in the center of the corridor.

Using the above information we performed both quantitative and qualitative evaluation of our learning approach. For the quantitative evaluation, we choose as metric the error between the distance that the robot kept to the left/right wall and that shown during the demonstration, averaged over each run (Figure 11, left). For the qualitative evaluation we considered the actual distances that the robot kept with respect to the left and respectively right walls after learning. For the center navigation experiments, as the distance between left and right walls should be equal, we consider the difference between the distances to the left and right walls (Figure 11, right). From these results we draw several conclusions. While the demonstrated distance to walls is not captured perfectly, the robot learns the preference for different navigation styles as shown by the trainer. The qualitative data also shows that the robot also exhibits a smooth navigation trajectory, without oscillations, as demonstrated during training. The laser readings that are outside the average values are due to temporary openings/obstacles that the robot is passing by in the corridor. The graphs also show that the robot exhibits a very smooth and fast recovery to its navigation style after all these perturbations.

3. Convergence of fusion weights. An important question for our learning algorithm is the robustness and convergence of the results. To test these factors we ran our algorithm repeatedly for 30 trials, on the same training data from Scenario 1 of the real-robot experiments. For each fusion primitive (corresponding to a goal) and for each different behavior applicability condition (BAC) within that goal, we computed the mean and standard deviation of the weights produced by the algorithm over the 30 runs. The results are presented in Figures 12, 13, and 14. During the training for the first goal, the robot encountered 8 (eight) different BACs, for the second goal 5 (five) and for the third goal 9 (nine), indicated by the separate subgraphs in each figure. The *x* axis gives the list of behaviors that are active during that BAC, as indicated also by the

fact that they have greater than zero weights.

For all these different BACs, the algorithm converged to a unique solution, fact indicated by the small values of standard deviation. While this does not constitute a formal proof, this result comes in support of our initial assumption, that the fusion of multiple behaviors by superposition is unimodal. The only larger values for standard deviation were recorded for the first BAC of the second goal (Figure 13, top left). By analyzing the data corresponding to that BAC we noticed that the corresponding sub-segment only contained 21 training samples (compared with several hundreds for the other sub-segments), which would account for the large variability in the resulting weights.

Another robustness metric refers to the robustness to varying demonstrations. During the course of our experiments we trained the task from Scenario 1 above in at least 5 different runs, by two different users. The results obtained from each of them were consistent with the results described above, in that the robot correctly captured the task sequence and navigation style. This indicates that our method is also robust to varying demonstrations. In future work, we plan to perform an extensive evaluation of this metric, by making a user study in which we rely on a larger number of users to perform the training runs.

Physical Scenario 2: Obstacle Navigation

In this scenario we performed four types of demonstrations. The goal of these experiments is to teach the robot a particular style of negotiating obstacles or people present in front of the robot. Figure 15 shows the experimental setup: along the center of a corridor in our building we placed three obstacles (trash cans) after which we blocked the remaining of the corridor. We performed the following four demonstrations:

1) from the *Start* position, *navigate in the center* of the corridor. When approaching an obstacle *turn to avoid it toward the right*.
2) from the *Start* position, *navigate on the right* side of the corridor. When approaching an obstacle *turn to avoid it toward the right*.
3) from the *Start* position, *navigate in the center* of the corridor. When approaching an obstacle *turn to avoid it toward the left*.
4) from the *Start* position, *navigate on the left* side of the corridor. When approaching an obstacle *turn to avoid it toward the right*.

In each of these runs, at the end of the obstacle course the robot was turned back (right or left, depending on the above case) and the demonstration continued until reaching the starting position. Thus, the robot visited a total of 6 (six) obstacles during each demonstration.

Results. We processed the data from the four training scenarios using our algorithm. This resulted in a different controller for each of the four runs, with behavior weights that encoded the obstacle negotiation technique. To validate these results we performed a set of 3 (three) runs for each of the four navigation controllers. The robot's behavior mirrored the demonstrations, in that it negotiated obstacles through the left when using the fusion primitives from experiments 3 and 4, and respectively through the right when using the fusion primitives from experiments 1 and 2. In addition, the robot stayed on the center corridor when using the primitives from experiments 1 and 3, on the right when using the primitive from experiment 2, and on the left when using the primitive from experiment 4. These results indicate furthermore the potential of using fusion primitives as a means to enable higher-level complexity from low-level robot behaviors.

VI. DISCUSSION

The experimental results presented above demonstrate main contributions of our proposed methods. First, we show that highly specific navigation behaviors can be learned as a superposition of multiple low-level primitives, in the form of *fusion primitives*. This is an important aspect of our work, in that we do not aim to map single, special-purpose behaviors with the demonstration of a teacher, but rather to learn such specialized behaviors from fusion of low-level, generic primitives. We demonstrated this capability in both simulated and physical environments, with a Pioneer 3DX mobile robot. Our results show that the robot is able to learn different modes of navigation using the same underlying set of behavior primitives. The difference in navigation styles was captured as different weights associated with each primitive behavior. These weights have the role of changing the magnitude of the response from each behavior before it is combined with the other behaviors, and provide a flexible and efficient way for robot controller design.

Second, we demonstrated that our hierarchical control architecture enables the learning of both *fusion primitives* and learning of task sequences, encoded as behavior networks within the same representation. In real-world environments our Pioneer 3DX mobile robot learned representations of multi-step tasks, with each step encoded as a fusion primitive. In our scenarios, each step was performed using a different navigation style, to show that both the goals of the task and the

modality in which the task should be executed are captured by our task representations.

In consideration of future work, the primary issues of interest regard scalability for learning control policies for unknown tasks on various robot platforms. Our approach to behavior fusion is one more step towards robot capabilities adapting to user demonstration, rather than human users adapting to preexisting robot capabilities. By using probabilistic inference, we estimate the combination of innate behaviors that most likely explain user demonstration.

Our approach has the potential to generalize across applications in other task domains, such as robot soccer, dexterous manipulation, and human action recognition. However, there are several caveats to consider. One consideration is that the robot's innate primitive behaviors must span a suitable decision space, providing coverage over representative set of control policies. Such primitives should have two properties: **1)** suitable preconditions to determine applicability to the current situation and **2)** command outputs amenable to fusion with other primitives. Given the limited capabilities of the Pioneer robot, defining primitive preconditions and fusion on robot heading was relatively straightforward. Other efforts have proposed methods for defining [48] or learning [36] appropriate sets of primitives in more sophisticated domains. As with any particle filter, fusion likelihood and state dynamics functions must be appropriately defined in order to properly estimate the weights.

Application of behavior fusion to the Manipulation Gaits work of Platt et al. [31] offers the ability to learn dexterous manipulation from demonstration. Manipulation Gaits is a behavior based architecture where primitive behaviors are schemas, as potential fields, and are coordinated by null-space subsumption. That is, the primitives are prioritized such that lower priority behaviors will not disturb the equilibrium of higher priority behaviors. Such subsumption compositionality is encapsulated by linear weighted fusion. Manipulation Gaits currently requires manual crafting of an MDP to describe desired robot behavior and its subsumptive compositionality. Through behavior fusion estimation, however, desired robot behavior can be learned from demonstration. Similarly, many tasks in robot soccer (e.g., the RoboCup 4-legged league) perform cruder and more rapid forms of manipulation with manually crafted controllers that often resemble MDPs. While robot soccer enjoys immense popularity, its true potential lies in the ability to allow mainstream users to develop controllers through demonstration. Towards this end, an architecture using behavior fusion estimation and Manipulation Gaits could be one viable avenue to realize robot soccer from demonstration.

Another caveat is the need for competent demonstration of the desired robot behavior. In order to estimate the control

policy latent in the user's mind, our method must be presented with representative examples of this policy in action. Towards this end, human teachers must be functional teleoperators. Teachers must have some knowledge of the user input to robot motor mapping, but not necessarily the robot's innate primitive behaviors. The quality of demonstration will certainly vary based on robot platform and teleoperation control mapping. When this quality is low, noise and ambiguity will be introduced into fusion inference process. Further ambiguity can occur when the teacher is inconsistent in their demonstration. Inconsistency can occur when the teacher acts differently in similar situations. Probabilistic inference, such as with the particle filter, is well suited for state estimation when such uncertainty is present.

Monocular tracking and action recognition of humans form video is another application where behavior fusion could be useful. Towards this task, Jenkins et al. [38] use particle filters with a vocabulary of schema-style human motion primitives describing actions. Each primitive is assumed to be active at all times and performing inference on human pose and action. Although their tracking is function, their inference of action assumes sequential performance, making hard classifications that are sometimes incorrect. Behavior fusion would allow for action estimates to be fused probabilistically and reduce classification mistakes.

VII. SUMMARY

We have proposed a method for learning learning robot tasks from demonstration through behavior fusion. We address the transfer of control policies to robots through segmenting, estimating fusion, and constructing behavior network representations from human users. Central to our approach is the estimation of behavior fusion to map the demonstrator's actions onto a robot existing control repertoire. Through behavior fusion, we provide a bridge for extending the intended capabilities of the a robot to new tasks demonstrated by human users. We envision that our approach to behavior fusion could be applicable to various other robot learning domains, such as robot soccer, dexterous manipulation, etc. In the greater context of task learning, our method has been demonstrated to capture not only the overall goals of the task, but also the specifics of the user's demonstration, without the need for specialized robot skills, thus enabling additional robot functionality.

VIII. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under Awards IIS-0546876 and IIS-0534858, a UNR Junior Faculty Award, and a Brown University Salomon Award.

References

[1] R. Voyles and P. Khosla, "A multi-agent system for programming robots by human demonstration," *Integrated Computer-Aided Engineering*, vol. 8, no. 1, pp. 59–67, 2001.

[2] P. K. Pook and D. H. Ballard, "Recognizing teleoperated manipulations," in *Proc., Intl. Conf. on Robotics and Automation*, Atlanta, Georgia, 1993, pp. 578–585.

[3] H. Tominaga, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Symbolic representation of trajectories for skill generation," in *Proc., Intl. Conf. on Robotics and Automation*, San Francisco, California, 2000, pp. 4077–4082.

[4] K. Ikeuchi, M. Kawade, and T. Suehiro, "Assembly task recognition with planar, curved and mechanical contacts," in *Proc., IEEE Intl. Conf. on Robotics and Automation*, Atlanta, Georgia, USA, 1993, pp. 688–694.

[5] R. P. Bonasso, R. J. Firby, E. Gat, D. K. D. Miller, and M. Slack, "Experiences with an architecture for intelligent, reactive systems," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2–3, pp. 237–256, 1997.

[6] R. J. Firby, "Task networks for controlling continuous processes," in *Proc., Intl. Conf. on Artificial Intelligence Planning Systems*, Chicago, Illinois, Jun 1994, pp. 49–54.

[7] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *Journal of Experimental and Theoretical AI*, vol. 9, no. 2-3, pp. 175–189, Apr-Sep 1997.

[8] S. Schaal, "Learning from demonstration," *Advances in Neural Information Processing Systems*, vol. 9, pp. 1040–1046, 1997.

[9] D. Wolpert, R. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends in Cognitive Science*, vol. 2, pp. 338–347, 1998.

[10] D. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, pp. 1317–1329, 1998.

[11] Y. Demiris and M. Johnson, "Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning," *Connection Science Journal*, vol. 15, no. 4, pp. 231–243, 2003.

[12] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition (hammer)," *Robotics and Autonomous Systems*, vol. 54, pp. 361–369, 2006.

[13] M. Arbib, "Schema theory," in *The Encyclopedia of Artificial Intelligence*, S.Shapiro, Ed. Wiley-Interscience, 1992, pp. 1427–1443.

[14] D. A. Norman and T. Shallice, "Attention to action: Willed and automatic control of behavior," *Consciousness and Self-Regulation: Advances in Research and Theory*, vol. 4, pp. 1–17, 1986.

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

[16] E. Gat, "On three-layer architectures," in *Artificial Intelligence and Mobile Robotics*, D. Kortenkamp, R. P. Bonnasso, and R. Murphy, Eds. AAAI Press, 1998, pp. 195–210.

[17] C. L. Nehaniv and K. Dautenhahn, "The correspondence problem," in *Imitation in animals and artifacts*. MIT Press, 2002, pp. 41–61.

[18] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Towards robot cultures?: Learning to imitate in a robotic arm test-bed with dissimilarly embodied agents," *Interaction Studies*, vol. 5, no. 1, pp. 3–44, 2004.

[19] P. Pirjanian, "Behavior coordination mechanisms - state-of-the-art," Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California, Tech Report IRIS-99-375, 1999.

[20] C. Breazeal and B. Scassellati, "A context-dependent attention system for a social robot," in *Proc., Intl. Joint Conference on Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1146–1151.

[21] J. G. Trafton, A. C. Schultz, D. Perznowski, M. D. Bugajska, W. Adams, N. L. Cassimatis, and D. P. Brock, "Children and robots learning to play hide and seek," in *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*. New York, NY, USA: ACM Press, 2006, pp. 242–249.

[22] S. Schaal, "Is imitation learning the route to humanoid robots," *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[23] M. Goodrich, D. Olsen, J. Crandall, and T.Palmer, "Experiments in adjustable autonomy," in *IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, Seattle, Washington, USA, 2001.

[24] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 12–20.

[25] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *International Symposium on Robotics Research*, Siena, Italy, 2004, pp. 561–572.

[26] R. C. Arkin, "Motor schema based navigation for a mobile robot: An approach to programming by behavior," in *IEEE Conference on Robotics and Automation*, 1987, pp. 264–271.

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

[28] P. Gaussier, S. Moga, J. Banquet, and M. Quoy, "From perception-action loops to imitation processes: A bottom-up approach of learning by imitation," *Applied Artificial Intelligence Journal*, vol. 12, no. 7/8, pp. 701–729, 1998.

[29] A. Billard and K. Dautenhahn, "Experiments in learning by imitation - grounding and use of communication in robotic agents," *Adaptive Behavior*, vol. 7, no. 3/4, pp. 415–438, 1999.

[30] M. N. Nicolescu and M. J. Matarić, "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: ACM Press, July 2003, pp. 241–248.

[31] R. Platt, A. H. Fagg, and R. R. Grupen, "Manipulation gaits: Sequences of grasp control tasks," in *IEEE Conference on Robotics and Automation*, New Orleans, LA, USA, 2004, pp. 801–806.

[32] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2002)*, vol. 4, May 2002, pp. 3404–3410.

[33] D. A. Pomerleau, "Alvinn: an autonomous land vehicle in a neural network," in *Advances in neural information processing systems 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 305–313.

[34] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[35] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Mulanda, "Tutelage and collaboration for humanoid robots," *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 315–348, 2004.

[36] O. C. Jenkins and M. J. Matarić, "Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion," *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 237–288, Jun 2004.

[37] O. C. Jenkins, "Data-driven derivation of skills for autonomous humanoid agents," Ph.D. dissertation, University of Southern California, 2003.

[38] O. Jenkins, G. González, and M. Loper, "Tracking human motion and actions for interactive robots," in *Proc., Intl. Conf. on Human-Robot Interaction*, Mar 2007.

[39] M. Nicolescu, "A framework for learning from demonstration, generalization and practice in human-robot domains," Ph.D. dissertation, University of Southern California, 2003.

[40] D. H. Grollman, O. C. Jenkins, and F. Wood, "Discovering natural kinds of robot sensory experiences in unstructured environments," *Journal of Field Robotics*, vol. 23, no. 11/12, pp. 1077–1089, 2006.

[41] M. N. Nicolescu and M. J. Matarić, "A hierarchical architecture for behavior-based robots," in *Proc., First Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002, pp. 227–233.

[42] P. Maes, "How to do the right thing," *Connection Science Journal, Special Issue on Hybrid Systems*, vol. 1, no. 3, pp. 291–323, 1990.

[43] A. N. Meltzoff and M. K. Moore, "Explaining facial imitation: a theoretical model," *Early Development and Parenting*, vol. 6, pp. 179–192, 1997.

[44] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[45] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[46] Z. Khan, T. R. Balch, and F. Dellaert, "A rao-blackwellized particle filter for eigentracking," in *IEEE Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 980–986.

[47] B. Gerkey, R. T. Vaughan, and A. Howard., "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc., the 11th International Conference on Advanced Robotics*, Coimbra, Portugal, June 2003, pp. 317–323.

[48] M. Huber, W. MacDonald, and R. Grupen, "A control basis for multilegged walking," in *IEEE Conference on Robotics and Automation*, Minneapolis, MN, USA, 1996, pp. 2988–2993.

Monica Nicolescu is an Assistant Professor of Computer Science with the Computer Science and Engineering Department at the University of Nevada, Reno and is the Director of the UNR Robotics Research Lab. Prof. Nicolescu earned her Ph.D. degree in Computer Science at the University of Southern California (2003) at the Center for Robotics and Embedded Systems. She obtained her M.S. in Computer Science at the University of Southern California (1999) and B.S. in Computer Science at the Polytechnic University Bucharest (Romania, 1995). She is a recipient of the NSF Early Development Career Award and a USC Academic Achievements Award. Prof. Nicolescu's research interests are in the areas of human-robot interaction, robot control and learning, and multi-robot systems.

Odest Chadwicke Jenkins has been an Assistant Professor of Computer Science at Brown University since 2004. Prof. Jenkins earned his Ph.D. in Computer Science at the University of Southern California (2003) at the Center for Robotics and Embedded Systems, M.S. in Computer Science at Georgia Tech (1998), and B.S. in Computer Science and Mathematics at Alma College (1996). Prof. Jenkins' research interests pertain primarily to human-robot interaction and robot learning. His other areas of research include humanoid robotics, machine learning, computer vision, computer animation, motion capture systems, neural prosthetics, and interactive entertainment. His dissertation pertained to model-free methods for motion capture and analysis of kinematic motion for modeling perceptual-motor primitives.

Adam Olenderski is a graduate student with the Computer Science and Engineering Department at the University of Nevada, Reno. He obtained his B.S. degree in Computer Science at the University of Nevada, Reno (2005). His research interests include robotics and artificial intelligence. He is a recipient of the Nevada Millennium Scholarship, a UNR Undergraduate Research Award, and a UNR Engineering Scholarship. He will obtain his MS degree from UNR in Summer 2007.

Eric Fritzinger is currently a software engineer at Hamilton Company. He obtained his M.S. degree in Computer Science (2006) and his B.S. degree in Computer Science (2003), both at the University of Nevada, Reno. His research interests include robotics, artificial intelligence, software engineering and computer graphics. He was a recepient of the Outstanding Teaching Assistant Award from the University of Nevada, Reno (2005).

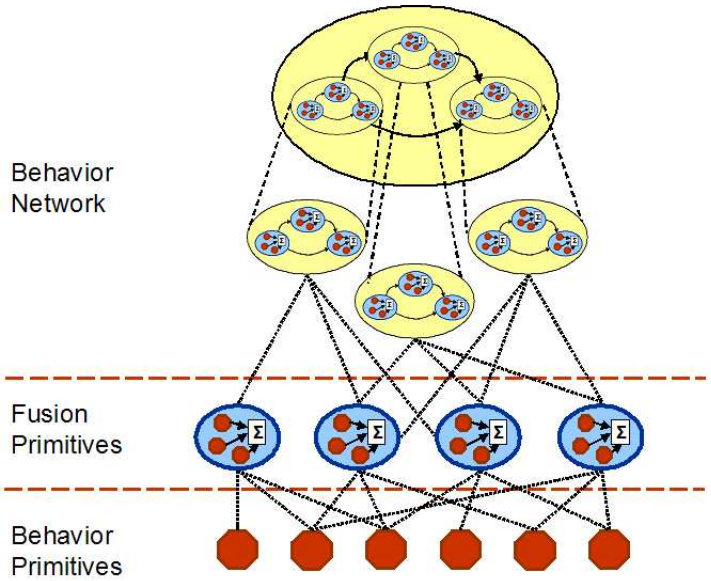

Fig. 1. A generic hierarchical task representation. The links between fusion primitives in the behavior network represent task-specific precondition-postcondition dependencies.

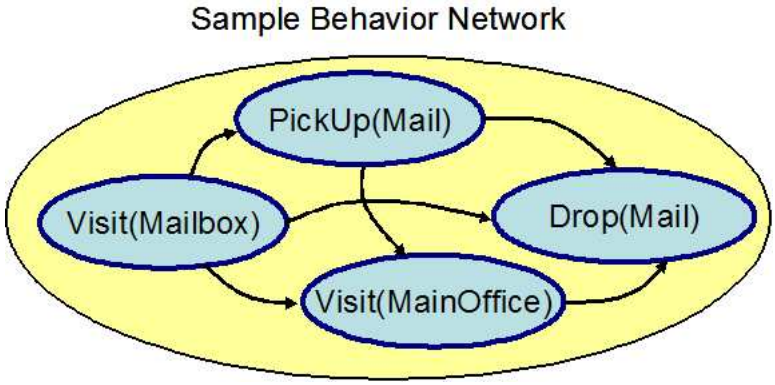

Fig. 2. Representation of a sample behavior network. The component nodes of the behavior network are fusion primitives. The links between fusion primitives in the behavior network represent task-specific precondition-postcondition dependencies.

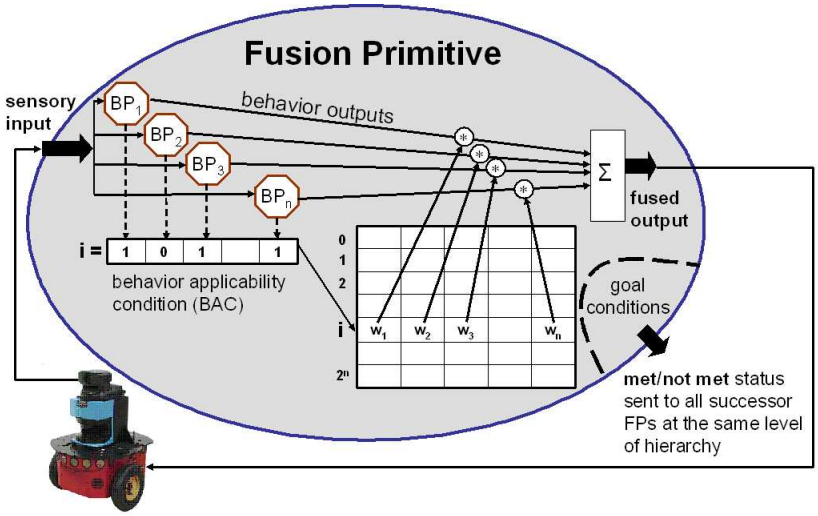

Fig. 3. Representation of a fusion primitive, the key component of our architecture. BP: behavior primitives.

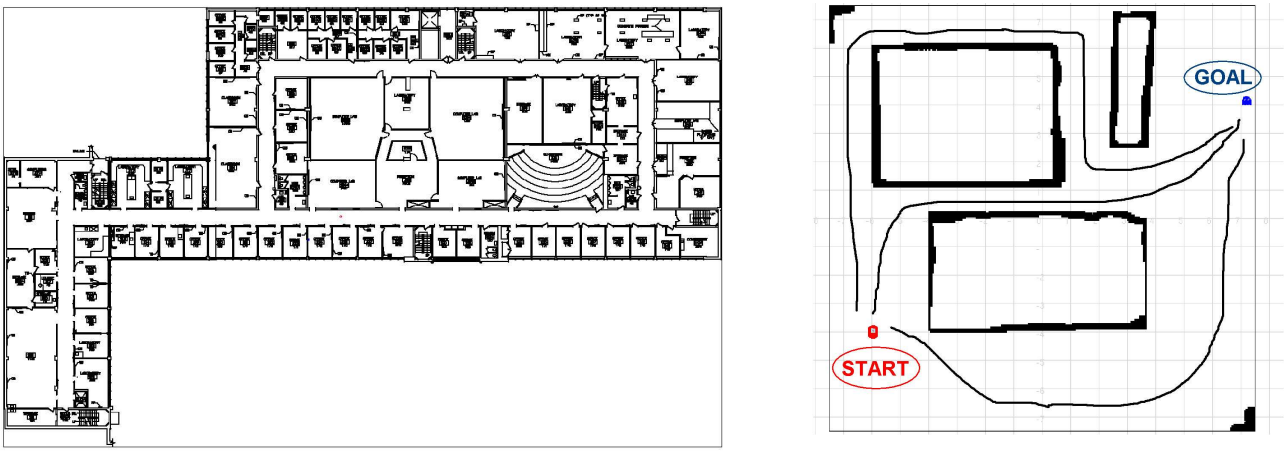

Fig. 4. Experimental layout. Left: sketch of building map; Right: small simulated environment.

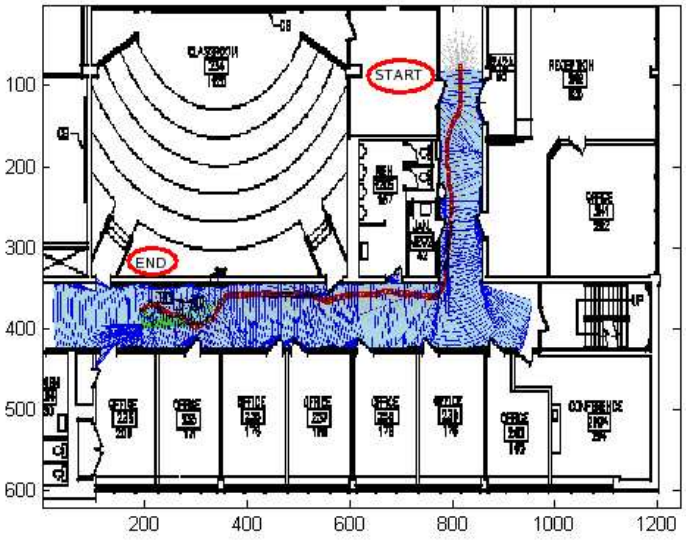

Fig. 5. Results from Scenario 1. The robot uses the weights from the center demonstration.

Avoid (LA):	produces a vector that is the sum of all repulsive vectors coming from obstacles closer than a threshold d_{max} distance, which are detected by the laser rangefinder.
Wander (WX):	produces a vector that is the sum of all vectors pointing in a direction in which the laser does not detect any obstacles closer than a threshold d_{max} distance.
Attract (BAX):	produces a vector pointing in the direction of the goal (which could be detected either by the camera or by the laser rangefinder).
Wall Attract (WFX):	produces a vector that is the sum of all vectors perpendicular to any detected objects, which are closer than a threshold d_{max} distance.
Sonar Avoid (SA):	similar to *avoid*, only that it considers the rear sonars of the robot.
Tangent Wall Follow (WSX):	produces a vector that is the sum of all vectors tangent to any detected objects, which are closer than a threshold d_{max} distance (for the left and respectively right 90-degree range in front of the robot).
Circular Avoid (CAX):	produces a vector pointing 30-degrees to the left and respectively right, if any obstacles are detected by the laser rangefinder in a 10-degree cone in front of the robot.

TABLE I. Listing of basic low-level behaviors used in our experiments. X - represents either *Right* or *Left*

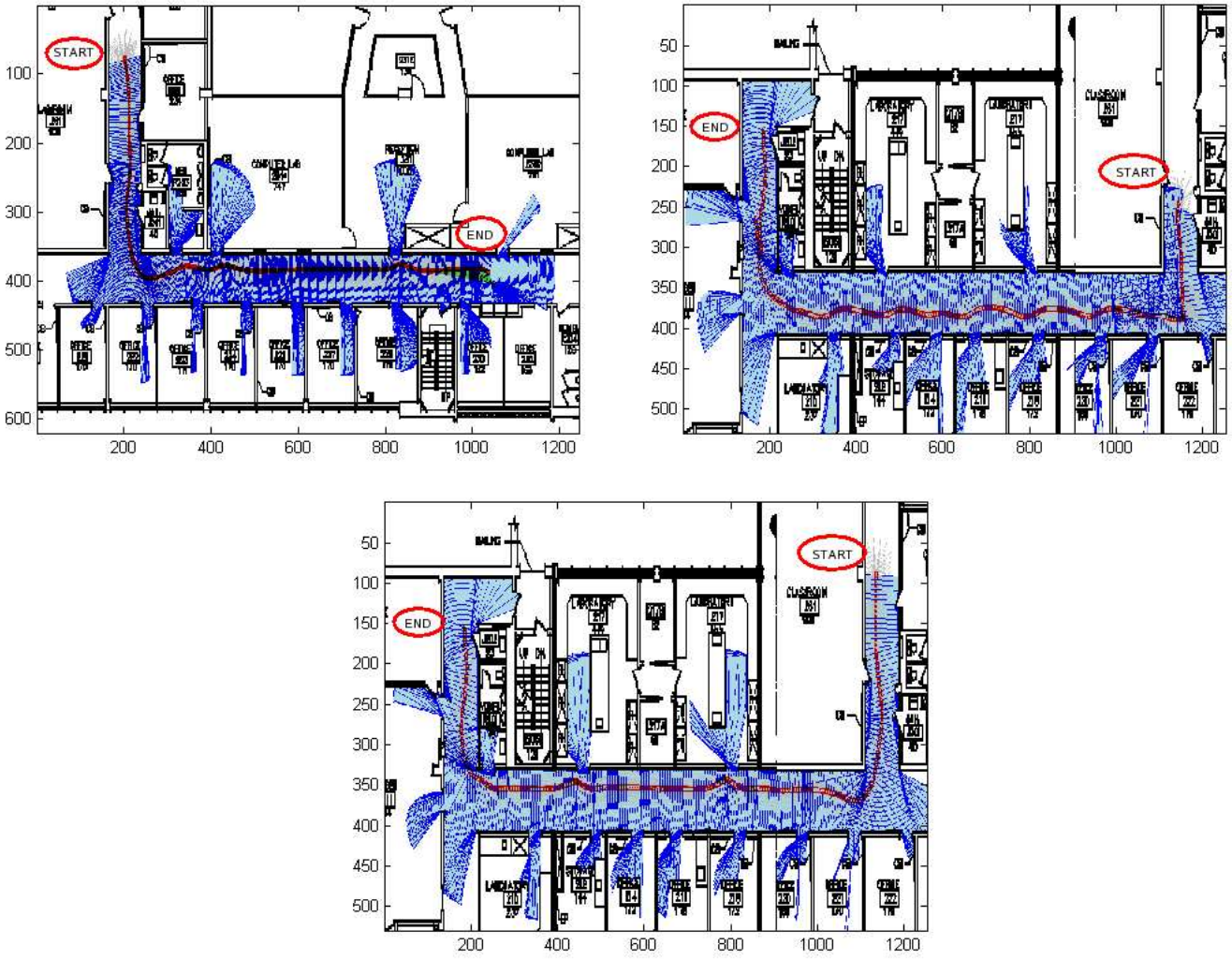

Fig. 6. Results from Scenario 3. The robot used learned fusion weights from the left, center and respectively right side demonstrations.

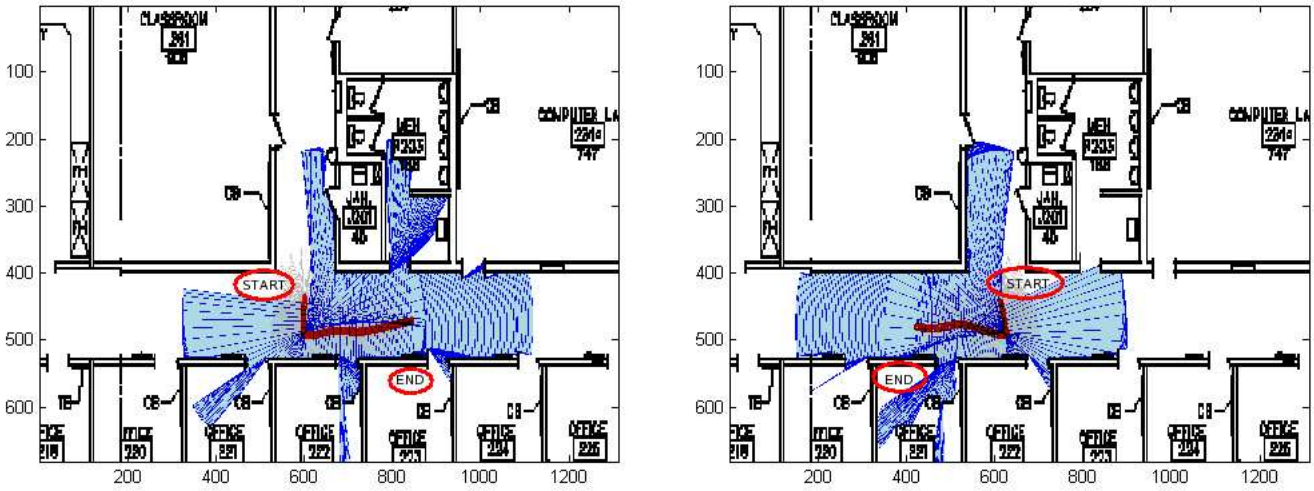

Fig. 7. The robot learns preferences of wandering in directions specified during the demonstration (top and bottom respectively).

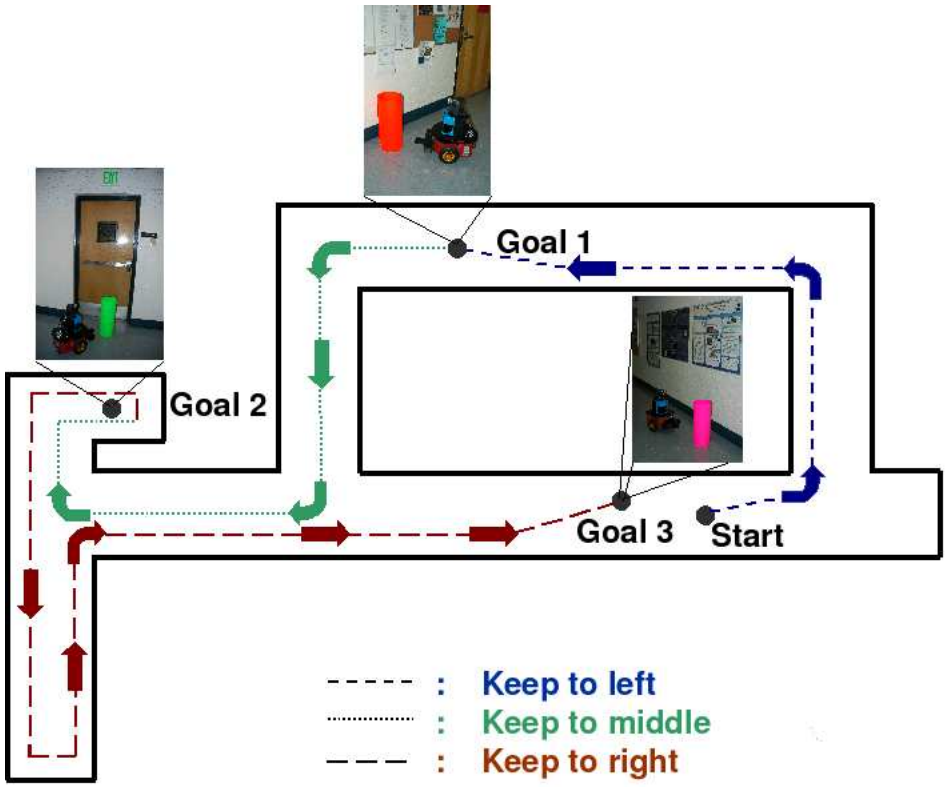

Fig. 8. Experimental setup for real-robot environment

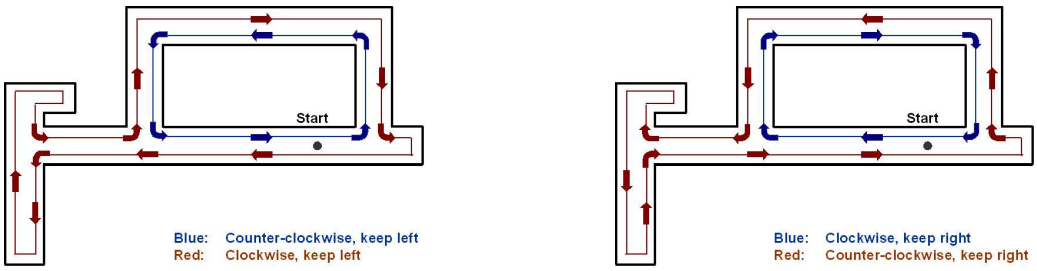

Fig. 9. Setup for validation of navigation styles. Left: navigation on left, Right: navigation on right

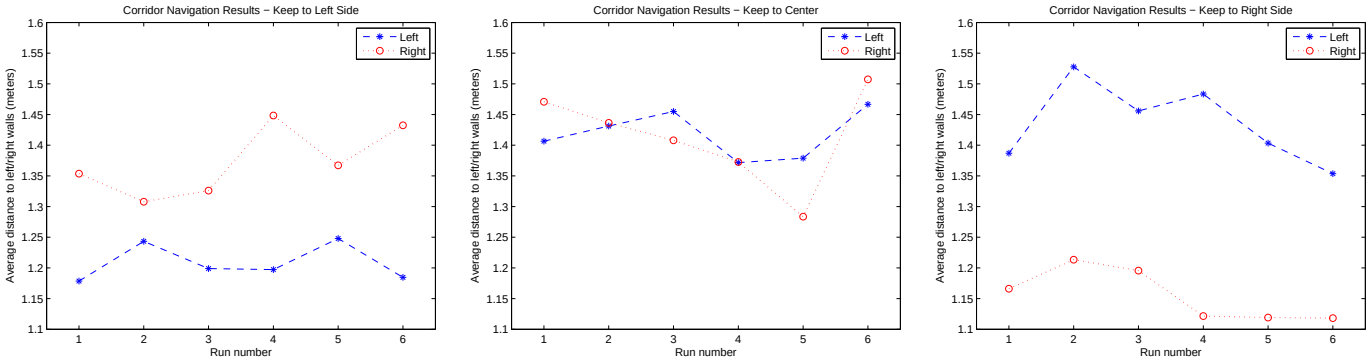

Fig. 10. Validation of navigation styles: average of distances to left and right walls (as reported by the laser)

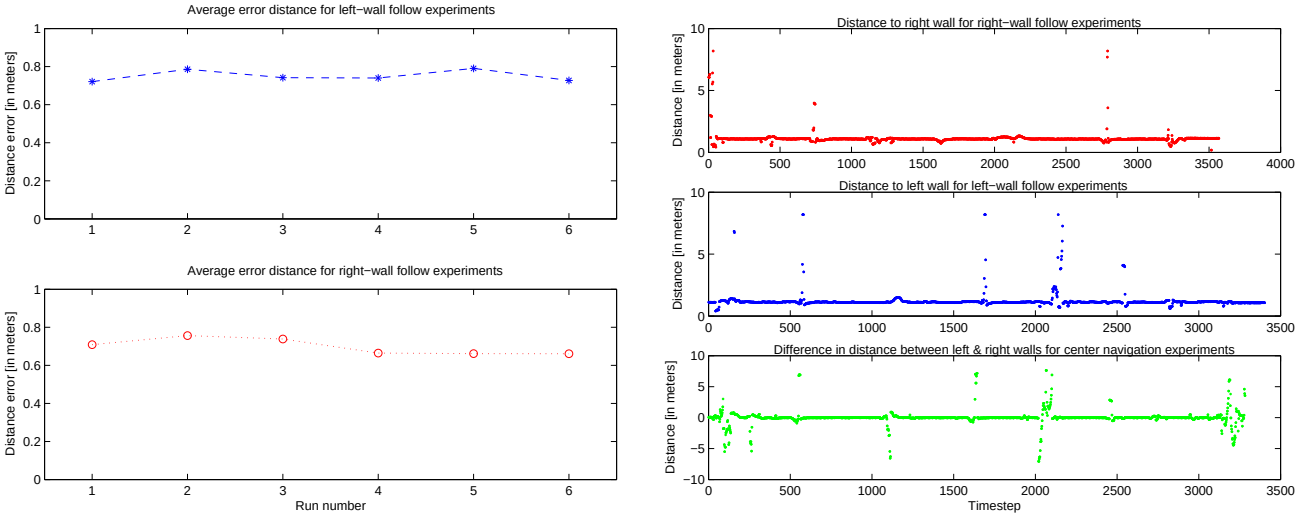

Fig. 11. Quantitative and Qualitative evaluation of learning different navigation styles. Left (for all runs): error between demonstrated and learned distance to left/right walls. Right (for a single run): Distance to left/right walls during runs for left/right navigation; also, error between distance to the left and right walls for center navigation.

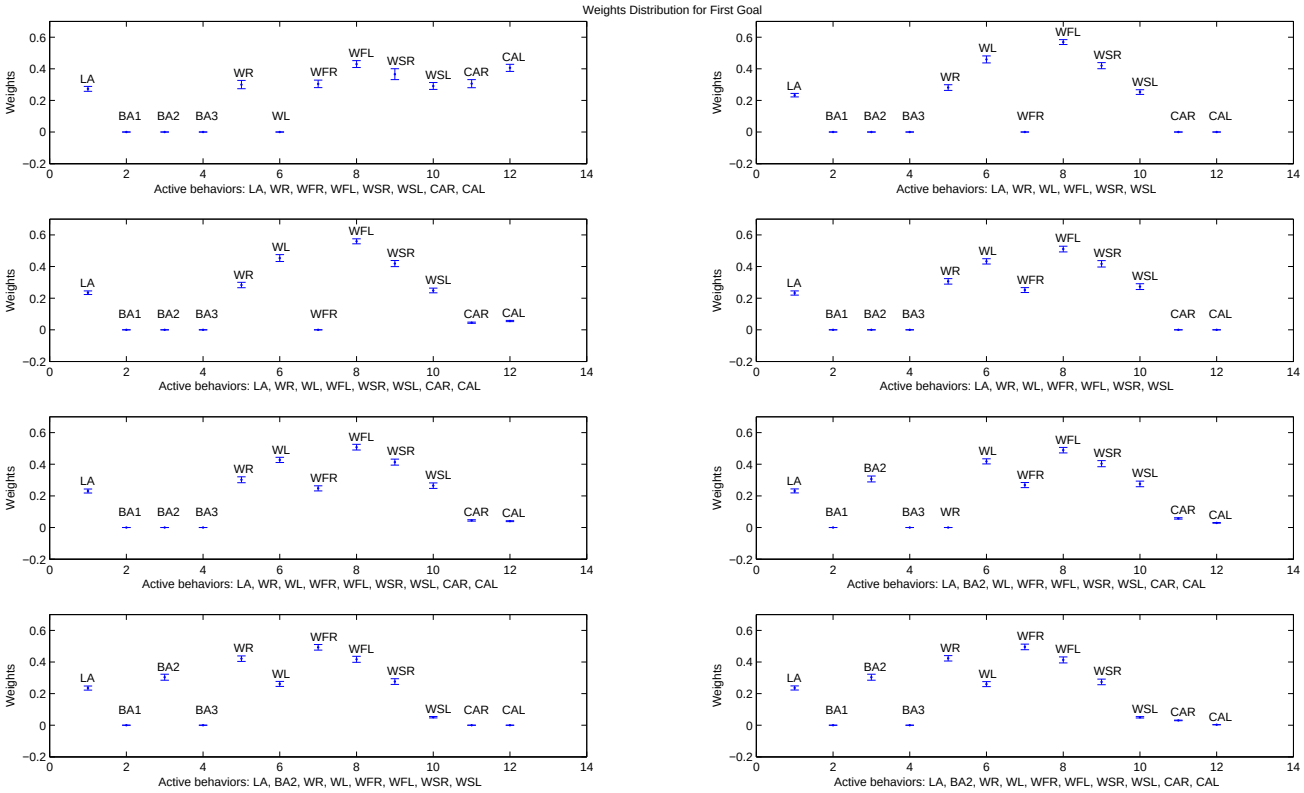

Fig. 12. Fusion weights learned for the First Goal of Scenario 1 (Goal 1 is detected by behavior BA2). Mean and standard deviation for each primitive behavior shown for 30 trials. Primitive behavior acronyms are listed in Table I.

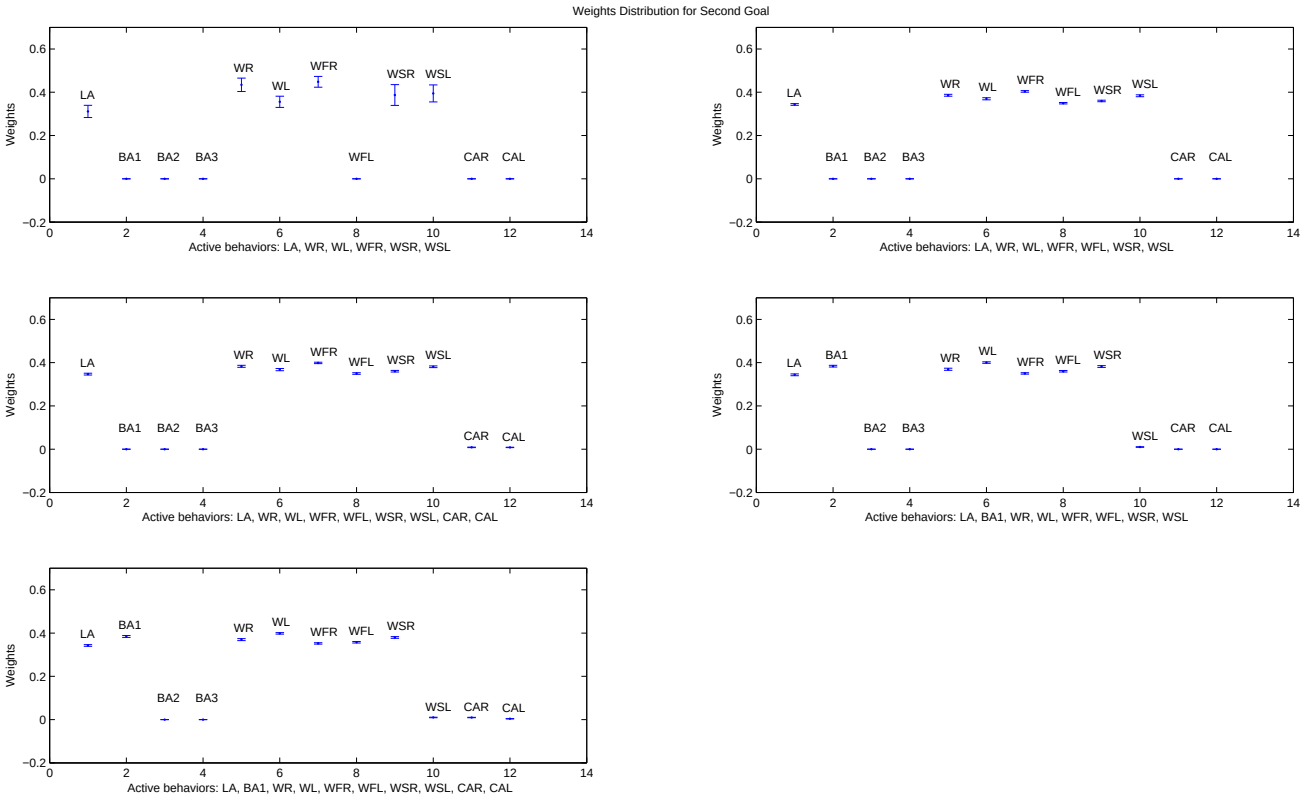

Fig. 13. Fusion weights learned for the Second Goal of Scenario 1 (Goal 2 is detected by behavior BA1). Mean and standard deviation for each primitive behavior shown for 30 trials. Primitive behavior acronyms are listed in Table I.

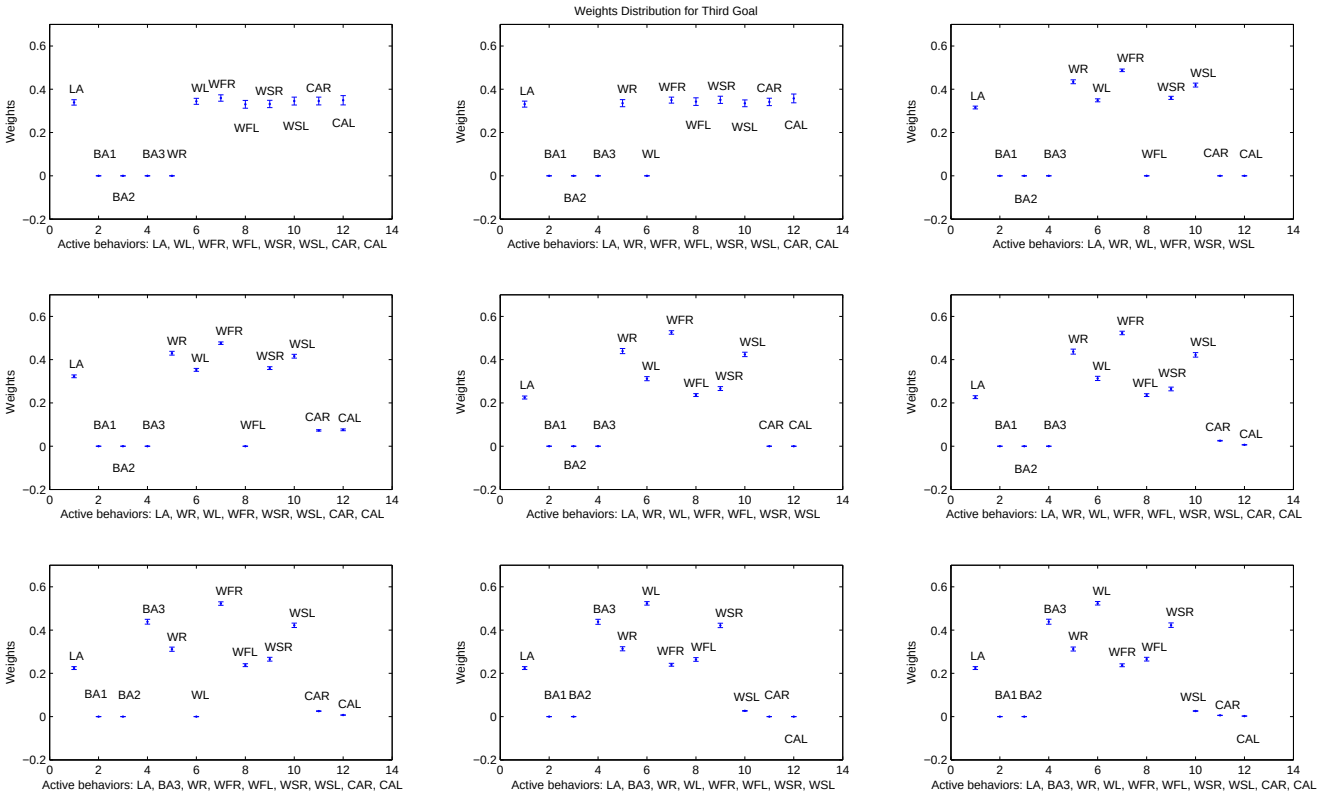

Fig. 14. Fusion weights learned for the Third Goal of Scenario 1 (Goal 3 is detected by behavior BA3)). Mean and standard deviation for each primitive behavior shown for 30 trials. Primitive behavior acronyms are listed in Table I.

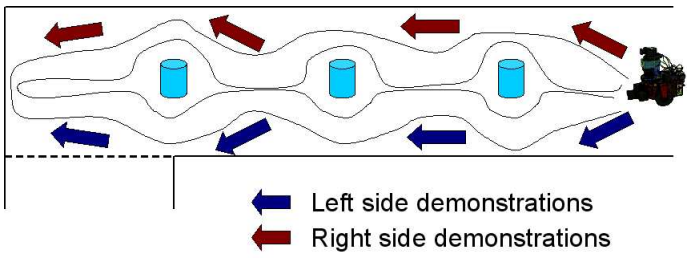

Fig. 15. Experimental setup for negotiating obstacles.