Fusing Robot Behaviors for Human-Level Tasks

Monica Nicolescu
University of Nevada, Reno
1664 N. Virginia St. MS 171
Reno, NV, 895231
monica@cse.unr.edu

Odest Chadwicke Jenkins
Brown University
115 Waterman St., 4th Floor
Providence, RI 02912-1910
cjenkins@cs.brown.edu

Austin Stanhope
University of Nevada, Reno
1664 N. Virginia St. MS 171
Reno, NV, 895231
stanhope@cse.unr.edu

***Abstract*— Behavior-based control is one of the most widely used approaches for autonomous robot control. However, in many robot systems, there is often a disconnect between a user's desired task-level behavior and a robot's preprogrammed (innate) capabilities. Typically, the space of robot behavior is limited to sequential performances, switching between the robot's available skills. Such limited expression does not necessarily overlap with the space of desired robot behavior, leaving users unable to express their true desired control policy to the robot. To bridge this divide, a new approach is proposed, which** *integrates state estimation (as a particle filter), learning by demonstration, and behavior-based control* **into an approach for robot learning. While these methods have typically been used in different contexts, we demonstrate the ability to use state estimation in order to learn a user's intended control policy from demonstration as a linear combination of innate behaviors. Through a specific navigation task, this method demonstrates how the same task-level behavior can be learned with different combinations of innate behaviors.**

I. Introduction

The problem of designing autonomous robot controllers 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.

This paper integrates multiple artificial intelligence techniques in order to address this *policy transfer* problem in a manner that facilitates task learning from demonstration. In particular, we synthesize *behavior-based control*, a typical approach to distributed robot coordination, with *Bayesian state estimation*, often used for robot localization and perception. The result is a novel approach to learning from demonstration, allowing a user's intended decision making policy to be imparted onto a robot. This synergy is formed by using state estimation to infer appropriate linear combinations of low-level, generic robot behaviors that accord to demonstrated human-level behavior.

A 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, in which multiple behavior models compete for prediction on the teacher's behavior [1]. The behavior with the most accurate prediction is the one said to match the observed action. These approaches rely on the assumption that there exists a unique corresponding behavior underlying any one of the demonstrator's actions [2]. The success of these methods is highly dependent on the existence of a special-purpose set of robot primitives, designed in particular for the target task. This limits the generality of these methods, as it requires that robot programmers have knowledge of the target task and be able to decompose it into appropriate functional modules.

This paper aims at overcoming these limitations, by taking a different perspective on the representation of robot controllers that can be learned by demonstration. Biological evidence, such as the schema theory [3], suggests that motor behavior is typically expressed in terms of concurrent control of multiple different activities. In addition, evidence from cognitive psychology [4] indicates the existence of a hierarchical model of motor control, in which a *willed behavior* control mechanism determines execution of activities through *concurrent activation of multiple motor primitives*. This view is also similar to the concept of *basis behaviors* [5], in which a complete set of elementary primitives can generate the entire span of behavior for a robot. Akin to the *basis behaviors*, typical innate behaviors in biological systems are known to be very simple responses to world stimuli: *reflexes*, *taxes*, *fixed-action patterns*, and *motivated behaviors* [6].

Considering these findings, this paper proposes a novel approach to learning a *task policy* from demonstration: using a set of low-level, generic behavior primitives expressed as schemas (or potential fields) [3], the method learns a coordination policy that linearly fuses the behaviors' combined output in a manner that matches the teacher's demonstration. The learning of this coordination is phrased as a fusion estimation problem, i.e., state estimation in the space of linear combinations of primitive behaviors. This is performed using

a particle filter that infers fusion estimates from robot sensory observations and motor commands. This method eliminates the need for task-specific knowledge in choosing the behavior primitives and it allows for increased generalization over the primitive set. The robot experiments described below demonstrate two significant results: **1)** the behavior fusion allows a robot to capture the intended policy of the human from demonstration without specialized underlying behaviors and **2)** the same task policy can be learned as the superposition of different underlying sets of innate primitives. These results validate the claim that special purpose behaviors and task knowledge are not necessary for learning a control policy from demonstration and also that increased robustness can actually be achieved with low-level, generic primitives, by providing multiple solutions to the policy learning problem.

II. Related Work

Successful approaches to learning from demonstration (LFD) [7] have demonstrated learning of reactive policies [8], trajectories [1], and sequential task descriptions [9]. Such sequential (or arbitrated) representations for coordination can be considered a subset of the approach proposed in this paper.

In making arbitration decisions, sequential methods represent all possible coordinations as discrete set of possibilities along each axis of fusion space. [10] 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, the comparison focuses only on the behavior coordination mechanism and not on the platform specifics. Other approaches to LFD use various forms of supervised learning to create a mapping from input states to output actions. [11] use piecewise statistical regression bootstrapped by demonstration and planning. [12] 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.

In terms of fusion estimation, neural networks have been previously employed in LFD. In [13] the fusion is done over input data from multiple sensory modalities. In this paper, fusion is performed over a set of "innate" robot primitives that are cooperatively coordinated from first or third person demonstration.

The choice of the particle filter [14] is only one of several methods available to infer behavior fusion. Alternatively, the problem can be cast as a parameter search. The most straightforward choice, linear least squares optimization, worked well when the number of primitives is small and likelihood ambiguity is negligible, based on our initial testing. Nonlinear methods, such as Levenberg-Marquardt or Nelder-Mead, could yield better results. However, the particle filter allows to account for ambiguity explicitly.

III. Behavior Representation

The behavior-based controllers used in this work are built from two components: *behavior primitives* (BPs) and *fusion primitives* (FPs), which are combined in *behavior network controllers*. A behavior network controller is a directed acyclic graph of FPs, in which the links between nodes represent task-specific activation conditions that enable the representation and execution of sequenced tasks.

The *behavior primitives* perform a set of actions under given (relevant) environmental conditions. 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 BP component brings its own contribution to the overall motor command. These contributions are weighted and fused through vector addition. These weights affect the magnitude of the individual vectors coming from each behavior, thus generating different modalities of execution for the task.

Each *fusion primitive* (Fig. 1) has a representation of the goals it achieves, expressed as abstracted environmental states. The state of the goals is continuously monitored and updated from sensory data. The component *behavior primitives* receive information from the sensors, which is first used to detect if the behavior is active or not, given its preconditions. The *active/not active* status of all behavior primitives is encoded in a N-dimensional vector, where N is the number of BPs. This vector, which we call a *behavior applicability condition* (BAC), contains for each behavior a 1 or a 0, depending on whether the behavior is active 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 preconditions. 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. For each possible BAC, the *fusion primitive* has a different set of fusion weights, which are used for behavior combination. The sets of weights for the multiple possible BACs are stored in a table, as shown in Fig. 1. The index of each row in the table is the decimal equivalent of the binary N-bit BAC value.

The weights from the corresponding BAC modulate the magnitude of control vector output by the individual primitives, thus influencing the resulting command from fusion. At each timestep t, each *behavior primitive* BP_i provides a response output vector v_i^t, which represents a desired heading for the robot. The *fusion primitive's* output V_r^t is a linear combination of the vectors $[v_1^t \cdots v_N^t]$, according to the BAC superposition weights $S^t = [s_1^t \cdots s_N^t]$:

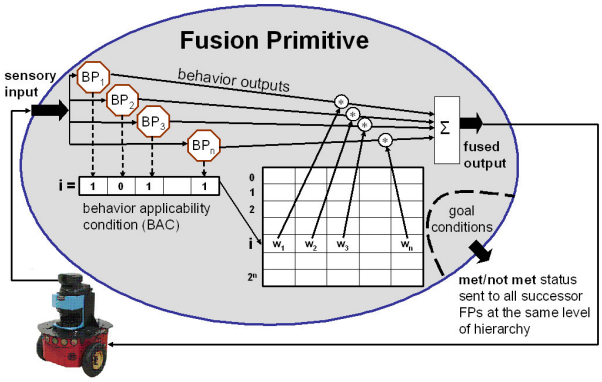


Fig. 1. Representation of a fusion primitive, the key component of the architecture. BP: behavior primitives.

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

We consider heading to be the most important consideration for behavior fusion in 2D navigation[1]. Consequently, command vectors are normalized to unit length.

IV. Learning by Demonstration Methodology

For this work, demonstrations consisted of guiding the robot through navigation tasks, using a joystick, while the robot's behaviors continuously provide a response command 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. The demonstration could also be performed by having the robot follow a human user, similar to work in [9]. However, instead of being translated into motor commands, the behaviors' outputs are recorded along with the turning rate of the robot, at that moment of time. This information is further processed, as described below.

Segmentation of observation traces. During training empty-shell FPs are created, which contain only information about the goals that the robot needs to detect, and whose only role during the demonstration is to monitor their goal status. Goals are abstracted environmental states, which can be monitored from sensory information. Every time the goal of a behavior is met, this generates a new segment in the demonstration, which will result in a new FP node in the task network. Thus, if the same goal is met multiple times during training, it will appear in multiple instances in the task representation.

Within each segment multiple *situations* are possible, in which different subsets of behaviors could be active (expressed as the BACs), depending on whether their preconditions are met or not. 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 and computed as follows.

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.

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 have been used for state and parameter estimation in several perception-oriented domains (such as robot localization [14] and insect tracking [15]).

Our method follows the same 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 behavior outputs and result vectors:

$$p(S^t|V_r^{1:t}, V_p^{1:t}) = \tag{2}$$
$$kp(V_r^t|V_p^t, S^t) \int p(S^t|S^{t-1}) p(S^{t-1}|V_r^{1:t-1}, V_p^{1:t-1})$$

where $p(V_r^{1:t}|S^t, V_p^{1:t})$ is the *likelihood* of observing a result vector given a vector of fusion parameters, V_p are the outputs of the primitives, $p(S^t|S^{t-1})$ 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.

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. Our likelihood function weights a particle i as the distance between the actual and the displacement direction given by the primitive behaviors:

$$\pi_{(i)}^t = p(V_r^t|V_p^t, S^t) = 2 - D(V_r^t, \hat{V}_{(i)}^t)/2 \tag{3}$$

where $D(a, b)$ is the Euclidean distance between a and b and the direction vector $\hat{V}_{(i)}^t$ normalizes to unit length a weighted sum over individual primitive outputs.

The process described above is performed on each of the recorded instances of time within a BAC sub-segment, resulting in a posterior distribution of fusion weights for every time

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

step during demonstration. To enable crisp decision making, a single state is extracted from the fusion posterior distribution, which is the mean of the posterior distribution.

Construction of Task Representation. For all the fusion primitives corresponding to the identified goals, the controller will include all sets of weights corresponding to different behavior applicability conditions (BACs) that were detecting during training. The fusion primitives resulting from the goal-achieving segments are encapsulated in a node in the behavior network and are linked in the order in which they occurred.

V. Experimental Results

In the validation experiments, a Pioneer 3DX mobile robot was trained to perform 3 different styles of navigation, in an office building environment. The robot was equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera, and was programmed using Player [16].

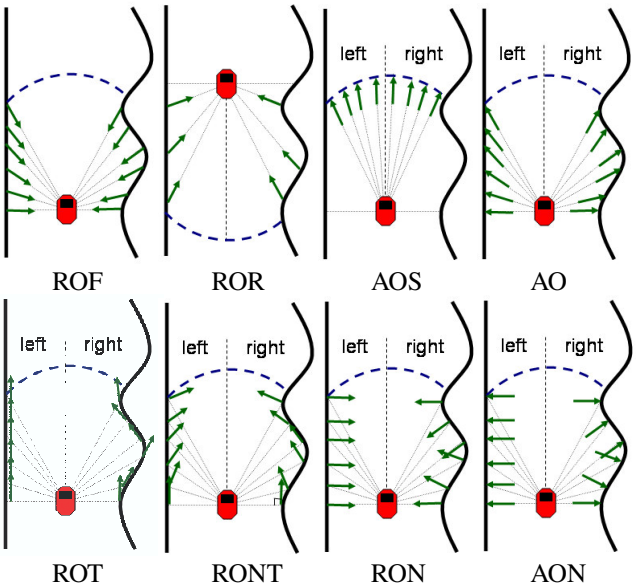


Fig. 2. The set of innate behaviors. The blue dashed line indicates the range d_{max}.

The robot's set of innate behaviors (Fig. 2) consists of 14 behaviors, whose output is a sum of vectors as follows:

- **Repulse obstacles front (ROF):** sum of all repulsive vectors from obstacles closer than d_{max}.
- **Repulse obstacles rear (ROR):** sum of all rear repulsive vectors from obstacles closer than d_{max}.
- **Attract open space (AOS):** sum of all vectors pointing toward open space farther than d_{max}.
- **Attract obstacles (AO):** sum of all vectors toward all detected objects closer than d_{max}.
- **Repulse obstacles tangent (ROT):** sum of all vectors tangent to any detected objects closer than d_{max}.
- **Repulse obstacle normal tangent (RONT):** sum of all vectors perpendicular to the direction toward obstacles closer than d_{max}.
- **Repulse obstacle normal (RON):** sum of all vectors normal to the tangent to any detected objects closer than d_{max}, pointing away from objects.
- **Attract obstacle normal (AON):** sum of all vectors normal to the tangent to any detected objects closer than d_{max}, pointing toward objects.

Except for ROF and ROR, all other behaviors have two versions, for the left and respectively right 90-degrees in front of the robot. Distance d_{max} was set to 2 meters and was detected with the laser rangefinder (in the front) and with the sonar array (in the back). The behaviors are activated by preconditions that test if certain laser readings are smaller or respectively larger than the d_{max} threshold.

The three different types of navigation consisted of: 1) navigating close to the right walls, 2) navigating close to the left walls, 3) navigating in the center of the corridor. For each type, a clockwise and counterclockwise experiment was performed, resulting in 6 training experiences (Fig. 3).

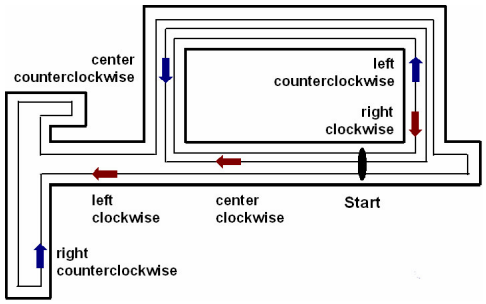


Fig. 3. Sketch of experimental setup.

The goal of these experiments is to demonstrate that the different navigation styles could be learned without the need for special purpose routines and furthermore, that the same task can be learned using different underlying sets of primitives. Toward this end, the training data was processed considering all possible subsets of primitives. That is, the particle filter was applied to all possible combinations of behaviors, resulting in $2^8 - 1 = 255$ controllers for each of the 6 training runs (the empty subset was ignored). For the *left wall* navigation, each of the 255 controllers (representing 255 fusion combinations) has been evaluated by running it on a robot in a simulated, accurate map of the building, using Stage [16]. The robot was started in a random position in the corridor (to determine if it is able to get into the correct navigation position) and allowed to navigate for a period of time (usually 2 minutes). For each controller, a qualitative evaluation was performed, to determine if the robot had captured the main aspects of the 4 navigation styles:

- Right follow (both runs): navigate on the right, take right turn at T or Y-junction.

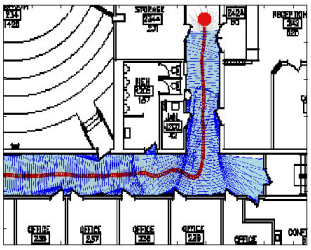
ROF	ROR	AOS	AO	ROT	RONT	RON	AON
				X	X		
			X		X		
		X			X		
		X	X				
X		X					
		X				X	X
	X				X		X
X					X		X

TABLE I

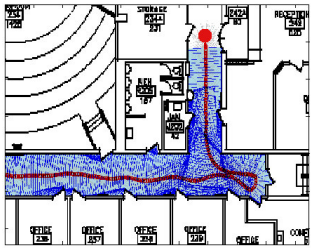
BEHAVIOR SUBSETS (OF 2 AND 3 BEHAVIORS) WHICH LEARNED THE LEFT-FOLLOW TASK. MULTIPLE BACS EXIST WITHIN EACH SUBSET.

- Left follow (both runs): navigate on the left, take left turn at T or Y-junction.
- Center follow (clockwise): navigate in center corridor, take right turn at T or Y-junction.
- Center follow (counterclockwise): navigate in center corridor, take left turn at T or Y-junction.

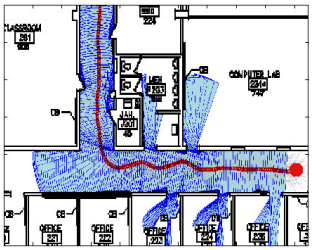
The evaluation of the 255 controllers lead to the expected results: a significant number of controllers, corresponding to different sets of underlying primitives, demonstrated successful learning of the navigation policies. Table I shows the combinations of 2 and 3 behaviors that were able to learn the task from the left-follow, clockwise demonstration. In addition to those, there were 14 other successful combinations of 4 behaviors or more, including the subset that contained all the behaviors. Also, the actual number of successful fusion combinations is significantly larger than $2+3+14 = 19$, due to the fact that there exist additional combinations, of more behaviors, which contain the groups of 2 and 3 behaviors shown in Table I.



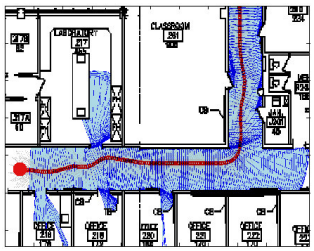
a) Center clockwise



b) Center counterclockwise



a) Right



b) Left

Fig. 4. Robot trajectories using controllers from the different navigation styles. The robot's path is indicated in red. The starting position is indicated by a red circle.

For the controllers resulting from the other 5 demonstrations, the Stage evaluation was performed for the fusion combinations that were found successful above and for a random set of previously unsuccessful combinations. The results show that with approximately 95% accuracy, the controllers' performance corresponded with their counterpart (same fusion combination) from *left wall* navigation task. These results indicate that the fusion process is consistent over the set of existing primitives. The small differences are expected due to variations in the environment (people, doors) during the individual demonstrations.

The robot trajectories shown in Fig. 4 demonstrate qualitatively that the robot had captured the correct tasks. From the *center clockwise* demonstration the robot learns to navigate in the center of the corridor and to take a right turn at junctions. From the *center counterclockwise*, it learns a similar navigation style, but it now takes left turns at junctions. From the *right* and *left* demonstrations, the robot learns to stay closer to right and respectively left walls, and to take the right/left turns at the junctions. Overall, these results also show that the robot exhibits a smooth navigation style, similar to what has been demonstrated.

As a quantitative evaluation, for the controllers that successfully learned the task (used in Fig. 4, and for a full run around the building), the distance from the robot to the left and right walls was recorded. The means of those values are as follows: a) Center right: mean left wall (MLW) = $1.99m$, mean right wall (MRW) = $2.13m$, b) Center left: MLW = $1.8m$, MRW = $2.03m$, c) Right: MLW = $2.81m$, MRW = $1.68m$, d) Left: MLW = $1.41m$, MRW = $3.13m$. For center navigation experiments, MLW and MRW have very close values. For right and left navigation, the mean distance to the wall the robot is supposed to follow is smaller. The standard deviation is irrelevant in this case, as the maximum values will always be the equivalent of an open space opening at junctions, which in this case is 8 meters.

The unsuccessful controllers exhibited behaviors such as getting stuck in a wall, oscillating (either in the middle of the corridor, or facing a wall), getting stuck in corners, or failing to take the correct turns at the junctions. The controllers' performance provided a wealth of information regarding the correlations between behaviors. For example, whenever ROT and RONT were used, behaviors RON, AON and ROR had no impact on the success of the controller (i.e., either if used or not, the robot still performed the task correctly). Similarly, whenever one of the RON or AON were used, if RONT was not used, the robot would always get stuck in a corner. A complete analysis of the behavior correlations is outside the scope of this paper, but will be performed in future work to gain insight into the role of each behavior for specific tasks.

VI. DISCUSSION OF RESULTS

A requirement for the applicability and success of this method, is that the robot must be presented with represen-

tative examples of this policy in action. Towards this end, human teachers must have some knowledge of controlling the robot, but not necessarily of its 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.

When used alone, none of the innate behaviors is able to learn the tasks or to produce any consistent behavior. However, even simple weighted combinations of 2 or 3 general purpose primitives are able to capture detailed aspects of a navigation policy, such as taking left/right turns at junctions and navigating closer to a left/right wall. If designed by hand, or by a learning from demonstration strategy that relies solely on behavior sequencing, controllers for the tasks demonstrated above would have required specialized behaviors for following walls and for dealing with junctions. In contrast, by expressing the learning problem as an estimation of behavior fusion weights, such tasks can be learned from low-level, generic primitives.

The advantage of the proposed method is twofold. First, it *removes the need for prior knowledge of the task* that needs to be learned, thus reducing the robot programmer's effort for designing the underlying behavior set. Second, it *provides increased generalization capabilities*, decreasing the discrepancy between the robot's innate capabilities and the users' intended task policy. The experimental results demonstrate these advantages and show how specific navigation tasks can be learned as a superposition of multiple low-level primitives, in the form of *fusion primitives*.

While only one-step tasks have been learned in this paper, the proposed fusion estimation method has also been employed in learning more complex, sequential task representations, thus indicating its potential for use in real-world, service robotics applications [17]. Future work will also seek to develop methods that would flexibly select or switch between the multiple solution policies of a task.

VII. Summary

The problem of designing autonomous robot controllers is typically restricted by the pre-existing innate robot capabilities. This paper proposes a novel approach to constructing such a task policy from demonstration. The method integrates particle filtering, learning from demonstration, and behavior-based robotics, in an approach that allows a robot to map the demonstrator's actions onto a linear combination of multiple, low-level behavior primitives. Central to this approach are the estimation of behavior fusion to map the demonstrator's actions onto the robot control repertoire and the use of general-purpose, low-level robot behaviors. Through fusion of these behaviors, the method removes the need for task knowledge and special-purpose robot primitives, increases the robustness of learning and provides extended generalization capabilities.

VIII. Acknowledgments

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

[2] 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.

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

[4] 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.

[5] M. J. Matarić, "Behavior-based control: Examples from navigaton, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2–3, pp. 323–336, 1997.

[6] R. D. Beer, H. J. Chiel, and L. S. Sterling, "A biological perspective on autonomous agent design," *Robotics and Autonomous Systems & 2, June 1990*, vol. 6, no. 1, pp. 169–186, 1990.

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

[8] 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.

[9] 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, July 2003.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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*, 2003, pp. 317–323.

[17] M. Nicolescu, O. C. Jenkins, A. Olenderski, and E. Fritzinger, "Learning behavior fusion from demonstration," *Interaction Studies Journal, Special Issue on Robot and Human Interactive Communication*, to appear, 2007.