Behavior Fusion Estimation for Robot Learning 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
monica@cse.unr.edu

Abstract

A critical challenge in designing robot systems that learn from demonstration is the ability to map the behavior of the trainer as sensed by the robot onto an existing repertoire of the robot's basic/primitive capabilities. Observed behavior of the teacher may constitute a combination (or superposition) of the robot's individual primitives. Once a task is demonstrated, our method learns a fusion (superposition) of primitives (as a vector of weights) applicable to situations encountered by the robot for performing the same task. Our method allows a robot to infer essential aspects of the demonstrated tasks without specifically tailored primitive behaviors. We validate our approach in a simulated environment with a Pioneer 3DX mobile robot. We demonstrate the advantages of our learning approach through comparison with manually coded controllers and sequential learning.

1. Introduction

Learning from demonstration is a natural method for extending a robot's repertoire of basic skills to more sophisticated task descriptions. A primary difficulty of this approach is the *symbols-from-signals* problem. That is, the interpretation of observation signals gathered by the robot's sensors during instruction and translation of said observations into appropriate *skills* or *task descriptions*. If the robot is endowed with a basic set of capabilities (or primitives), interpreting the demonstration becomes a problem of creating a mapping of the perceived behavior of the teacher onto a set of existing primitives [18].

Behavior-based control [4] is particularly well suited for autonomous robot control and learning from demonstration [15] due to its modularity and real-time response properties. While behavior-based control emphasizes the concurrent and parallel coordination of behaviors, existing learning by demonstration strategies have mostly focused on learning serially sequential mappings of observed behavior, in which only one primitive module can be active at a given time. However, this serial coordination assumption is not valid in a significant number of situations and greatly limits the applicability and autonomy of an established set of controllers. Biological evidence, such as schema theory [2], suggests that motor behavior is typically expressed in terms of concurrent control of multiple different activities. This view is also similar to the concept of *basis behaviors* [14], in which a complete set of underlying primitives can generate the entire span of behavior for a robot.

Based on these considerations, it becomes of significant interest to softly decompose a demonstrator's behavior with respect to a robot's perceptual-motor repertoire, avoiding hard binary decompositions. We argue that such a superposition-based strategy will enable robots to learn aspects of the demonstrated task that could not otherwise be captured by standard approaches to learning from demonstration that treat primitives as mutually exclusive control mechanisms. For our approach to superposition for coordination, we assume that a robot has been endowed with an *a priori* repertoire of primitive control modules (behaviors), encoded using a schema-based representation [3]. This representation uses a vector-based output format, allowing for cooperative behavior coordination by fusing commands from multiple behaviors through vector addition.

In this paper, we present *behavior fusion estimation* as a method to learn from demonstration fusion weights for coordinating concurrently executing primitives (as dynamical forward models) through superposition. We aim to find the contribution, expressed as a weight, of each of the primitive behaviors across the demonstrated task. These weights incorporate essential information about coordination and can capture subtle differences between tasks, which, even if achieving the same goals, can lead to very different executions of a task. Our method uses a particle filter to infer superposition weighting parameters for an arbitrary number of behaviors in a Bayesian fashion.

We demonstrate that the proposed approach learns con-

Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)
0-7695-2589-X/06 $20.00 © 2006 IEEE

trollers that have advantages over both hand-coded controllers and controllers built by a sequential learning by demonstration approach.

2. 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.

In the mobile robot domain, successful first person approaches have demonstrated learning of reactive policies [9], trajectories [7], or high-level representations of sequential tasks [15]. These approaches employ a *teacher following* strategy, where the robot learner follows a human or a robot teacher. Our work is similar to that of [1], who perform the demonstration in a simulated, virtual environment. With respect to humanoid robotics, Platt et al. [16] have demonstrated null-space composition as a fusion coordination mechanism limited to control states where behaviors do not affect each other.

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. [6]. One successful approach to this problem has been to match observations to robot behaviors based on *forward models* [17, 12], in which multiple behavior models compete for prediction of the teacher's behavior [20, 11]. The behavior with the most accurate prediction is the one said to match the observed action.

In this paper, our aim is to avoid hard binary decisions in coordination of and fusing control commands from different behaviors. We attempt to estimate the *fusion state*, in a manner similar to Monte Carlo Localization, of the robot during the demonstration.

3. BEHAVIOR REPRESENTATION

Behavior-Based Control (BBC) has become one of the most popular approaches to embedded and robotic system control both in research and in practical applications. We utilize a schema-based representation in the context of BBC, similar to approaches in [3]. 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.

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

4. 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 joint probability distribution of the fusion space (i.e., across weighting combinations) over the demonstration duration. 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. Thus, for each timestep t, we are provided with 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 known 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. 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.

4.1. Incorporating Behavior Preconditions

At any given time during the demonstration, multiple behaviors could be active, depending on whether their preconditions are met or not. We segment the demonstration into intervals based on the binary decisions set by the preconditions of each behavior. A 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 segments represent different environmental situations, since different behaviors become "applicable" at the transition points. The weights of behaviors within each segment encode the mode of performing the current task given the situation and, thus within each 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.

4.2. Fusion Parameter Estimation

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 [5] have been used for state and parameter estimation in several different domains (such as robot localization [19], pose estimation [10], and insect tracking [13]). Restating these methods, mostly following [13], 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:

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

where $p(V_r^{1:t}, V_p^{1:t}|S^t)$ 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:

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

The resulting Bayes filter:

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

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:

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

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:

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

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

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

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

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

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. Current particle filtering methods have three approaches to selecting appropriate parameters from this distribution as 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. 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.

5. EXPERIMENTAL RESULTS

To validate the proposed learning algorithm we performed experiments using the Player/Stage simulation environment [8] and a Pioneer 3DX mobile robot equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera. The robot is equipped with the following primitive behaviors implemented as potential fields: laser obstacle avoidance (*avoid*), attraction to a goal object (*attract*), attraction to unoccupied space (*wander*),

attraction to walls (*wallAttract*), sonar obstacle avoidance (*sonarAvoid*). *sonarAvoid* is distinct from the *avoid* due to differences in sensor specifications. All of these behaviors use information from the laser rangefinder and the camera (for goal detection).

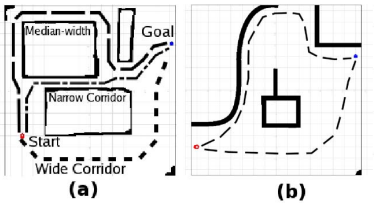

Figure 1. Experimental setup with possible paths: (a) Corridor traversal experiment, (b) Wander/wall follow experiment

5.1. Experimental Setup

We tested the learning algorithm described above in two scenarios, performed in the environments shown in Figure 1. For each scenario we performed two types of experiments, comprising the teaching by demonstration and the validation stages respectively. The following training experiments have been performed:

- **Corridor traversal.** We performed three types of training sessions, consisting of taking one of the three different paths to the goal: traverse the narrow, medium-width and wide corridors respectively. From this training, it is expected that the learning algorithm will derive a set of weights for the behaviors that will enable the robot to traverse the corridors and reach the goal.
- **Wander/wall follow.** We performed two types of training sessions, consisting of taking two different approaches for reaching the goal: navigate close to the walls in the environment (upper path in Figure 1) or through open spaces (lower path in Figure 1). From this training, it is expected that the robot will learn two different navigation strategies: one that keeps it very close to walls and one that only loosely attracts it to them.

To evaluate the performance of the learning algorithm we place the robot at various locations in the environment and equip it with an autonomous controller which uses the derived weights. More specifically, for each situation the robot encounters it will use the set of weights learned from the demonstration for that particular situation. Thus, the robot flexibly changes the contribution of each behavior as the environmental conditions are changing. If the robot successfully reaches the goal and performs the same strategy as demonstrated by the user (e.g., stays closer or farther from walls), the experiment is considered a success.

5.2. Results

Corridor Traversal Experiment. For this scenario we performed three separate experiments, one for each path indicated in Figure 1 (a). Furthermore, each of these experiments was repeated three times. Table 1 shows the weights that the robot had learned in one of the training sessions for the three paths. Each table shows the different situations that the robot encountered, as indicated by the combination ID. For example, an ID of 14 represents a situation in which [*avoid = 0, sonar = 1, goalAttract = 1, wander = 1, wallAttract = 0*], $01110_{binary} = 14$, that is the *sonar*, *goalAttract*, and *wander* behaviors were applicable. Two subsequent repetitions of these experiments (both learning and validation) led to similar results (with slight differences due to variability in the user's demonstration).

Table 1. Behavior weights learned for the narrow, medium width and wide paths

ID	Avoid	Sonar	GoalAttract	Wander	WallAttract
6	0	0	0.1013	0.9000	0
14	0	0.5610	0.4810	0.6103	0
19	0.4012	0	0	0.3391	0.7576
23	0.4198	0	0.4420	0.5962	0.2302
27	0.3063	0.3487	0	0.4742	0.6646
31	0.3075	0.3717	0.4216	0.6311	0.0870

ID	Avoid	Sonar	GoalAttract	Wander	WallAttract
14	0	0.2333	0.7304	0.5472	0
19	0.3694	0	0	0.5435	0.7095
23	0.3308	0	0.4403	0.3863	0.5597
27	0.2751	0.4094	0	0.4862	0.5993
31	0.2866	0.4127	0.4810	0.5877	0.0205

ID	Avoid	Sonar	GoalAttract	Wander	WallAttract
10	0	0.0425	0	0.9945	0
14	0	0.0453	0.9346	0.0749	0
19	0.4394	0	0	0.5607	0.6471
23	0.4285	0	0.5328	0.6324	0.1106
27	0.3217	0.4206	0	0.5479	0.4537
31	0.3325	0.4020	0.5544	0.3683	0.1478

For each one of the three sets of weights from above we performed three validation experiments, for a total of nine runs. In all these experiments, the robot successfully traversed the environment and reached the goal. Irrespective of the set of weights that was used, the robot was able to navigate through the median-width and the wide corridors, but avoided entering the narrow-width corridor by itself. The robot chose to stay relatively closer to the wall when using the weights from the narrow and medium-width paths, versus those from the wide corridor training. No other significant differences in behavior were detected among the three sets of weights, which is expected since the task was similar in all situations.

The fact that the robot did not learn to traverse the narrow corridor can be justified by two properties of the training data. First, since the narrow corridor is much shorter than both other paths, the training data for this situation was very sparse. Second, the training data for the behavior combination relevant to this segment was combined with readings from wider spaces, in which the same behaviors were applicable (such as the zone in which the narrow and medium-width paths meet before leading to the goal).

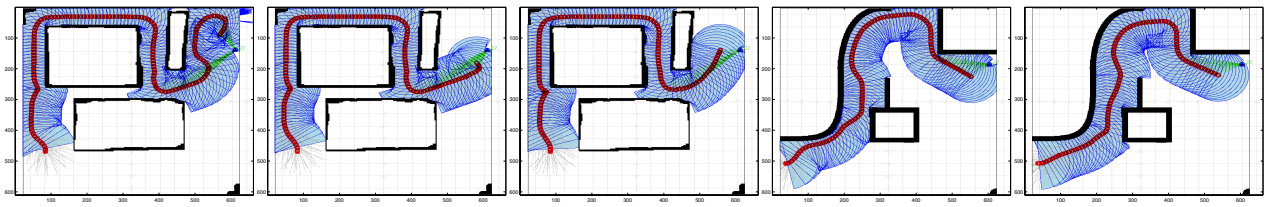

Figure 2. Results from corridor traversal (left three) and wandering/wall following (right two). Red: robot path, blue: laser range readings, green: detection of the goal. The corridor traversal used learned fusion weights from demonstrations in narrow (left), medium (middle), and wide (right) width corridors. The wall following used fusion weights from close (left) and wide (right) wall following demonstrations.

Wander/Wall following experiments. The experiments for the second scenario consisted of two different types of demonstrations, each repeated three times. In the first demonstration, the user drove the robot to the goal by closely following the walls in the environment (upper path in Figure 1). In the second demonstration, the user followed wider spaces and lead the robot to the goal by taking the lower path in Figure 1. Table 2 shows the situations (behavior combinations) that occurred during the two demonstrations and the weights that the robot learned for each of them.

For validation of the results, the robot was equipped with a controller that used the learned weights and was placed in the training environment. The weights obtained from the training experiment generated the expected behavior.

Table 2. Behavior weights learned for the wander/wall follow scenario

ID	Avoid	Sonar	GoalAttract	Wander	WallAttract
10	0	0.0087	0	0.9542	0
19	0.3600	0	0	0.4696	0.7545
23	0.4510	0	0.4415	0.4649	0.5012
27	0.2669	0.3599	0	0.4032	0.7191
31	0.2564	0.3493	0.4010	0.7012	0.0441

ID	Avoid	Sonar	GoalAttract	Wander	WallAttract
2	0	0	0	0.9996	0
6	0	0	0.1836	0.8591	0
10	0	0.1985	0	0.8447	0
19	0.5515	0	0	0.5601	0.5349
23	0.5253	0	0.5577	0.4591	0.1633
27	0.5035	0.4665	0	0.4795	0.3769
31	0.5340	0.3906	0.4019	0.4248	0.0213

Using the weights from the first training experiment, the robot kept close to the walls until reaching the goal, when it stopped. The same behavior was observed in five different evaluation runs. Additional observations of the robot's behavior demonstrate that the learning algorithm has fully captured the information provided by the demonstration. First, when started or moved into open space, the robot showed its preference for wall following by getting close to a wall as soon as one was encountered. Second, if the goal was moved or was away from the wall that the robot was following, the combination of weights that was learned allowed the robot to pull away from the wall and move toward the goal location.

The weights learned from the second experiment generated a robot behavior consistent with the open space demonstration. During the execution of the task, the robot would prefer open spaces to staying close to walls. The robot would go along the walls when they were encountered, but at a much larger distance. In addition, the robot could move away from the walls even in the absence of an external attractor (such as the goal), which was not possible in the previous experiment. The robot found the goal in all five experiments.

The trained controllers are not restricted to a a particular path or execution sequence and are therefore general enough to exhibit meaningful behavior in any evaluation environment. For example, if choosing the controller learned from the first training experiment, the robot will closely follow the walls and will be similarly attracted to goals, irrespective of the environment it is placed in. The learned weights incorporate the specifics of executing the task, which the robot will demonstrate in all circumstances.

Comparison with Hand-coded Controller. A typical hand-coded approach to behavioral fusion employs equal weights (usually unit-valued) for each of the behaviors to be combined [3]. While this approach is suitable for a large range of problems, by giving equal importance to all behaviors at any given time, the method lacks the flexibility needed to deal with different environmental conditions.

To demonstrate this claim, we endowed the robot with a controller that uses unit weights for all behaviors applicable to the robot at any given time. The results we obtained confirmed our assumption: with the uniform weights solution the robot wandered in the environment with no observable purpose. In the proximity of the goal the robot had a tendency to move toward it, but eventually wandered away, never being able to reach it. Due to the fact that all the robot's behaviors have equal weights, the vector provided by the goal attraction behavior was never strong enough to overcome the influence from the other behaviors.

Comparison with Sequential Learning Approach. The sequential approach to learning by demonstration identifies a single behavior that matches the observed actions, typically the one whose prediction most closely resemb[illegible]

teacher's demonstration, thus only one behavior being active at any given time.

The main drawback of this method is that, by choosing only one behavior as a correspondent to the teacher's actions, it ignores information that would be highly relevant for the task. The results that we obtained from the particle filtering approach can be directly transferred to the sequential approach. This is achieved by taking, for each behavior combination, the behavior with the highest weight, since it represents the behavior that was closest to the demonstrated action. Through simple inspection of the resulting controller, it is observed that it would fail to achieve our goal seeking task. For example, whenever the robot finds itself close to a wall (combination 19 or 27 in Table 2 top), it would stop and turn toward the wall, since the wall attract behavior has the highest weight. Similarly, when the robot sees the goal and is close to obstacles from all directions (combination 31 in Table 2 bottom), it will choose to simply avoid the obstacles, since *avoid* is the behavior with the highest weight.

6. SUMMARY

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

References

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

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

[3] 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*, pages 264–271, 1987.

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

[5] 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*, 50(2):174–188, Feb. 2002.

[6] 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*, 1(2), 2004.

[7] 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*, 12(78):701–729, 1998.

[8] 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*, pages 317–323, 2003.

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

[10] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[11] O. C. Jenkins. *Data-driven Derivation of Skills for Autonomous Humanoid Agents*. PhD thesis, The University of Southern California, 2003. Also listed as, Center for Robotics and Embedded Systems Technical Report ¡a href=”http://cres.usc.edu/Research/publications.php”¿CRES-03-014¡/a¿.

[12] O. C. Jenkins and M. J. Matarić. Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion. *International Journal of Humanoid Robotics*, 1(2):237–288, Jun 2004.

[13] Z. Khan, T. R. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 980–986, 2004.

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

[15] 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.

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

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

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

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

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