Learning Robot Soccer from Demonstration: Ball Grasping

Daniel H Grollman and Odest Chadwicke Jenkins
Department of Computer Science
Brown University
Providence, RI 02906
{dang,cjenkins}@cs.brown.edu

***Abstract*— We approach floor-level robotic manipulation from a Learning from Demonstration direction. In this paradigm, a robot learns new tasks in new environments from observations of the task itself. Many current robot learning algorithms require the existence of basic behaviors that can be combined to perform the desired task. However, robots that exist in the world for long timeframes may exhaust this basis set. In particular, a robot may be asked to perform an unknown task for which its built in behaviors may not be appropriate. We demonstrate a learning framework that is capable of learning both low-level motion primitives and high-level tasks built on top of them from interactive demonstration. We apply nonparametric regression within this framework towards learning a complete robot soccer player and successfully teach a robot dog to first walk, and then to seek and grasp a ball.**

I. Introduction

As functional robots become more common, users may have the desire to adapt robot behavior, but may be unwilling or unable to program a robot in a traditional manner. Instead, robot adaptability will rely upon programs designed by the robot's creators. One approach to maximizing robot flexibility is to use learning to deduce from the user what tasks the user wants performed, and how to perform them. In particular, we are interested in how a robot's behaviors will need to change over its entire lifetime (measured in years, not minutes), especially as the user asks it to perform a task never before encountered, **"The Unknown Task"**.

We consider the Sony Aibo robotic dog. First introduced as an entertainment robot, they were quickly adapted for robot soccer. Manipulation of the ball in this setting is made difficult due to physical limitations of the robot as well as uncertainty and rapid changes in the environment. Due to the programming required to overcome these issues, it is only the technical elite (people with years of training in technology and programming) that have been able to participate in these exciting games. Our goal is to enable common users of technology to train robots to perform such complex, team-based tasks (soccer) using teleoperation interfaces familiar to them from video games and compete against the best manually programmed teams.

We cast robot soccer as a floor-level manipulation task and have implemented a learning from demonstration framework with nonparametric regression to learn the soccer skills of walking, trapping, and ball acquisition. These skills form the analogue of grasping in this domain (positioning the effector and forming a closure) and comprise the first half of a viable soccer attacker. Our system successfully learns them from human-mediated demonstration.

II. Related Work

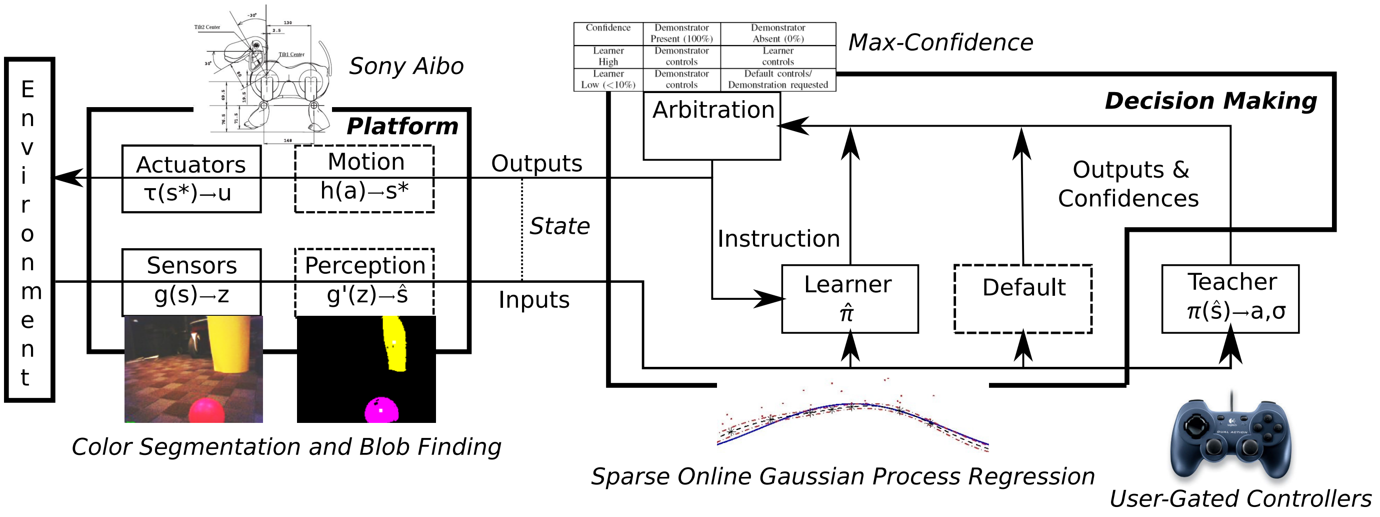
Learning from Demonstration (LfD) is an approach that enables robots to learn tasks simply by observing performances by skilled teachers [1], [2]. In LfD, the robot gathers information about the task in the form of perceptual inputs ($\hat{\mathbf{s}}$) and action outputs ($\mathbf{a}$) and estimates the latent control policy ($\pi : \hat{\mathbf{s}} \rightarrow \mathbf{a}$) of the demonstrator. The estimated policy ($\hat{\pi}$) can then be used to drive the robot's autonomous behavior.

Policies thus learned often require further tuning by the instructor to achieve acceptable task performance. During adjustment, it is important to reveal the learner's policy to the the teacher [3]. We follow the approach of interactive teaching of [4], where teaching and autonomous behaving are interleaved and the robot's behavior serves as feedback.

Interactive teaching is related to **Mixed Initiative Control** (MIC), an area of research that deals with sharing control of one physical robot between several concurrent controllers. Previous work has dealt with static autonomy - autonomous behaviors that are either preprogrammed, or learned once, and do not change over the course of operation [5]. In contrast, we want the user's input to guide the robot's learning process as in [6] so that its autonomous behavior can change over time. We perform arbitration between the multiple controllers (user and autonomy) by means of confidences, similar to [7].

As described in [8], robotic **Lifelong Learning** deals with a single robot learning multiple, predetermined tasks over time. We extend this concept to the unknown task, one which the learner knows nothing about. Approaches to task learning such as [9] depend on the presence of basis behaviors that can be combined to perform new tasks. For long-lived robots it is possible that these low-level abilities may need to change during the robot's lifetime as new tasks are attempted.

Learning in LfD can take many forms, but we treat it as a nonlinear regression from perceived state inputs to action outputs. Locally Weighted Projection Regression (LWPR) is a popular nonlinear regressor that uses the concept of receptive fields to form a piecewise approximation of the mapping and update it incrementally [10]. Gaussian Processes (GP) [11] are a powerful statistic technique, but suffer from memory and



Confidence	Demonstrator Present (100%)	Demonstrator Absent (0%)
Learner High	Demonstrator controls	Learner controls
Learner Low (<10%)	Demonstrator controls	Default controls/ Demonstration requested

Fig. 1. The Dogged Learning framework as applied to our tasks; pictures and italics show our instantiations and modifications. The platform (robot) provides inputs ($\hat{\mathbf{s}}$) in the form of raw motor readings or color-segmented and blobbed vision data. The user-gated demonstrating controller, the learning algorithm (Sparse Online Gaussian Processes) and a self-protective default controller produce outputs ($\mathbf{a}$) and confidence values (σ) in response. The arbitrator selects the output with the maximum confidence to pass back to the platform and the learner. The learner then updates its I-O mapping and the platform converts walking parameters to actuator values and performs the output. We explicitly expose state in the outputs and inputs of the system to enable stateless learning.

computational limitations. We seek to learn the entire decision making function in real-time on a memory-constrained system and thus consider a sparse approximation [12].

III. Methodology

Fig. 1 shows our decision-making learning framework. It has been designed to run on many robot platforms, and be compatible with multiple learners, demonstrators, and arbitrators. We assume that the platform (robot) performs low level perception and generates perceived states ($\hat{\mathbf{s}}$) for decision making. The learned policy of the robot then processes these inputs to produce an appropriate control output ($\mathbf{a}$). A demonstrator, if present, produces an output that can be used to teach the desired task. These outputs are arbitrated in a mixed initiative manner (using confidences) and one is sent to the platform. The platform performs motion control, turning the control outputs into actuator commands. A default controller instantiates "instinctive", or self-protective behavior.

We use this framework with a Sony Aibo robotic dog, shown in Fig. 2. Our unknown task is to perform basic soccer: find, acquire and kick a ball into a goal. To support this task we have filled the perception box with a simple color-segmentation based vision system that is capable of discerning the relevant entities in the world (ball, goal, field, etc.) We denote the set of C detected blobs by $\beta = \{h_c, v_c, s_c\}, c \in 1 : C$, where $\mathbf{h}$ and $\mathbf{v}$ are the blob center locations in image coordinates and $\mathbf{s}$ is the sizes of the blobs, measured in pixels.

The robot itself implements the motion box, which consists of PD servoing to commanded motor angles. These angles themselves are sensed and returned as input data. For our experiments, it is advantageous to separate the motor angles by robot appendage, thus $\Theta = \{\theta_h, \theta_t, \theta_l, \theta_r\}$ are the angles of the motors in the head, tail, left and right legs.

In the remainder of this section we describe our choices for the main components of the framework: The learning algorithm, the teaching interface, and the arbitration matrix.

A. The Learning Box

The learning box observes input-output pairs generated by the teacher and learns a representative decision making mapping which can then be used to perform the demonstrated behavior autonomously. We initially take the desired mapping to be functional, but plan to consider non-functional mappings as well. That is, for each input we currently make the strong assumption that there is only one correct output that centers a unimodal distribution of observed outputs (corrupted by noise, ϵ), although multiple inputs can generate the same output. Our target function,

$$\pi(\hat{\mathbf{s}}) = \mathbf{a} - \epsilon \tag{1}$$

is the demonstrator's mapping of perceived states to actions. We denote the learned approximation to this mapping as $\hat{\pi}$.

Our initial experiments used LWPR to learn this mapping for two tasks: mimicry and simple ball chasing [13]. Experiments with more complex soccer behaviors revealed several issues that prevented successful learning: Model selection, the number and placement of receptive fields in the input space, and storage requirements, the amount of training data that must remain in memory. Limiting necessary storage (and thus the amount of computation) is crucial if we are to ensure real-time interaction between the robot and a user.

In this paper we investigate Sparse Online Gaussian Processes (SOGP) for our learning box. We refer the reader to [11] and [12] for full details, but briefly review it here.

Gaussian Process Regression (GPR) consists of learning a mapping from a set of N inputs $\mathbf{X} = \{\mathbf{x}_i\}_1^N$ to a set of noise-corrupted outputs $\mathbf{Y} = \{\mathbf{y}_i\}_1^N$. In our experiments, inputs are

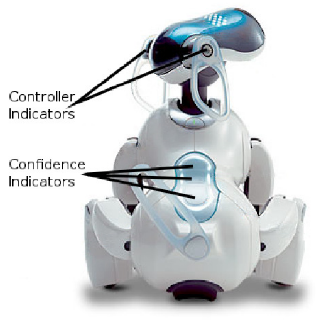


Fig. 2. Our robot. Feedback to the user was given via LEDs on the robot's surface. The robot's ears change color based on which output the arbitrator has selected (red for teacher, blue for student, green for default). The robot's back indicates the confidence value associated with the current output.

perceived states ($\mathbf{x}_i = \hat{\mathbf{s}}_i$) and outputs are actions ($\mathbf{y}_i = \mathbf{a}_i$). We use a radial basis kernel function (q) to compare inputs:

$$q(\mathbf{x}_i, \mathbf{x}_j) = \exp \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_k^2} \tag{2}$$

where the kernel width (σ_k^2) is a hyperparameter, and compute the covariance matrix:

$$C_{ij} = q(\mathbf{x}_i, \mathbf{x}_j) + \delta_{ij}\sigma_\nu^2 \tag{3}$$

where $\delta_{ij} = 1 \texttt{ iff } i = j$. σ_ν^2 is another hyperparameter representing the variance of the observation noise, ϵ.

Prediction of $\hat{\mathbf{y}}$ from a new datapoint $\mathbf{x}'$ is straightforward:

$$k_i = q(\mathbf{x}', \mathbf{x}_i) \tag{4}$$

$$\hat{\mathbf{y}} = \mathbf{k}^\top \mathbf{C}^{-1}\mathbf{Y} \tag{5}$$

with variance:

$$\sigma_{\hat{\mathbf{y}}}^2 = (1 + \sigma_\nu^2) - \mathbf{k}^\top \mathbf{C}^{-1}\mathbf{k} \tag{6}$$

New data pairs ($\mathbf{x}_{N+1}$ and $\mathbf{y}_{N+1}$) can be incorporated into the model by extending $\mathbf{C}$, $\mathbf{X}$ and $\mathbf{Y}$. Efficiency is achieved by storing and extending $\mathbf{C^{-1}}$ directly, through application of the partitioned inverse equations:

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^\top & \mu \end{bmatrix} \tag{7}$$

$$\mu = ((1 + \sigma_\nu^2) - \mathbf{k}^\top \mathbf{C}_N^{-1}\mathbf{k})^{-1} \tag{8}$$

$$\mathbf{m} = -\mu \mathbf{C}_N^{-1}\mathbf{k} \tag{9}$$

$$\mathbf{M} = \mathbf{C}_N^{-1} + \tfrac{1}{\mu}\mathbf{m}\mathbf{m}^\top \tag{10}$$

Prediction and update in this model both grow as N^3, severely limiting the amount of data usable in a real-time scenario.

Ref. [12] introduces a sparse approximation, only storing a subset of the datapoints of size P, leading to a runtime of order NP^2. We use a simplified version of this method to achieve real-time operation. We first include a new datapoint in the usual way using Eqns. 7-10 and then compute the variance bounds around all stored points:

$$\sigma_p^2 = C(p, p) \tag{11}$$

The point corresponding to the lowest variance (which may be the point just added) is selected for removal. The point is removed from the stored data set ($\mathbf{X}$ and $\mathbf{Y}$), and the partitioned inverse equations are run in reverse to remove the corresponding row and column from $\mathbf{C}^{-1}$.

B. The Demonstration Box

We use a hand-coded controller to perform the desired tasks and let a human user modulate its activity. That is, while the *content* of the teacher's output is controlled by a hand-coded program, the *presence* of the instruction is controlled by a human. This enforces the mathematical functionality of the teacher's input-output mapping while still maintaining a human in the loop. By toggling the controller, the user dictates when teaching occurs, and therefore what portions of the task are taught. The user may enable teaching for an arbitrarily short amount of time, to teach the student specific sub-portions of the task, or to correct errant behavior. We call this teaching paradigm interactive teaching via human-gated controllers.

C. The Arbitration Box

We have chosen to use a simple confidence-based arbitration scheme where the most confident output gets passed to the platform. We set our teacher's confidence to 100% and introduce a self-protecting default box with a confidence of 10%. This default box prevents an nonconfident student from causing damage to the robot.

Thus, if active, the teacher's output will always be executed. Otherwise, the student or default controller will be in charge. Table I shows the arbitration matrix, and Fig. 2 shows how feedback is given to the user. By observing the robot the user can easily determine if the student is acting and how confident it is in its actions. If the student's confidence is low, the human may decide to teach it more, or they may let the student continue uneducated.

IV. Results

We focus on the learning of 4-legged league Robocup style behavior from demonstration. Our robocup team competed in the 2006 Robocup US open using a basic attacker tactic that can be logically decomposed into a set of skills as in Fig. 3. We seek to learn these skills, and the entire tactic, from demonstration as a test of our proposed methodology for learning unknown tasks.

We have learned the first half of the tactic, up to and including the Ball-Acquire skill. We first learned the lowest-level primitive skills (Walk and Trap), but due to computational considerations we used the hand-coded walk controller

TABLE I
Our arbitration matrix.

Confidence	Demonstrator Present (100%)	Demonstrator Absent (0%)
Learner High	Demonstrator controls	Learner controls
Learner Low (<10%)	Demonstrator controls	Default controls/ Demonstration requested

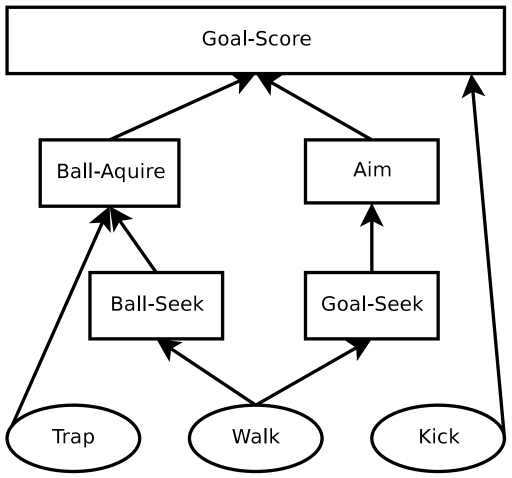


Fig. 3. The desired goal-scoring tactic decomposed into its constituent skills. We seek to learn the lower-level behaviors first and then learn to combine them into a complete system.

in place of the learned behavior during learning of the Ball-Acquire skill.

We use SOGP as our learning box with fixed parameters across all experiments. Our observation noise parameter (σ_ν) is .1 and we use the Radial Basis Function with a width of .1 (σ_k) as our kernel function. Our memory limitations constrain us to keeping only 900 datapoints (roughly 30 seconds of data) in memory in order to achieve real-time performance.

These studies involved one human user (the author) conducting multiple trials with each behavior. Learning success was evaluated by a visual inspection of the robot's behavior and comparison with the stated task goals, i.e. walking, approaching the ball, trapping the ball.

A. Learning to Walk

In our robot soccer setting, the object to be manipulated (the ball) is on the same scale as the robot itself. Thus walking, locomotion, is a vital portion of manipulation. Not only is it necessary to move the entire robot into position to grasp the object but, once grasped, moving the object requires moving the whole robot as well.

We use the University of Pennsylvania walking gait engine as our teaching program and learn to recreate it. A full understanding of the code was not achieved, only enough to wrap it for use in our system.

We enable users to control the walk direction via a control pad. This allows them to test/train the different walks in an much more interactive manner. The user can steer the robot, examining its ability to walk, and then focus training on trouble areas. The inputs to the learning system were thus the sensed motor angles and three control variables taken from the human-operated control pad. Outputs are the new desired motor angles. That is, we define

$$\hat{\mathbf{s}} = \{\Theta, \Omega\}$$

$$\mathbf{a} = \Theta$$

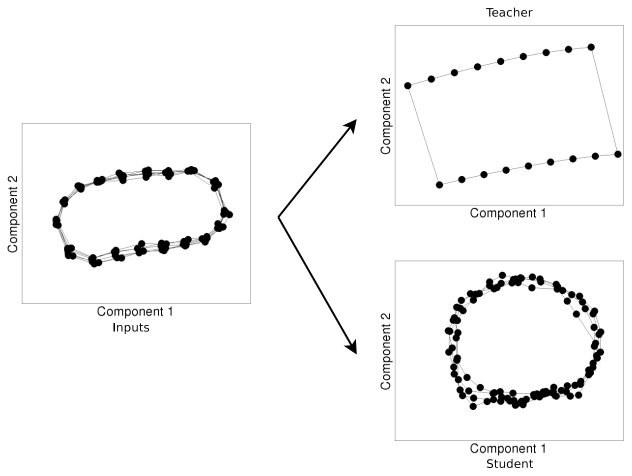


Fig. 4. The walking behavior as demonstrated and learned. Only the forward walking gait is shown for clarity, but our system learned forward/back, side to side, and turning gaits. The teacher (top), which is open loop, outputs the exact same motor angles each walk cycle. Due to effects such as gravity and friction, the motors do not obtain the commanded angles. The learning algorithm (bottom) learns to reproduce the motion as seen. All results are shown to the same scale and projected onto the same first two principal components of the input motor space.

where Ω are the analog inputs from the control pad's two joysticks, corresponding to desired walk directions.

Of interest is that the teacher is partially open-loop. That is, the sensed motor inputs do not affect the outputs. Instead, the controller uses internal state to keep track of where in the gait the robot is and combines it with the control pad inputs to generate the next pose. Our learning algorithm does not have access to this state, and has no internal state of its own, but is able to learn to walk function just from the input-output pairs.

Fig 4 shows the sensed motor values for a few cycles of the walking forward gait (data has been projected onto the first two principal components of the input space). The demonstrated and learned outputs are shown as well. Note the open loop nature of the outputs as discussed above. In contrast, the learned output is smoother and better captures the actual walking behavior that is performed.

B. Learning to Trap

In Aibo soccer, successful teams acquire and manipulate the ball by 'trapping' it under the chin of the robot. This is equivalent to forming a closure as the robot can then turn with the ball to line up a kick and score a goal. A key component of the trap maneuver is to detect when the ball has been successfully acquired.

A stateless control program to perform the trapping motion and detection was written by hand and used to train the learner. Inputs were the 3 motors that defined head pose (neck angle, chin angle, and mouth angle) and outputs were the desired locations of these motors as well as an indicator variable. This ternary indicator showed if the ball was successfully trapped, if the trap was unsuccessful, or if it could not be determined at this time (the sensed motor angle of the mouth is what actually

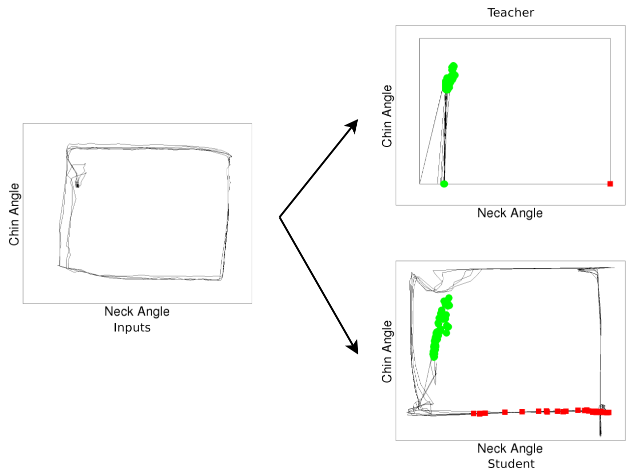


Fig. 5. The trapping behavior, demonstrated and learned. Green circles indicate trap success, red squares, failure. All other points are undecided.

detects trap success).

$$\hat{\mathbf{s}} = \theta_h$$

$$\mathbf{a} = \{\theta_h, r\}; r \in \{\texttt{Trap, NoTrap, Unknown}\}$$

We were able to learn both the trap motion and the indicator variable function from only a few examples of both successful and unsuccessful traps. The demonstrated and learned trap behaviors are shown in Fig 5. Note again how a smoother function than the one demonstrated is learned.

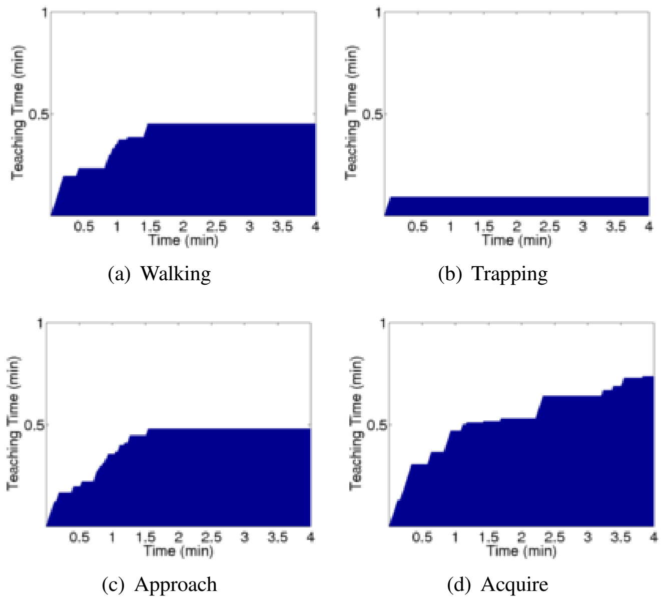
C. Approaching the ball

The approach portion of the goal-score task builds directly upon the walking behavior learned above, simply using the sensed ball location to derive the walking direction. As mentioned before we use color segmentation and blob finding to detect the ball in an image captured from the robot's camera. However, in addition to walking towards the ball, the hand-coded controller brings the robot's head down as it approaches the ball to keep it in sight. Lastly, if the ball is not detected, the robot circles while bobbing it's head up and down to find the ball. Due to computational constraints, we used the hand-coded walking skill, and simply learned walking parameters.

$$\hat{\mathbf{s}} = \{\beta, \theta_h\}$$

$$\mathbf{a} = \{\Omega, \theta_h\}$$

This task, as well as the walking task discussed earlier, made heavy use of the interactive nature of the training paradigm. After an initial successful demonstration (the robot approached and stopped near the ball), the ball was moved and the student was allowed to act autonomously. Due to the high variability of the inputs, the student kept leaving its area of expertise and its confidence fell. This often resulted in improper behavior (failing to slow down or bob the head). The user reactivated the teacher to correct these mistakes. The training profile (total teaching time over session) is show in Fig. 6(c).



(a) Walking (b) Trapping (c) Approach (d) Acquire

Fig. 6. Teaching profiles for the various tasks learned herein. 4 minute sessions are shown, with total teaching time on the Y axis. Note that trapping was learned successfully from one training session.

D. Approach and Acquire

The full approach-and-acquire task is shown in Fig. 7. It is to approach the ball and then transition to the trap behavior. Unfortunately, during the trap motion the robot enters an ambiguous state, where it cannot be determined from the inputs (Θ, β) if the robot is attempting to trap the ball or is still seeking it. To get the teaching program to perform the task we had to introduce an internal state variable.

Attempts to learn with this demonstrator were unsuccessful. We posited that it was due to the internal state of the controller being hidden from the learning algorithm, and our regression technique's inability to deal with that fact due to the functionality assumption. To address this issue we added the state to both the input and output vectors of the learning system ($\Phi \in \texttt{Trapping}, \texttt{Seeking}$). That is, we effectively passed the state out into the world and then read it back in as shown by the dashed line labeled 'State' in Fig. 1. This exposed the state to the learning box and enabled the box to learn based on the state. By doing so we were able to learn the full approach and trap behavior. The inputs and outputs that were used for learning, with exposed state, were

$$\hat{\mathbf{s}} = \{\theta_h, \beta, \Phi\}$$

$$\mathbf{a} = \{\Omega, \theta_h, \Phi\}$$

This task, the most complex of the ones discussed here, took the most time to teach and made the heaviest use of the interactive nature of our training paradigm. Often, while behaving autonomously, the robot would enter a 'stuck' state, where it was unsure of what to do (had low confidence) and thus waffled. Short teaching demonstrations delivered at these

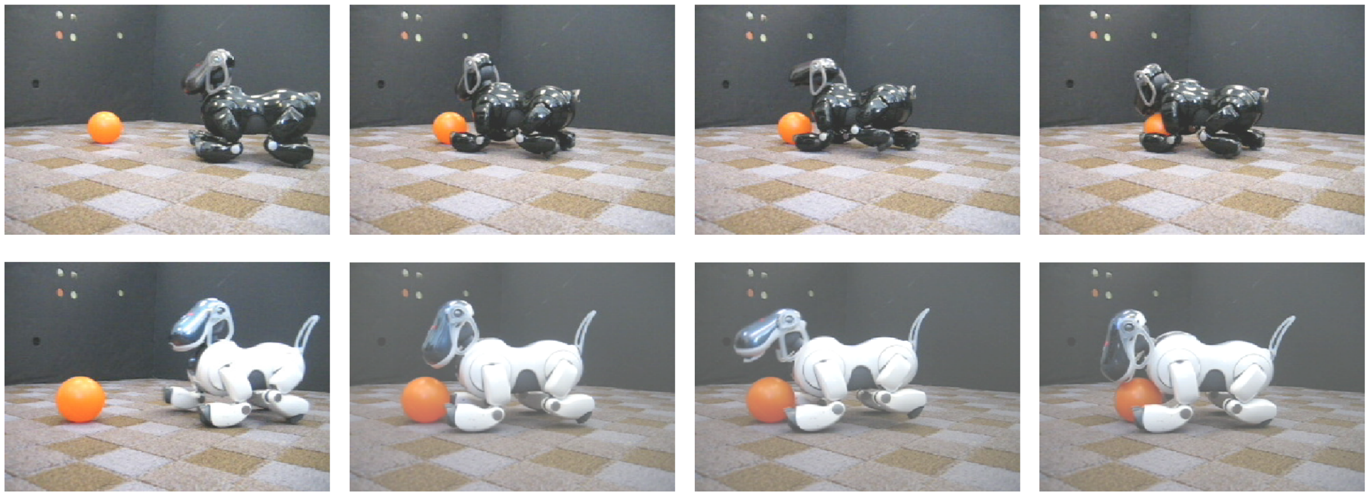


Fig. 7. The approach-and-acquire task as demonstrated (top) and learned (bottom). To perform this task, which is equivalent to floor-level grasping, basic walking is modulated to have the robot approach the ball, and the trap behavior is used to successfully trap it beneath the chin of the robot.

times provided the needed education and allowed the robot to continue behaving as seen in Fig. 6(d). [1]

V. Discussion And Future Work

We have presented results showing that both simple and compound decision making tasks can be learned in an interactive manner from demonstration. These results set the stage for much advanced work. The first step would be to learn the rest of the goal-scoring task. Once done, a direct comparison with hand coded controllers can be assessed as both the teacher and the student attempt to score goals in robot soccer.

We would also like to learn directly from human control instead of from hand-coded controllers. To do this however, we must remove our dependence on explicit state and thus relax the strong assumption of unimodal outputs. Other possibilities for the learning box that incorporate the concept of hidden state such must be investigated. Once this is accomplished, we can evaluate our teaching paradigm with more users to examine the benefits of teaching robots without explicit programming. Development and testing of the user interface will be necessary as well.

We note that one advantage of this paradigm when compared with other learning from demonstration techniques is that learning by parts comes free. That is, portions of the task can be demonstrated and practiced before the entire task is learned. We would like to generalize this and learn when a task has repeated structures that can serve as basis behaviors. Those basis behaviors can then be learned, stored, and used when learning other tasks. Of course, the motor primitives themselves would be amenable to future learning.

VI. Conclusion

We have used a robot learning from demonstration paradigm to successfully learn portions of a floor-level robot manipulation task (soccer) from human-mediated demonstration. Prior knowledge of the desired task was limited, as both the task itself and the low-level behaviors that compose it were learned. The technique used promises to be a flexible approach that will enable robot behavior modification without explicit programming.

[1]Videos of the full approach-and-acquire task being taught and autonomously demonstrated can be found at http://cs.brown.edu/∼dang/lrsfd

References

[1] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *14th International Conference on Machine Learning (ICML)*, D. H. Fisher, Ed., Nashville, TN, 1997, pp. 12–20.

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

[3] A. L. Thomaz and C. Breazeal, "Transparency and socially guided machine learning," in *5th Intl. Conf. on Development and Learning (ICDL)*, 2006.

[4] T. Inamura, M. Inaba, and H. Inoue, "Acquisition of probabilistic behavior decision model based on the interactive teaching method," in *9th Intl. Conf. on Advanced Robotics (ICAR)*, 1999, pp. 523–528.

[5] J. A. Adams, P. Rani, and N. Sarkar, "Mixed initiative interaction and robotic systems," Vanderbilt, Tech. Rep. WS-04-10, 2004.

[6] M. Rosenstein and A. Barto, *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. New York: John Wiley and Sons, Inc., 2004, ch. Supervised actor-critic reinforcement learning, pp. 359–380.

[7] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

[8] S. B. Thrun and T. M. Mitchell, "Lifelong robot learning," University of Bonn, Tech. Rep. IAI-TR-93-7, Jan 1993.

[9] M. Nicolescu and M. J. Matarić, "Natural methods for robot task learning: Instructive demonstration, generalization and practice," in *2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Melbourne, AUSTRALIA, 2003, pp. 241–248.

[10] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[11] D. J. C. Mackay, *Neural Networks and Machine Learning*. Springer-Verlag, 1998, ch. Introduction to Gaussian Processes, pp. 84–92.

[12] L. Csató and M. Opper, "Sparse online gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–669, 2002.

[13] D. H. Grollman and O. C. Jenkins, "Dogged learning for robots," in *2007 IEEE International Conference on Robotics and Automation (ICRA)*, 2007.