Learning Robot Soccer Skills from Demonstration

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

***Abstract*— We seek to enable users to teach personal robots arbitrary tasks so that the robot can better perform as the user desires without explicit programming. Robot learning from demonstration is an approach well-suited to this paradigm, as a robot learns new tasks 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 and learn many tasks over their lifetime may exhaust this basis set and need to move beyond it. In particular, we are interested in a robot that must learn to perform an unknown task for which its built in behaviors may not be appropriate. We demonstrate a learning paradigm that is capable of learning both low-level motion primitives (locomotion and manipulation) 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 acquire a ball.**

I. Introduction

Users of consumer personal robots may have the desire to adapt robot behavior, but may be unwilling or unable to sit down and program a robot in a traditional manner. Instead, users will rely upon whatever functional flexibility has been built into the robot by the creators. One method for maximizing robot flexibility is to use learning to deduce from the user what tasks to perform and how to do them, and thus adapt the robot's behavior to the user's needs. In particular, we are interested in how a robot's behaviors will need to change over its entire lifetime (measured in years, not minutes). This adaptation may be required because the robot's capabilities have changed, either due to hardware failure or upgrades, but especially as the user asks it to perform a task never before encountered, **"The Unknown Task"**.

A concrete example would be the Sony Aibo robotic dog: First introduced as an entertainment robot, they were quickly adapted for the field of robotic soccer. However, only the technical elite (people with years of training in technology and programming) have been able to participate in these exciting games due to the high programming requirement. Our goal is to enable common users of technology to train robots using teleoperation interfaces familiar to them from video games and compete against the best manually programmed teams.

Here, we present work using nonparametric regression to perform robot learning from interactive demonstration for the soccer skills of walking, trapping, and ball acquisition. Our system learns these tasks, which comprise the first half of a viable soccer player, from human-mediated demonstration.

II. Related Work

Learning from Demonstration (LfD) is an approach that enables robots to learn tasks simply by observing the performance by a skilled teacher [1], [2]. In LfD, the robot gathers information about the task in the form of perceptual inputs and action outputs and estimates the latent control policy of the demonstrator. The estimated policy can then be used to drive the robot to behave autonomously.

Demonstration to a robot can take many forms. A robot could extract information from a sensor-laden demonstrator, or be guided through the task and gain information from its own embodiment. Information from multiple sources can be combined to improve learning performance [3]. Likewise, the teacher can be realized in multiple ways and multiple teachers can teach the same robot. In our consumer-based scenario, the robot would learn from the user, but it could also learn from another robot, or a piece of custom designed control software.

Policies learned from demonstration often require further tuning to achieve acceptable task performance. This modification can be accomplished by introducing a 'critiquing' phase, where the robot performs the task whilst receiving feedback from the teacher [4]. Ref. [5] has shown the importance of transparency (revealing what the learner is thinking) to teaching via demonstration. This approach is taken further in [6] where the task as understood by the robot is actually recited back to the teacher. We interleave teaching and critiquing to achieve interactive instruction as in [7].

Mixed Initiative Control (MIC) 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 [8]. Recently, [9] showed a system that used sliding autonomy to enable one user to control three robots by taking control from, or giving it to, a static autonomous controller. In contrast, we want the users input to guide the robot's learning process as in [10] so that its autonomous behavior can change over time to react to and incorporate the user's input.

As described in [11], 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. Information from previous, simpler, tasks can be leveraged to assist in learning [12]. Approaches such as [13] make this assistance explicit by

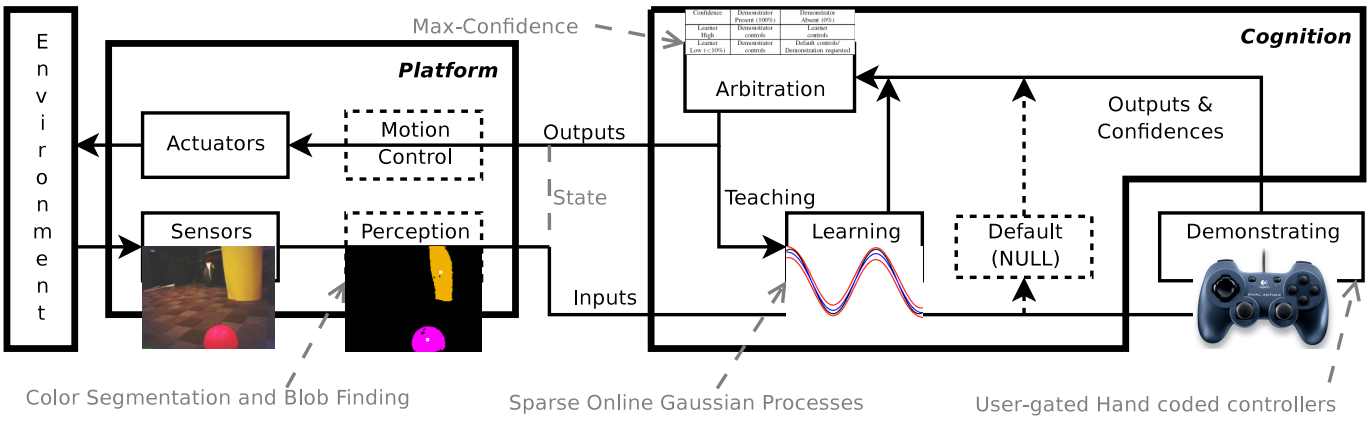


Fig. 1. Our learning framework as applied to our tasks; dotted lines indicate optional portions of the framework and grey, dashed arrows show our instantiations. The platform (robot) provides inputs in the form of raw motor readings or color-segmented and blobbed vision data. These inputs are passed to the user-gated demonstrating controller, the learning algorithm (Sparse Online Gaussian Processes) and a self-protective default controller. Each box generates an output and a confidence value. The arbitrator receives these outputs and confidences and chooses 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.

depending 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 here as a nonlinear regression from perceived state inputs to action outputs. Locally Weighted Projection Regression (LWPR) is a popular nonlinear regressor that instantiates the concept of receptive fields to form a piecewise approximation of the mapping and update it incrementally [14]. Gaussian Processes (GP) [15] are a powerful statistic technique that have been successfully used to learn residual error in autonomous robot controllers [16]. We instead learn the entire control function in real-time on a memory-constrained system and thus consider a sparse approximation [17].

III. Methodology

Shown in Fig. 1, we use a cognitive learning framework which 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 for decision making. The learned decision making system of the robot then processes these inputs to produce an appropriate control output. If a demonstrator is present, it produces a training output that can be used to teach the desired task. These outputs are arbitrated in a mixed initiative manner (using confidences) to determine what is sent to the platform. The platform then performs motion control, turning the control outputs into actuator commands. A default controller can also be present to instantiate "instinctive", or self-protective behavior.

We have implemented this framework for use with a Sony Aibo robotic dog shown in Fig. 2. Unknown to the system, the desired task is to perform basic robot soccer: find, acquire and shoot 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.) The robot itself implements the motor control box, which consists of PD servoing to commanded motor angles. These angles themselves are sensed and returned to complete the set of input data.

In the remainder of this section we describe our choices for the frameworks's main components: The learning algorithm, the teaching interface, and the arbitration between them.

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. That is, for each input we assume that there is only one correct output that centers a unimodal distribution of observed outputs (corrupted by noise), although multiple inputs can generate the same output.

Initial experiments in [18] used LWPR to learn two tasks: mimicry and simple ball chasing. Each of these tasks was conceptually simple to perform and took less than 30 seconds to demonstrate and learn. As we began experimenting with learning robot soccer from demonstration we encountered several issues that prevented successful learning: Model selection, the placement of receptive fields in the input space, and storage requirements, the amount of training data that must remain in memory. While LWPR is nonparametric and does not require all previous datapoints to be stored, we found it difficult to enforce a maximum storage size for the learned mapping. Limiting necessary storage 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) in our learning box. We refer the reader to [15] and [17] for full details, but briefly review it here.

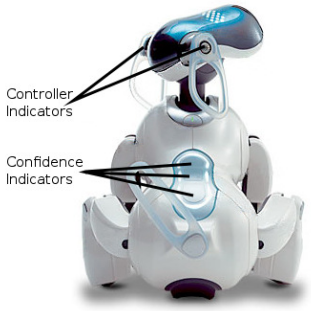


Fig. 2. The robot used in our experiments. Feedback to the user was given via LEDs on the robot's surface. The robot's ears change color based on which box's output the arbitrator is currently running on the robot (red for teacher, blue for student, green for default). The robot's back indicates the confidence value associated with the current output.

1) SOGP: 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 targets $\mathbf{T} = \{\mathbf{t}_i\}_1^N$. We use a radial basis kernel function (q) to compare pairs of inputs:

$$q(\mathbf{x}_i, \mathbf{x}_j) = \exp \frac{||\mathbf{x}_i = \mathbf{x}_j||^2}{2\sigma_k^2} \tag{1}$$

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

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

$\delta_{ij} = 1$ iff $i = j$ and σ_ν^2 is another hyperparameter representing the observation noise.

Prediction of $\hat{\mathbf{t}}$ from a new datapoint $\mathbf{x}'$ is straightforward:

$$k_i = q(\mathbf{x}', \mathbf{x}_i) \tag{3}$$

$$\hat{\mathbf{t}} = \mathbf{k}^\top \mathbf{C}^{-1}\mathbf{T} \tag{4}$$

with variance:

$$\sigma_{\hat{\mathbf{t}}}^2 = (1 + \sigma_\nu^2) - \mathbf{k}^\top \mathbf{C}^{-1}\mathbf{k} \tag{5}$$

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

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^\top & \mu \end{bmatrix} \tag{6}$$

$$\mu = ((1 + \sigma_\nu^2) - \mathbf{k}^\top \mathbf{C}_N^{-1}\mathbf{k})^{-1} \tag{7}$$

$$\mathbf{m} = -\mu \mathbf{C}_N^{-1}\mathbf{k} \tag{8}$$

$$\mathbf{M} = \mathbf{C}_N^{-1} + \tfrac{1}{\mu}\mathbf{m}\mathbf{m}^\top \tag{9}$$

Prediction and update in this model both grow as N^3, severely limiting the size of the data set that can be used in a real-time scenario. Speedups can be obtained by using a sparse approximation to GPR and only storing a subset of the datapoints of size P, leading to a runtime of order NP^2.

Ref. [17] introduces such an approximation, and we use a simplified version to achieve real-time operation. We first include a new datapoint in the usual way using Eqns. 6-9 and then compute the variance bounds around all stored points:

$$\sigma_p^2 = C(p, p) \tag{10}$$

The point corresponding to the lowest variance (which may be the point just added) is selected for removal. Deletion consists of removing the point from the stored data set ($\mathbf{X}$ and $\mathbf{T}$), and running the partitioned inverse equations 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 input-output mapping 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, but the *presence* of the instruction is controlled by a human, meaning that the target control policy can only be queried if the human allows it. This human-gated control policy 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.

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 by always returning a null output. That is, if the learner is less than 10% confident, the robot does nothing and awaits instruction or a change in the environment.

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. Experiments

We focus on the learning of 4-legged Robocup style behavior from demonstration. Our University's team successfully competed in this competition using a basic attacker behavior

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

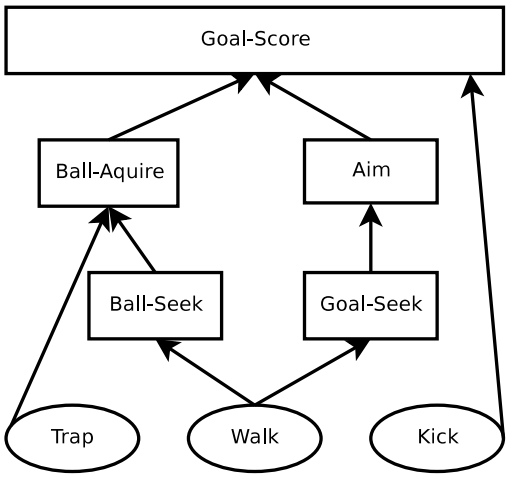


Fig. 3. The desired goal-scoring behavior decomposed into its constituent behaviors. We seek to learn the lower-level behaviors first and then learn to combine them into a complete system.

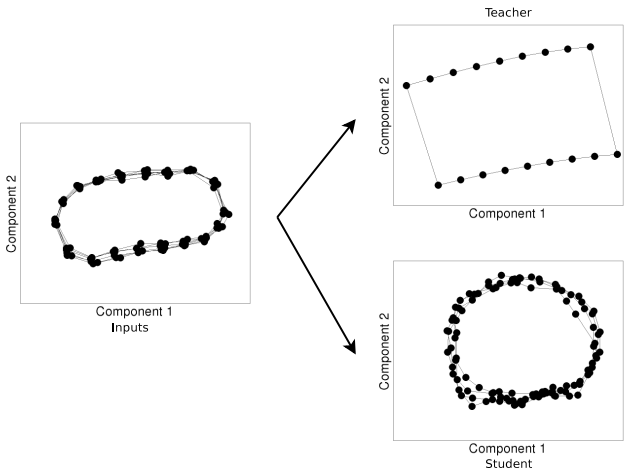


Fig. 4. The walking behavior as demonstrated and learned. Only the forward walking gait is shown for clarity, but our system learned forward/back, side to side, and turning gaits. The teacher (top), which is open loop, outputs the exact same motor angles each walk cycle. Due to effects such as gravity 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.

that can be logically decomposed as in Fig. 3. We seek to learn this behavior as a demonstration of our proposed methodology for learning unknown tasks.

As of publication, we have learned the first half of the full task, up to and including the acquire-ball behavior. We first learned the lowest-level primitive behaviors (walk and trap) and then the compound behavior Ball-Acquire. Due to computational considerations, we replace the learned walk behavior with its hand-coded controller during high-level learning.

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

The University of Pennsylvania has developed a walking gait for the Aibo that is widely used in the Robocup 4-legged league. We use their walk engine as the teacher in our framework and learn to recreate it. A full understanding of the code was not achieved, only enough to wrap it for use in our system.

Instead of following a pre-programmed trajectory for walking, we enabled the user to control the walk direction via a control pad. This allowed 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 from the cognitive learning are are the new desired motor angles.

Of interest is that the teacher in this experiment is partially open-loop. That is, the sensed motor inputs do not affect the outputs of the teacher, 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. 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. As the region under the robot's chin is hidden from its camera, other sensors must be employed.

A stateless control program to perform just the trapping motion 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 (the sensed motor angle of the mouth is what actually detects

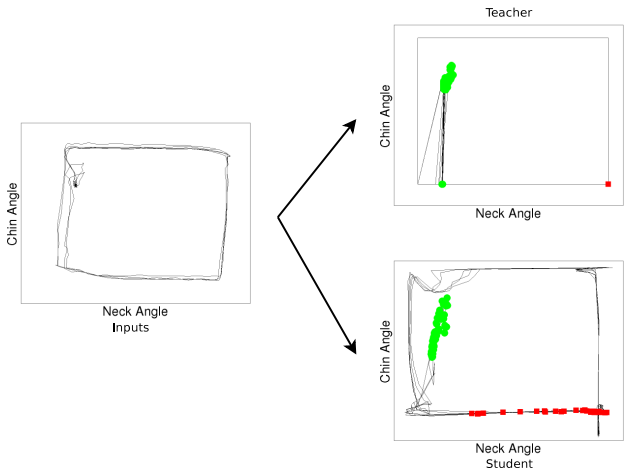
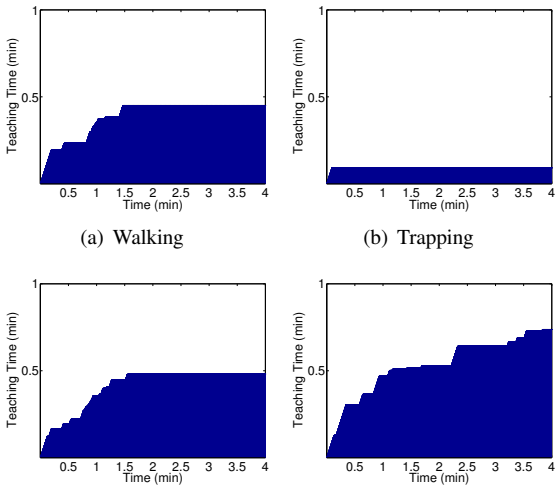


Fig. 5. The trapping behavior as demonstrated and learned. Trap success is indicated by a green circle, and failure by a red square. All other points are undecided.



(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 (confidence never fell below 10%) from one training session, and thus the teacher was never reactivated.

trap success). 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. Our system was able to learn both the trap motion and the indicator variable function from only a few examples of both successful and unsuccessful traps.

Fig 5 shows the demonstrated and learned trap behaviors. Note again how a smoother function than the one demonstrated is learned.

C. Approaching the ball

The approach portion of the goal-score can be built directly upon the walking behavior learned above, it simply uses the sensed ball location to derive the parameters for walking. As mentioned before we use simple 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.

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).

D. Approach and Acquire

The full approach-and-acquire's goal is to reach the state shown in Fig. 7. That is, approach the ball and then transition from to the trap behavior. Unfortunately, during the trap motion the robot enters an ambiguous state, where it cannot be determined from the inputs 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. To address this issue we added the state to both the input and output vectors of the cognitive system. That is, we effectively passed the state out into the world and then read it back in as shown by the grey line labeled 'State' in Fig. 1. This exposed the state to the learning box and enabled learning based on the state. By doing so we were able to learn the full approach and trap behavior.

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 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 a full motor system (both simple and compound tasks) can be learned in an interactive manner from demonstration using the same framework. This sets 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

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

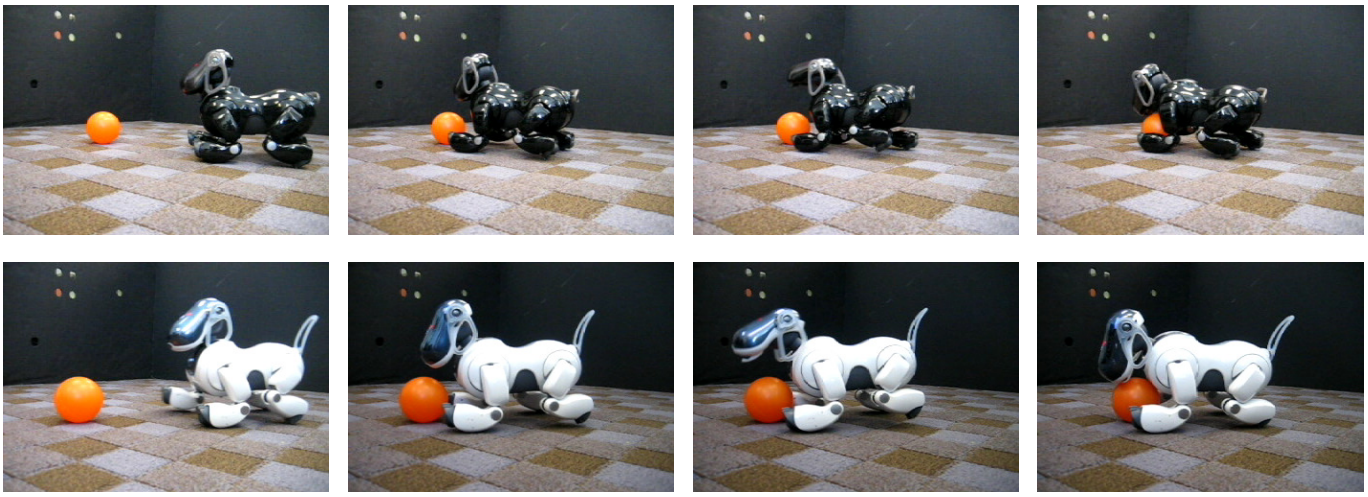


Fig. 7. The approach-and-acquire task as demonstrated (top) and learned (bottom). Basic walking is modulated to have the robot approach the ball, and the trap behavior is used to successfully trap it beneath the chin of the robot.

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. 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.

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 basis behaviors themselves would be amenable to future learning.

VI. Conclusion

We have used a robot learning from demonstration paradigm (Dogged Learning) to successfully learn portions of a robot soccer task 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.

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] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *2nd Conf. on Human-Robot Interaction (HRI)*, Arlington, Virginia, 2007, pp. 255–262.

[4] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *2nd Conf. on Human-Robot Interaction (HRI)*, Arlington, Virginia, 2007, pp. 57–64.

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

[6] P. E. Rybski, K. Yoon, J. Stolarz, and M. Veloso, "Interactive robot task training through dialog and demonstraion," in *2nd Conf. on Human-Robot Interaction (HRI)*, Arlington, Virginia, 2007, pp. 255–262.

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

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

[9] F. W. Heger and S. Singh, "Sliding autonomy for complex coordinated multi-robot tasks: Analysis and experiments," in *Robotics: Science and Systems II (R:SS)*, Philidelphia, PA, 2006.

[10] 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.

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

[12] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "Teaching robots by moulding behavior and scaffolding the environment." in *1st Conf. on Human-Robot Interaction (HRI)*, 2006, pp. 142–150.

[13] 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.

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

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

[16] J. Ko, D. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2007.

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

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