In *IEEE International Conference on Machine Learning Applications* (ICMLA-11), Honolulu, Hawaii, USA, December 18-21 2011.

Augmented Reinforcement Learning for Interaction with Non-expert Humans in Agent Domains

Mohan Sridharan
Stochastic Estimation and Autonomous Robotics Lab
Department of Computer Science
Texas Tech University
Lubbock, TX 79409
mohan.sridharan@ttu.edu

***Abstract*—In application domains characterized by dynamic changes and non-deterministic action outcomes, it is frequently difficult for agents or robots to operate without any human supervision. Although human feedback can help an agent learn a rich representation of the task and domain, humans may not have the expertise or time to provide elaborate and accurate feedback in complex domains. Widespread deployment of intelligent agents hence requires that the agents operate autonomously using sensory inputs and limited high-level feedback from non-expert human participants. Towards this objective, this paper describes an augmented reinforcement learning framework that combines bootstrap learning and reinforcement learning principles. In the absence of human feedback, the agent learns by interacting with the environment. When high-level human feedback is available, the agent robustly merges it with environmental feedback by incrementally revising the relative contributions of the feedback mechanisms to the action choice policy. The framework is evaluated in two simulated domains: *Tetris* and *Keepaway soccer*.**

I. INTRODUCTION

Intelligent agents or robots interacting with humans in dynamic domains need the ability to operate reliably, efficiently and autonomously [1], [2]. Existing approaches to human-computer or human-robot interaction (HCI/HRI) predominantly focus on enabling the agent to operate autonomously based on sensory inputs [3], [4], or to learn from extensive manual training and domain knowledge [5], [6], [7], [8], [9]. In dynamic domains characterized by partial observability and non-determinism, it is typically difficult for an agent to operate without any human input [10], [11]. On the other hand, although human feedback can help an agent learn a rich representation of the task and domain, humans frequently do not possess the expertise or time to provide elaborate, accurate and real-time feedback in complex domains.

Widespread deployment of learning agents requires that these agents be accessible to non-expert users who can provide limited high-level feedback in response to the agent's observed performance, e.g., positive/negative reinforcement of the agent's actions or a selection from multiple options posed by the agent. Recent research has focused on enabling a robot or an agent to acquire human feedback when needed (or available) and merge it with the information extracted from sensory cues. However, these methods do not model the unreliability of human inputs and require elaborate knowledge of the domain, limiting their use to simple simulated domains or specific robot tasks [12], [13], [14]. This paper presents an augmented reinforcement learning (ARL) framework that enables an agent to merge limited and unreliable high-level human feedback with the reinforcement obtained by interacting with the environment. The ARL framework uses bootstrap learning to enable the agent to continuously and incrementally revise the relative contributions of environmental feedback and human feedback to the agent's action choices. The proposed approach is evaluated in two simulated domains: (a) *Tetris*, which consists of a single agent; and (b) *Keepaway soccer*, which consists of multiagent teams.

The remainder of the paper is organized as follows. Section II describes related work and Section III describes the proposed scheme and test domains. Experimental results are presented in Section IV, followed by conclusions in Section V.

II. RELATED WORK

Sophisticated approaches have been developed for key human-computer and human-robot interaction (HCI and HRI) challenges such as autonomous operation, engagement, safety, acceptance and interaction protocol design [1], [2]. Many algorithms have been developed to enable autonomous operation in HCI/HRI, the focus of this paper, using a variety of sensory inputs (e.g., visual, verbal and range data) to model social and environmental cues [15]. Considerable work has been done on using embodied relational agents and virtual agents in applications such as health care [16]. However, existing methods typically require a significant amount of domain knowledge, limiting their use to specific applications.

Significant research has also been performed on enabling a robot or a simulated agent to learn from demonstrations provided by a human observer [6], [7], [17], [18]. Many of these methods focus on building sophisticated mathematical models using recent research findings in a wide range of related fields such as control theory, biology and psychology. Some approaches have been based on theories of social interactions among humans and an understanding of the human learning process. A key constraint of these schemes is that the associated feedback and information can only be provided by human participants who possess substantial knowledge of the domain and the agent's capabilities.

Researchers are increasingly focusing on using limited high-level human feedback in robot/agent domains based on need and availability. For instance, Rosenthal et al. [14] developed a *CoBot* that associates each action with probability functions of success and failure, and seeks human help (on failure) to localize and navigate to desired locations. Knox and Stone [13] developed the *TAMER* (Training an Agent Manually vis Evaluative Reinforcement) framework to enable a human to train a learning agent, and used different linear functions to combine human and environmental feedback and maximize a reward function in simulated domains. The work described in this paper is also based on reinforcement learning and a scheme to combine human and environmental feedback. The key difference is that the two feedback mechanisms bootstrap off of each other to continuously revise their relative contributions to the agent's action choice policy, thereby making best use of all the available information.

III. Problem Formulation

This section describes the framework that combines bootstrap learning with reinforcement learning, followed by a description of the specific test domains.

A. The RL Framework and Bootstrap Learning

Reinforcement learning (RL) is a computational goal-oriented approach, where an agent repeatedly performs actions on the environment and receives a state estimate and a reward signal [19]. It is common to model an RL task as a Markov decision process (MDP). In this paper, the standard formulation is augmented to include the human feedback signal, resulting in the tuple $\langle S, A, T, R, H \rangle$:

- S is the set of states.
- A is the set of actions.
- $T : S \times A \times S' \rightarrow [0,1]$, is the state transition function.
- $R : S \times A \rightarrow \Re$ is the environmental reward function.
- H is the human reward signal.

At each step, the agent uses a policy to probabilistically select an action $a \in A$ in state $s \in S$:

$$\pi : S \times A \rightarrow [0,1] \tag{1}$$

The goal is to compute the policy that maximizes the expected future reward over a planning horizon. One of many different schemes such as policy iteration, value iteration and policy gradient algorithms can be used to compute this policy. The key difference with respect to the standard MDP formulation is the inclusion of high-level human feedback that (like environmental feedback) can be unreliable. In addition, though the environmental feedback is obtained instantaneously for a given state and action, the human feedback can be a complex function of current, past or future states and actions.

As shown in Figure 1, a bootstrap learning scheme models the action choice policy as a function of the feedback signals:

$$a = \underset{a \in A}{\operatorname{argmax}} f(R, H) \tag{2}$$

where R is the environmental feedback, H is the human feedback and a is the action choice that maximizes the function

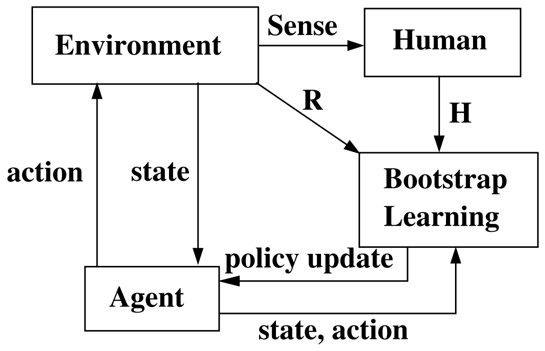

Fig. 1: Augmented RL framework with bootstrap learning.

of R and H. In the experimental domains described below, the following functions were evaluated:

$$a = \underset{a \in A}{\operatorname{argmax}} \{w_r \cdot R + w_h \cdot H\} \tag{3}$$
$$a = \underset{a \in A}{\operatorname{argmax}} \{w_r \cdot R(1 + H^{w_h})\}$$

where w_h is the "weight" assigned to the human feedback and w_r is the weight assigned to the environmental feedback. Since the weights represent relative importance, we can set $w_r = 1$ and use w_h as the relative importance of human feedback. A linear function was considered because a similar function resulted in best performance in the *Mountain Car* domain when the weight assigned to human input was randomly annealed at the end of each iteration [13]. The *novelty* of our approach is that the two feedback mechanisms bootstrap off of each other. The weights are continuously revised based on the relative ability of the feedback signals to maximize global performance measures (e.g., time to complete task), resulting in improved performance in more complex domains (Section IV). The other combination scheme, called the *exponential* scheme because the weight is used as an exponent, was used to investigate a stronger correlation between the feedback signals—it provided best results among a set of similar functions. The bootstrap learning scheme to update the weights proceeds as follows:

- In the absence of human feedback, the agent generates different policies by varying the parameters of the underlying RL algorithms (e.g., policy gradient) and evaluates the policies using a suitable "performance measure"—see Sections III-B, III-C for examples.
- At any given time, the agent keeps track of the top N policies: $\pi_i, i \in [1,N]$, sorted in decreasing order of value of the performance measures: $pm_i, i \in [1,N]$. Actions are chosen based on one of these policies (with $w_r = 1, H = 0$ in Equation 3), where the probability of selecting a policy is proportional to the value of its performance measure relative to those of other policies.
- When human input is provided in the form of positive/negative reinforcements, the agent maintains a separate policy based on these inputs. The agent also computes the degree of match between the action chosen by the current environmental feedback-based policy and action that would be chosen based on the human feedback-based policy. The degree of match can, for instance, be a count of the number of times the two policies result in the same action choice.

- If $m_i, i \in [1,N]$ represent the degree of match between the human feedback-based policy and the best environmental feedback-based policies, the weight associated with the human feedback can be estimated as:

$$w_h = \frac{\sum_i pm_i \times m_i}{\sum_i pm_i} \tag{4}$$

where a high value represents a high degree of belief associated with the human feedback.
- A similar scheme can be used to weight the environmental feedback with respect to one or more sources of human feedback if substantial human feedback is available. In addition, it is possible to update the weights incrementally across different episodes. For instance, the weight w_h can be updated after episode k:

$$w_h^k = \frac{pm^{k-1} w_h^{k-1} + pm^k m^k}{pm^{k-1} + pm^k} \tag{5}$$

based on the performance measure in this and the immediately previous episode $(k-1)$.

The core component is the continuous and online update of the action policy, where the agent alternately assumes the policy (or policies) based on each feedback mechanism to be ground truth, in order to update the weight of the policy based on the other feedback mechanism. Since action choices are based on the combined policy (Equation 3), the agent quickly adapts to different humans, unreliability of environmental feedback, and dynamic changes, e.g., the human observer gets tired or bored. Specific instances of this learning scheme are described below for the two simulated domains.

B. Tetris Domain

Tetris is a game played on a $w \times h$ grid in which "tetrominoes" (i.e., shapes) of four blocks fall one at a time from the top of the grid, stacking up on the grid's base or any blocks below. If the blocks fall such that a row is completely filled with blocks, then that row is cleared. All the blocks in that row disappear and all the blocks in higher rows shift down by a row. When the blocks stack up to the top of the grid, the game ends. The goal of a Tetris player is to maneuver the falling blocks to clear as many lines as possible and maximize episode duration. The *performance measure* in this domain is hence the number of lines cleared per game (i.e., per episode). A screenshot of the domain is shown in Figure 2.

One challenge in this domain is the size of the state space—the 20×10 board shown in Figure 2 has a state space of $\approx 2^{200}$. The action space consists of four actions: *move left, move right, rotate clockwise, drop*, and the feedback signal from the environment (or human) is assumed to be instantaneous. Knox and Stone [13] developed a much smaller set of 21 features for this domain, e.g., column heights, maximum height of columns, number of holes and difference in heights of adjacent columns. An analysis of the significance of the feature set shows that it is in fact possible to obtain similar performance with just 12 features: maximum and average height of columns, number of holes and difference in column heights. However, for ease of comparison, we use the set of 21 features in our experiments. Two different RL algorithms were used: the policy gradient (PG) algorithm implemented in the libPG library [20] and the cross-entropy (CE) method [21] as used in [13]. PG methods parametrize the policy function and use gradient descent to converge on a stochastic policy that optimizes the long-term reward—they are more robust to changes in policy parameters. The CE method learns weights for the feature vectors using sampling techniques in order to maximize the reward—it has been shown to outperform many RL methods in the Tetris domain.

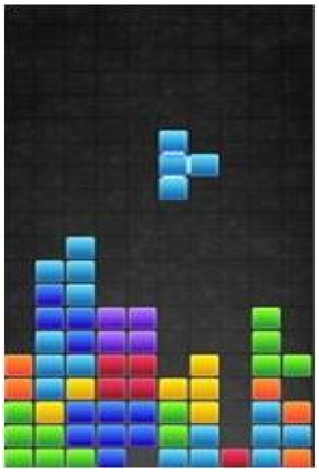
Fig. 2: The Tetris application domain

C. Keepaway Soccer Domain

Keepaway is a subtask of robot soccer involving a small number of players. One team, the keepers, tries to maintain possession of a ball within a limited region, while another team, the takers, tries to gain possession. Whenever the takers take possession or the ball leaves the region, the episode ends and the players are reset for another episode with the keepers being given possession of the ball. This domain is implemented within the RoboCup soccer simulator [22]. Parameters of the task include the size of the region, the number of keepers and the number of takers. Figure 3 shows a screen shot of an episode with 3 keepers and 2 takers (called $3vs.2$ or $3v2$ for short) playing in a $20m \times 20m$ region.

Keepaway is a challenging domain because the state space is too large to explore exhaustively, each agent has partial state information, and all agents in the team have to learn simultaneously. Since it is based on the RoboCup simulator, agents receive (noisy) visual perceptions every 150msec indicating the relative distance and angle to visible objects. In each episode, agents choose from high-level macro actions based on available skills, e.g., *HoldBall(), PassBall(k), GetOpen(), GoToBall()* and *BlockPass(k)*, instead of executing parametrized primitive actions, e.g., *Turn(angle)* and *Dash(power)*. The focus is on enabling the keepers to retain possession of the ball for as long as possible—the episode length is hence used as the *performance measure*. As in [22], the state space for keepers is discretized to 13 variables that capture the distances and angles

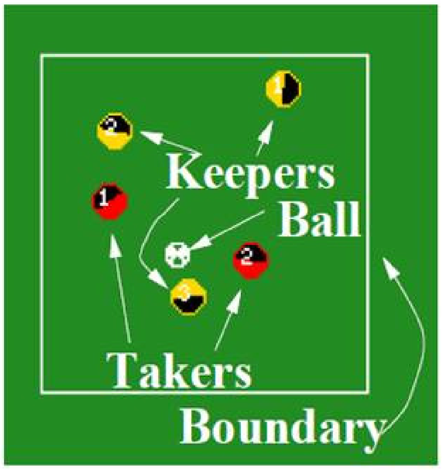

Fig. 3: Keepaway (3v2) soccer domain.

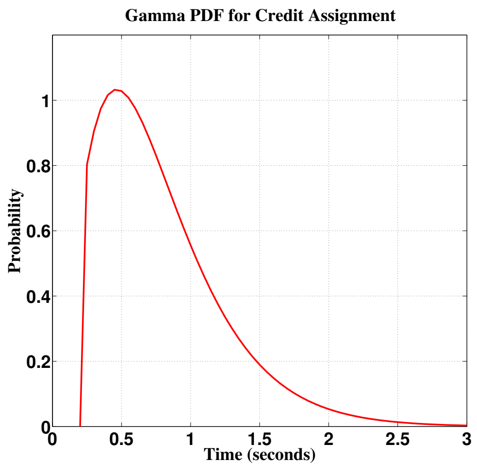

Fig. 4: Probability density function $p(x)$ for credit assignment estimated as $gamma(2.5, 0.3)$. The x-axis denotes time before reinforcement (i.e., in the past).

between the keepers, takers and the center of the region. The takers follow a policy of moving to the ball. The semi-Markov decision process (SMDP) version of $Sarsa(\lambda)$ is used as the RL algorithm to modify the behavior of the keepers. Despite the reduction in state and action space, learning by exploration is still a challenge. Tile coding is hence used to make learning feasible, and the associated parameters are assigned values based on those reported in [22].

Reinforcement signals in the keepaway domain are based on the performance of the team of keepers. Though environmental feedback can be encoded to occur instantaneously, the domain changes too quickly for a human to provide feedback for a specific state and action, i.e., human feedback is not instantaneous and is likely to be a function of a set of prior states and actions. Based on prior work [23], [24], the credit assignment of human feedback is modeled as a gamma distribution, as shown in Figure 4. The parameters of this function were estimated experimentally by conducting a study of the reaction times of a set of human participants in standard cognitive experiments. The resultant distribution is similar to that reported in [24]. If $p(x)$ is the gamma distribution-based probability density function (PDF), the credit assigned to a time interval as a result of a unit reinforcement is computed by integrating the PDF over the interval. In the experiments below, for a human feedback at time t, the mean of the gamma PDF is located at $\approx (t - 0.5)$ to assign credit to a set of states and actions. The PDF indicates that human input is delayed and that the credit drops off exponentially as prior time steps are considered.

IV. Experimental Setup and Results

This section reports the results of experiments performed in the Tetris and Keepaway domains. Two hypotheses were evaluated: (a) combining human and environmental feedbacks results in better performance than with the individual feedback mechanisms; and (b) using the bootstrap learning scheme significantly improves the performance obtained with the underlying reinforcement learning algorithms. In the remainder of this paper, *ARL* refers to the use of bootstrap learning in the modified RL framework. All results are statistically significant (at 99% level) unless otherwise stated.

Human Participants: The experiments involved four non-expert human participants who had a high-level description of the test domains, available action choices and the performance measures to be maximized by the agent(s). The participants had no knowledge of the states or the underlying algorithms, and very little knowledge of the domains prior to these experiments. Human feedback was in the form of positive or negative reinforcement of the agent's actions.

Experiments were conducted in the Tetris domain using cross-entropy as the underlying RL algorithm. The ARL approach and the linear combination function of Equation 3 were used to merge human feedback with the environmental feedback. Performance was compared against the scheme that combines the feedback signals by annealing the weight assigned to human feedback at the end of each episode—this scheme provided the best performance in the simple *Mountain Car* domain [13]. Figure 5 shows experimental results, with each data point obtained by averaging the results over a set of 20 trials. Human input was provided at infrequent intervals—no more than 5 times in any episode and on average 2 times per episode. The participants had breaks between trials.

The results show that the ARL approach significantly improves performance (i.e., increases the number of lines cleared) in comparison to the default CE approach and the combination scheme that anneals the weight assigned to human feedback between episodes [13]. The performance improvement is due to the ARL approach's ability to adapt to the unreliability of feedback signals, thereby exploiting their complementary properties. The performance of the ARL approach is also much better than that obtained with just the human feedback—these results are not included in Figure 5 because it is infeasible to provide human feedback over different states and actions for a large number of episodes.

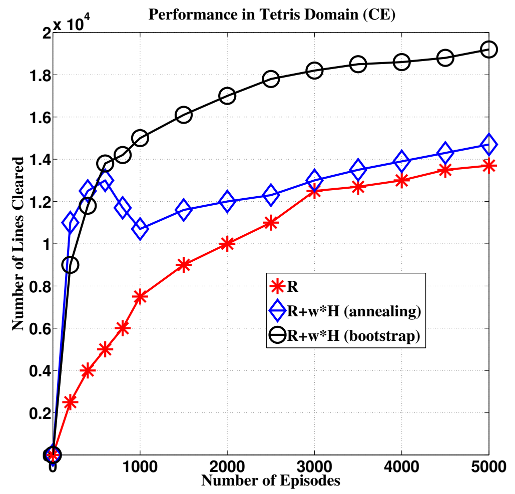

Fig. 5: Performance in the Tetris domain using cross entropy as the default RL algorithm. The ARL approach performs significantly better than CE and the scheme that anneals the weight factor for human feedback.

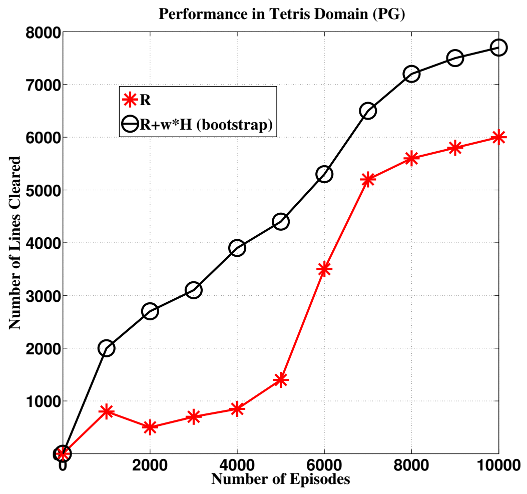

Fig. 6: Performance in the Tetris domain using policy gradient as the default algorithm. The ARL approach significantly improves the performance of the PG algorithm.

Next the performance of the ARL approach was evaluated using policy gradient (PG) as the underlying RL algorithm. The results of these experiments are shown in Figure 6. As expected, the default PG algorithm does not result in as many lines being cleared as with the CE algorithm. However, the ARL approach for merging the feedback mechanisms still performs significantly better, i.e., clears a much larger number of lines than the default PG algorithm.

Finally, experiments were conducted in the Keepaway soccer domain, where the performance of the keepers was measured in terms of their ability to retain possession of the ball for as long as possible. The underlying RL algorithm was the SMDP version of $Sarsa(\lambda)$. The ARL approach was used to incrementally determine the best combination of human feedback and environmental feedback to be used to determine the action choice policy. Here, both the linear combination function and the exponential combination function (Equation 3) were evaluated. As described in Section III-C, it is not possible to provide instantaneous human feedback in this domain. In order to measure the suitability of the gamma PDF (shown in Figure 4) for credit assignment when human feedback is provided, performance was measured with and without the use of this PDF. Figure 7 summarizes the results, with each data point (as in the Tetris domain) representing the average over 20 trials. Given that episode times can vary (as seen in Figure 7) the human participants provided feedback infrequently—no more than two times in any episode. The human participants also provided (intentionally) incorrect feedback—on average, one in every ten inputs is incorrect.

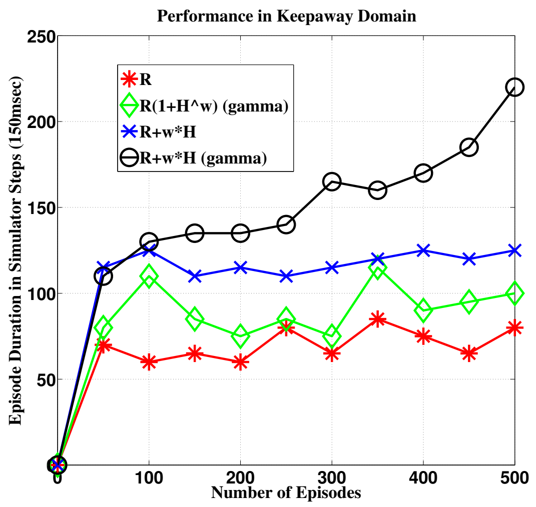

Fig. 7: Performance in the Keepaway soccer domain using policy gradient. The ARL approach performs better than the default $Sarsa(\lambda)$ algorithm. Using the learned gamma distribution for credit assignment significantly boosts performance.

The plots in Figure 7 indicate that all schemes that combine the feedback mechanisms using the ARL approach perform better than the default RL algorithm without any human feedback, despite the unreliability in the human feedback. When a human participant provides incorrect feedback, bootstrap learning quickly revises the weights appropriately. The performance is also better than using just the human feedback, which is not shown in Figure 7 because it is difficult to provide human inputs over a large number of episodes. The weighted linear combination of the two feedback mechanisms results in significantly better performance than the default RL algorithm. The best performance is achieved when the gamma PDF is used for credit assignment along with the linear combination function. The exponential combination function (even with the gamma PDF) does not improve performance substantially, which indicates that this function does not reflect the true relationship between the feedback signals. The experimental results in the two test domains indicate that using the gamma PDF-based credit assignment and the bootstrap learning scheme within the augmented reinforcement learning framework is a promising option for merging human and environmental feedbacks in other domains [25].

Threats to Validity: When human feedback is used in domains with intelligent agents or autonomous robots, the performance may be sensitive to the capabilities of the human participants involved in the study. The experiments reported in this paper used inputs provided by four human participants at infrequent intervals. Though the performance of the participants (when considered individually) were consistent across the two different domains, additional trials may be required in other domains to further substantiate the results reported here. Specifically, future experiments will consider a larger number of human participants, larger number of episodes, varying amounts of added noise, other combination functions, other appropriate application domains and thoroughly analyze the corresponding experimental results.

V. Conclusions and Future Work

Human participants can enable agents or robots to learn a rich representation of the task and domain, thereby operating reliably and efficiently in dynamic domains. It may however be infeasible for a human to possess the time and expertise to provide elaborate, accurate and real-time feedback to agents in complex domains. This paper described an approach that uses bootstrap learning in an augmented reinforcement learning framework to enable an agent to effectively merge the limited and unreliable high-level feedback from a human with the reinforcement signals obtained through interactions with the environment. The agent incrementally and continuously revises weights that determine the relative contribution of each feedback mechanism to its action choice policy. The agent is hence able to make best use of the available information. Experimental results indicate that the approach described in this paper outperforms the individual feedback mechanisms and the existing schemes to combine these feedback signals.

The aim of the research reported in this paper is to robustly combine human inputs and sensory cues. One direction of further research is to incorporate an underlying probabilistic belief representation to enable an agent (or a robot) to operate in partially observable domains and automatically acquire relevant human input when needed. Future research will also focus on multiple agents or robots collaborating towards a shared objective in real-world domains.

Acknowledgment

This work was supported in part by the ONR Science of Autonomy award N00014-09-1-0658. The author also thanks Mamatha Aerolla for her help in gathering some of the experimental results reported in this paper.

References

[1] M. A. Goodrich and A. C. Schultz, "Human-Robot Interaction: A Survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[2] A. Tapus, M. Mataric, and B. Scassellati, "The Grand Challenges in Socially Assistive Robotics," *Robotics and Automation Magazine, Special Issue on Grand Challenges in Robotics*, vol. 14, no. 1, pp. 35–42, March 2007.

[3] M. Cakmak, N. DePalma, R. Arriaga, and A. Thomaz, "Exploiting Social Partners in Robot Learning," *Autonomous Robots*, vol. 29, pp. 309–329, 2010.

[4] J. Forlizzi and C. DiSalvo, "Service Robots in the Domestic Environment: A Study of the Roomba Vacuum in the Home," in *International Conference on Human-Robot Interaction, HRI-06*, Salt Lake City, USA, March 2-4 2006, pp. 258–266.

[5] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning from Demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] C. Breazeal and A. Thomaz, "Learning from Human Teachers with Socially Guided Exploration," in *International Conference on Robotics and Automation*, 2008, pp. 3539–3544.

[7] D. Grollman, "Teaching Old Dogs New Tricks: Incremental Multimap Regression for Interactive Robot Learning from Demonstration," Ph.D. dissertation, Department of Computer Science, Brown University, 2010.

[8] D. Perzanowski, A. Schultz, W. Adams, E. Marsh, and M. Bugajska, "Building a Multimodal Human-Robot Interface," *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 16–21, January-February 2001.

[9] P. Zang, A. Irani, P. Zhou, C. Isbell, and A. Thomaz, "Using Training regimens to Teach Expanding Function Approximators," in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010, pp. 341–348.

[10] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A Survey of Socially Interactive Robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[11] S. Thrun, "Toward a Framework for Human-robot Interaction," *Human-Computer Interaction*, vol. 19, p. 2004, 2004.

[12] J. Biswas and M. Veloso, "WiFi Localization and Navigation for Autonomous Indoor Mobile Robots," in *International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, May 3-8 2010.

[13] W. Knox and P. Stone, "Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning," in *International Conference on Autonomous Agents and Multiagent Systems*, May 2010.

[14] S. Rosenthal, J. Biswas, and M. Veloso, "An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction," in *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Toronto, Canada, May 2010.

[15] J. Fasola and M. Mataric, "Robot Motivator: Increasing User Enjoyment and Performance on a Physical/Cognitive Task," in *International Conference on Development and Learning*, Ann Arbor, USA, August 2010.

[16] A. Rizzo, T. Parsons, G. Buckwalter, and P. Kenny, "A New Generation of Intelligent Virtual Patients for Clinical Training," in *The IEEE Virtual Reality Conference*, Waltham, USA, March 21 2010.

[17] A. Billard and K. Dautenhahn, "Experiments in Social Robotics: Grounding and Use of Communication in Autonomous Agents," *Adaptive Behavior*, vol. 7, no. 3-4, pp. 415–438, 1999.

[18] D. W. Franklin, T. W. Milner, and M. Kawato, "Single Trial Learning of External Dynamics: What can the Brain Teach Us about Learning Mechanisms in Brain Inspired IT," *International Conference on Brain-Inspired Information Technology*, pp. 67–70, 2007.

[19] R. L. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[20] O. Buffet and D. Aberdeen, "The Factored Policy-Gradient Planner," *Artificial Intelligence*, vol. 173, no. 5-6, pp. 722–747, 2009.

[21] I. Szita and A. Lorincz, "Learning Tetris using Noisy Cross Entropy Method," *Neural Computation*, vol. 18, pp. 2936–2941, 2006.

[22] P. Stone, R. Sutton, and G. Kuhlmann, "Reinforcement Learning for RoboCup Soccer Keepaway," *Adaptive Behavior*, vol. 13, pp. 165–188, 2005.

[23] W. E. Hockley, "Analysis of Response Time Distributions in the Study of Cognitive Processes," *Journal of Experimental Psychology. Learning, Memory and Cognition*, vol. 10, 1984.

[24] W. Knox, I. Fasel, and P. Stone, "Design Principles for Creating Human-Shapable Agents," in *AAAI Spring Symposium on Agents that Learn from Human Teachers*, 2009.

[25] M. Aerolla, "Incorporating Human and Environmental Feedback for Robust Performance in Agent Domains," Master's thesis, Computer Science, Texas Tech University, May 2011.