2011 RO-MAN
20th IEEE International Symposium on Robot and
Human Interactive Communication,
July 31 - August 3, 2011, Atlanta, GA, USA

Effect of Human Guidance and State Space Size on Interactive Reinforcement Learning

Halit Bener Suay and Sonia Chernova

Abstract— **The Interactive Reinforcement Learning algorithm enables a human user to train a robot by providing rewards in response to past actions and anticipatory guidance to guide the selection of future actions. Past work with software agents has shown that incorporating user guidance into the policy learning process through Interactive Reinforcement Learning significantly improves the policy learning time by reducing the number of states the agent explores. We present the first study of Interactive Reinforcement Learning in real-world robotic systems. We report on four experiments that study the effects that teacher guidance and state space size have on policy learning performance. We discuss modifications made to apply Interactive Reinforcement Learning to a real-world system and show that guidance significantly reduces the learning rate, and that its positive effects increase with state space size.**

I. Introduction

The ability to acquire and customize robot behavior through learning is vital for the successful deployment of robots in a broad range of consumer and service applications. In order for robots to become effective assistants in diverse human environments we must develop the capability to quickly and efficiently customize robot behavior and learn new tasks.

Numerous works have addressed the challenges of learning in complex real-world environments [1], [2]. One popular technique is Reinforcement Learning (RL) [3], which enables a robot to learn from rewards it obtains while attempting to perform its task. Although RL algorithms have had great success in offline learning and software applications, the large amount of data and high exploration times that such algorithms require make them intractable for many real-world robotic domains.

To speed up learning time, a number of recent works have shown that in the presence of a human, robotic systems can be designed to take advantage of human input [4], [5]. In the case of Reinforcement Learning, one example is the work of Kaplan et al. where they present a technique in which user-generated reward and punishment signals are used to train the robot new tasks in a form of clicker training [6]. Another related work was presented by Smart and Kaelbling, where human demonstrations of robot actions can be used to make learning of otherwise intractable policies possible [7]. Also, Knox and Stone introduce the TAMER framework that allows a human trainer to interactively shape an agents policy via reinforcement signals [8].

H. B. Suay and S. Chernova are with the Robotics Engineering Program, Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA 01609, USA. benersuay@wpi.edu, soniac@wpi.edu

The motivation behind our work is similar to these past examples. Here we study how Reinforcement Learning can be made more efficient for real-world robotic systems, by enabling human teachers to provide feedback at run-time. Our feedback differs from the related work in the sense that the teacher can provide not only reward for the preceeding action but also guidance for the subsequent action. Our work builds upon a series of studies by Thomaz et al. that show that in a reinforcement learning scenario users tend to not only reward past actions but also provide anticipatory rewards to guide the learner in future actions [9]. Based on this analysis, Thomaz and Breazeal present *Interactive Reinforcement Learning* (Interactive RL), an algorithm that enables a human trainer to 1) provide positive and negative rewards in real time in response to robot actions, and 2) provide anticipatory guidance input that restricts action selection choice and guides the learner towards performing the desired behavior [10]. The authors show that incorporating user guidance into the policy learning process significantly improves the learning rate of a software agent by reducing the number of states it explores.

In this paper, we report the results of a case study in which Interactive Reinforcement Learning was applied to teaching an Aldebaran Nao robot. To our knowledge, this paper is the first to apply this type of learning algorithm to a real-world robotic system. To evaluate the approach, we closely reproduce the interactive reward interface proposed in [10] and perform four experiments comparing Interactive RL with and without anticipatory guidance for two different state space sizes. We discuss modifications made to apply this learning approach to a real-world system. Our results show that guidance significantly reduces the learning rate, and that its positive effects increase with state space size. We conclude with a discussion of the benefits and challenges of using Interactive Reinforcement Learning in real-world robotic applications.

II. Reinforcement Learning

Reinforcement learning algorithms enable a robot to learn from its experience. We define reinforcement learning using the standard notation of Markov decision processes (MDPs) [11]. At every time step the robot observes its state $s \in S$ as a vector of k state variables such that $s = \langle x_1, x_2, \ldots, x_k \rangle$, and selects an action from the set of available actions A. The MDP's reward function $R : S \times A \mapsto \mathbb{R}$ and transition function $T : S \times A \mapsto S$ fully describe the system's dynamics. The robot attempts to maximize the long-term reward determined by the (initially unknown) reward

978-1-4577-1573-0/11/$26.00 ©2011 IEEE

and transition functions.

A learner chooses which action to take in a state via a policy, $\pi : S \mapsto A$. Policy π is modified by the learner over time to improve performance, which is defined as the expected total reward. Instead of learning π directly, many RL algorithms instead approximate the action-value function, $Q : S \times A \mapsto \mathbb{R}$, which maps state-action pairs to the expected real-valued return. One example of this kind of learning algorithm is Q-learning [12], which forms the basis for the reinforcement learning algorithm used in this paper.

Although RL approaches have enjoyed multiple past successes (e.g., TDGammon [13], inverted Helicopter control [14], and robot locomotion [15]), they frequently take substantial amounts of data to learn a good control policy. In robotic applications, collecting such data may be slow, expensive, or infeasible, motivating the need for finding ways of making RL algorithms more efficient.

III. Interactive Reinforcement Learning with Human Guidance

The Interactive Reinforcement Learning algorithm with human guidance, as described in [10], modifies traditional Q-learning by integrating human input into the reward and action selection mechanisms. In Interactive RL, all *reward input* to the robot comes from the human teacher instead of the environment; the teacher is given a short period of time following each action to provide positive or negative feedback. Additionally, during a short delay period between actions, the user can provide *guidance input*. Guidance is given with respect to an object or location (e.g., table), with the effect of restricting the choice of immediately available actions to those related to the target object (e.g., look at table, put down object on table). This technique can be thought of as a means of directing the robot's attention to the target object or area. In the following sections, we first present the details of the Interactive RL algorithm and then describe the interaction interface used by the human teacher.

A. Implementation Details

Algorithm 1 presents the pseudo-code for Interactive RL. The algorithm extends traditional Q-learning with the addition of human reward input (lines 18-21) and human guidance (lines 6-8). When learning begins, the agent has no knowledge of the task and the Q-table contains only the robot's current state. The table expands (line 17) as the robot performs actions, receives rewards and encounters new states, and the Q-values are updated (lines 23-24). In our implementation, we set the learning rate $\alpha = 0.3$ and the discount factor $\gamma = 0.75$. We initialize the Q-values for every new state that the agent discovers to 0.5.

Our initial implementation of Interactive RL followed the exact description provided in [10]. Following preliminary testing, we implemented a change to the action selection mechanism. Specifically, in the original implementation, in the absence of teacher guidance the agent uses the $Q[s, a]$ values to weight available actions and probabilistically select the next action to perform. During preliminary experiments, we found that this approach resulted in very long learning times for the robot (over 80 minutes for the small domain described in Section IV-B). In followup experiments, we found that ε-greedy action selection (lines 1-2, 12-15) results in significantly improved learning times. In this selection method, the robot performs the optimal action with probability $1 - \varepsilon$ and a random action with probability ε. Over time, the value of ε decays (line 25) until eventually the robot always performs the optimal action (i.e., policy exploitation).

The ε-greedy action selection method is used to select the next action both when a guidance signal is present and when it is absent. If none of the actions specified by the guidance message are valid (not physically possible), the robot ignores the guidance message and uses ε-greedy selection to pick among all available actions. For example, in the domain described in Section IV-A, the *PickUp* action is not valid before the robot determines whether there is an object in front of it using the *TakePicture* action.

Algorithm 1: Interactive Q-Learning with human generated reward and guidance.

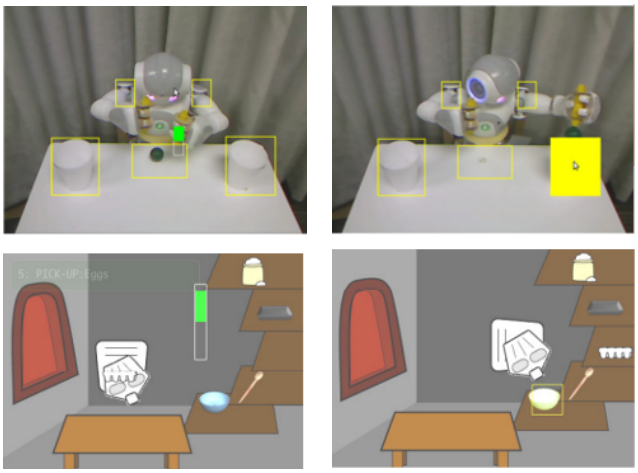
s = last state, s' = current state, a = last action

```
(1)  ε = initial value
(2)  δ_ε = ε/time steps to stop exploration
(3)  comment: Initialize Q-table with starting state
(4)  while learning do
(6)      while waiting for guidance do
(7)          if received human guidance
(8)              g = guidanceTarget fi od
(9)      if received guidance
(10)         A = legal actions that contain g only
(11)     else A = all legal actions for s fi
(12)     rand = random number[0, 1]
(13)     if rand > ε
(14)         a = arg max_a Q[s, a]
(15)     else a = randomly selected action in A fi
(16)     execute a and transition to s'
(17)     if s' not in Q then add s' to Q-table fi
(18)     while waiting for reward do
(19)         if receive human reward
(20)             r = human given reward
(21)         else r = 0 fi od
(22)     comment: update Q values:
(23)     Q[s, a] ←
(24)     Q[s, a] + α(r + γ(max_a' Q[s', a']) − Q[s, a])
(25)     if ε > 0 then ε ← ε − δ_ε fi od
```

B. Interaction Interface

A key contribution of the original Interactive RL paper is the interactive reward interface that enables an online user to train a virtual robot to bake a cake in a domain called Sophie's Kitchen*. The human trainer uses the left

*The original Sophie's Kitchen web application is available at http://www.cc.gatech.edu/∼athomaz/sophie



(a) (b)

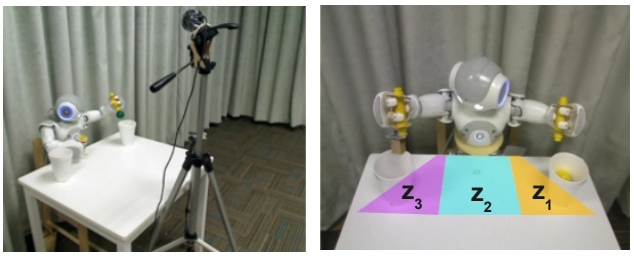
Fig. 1: Screen-shots of our interface (*Top row*) and Sophie's Kitchen (*Bottom row*). (a) shows positive reward given with a left click and drag upwards; (b) shows guidance given with a right click on the object that the teacher wants the agent to interact with.

mouse button to provide reward signals $r = [-1, 1]$ for the agent's actions, and the right mouse button to provide guidance for future actions. In this work, we recreate the interactive reward interface for a real robot by providing the human trainer with a fixed view of the environment via a web-cam (Figure 2a). As with the algorithm, we began by implementing the interface described in [10] in order to evaluate its use in real-world systems. Following preliminary evaluations we found the need to make one modification to the interface in order to improve the ease of interaction for our domain.

In our evaluation, the reward interface functions as in the original web application, in which left clicking anywhere in the image brings up a reward bar (Figure 1a) that the user can fill with a positive or negative reward value representing the desirability of the current state. The guidance interface differs slightly; instead of allowing the user to click anywhere, our interface highlights five rectangular regions as potential guidance targets. The teacher provides guidance by right-clicking within the highlighted boxes (Figure 1b). Table I lists the five regions and their effect on action selection.

IV. Experimental Setup

To investigate the ways in which human guidance impacts the policy learning rate in real-world systems we developed an object sorting domain using an Aldebaran Nao humanoid robot. The domain setup consists of a stationary work table where the Nao is seated on a chair. Magnetic objects are placed in front of the robot one at a time. The robot must learn to use its camera at the appropriate time to identify the characteristics of the object and then to pick up and place the object in one of two cups located on the table. The user observes the robot's actions via a web-cam (see Figure 2a) and provides feedback to the robot through the graphical user interface. The described setup enables us to evaluate



(a) (b)

Fig. 2: Experiment Setup. (a) Web camera captures the table and the robot for the user interface. (b) The table is divided in three zones. Orange, cyan blue and purple show z_1, z_2, and z_3 respectively. The robot has an object at the tip of its right hand and is about to drop it in the right cup.

the impact that human guidance input has on robot learning performance and the usability of the user interface.

A. Domain

The experimental domain $D = (Z, S, A, T)$ is defined by a finite set of zones Z, a finite number of world states S, a finite set of possible robot actions A and the transition function $T : S \times A \rightarrow S$ that determines transitions between states by way of actions. Zones Z represent the regions where the robot's arms or an object can be located. In our domain, $Z = \{z_1, z_2, z_3\}$ corresponds to the left, front and right side of the robot respectively (Fig. 2b).

The state of the world $S = \{S_r \cup S_o\}$ consists of the union of the state of the robot S_r and state of the object being classified S_o. The robot state is defined by $S_r = (z_{lh}, z_{rh}, hand_o)$, where $z_{lh}, z_{rh} \in Z$ correspond to the current zone of the left and right hands, respectively, and $hand_o \in \{left, right, none\}$ represents whether an object is located in either of the robot's hands, or not. For example, in Fig. 2b the robot's state is represented by $S_r = (z_1, z_3, right)$.

The object state is defined by $S_o = (I, o_z, o_p)$, where I is an object descriptor that specifies the characteristics of the object using a set of features described in Section IV-B. $o_z \in Z$ is the current zone of the object, and $o_p \in P$ is the placement of the object such that $P = \{z_1, z_2, z_3, right\_cup, left\_cup\}$. For example, in Fig. 2b the object state is represented by $S_o = (pattern, z_3, z_3)$.

The robot has eleven possible actions, $TakePicture$, $xPickUp$, $xPutDown$, $xDrop$, $xLeft$, and $xRight$ where $x \in \{L, R\}$ determines the arm with which the action is performed. The $TakePicture$ action makes the robot take a snapshot of the table in front of it and extract features from the image to determine the characteristics of the object currently placed there (if any) (see Fig. 3). $xRight$ and $xLeft$ actions move the hands between zones (e.g. $LLeft$ moves left hand left, that is, to Zone 1 and $RLeft$ moves right hand left, that is, Zone 2. Note that the robot's left hand can operate only in $z_{1,2}$, and the right hand can operate only in $z_{2,3}$, which enables the robot to pick up an object with either hand but allows only one hand to reach each cup and $Drop$ the object into it. If the robot initially picks

Guidance	ActionSet
Zone 1	$LLeft$, $LDrop$
Zone 2	$LPutDown$, $LPickUp$, $RPutDown$, $RPickUp$, $TakePicture$
Zone 3	$RRight$, $RDrop$
Left Shoulder	$LPickUp$, $LRight$, $LLeft$, $LPutDown$, $LDrop$
Right Shoulder	$RPickUp$, $RRight$, $RLeft$, $RPutDown$, $RDrop$

TABLE I: Guidance messages restrict action selection to the set of available actions as shown. The first letter *L* and *R* denotes either Left or Right hand depending on the hand with which the action is executed.

up the object with the wrong hand, it can switch hands by performing a $xPutDown$ action followed by a $xPickUp$.

Table I shows how different guidance messages restrict the set of available robot actions. Note that the degree to which guidance reduces the set of actions has a significant effect on the learning rate. If guidance does not significantly reduce the choice of available actions, then the effect of this input method is reduced since the robot will have to choose randomly among them. In contrast, if the guidance message limits the robot to a single action, then all choice is taken away from the robot; this approach would be equivalent to the human demonstration techniques applied in *learning from demonstration* approaches [16], [7]. The effect on learning would be to eliminate exploration, resulting in a policy that is only as good as the decisions demonstrated by the human, and is poorly defined in areas of the state space not encountered during training. Further studies are needed to determine the precise impact of these choices. In our implementation we chose to design guidance as an effective teaching method that always restricts the list of available actions but never reduces action selection to a single choice.

B. Experimental Conditions

We performed four sets of experiments to evaluate the Interactive RL algorithm and study the effects that 1) teacher guidance, and 2) size of the state space have on learning performance. The same object sorting task was used for all experimental conditions. In the no-guidance condition, the user was restricted to only the reward input, and the guidance condition allowed for both types of input. To vary the state space size we changed the number of features used to describe the object being sorted. Specifically we used the following two experimental conditions:

- **Small deterministic state space:** In the small state space condition, the object descriptor I of $S_o = (I, o_z, o_p)$ consists of a single variable with two possible values, $plain$ or $pattern$, representing the color class of object. This mapping is determined based on the number of Speeded Up Robust Features (SURF) [17] identified in the image of the object and identifies the object as either a solid colored ball or a character magnet, respectively. Specifically, objects for which over 50 SURF features were identified were characterized as *pattern*, otherwise the object was *plain*. In our tests we found that the number of SURF features deterministically separate these two object types. Combining this binary object representation with other state information, such as object location and robot state, results in 360 possible states.
- **Large, non-deterministic state space:** In the large state space condition, the object descriptor I consists of four elements, where measured values are rounded to the closest defined category: the number of SURF features (50, 100, or 150), smoothness (0.05 or 0.1), entropy (5.0 or 10.0) and area of the bounding box of the object (15,000, 20,000 or 25,000). We calculate smoothness and entropy as described in [18]. After image processing, the value for each descriptive feature is thresholded into the categories listed above. This representation was purposefully designed to reduce the size of the state space by providing a small set of possible values for each variable, while at the same time making sure that none of the descriptive elements alone is sufficient for distinguishing between the plain and patterned object types. For example, although they are the same type of object, the plain green and plain yellow magnets have different smoothness, perceived area and entropy states. To the robot these objects therefore appear distinct and it must learn that they belong to the same group. Furthermore, this representation is non-deterministic and variations in object placement will result in slightly different state representations, allowing us to evaluate how Interactive RL performs under these conditions. In total, the large state space representation results in 6480 states.

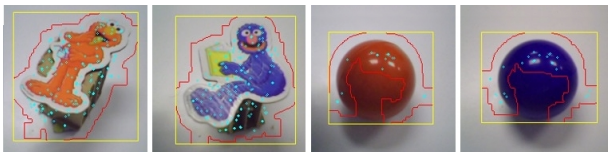


Fig. 3: Images of different objects taken by robot's camera. Light blue spots show SURF, red lines show the outline, and yellow rectangles show the bounding box or rectangle of interest of the object found.

During the experiments all processing was performed on a PC connected to the robot over the network. For image processing we used OpenCV [19], examples of the processed images are shown in Figure 3. All experiments were performed by the same teacher who is one of the authors. The teacher used the following consitent guidance and reward shaping protocol. For the first 20 actions of the experiments, all actions were given reward and guidance (in case of experiments with guidance). After 20 actions, reward and guidance were given only after the incorrect action that the robot performed. After each incorrect action,

the following 5 actions were rewarded and guided. If the action was correct, no reward or guidance was given until the next incorrect action. In the small state space condition we used $\varepsilon = 0.1$ and $\delta_\varepsilon = 0.002$ (ε reaches 0 after 50 time steps), and in the large state space condition $\varepsilon = 0.1$ and $\delta_\varepsilon = 0.001$ ($\varepsilon = 0.1$ reaches 0 after 100 time steps). We terminate learning once the robot is able to correctly sort three objects into each cup without guidance or reward. Objects were presented to the robot in random order. A video of the experiment and the interface can be found online[†].

V. Results

We ran three trials for each of the four experimental conditions, the averaged results are presented in Figure 4. Graphs plot the level of teacher involvement over time in terms of the percentage of actions for which the teacher provided reward and/or guidance. A summary of experimental statistics is given in Table II.

A comparison of plots Fig. 4a to 4c and Fig. 4b to 4d, shows that the use of guidance remarkably reduces the learning time (small state space: 28%, large state space: 44%) as well as the number of states explored (small state space: 51.5%, large state space: 38%). In effect, the robot learns more quickly with guidance because it selects the correct action more frequently and therefore avoids exploring irrelevant areas of the state space. As a result, we observe that in the presence of guidance the teacher reduces the number of interactions earlier in the training process compared to the no-guidance condition.

We also observe differences in teacher behavior between the guidance and no-guidance conditions. The cumulative reward for the trials is positive in the presence of guidance and negative in its absence. This is a direct consequence of the fact that the robot selects more inappropriate actions when learning without guidance and the teacher spends more time giving praise to the robot than punishment. This is an interesting result given that a number of studies show that given the option between reward and punishment most human teachers prefer to use positive reward and avoid punishment when training an agent [9].

In summary, the comparison study shows that teacher guidance significantly reduces learning time by reducing the number of states explored. This result supports the findings of [9], in which average reductions of 30% in case of expert teachers and 49% in case of non-expert teachers were observed in the number of trials. Furthermore, we find that the impact of teacher guidance increases as the size of the state space grows. We hypothesize that similar benefits would be observed with respect to growth in the size of the action space, although further studies are required to verify this effect.

VI. Discussion

The Interactive Reinforcement Learning approach has several key strengths: 1) its simple action selection mechanism can be integrated into a broad range of existing RL approaches, 2) its application results in a flexible learning system that can learn independently but is able to take advantage of human guidance when it is available, and 3) guidance is combined with independent exploration in a way that enables the robot to not only imitate the demonstrated behavior but also to surpass the performance of the teacher if necessary.

However, there are a number of challenges to deploying this approach in real-world applications. The current implementation of the teaching interface assumes a fixed-camera setup that enables the human user to select relevant areas of the state space for guidance. While this interface may translate into some service and factory applications, it is not suitable for mobile robotic systems or unconstrained environments. Further work is required to evaluate how other input methods, such as portable tablets, speech, and gestures, can be used to interact with the robot in this setting.

The running time of the algorithm is another area for potential improvement. Currently, the robot requires approximately half an hour to learn a relatively simple task. Much of this learning time is dependent on the time required

[†]http://www.youtube.com/WPIrail

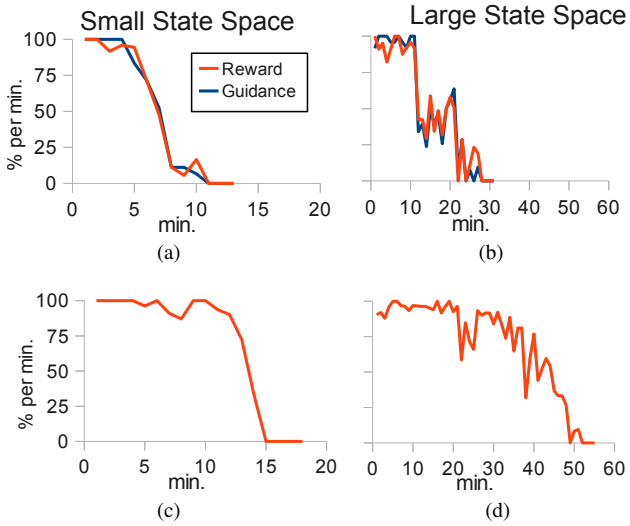


Fig. 4: User interaction and training time with (a,b) and without (c,d) guidance input. Figures (a,c) and (b,d) show the results obtained in small and large state space respectively. The y-axis and legend are common for all graphs.

	Fig.4a	Fig.4b	Fig.4c	Fig.4d
#States	11.33	42.67	22	68.67
#Actions	82	201	134.67	366.67
Cumul. Rew.	16.67	58	-44.67	-74.67
Total #G.	44	113.67	0	0
Training Time [min.]	14	31	18	55

TABLE II: Experiment statistics. All results are shown as the average of three trials for each experiment. Number of states discovered (#States), total number of actions performed (#Actions), sum of the positive and negative rewards given (Cumul. Rew.), total number of guidance given (Total #G.) and training time.

to execute each action. For example, an average of 201 actions was required to learn the large task, which is a relatively small number compared to the total number of states in the domain, but each action had an average time of 9.8 seconds. Other critical factors include the exploration rate, state representation, and the degree to which guidance reduces the choice of available actions. In order to enable this approach to scale to more complex domains, we must explore the effects that these factors have on learning time, as well as study how Interactive RL can be applied to RL algorithms that utilize continuous state space representations which model complex world state more effectively [20], [21].

We compared two conditions, the large state space, in which the generalization across multiple sensory inputs had to be learned, and the small state space, in which a single sensory input automatically generalized across the two object types. Both conditions were designed through manual selection of the appropriate state features in a preliminary testing stage. In the case of the small state space, the chosen representation enabled us to teach the task more effectively because instructing the robot through the use of just one to two patterned magnets enabled it to automatically generalize the behavior to all patterned shapes. In the case of the large domain, each patterned item was described by a unique set of features, and more examples were required for the algorithm to learn the appropriate generalization. The large state space more closely represents a real-world scenario; robots designed for operation in real-world environments in the presence of non-expert users are likely to have an even larger number of sensory inputs, possibly with many irrelevant features, which will lead to longer learning times. This motivates the need for semi-autonomous feature selection methods that reduce the size of the state space and enable the robot to learn more effectively.

Finally, when designing interactive learning algorithms we must consider their usability for non-expert users. This work focused on empirical evaluation of the best-case scenario in which the robot was taught by a roboticist after numerous practice trials. In future work we plan to perform a user study in order to examine how intuitive the training interface and teaching technique are for people who are not familiar with programming or robots. Particularly important to examine in this context is the effect of the quality of teacher demonstrations on learning performance.

VII. Conclusions

The ability for robots to quickly and efficiently learn new tasks is critical for their successful operation in human environments. In this paper we explored how the Interactive Reinforcement Learning algorithm that enables a human trainer to provide both rewards and anticipatory guidance for the learner can be applied to a real-world robotic system. We reported the results of four experiments examining the effect that human guidance and state space size have on policy learning time. Our findings show that guidance significantly reduces the number of explored states and the learning time, under the condition that each guidance message restricts the set of choices for action selection to a small number of actions. We also show that guidance leads the teacher to provide more positive reinforcement signals, and that its positive effects increase with respect to the size of the state space. Overall, we believe Interactive Reinforcement Learning can be used to improve the learning rate of many Reinforcement Learning systems. However, we note a number of important directions for future work that will lead to further improvements in performance and usability.

References

[1] D. Gu, R. J. Howlett, and Y. Liu, *Robot Intelligence: An Advanced Knowledge Processing Approach*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, ser. Intelligent robotics and autonomous agents. MIT Press, Sept. 2005.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, London, England: The MIT Press, 1998.

[4] S. Calinon, *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.

[5] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *J. Artificial Intelligence Research*, pp. 1–25, 2009.

[6] F. Kaplan, P. Y. Oudeyer, E. Kubinyi, and . Miklsi, "Robotic clicker training," *Robotics and Autonomous Systems*, vol. 38, no. 3-4, pp. 197–206, 2002.

[7] W. D. Smart, "Making reinforcement learning work on real robots," Ph.D. dissertation, Department of Computer Science, Brown University, Providence, RI, 2002.

[8] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *The Fifth International Conference on Knowledge Capture*, September 2009.

[9] A. Thomaz, G. Hoffman, and C. Breazeal, "Reinforcement learning with human teachers: Understanding how people want to teach robots," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, 2006, pp. 352 –357.

[10] A. L. Thomaz and C. Breazeal, "Adding guidance to interactive reinforcement learning," in *In Proceedings of the Twentieth Conference on Artificial Intelligence (AAAI)*, 2006.

[11] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[12] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[13] G. Tesauro, "TD-Gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computation*, vol. 6, no. 2, pp. 215–219, 1994.

[14] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Inverted autonomous helicopter flight via reinforcement learning," in *International Symposium on Experimental Robotics*, 2004.

[15] M. Saggar, T. D'Silva, N. Kohl, and P. Stone, "Autonomous learning of stable quadruped locomotion," in *RoboCup-2006: Robot Soccer World Cup X*, ser. Lecture Notes in Artificial Intelligence, G. Lakemeyer, E. Sklar, D. Sorenti, and T. Takahashi, Eds. Berlin: Springer Verlag, 2007, vol. 4434, pp. 98–109.

[16] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, pp. 469–483, May 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1523530.1524008

[17] T. T. Herbert Bay, Andreas Ess and L. V. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[18] R. E. W. Rafael C. Gonzalez and S. L. Eddins, *Digital Image Processing using Matlab*. Prentice Hall, 2003.

[19] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, 2008.

[20] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 2000, pp. 903–910.

[21] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *Proceedings of American Association for Artificial Intelligence(AAAI)*, 1998.