In: Proceedings of the Fourth Joint IEEE ICDL-EpiRob '14, pp. 125-130, Genoa, Italy (2014)

Improving Reinforcement Learning with Interactive Feedback and Affordances

Francisco Cruz, Sven Magg, Cornelius Weber and Stefan Wermter
Knowledge Technology Group, Department of Informatics, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
Email: {cruz, magg, weber, wermter}@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de/WTM/

***Abstract*—Interactive reinforcement learning constitutes an alternative for improving convergence speed in reinforcement learning methods. In this work, we investigate inter-agent training and present an approach for knowledge transfer in a domestic scenario where a first agent is trained by reinforcement learning and afterwards transfers selected knowledge to a second agent by instructions to achieve more efficient training. We combine this approach with action-space pruning by using knowledge on affordances and show that it significantly improves convergence speed in both classic and interactive reinforcement learning scenarios.**

I. Introduction

In learning robots different tasks such as navigation, grasping, vision, speech recognition, and pattern recognition among others, can be tackled by different machine learning paradigms, like supervised, unsupervised or reinforcement learning [1], [2]. These tasks are often needed in domestic scenarios with active human participation in order to execute them collaboratively.

This paper focuses on Reinforcement Learning (RL onwards) [3] which uses sequential decisions where an agent interacts with its environment. In each state, the agent selects an action to be performed and in some states receives either a reward or a punishment. It attempts to get the highest reward over time, therefore the problem is reduced to finding a proper policy that allows to associate actions to states in order to get maximal future reward.

RL has demonstrated to be a very useful learning approach, but the largest problem for RL agents is often the time spent for the learning process, mainly due to large and complex state spaces which lead to excessive computational costs in order to find a suitable policy [4]. There are different approaches that attempt to speed up RL including Interactive Reinforcement Learning (IRL) where RL is supported by an external trainer who provides some instructions on how to tackle the problem [5], [6], [7].

A promising alternative method to improve convergence speed is the use of affordances [8], [9], which means that cognitive agents favor specific actions to be performed with specific objects. Affordances represent neither agent nor object characteristics, but rather the characteristics of the relationship between them [10]. In this paper, we are not yet looking at learning affordances automatically, but we want first to investigate how beneficial they are in our proposed scenario.

In this work we integrate IRL and affordances to enhance the performance of classic RL methods, demonstrating that we get better results when using affordances with classic RL as well as when combined with IRL with instructions coming from a previously trained agent.

Our paper is organized as follows: first, we describe the main characteristics of the IRL paradigm and different strategies to combine the RL approach with external trainer interaction. Next, we give a description of the concept of affordance and explain how they are used in our approach. Then, we define our robotic agent scenario for a domestic task and state our strategy to speed up RL with both instructions and affordances. Moreover, we show and compare our main results in both RL and IRL approaches. Finally, we present our main conclusions and describe future research.

II. Interactive Reinforcement Learning

RL attempts to maximize received reward in a specific scenario through the interaction that is produced between a robotic agent and its environment. Such actions are selected yielding a transition to a new state and getting a reward. The optimal action-value function can be solved through the Bellman equation for q^*:

$$q^*(s,a) = \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma \max_{a'} q^*(s',a')] \quad (1)$$

where s is the current state, a is the taken action, s' is the next state reached by performing action a in the state s, a' are possible actions that could be taken in s'. In the equation, p represents the probability of reaching the state s' given that the current state is s and the selected action is a, and r is the received reward for performing action a in the state s for reaching the state s'. The parameter γ is known as discount rate and represents how influential future actions will be [3].

There are different strategies of interaction between a robotic agent and an external trainer for developing collaborative tasks, for instance, they could interact through demonstration [11], [12], imitation [13] or feedback [5], [14]. Interaction through feedback is used in our approach. So far two different strategies have mainly been used to give feedback. These consist of manipulating the received reward r or the selected action a to be performed in the state s. In both cases, shown

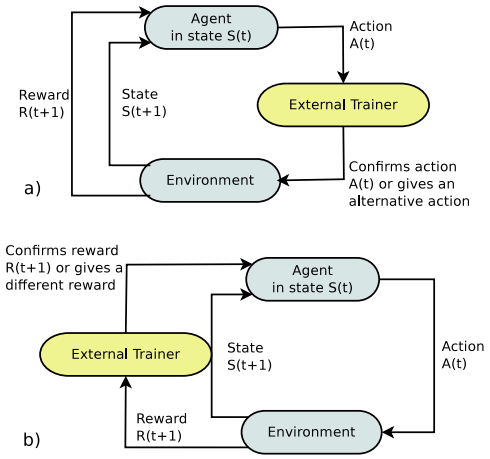

Fig. 1. a) First approach to interaction between a robotic agent and an external trainer by feedback. In this case, the external agent is able to change a selected action to be performed in the environment b) Second approach to interaction between a robotic agent and an external trainer by feedback. In this second case, the external agent is able to modify the proposed reward.

in figure 1, an action is carried out in the environment, and thus a new state is reached and a new reward obtained. When the external trainer does not give feedback, acceptance of the action a or reward r is assumed. Novel strategies can emerge from mixing both, namely, decision on the action a to be performed and on manipulating the received reward r as well.

Figure 1a shows the first approach in IRL through feedback where interaction from an external trainer is given during the robot's action selection. Manipulating actions is a way to tell the agent that what it is currently doing is wrong and should be corrected in the future [15]. The second approach is shown in figure 1b. In this case, the external trainer may modify the reward r and send its own reward to the agent, specifying how good or how bad the latest performed action a is. Examples of this approach were developed in [5], [16].

In a real domestic scenario an agent is expected to work with humans as external trainers. For them, delivering directions is more natural instead of quantifying a reward; hence, we decided to use the first feedback method of IRL in this paper, shown in figure 1a. For this, we created a simulated environment where the external trainer was previously trained using classic RL. It was therefore an artificial trainer agent that had full knowledge about all possible actions and delivered it to a second artificial agent which was trained with IRL.

For the training in RL we use the on-policy method SARSA [17] where the equation 1 is solved considering transitions from state-action pair to state-action pair instead of transitions from state to state only. Thus, every state-action value can be updated using the following equation 2:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2)$$

Algorithm 1 shows the SARSA method using interaction. The conditional statement in line 8 represents the fact that the external agent delivers feedback and changes the next action a'. Line 11 represents the regular update of the next action.

The policy used in lines 3 and 11 corresponds to ϵ-greedy selection with $\epsilon = 0.1$. Thus, most of the time the action with the highest estimated state-action pair value is selected according to:

$$a = \underset{a}{\operatorname{argmax}}\ Q(s_t, a) \quad (3)$$

where s_t is the current state at time t, and a corresponds to the action. Later, we will modify this policy slightly to use affordances which is explained in the next section.

Algorithm 1 On-policy SARSA algorithm using interaction and affordances

Require: Previous definition of states and actions
1: Initialize $Q(s, a)$ arbitrarily
2: Filter actions avoiding failed states using affordances
3: Choose a from s using ϵ-greedy action selection
4: **repeat**
5: Take action a
6: Observe reward r and next state s'
7: Filter actions avoiding failed states using affordances
8: **if** external trainer gives an alternative next action **then**
9: Choose a' given by the external trainer
10: **else**
11: Choose a' from s' using ϵ-greedy action selection
12: **end if**
13: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
14: $s \leftarrow s'$
15: $a \leftarrow a'$
16: **until** s is terminal

III. Use of Affordances

The term of affordance was coined by Gibson as action opportunities for an observer who is aware of its environment or an object in it [10]. For instance, a cup and a sofa afford different actions to a person who is able to grasp the cup and sit on the sofa, but cannot do it the other way around. Thus, an agent using its prior knowledge, its experience, and its perceived information is able to determine some object affordances in advance or the caused effect after a specific action is performed.

The use of affordances in robotics allows to address much more interesting problems by reducing action space due to the retrieved relevant information from the world allowing to identify what actions are possible for a robot to perform [8]. Affordances have been mainly used to relate actions to objects and we use them as a way to represent prior object/action information.

Affordances represent characteristics of the relation between an agent and an object. More specifically, one affordance can be defined as the relationship between an object, an action, and an effect [8], [18] as a triplet shown in equation 4:

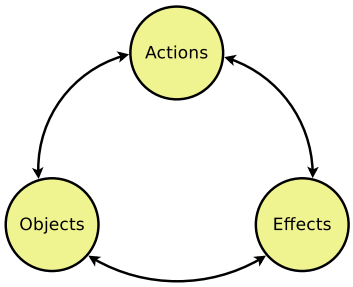

Fig. 2. Affordances as relations between objects, actions, and effects where objects are entities which the agent is able to interact with, actions represent the behavior that can be performed with the objects, and the effects are the results caused by applying an action.

$$Affordance = (Object, Action, Effect) \quad (4)$$

Figure 2 shows the relationship between these three components where objects are entities which the agent is able to interact with, actions represent the behavior or motor skills that can be performed with the objects, and the effects are the results of an action using an object [19].

We use affordances to provide knowledge about actions which lead to undesirable or failed states which means that it is not possible to reach the final state from them. Therefore, the action space is reduced by avoiding these states. In our approach, the set of possible actions is filtered every time for the current state that the agent is in. Thus, every agent has a preprogrammed behavior to avoid undesirable actions in all states. Algorithm 1 shows the use of affordances in lines 2 and 7 where the filter is applied.

Although this preprogrammed use of affordances can only be used in our abstracted scenario, it enables us to investigate the effects of affordances on our learning approach. In future work the affordance triplets have to be retrieved automatically during learning.

IV. Domestic Scenario

In this work, we have defined a domestic scenario focused on cleaning a table which can be learned by a robot supported by an external agent for reducing the training time. To perform this, we have defined objects, locations and actions inside this scenario. Initially, a robotic agent which has a *sponge* will stand in front of a table, in particular in front of a specific area of the table which is desired to be cleaned. The table has an additional object like a *cup*. We have defined three locations, *left*, *right* and *home*, the *left* and *right* side on the table within the reachable area for the robot's arm, and the location *home* is the initial position of the robot's hand which is also the storage place of the object *sponge*.

In this scenario four actions are possible: (i) *get* an object, which allows the robotic agent to pick up an object which is placed in the current robot hand location, (ii) *drop* an object, which allows the robot to put down the object that is currently kept in the robot's hand in the location where it is positioned, (iii) *go*, which allows to move the robotic hand towards any location, and (iv) *clean*, which cleans only the current location where the hand is positioned.

TABLE I
List of defined objects, locations and actions for cleaning-table scenario.

Objects	Locations	Actions
sponge	left	get <object>
cup	right	drop
	home	go <location>
		clean

Let us suppose that the *cup* is located on the *left* side of the table at the beginning. The initial position of the robot's hand is the location *home*, and we want to finish with the hand free and over *home* with both sides clean. In this context, an episode is defined as an attempt to reach the goal. The following example shows an episode to complete the task successfully: *get sponge, go right, clean, go home, drop, go left, get cup, go right, drop, go home, get sponge, go left, clean, go home, drop*. The minimal number of actions is 15 to complete the task. Table I shows a summary of objects, locations and actions defined for this domestic scenario, which for this first approach has been developed in a simulated robot environment. Figure 3 shows an example of a real scenario where a robot is standing in front of the table and Figure 4 shows a simulated collaborative scenario in the V-REP simulation environment.

V. Proposed Model

To implement this domestic scenario we develop a state machine with one final state. The number of possible states is obtained by considering the possible combinations among (i) the robotic agent's hand position, (ii) whether the hand is free or holding an object, and which, (iii) the cup location, and (iv) the current condition of every side of the table surface, that is, if the table has already been cleaned or not (see Table II). However, this leads to an explosion of the number of states where many of them are failed states from where it is not possible any more to reach the final state. For instance, let us assume that the robot has just performed the action *get cup*; therefore the cup is held in its hand. If the robot then cleans a section of the table with the *cup* in its hand instead of a *sponge*, it may shatter the cup.

Failed states are not necessarily a problem; they can be handled and controlled by an RL approach by giving punishment (or negative reward). In this case, the agent is discouraged to perform that action in the future from the same state. Nevertheless, a better strategy is to consider the use of affordances which are motivated from psychology [10] and which have been examined for improving the convergence speed of learning algorithms [20], [21]. In our work, we are interested in reducing the needed episodes to reach a reasonable performance by using affordances in both approaches, RL and IRL. This becomes especially important when it is desired to work in real scenarios, because, while in simulated environments it is feasible to run many episodes, in a real

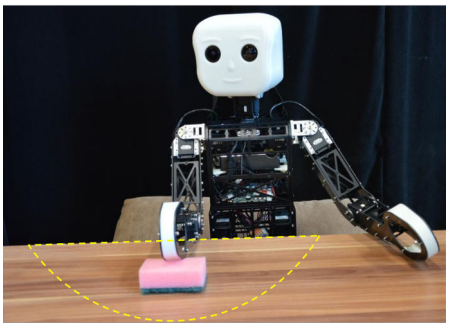

Fig. 3. A humanoid robot stands in front of the table, the yellow area contains the left and right locations within the reachable zone for the robot's arm.

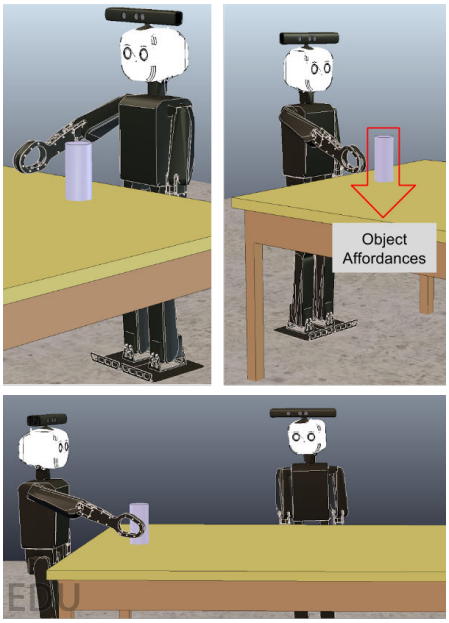

Fig. 4. The scenario in the upper left shows a simulation where the robot stands in front of the table. The simulated scenario in the upper right shows the task where object affordances are being used. The final scenario shows two humanoid robots in a simulated collaborative domestic scenario. After one agent learns the cleaning-table task it can transfer selected knowledge by interaction to the second agent.

environment one cannot afford to run excessive episodes until reaching a suitable policy.

Considering affordances, it is possible to define 46 regular states, many of which are shown in Table II where the initial state is the number 1 and the final state is the number 46. The states between number 15 and 28, and states between 29 and 42 are not shown since they are exactly the same as the first 14 states apart from the *side conditions* which are *clean-dirty* and *dirty-clean* respectively. Figure 5 summarizes the state transitions and we can observe an initial state where both sides of the table are dirty. In this context, a regular state means that it is still possible to reach the final state. Internally, in our algorithm we do not use *left* or *right*, but rather *side1* and *side2*, where *side1* is the side of the table where the *cup* is at the beginning of an episode, and it is feasible to start cleaning any side of the table because both paths will lead to the final

TABLE II
REGULAR STATES DEFINED FOR CLEANING-TABLE SCENARIO. THE LAST FOUR STATES ARE INDEPENDENT OF THE CUP POSITION SINCE THEY REPRESENT ACTIONS FOR RETURNING THE SPONGE TO THE HOME POSITION ONCE BOTH SIDES ARE ALREADY CLEANED.

Side condition	Cup position	Hand position	Held object	Nr.
dirty-dirty	left	home	free	1
			sponge	2
		left	free	3
			sponge	4
			cup	5
		right	free	6
			sponge	7
	right	home	free	8
			sponge	9
		left	free	10
			sponge	11
		right	free	12
			sponge	13
			cup	14
...	...	...	...	...
clean clean-	left or right	left	sponge	43
		right	sponge	44
		home	sponge	45
			free	46

Fig. 5. Given that the cup is on the left side, there are two feasible paths for reaching the final state from the initial state. The lower path would include 17 possible different states and consists of cleaning first the empty side of the table, and then moving the cup in order to clean the second side. The upper path would include 24 possible different states and consists of moving the cup first to the empty side and cleaning this side, and after that returning the cup to its original side for cleaning the second one. In both cases the final state is reached involving different numbers of intermediate states. The ending of the task consists of four states including those states for returning the robot's hand to the home location and for dropping the sponge.

successful state. As we already stated in the previous section, the shortest path is composed of 15 actions for reaching the final state.

VI. SIMULATIONS AND RESULTS

To carry out this experiment, first, we used a classic RL approach to train a simulated agent for reaching the final state. Afterwards, we introduced affordances and were able to reduce the needed episode numbers to obtain a satisfactory performance in terms of the performed actions for reaching the final state. Finally, a second agent was trained using IRL and receiving feedback from the previous trained agent that sometimes showed which action to choose in a specific state. Each set-up was carried out 100 times using the obtained

average values in figures. Moreover, involved parameter values were selected empirically with learning rate $\alpha = 0.1$, discount factor $\gamma = 0.9$, and ϵ-greedy action selection with $\epsilon = 0.1$. The Q-values were initialized randomly using an uniform probability distribution between 0 and 1. These three set-ups will be explained in detail in the next subsections.

A. Training an Agent Using RL

In the first step, simulations were performed to train the first agent with the SARSA algorithm. On the one hand, our reward function delivered a positive reward equal to 1 to the agent every time that it reached the final state. On the other hand, it delivered a negative reward or punishment equal to -1 every time that a failed state was reached to discourage accessing this state in the future again. Equation 5 shows the reward function defined in this scenario.

$$r(s) = \begin{cases} 1 & \text{if s is the final state} \\ -1 & \text{if s is a failed state} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Due to the large number of states, and since many of them were actually failed states, the agent needed around 700 episodes of training until reaching at least once the final state in 100 attempts. Figure 6 shows the average number of actions involved in every episode until reaching the final state with green crosses. Here, the number of actions are only shown when the final state was reached; so in the episodes where no cross is shown always a failed state was reached. In the first half of the training the final state was reached just a few times, but in the last part the system became more successful and furthermore, in those cases close to 15 actions were performed which in fact is the minimal number of possible actions as mentioned above. The dashed green line shown in figure 6 with a different scale in the y axis represents the percentage of successful runs. This curve is calculated by a convolution using 50 neighbors to make it smoother and we observe that the initial percentage of success is very low. Nevertheless, additional tests have shown that the curve keeps growing until approximately 4000 episodes, reaching only success rates of 40%. This clearly shows the difficulty to obtain a stable behavior by RL and the corresponding long training times.

B. Training an Agent Using RL with Affordances

As mentioned in the previous subsection, excessive episodes were required in order to reach a stable system. Even though this is computationally expensive it would be feasible in a simulated environment. Nevertheless, it would be unfeasible to perform those quantities of episodes in a real scenario. Therefore, we decided to explore the benefit of affordances which were implemented by reducing the valid action space for the agent to avoid failed states. Using this approach, we managed to reduce the number of episodes considerably, i.e. we needed less than 150 episodes to get a stable behavior and a number of performed actions close to the minimum. Figure 6 shows the average number of actions in each episode with this set-up.

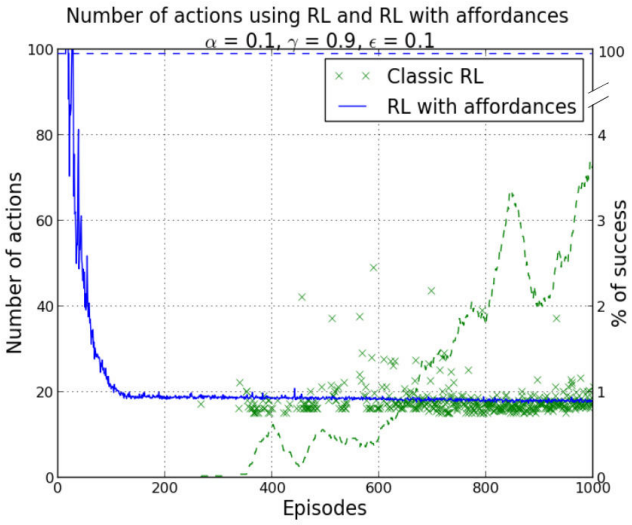

Fig. 6. Average number of actions needed for reaching the final state for classic RL approach (green) and RL with affordances (blue) from 100 runs are shown. For classic RL, the average number of actions is shown as green cross if at least one run was successful. Dashed lines show the percentage of runs that have reached the final state which is always 100% in case of RL with affordances since failed states can not be reached anymore. Success rate was smoothed by a moving average with window size 50.

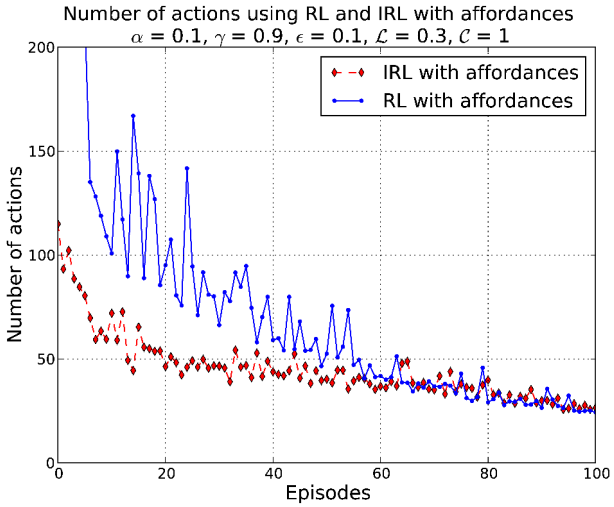

Fig. 7. Average number of actions needed for reaching the final state in each episode in RL and IRL with affordances. The convergence speed was improved by means of interaction from the first trained agent. To obtain less than 50 performed actions averaged over 100 runs 49 episodes were needed in RL with affordances and in those first 49 episodes 5449.55 actions were performed in RL with affordances while 2612.56 actions were performed in IRL with affordances.

Whereas in the previous scenario the probability of success was still low in the first episodes of training, in this set-up no episode ended in a failed state because of the use of affordances, since an episode could only end when the agent reached the final state (see dashed blue line in figure 6), which also produced considerably lower variation in comparison with the preceding scenario.

C. Training a Second Agent Using IRL with Affordances

Once a first agent had been trained, a second agent was trained with an IRL approach that allowed to manipulate selected actions as shown in figure 1a. In our method, the external trainer that provided feedback was the trained agent which already had knowledge about the task to be performed. Furthermore, we based the interaction model on a method called *advise* proposed by Griffith et al. [22]. In their work the authors use two likelihoods, $\mathcal{C}$ to refer to the consistency of feedback which comes from an external human agent, and $\mathcal{L}$ to refer to the probability of receiving feedback, i.e. a likelihood that an external human agent delivers guidance at some point.

In our work, we use probability of feedback $\mathcal{L} = 0.3$ and consistency of feedback $\mathcal{C} = 1$, the latter because in our case the interaction is provided from another simulated agent rather than a human trainer and this external agent has full knowledge about the task.

Figure 7 shows the training for this approach. Shown are the average number of actions performed in RL versus IRL for 100 episodes, where convergence speed in IRL is significantly improved by receiving instructions from the first trained agent in 30% of the time. In figure 7, the axes are just displayed until 100 episodes and 100 actions to highlight the behavior of both approaches in initial episodes.

VII. Conclusions and Future work

We have shown affordances and IRL to be an efficient method to improve the convergence speed of RL. We have presented a method that was able to successfully train a simulated agent and then to transfer this knowledge by IRL to a new agent which led to a reduction in the number of the required episodes during training as well as a reduction of the number of actions performed in each episode. The use of affordances also allowed to reduce the amount of iterations or needed episodes for training which is fundamentally important considering real scenarios where running through a large number of episodes would be impracticable.

As next step, we are going to consider alternative evaluations with different set-ups of parameters for getting our model as robust as possible. The probability of feedback $\mathcal{L}$ deserves special attention because it could possibly have an impact in scenarios where feedback comes from human trainers. In addition, a recently proposed method by Torrey and Taylor [23] where interaction is delivered at some specific points of the training could provide enhancements to our approach. Moreover, an important task involves to transfer our method to a real scenario where a humanoid robot will perform the actions given by a human trainer.

Acknowledgment

The first author gratefully acknowledges the support by *Universidad Central de Chile* and *CONICYT*.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 2nd printing edition, Springer, 2011.
[2] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems*, Springer Berlin Heidelberg, 2011.
[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, A Bradford Book, 1998.
[4] H. Ammar, M. Taylor, K. Tuyls, and G. Weiss, *Reinforcement Learning Transfer Using a Sparse Coded Inter-Task Mapping*, in Multi-Agent Systems Vol 7541 of LNCS, Springer Berlin Heidelberg, 2012, pp. 1–16.
[5] A. L. Thomaz, G. Hoffman, and C. Breazeal, *Real-Time Interactive Reinforcement Learning for Robots*, in Proceedings of AAAI Workshop on Human Comprehensible Machine Learning, 2005.
[6] W. Knox, B. Glass, B. Love, W. Maddox, and P. Stone, *How Humans Teach Agents*, in International Journal of Social Robotics, Vol. 4, Issue 4, 2012, pp. 409–421.
[7] J. Grizou, M. Lopes, and P.-Y. Oudeyer, *Robot Learning Simultaneously a Task and How to Interpret Human Instructions*, in Proceedings of ICDL-Epirob, 2013, pp. 1–8.
[8] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, *Learning Object Affordances: From Sensory-Motor Coordination to Imitation* in IEEE Transactions on Robotics, Vol. 24, No. 1, 2008, pp. 15–26.
[9] C. Wang, K. V. Hindriks, and R. Babuska, *Robot Learning and Use of Affordances in Goal-Directed Tasks*, in Proceedings of IROS International Conference, 2013, pp. 2288–2294.
[10] J. J. Gibson, *The Ecological Approach to the Visual Perception of Pictures*, in Leonardo, Vol. 11, No. 3, MIT Press, 1978, pp. 227–235.
[11] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, *Robot Learning From Demonstration by Constructing Skill Trees*, in The International Journal of Robotics Research, Vol. 31, No. 3, 2012, pp. 360–375.
[12] L. Rozo, P. Jiménez, and C. Torras, *A Robot Learning From Demonstration Framework to Perform Force-Based Manipulation Tasks*, in Intelligent Service Robotics, Vol. 6, No. 1, 2013, pp. 33–51.
[13] J. P. Bandera, J. A. Rodríguez, L. Molina-Tanco, and A. Bandera, *A Survey of Vision-Based Architectures for Robot Learning by Imitation*, in International Journal of Humanoid Robotics, Vol. 09, No. 1250006, 2012, pp. 1–40.
[14] W. B. Knox, P. Stone, and C. Breazeal, *Teaching Agents With Human Feedback: A Demonstration of the TAMER Framework*, in Proceedings of International Conference on Intelligent User Interfaces Companion, New York USA, ACM, 2013, pp. 65–66.
[15] A. L. Thomaz and C. Breazeal, *Asymmetric Interpretations of Positive and Negative Human Feedback for a Social Learning Agent*, in RO-MAN 2007 IEEE, 2007, pp. 720–725.
[16] W. B. Knox and P. Stone, *Reinforcement Learning From Human Reward: Discounting in Episodic Tasks*, in RO-MAN 2012 IEEE, 2012, pp. 878–885.
[17] G. A. Rummery and M. Niranjan, *On-Line Q-Learning using Connectionist Systems*, in Cambridge University Engineering Department Technical Report, 1994
[18] M. Kammer, T. Schack, M. Tscherepanow, and N. Yukie, *From Affordances to Situated Affordances in Robotics - Why Context is Important*, in Frontiers in Computational Neuroscience (Conference Abstract: IEEE ICDL-EPIROB 2011), Vol. 5(30), 2011.
[19] İ. Atıl, N. Dağ, S. Kalkan, and E. Şahin, *Affordances and Emergence of Concepts*, in Proceedings of 10th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, 2010, pp. 149–156.
[20] H. S. Koppula, R. Gupta, and A. Saxena, *Learning Human Activities and Object Affordances From RGB-D Videos*, in The International Journal of Robotics Research, Vol. 32(8), 2013, pp. 951–970.
[21] J. Kober and J. Peters, *Reinforcement Learning in Robotics: A Survey*, in Reinforcement Learning, Vol. 12, Springer Berlin Heidelberg, 2012, pp. 579–610.
[22] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. Thomaz, *Policy Shaping: Integrating Human Feedback With Reinforcement Learning*, in Advances in Neural Information Processing Systems, 2013, pp. 2625–2633.
[23] L. Torrey and M. Taylor, *Teaching on a Budget: Agents Advising Agents in Reinforcement Learning*, in Proceedings of AAMAS 2013, Richland, 2013, pp. 1053–1060.