Training Agents With Interactive Reinforcement Learning and Contextual Affordances

Francisco Cruz, Sven Magg, Cornelius Weber, and Stefan Wermter

***Abstract*—In the future, robots will be used more extensively as assistants in home scenarios and must be able to acquire expertise from trainers by learning through crossmodal interaction. One promising approach is interactive reinforcement learning (IRL) where an external trainer advises an apprentice on actions to speed up the learning process. In this paper we present an IRL approach for the domestic task of cleaning a table and compare three different learning methods using simulated robots: 1) reinforcement learning (RL); 2) RL with contextual affordances to avoid failed states; and 3) the previously trained robot serving as a trainer to a second apprentice robot. We then demonstrate that the use of IRL leads to different performance with various levels of interaction and consistency of feedback. Our results show that the simulated robot completes the task with RL, although working slowly and with a low rate of success. With RL and contextual affordances fewer actions are needed and can reach higher rates of success. For good performance with IRL it is essential to consider the level of consistency of feedback since inconsistencies can cause considerable delay in the learning process. In general, we demonstrate that interactive feedback provides an advantage for the robot in most of the learning cases.**

***Index Terms*—Contextual affordances, developmental robotics, domestic cleaning scenario, interactive reinforcement learning (IRL), policy shaping.**

I. Introduction

THERE has been considerable progress in robotics in the last years allowing robots to be successful in diverse scenarios, from industrial environments where they are nowadays established to domestic environments where their presence is still limited [1]. In domestic environments, tasks often require active human participation in order to execute the tasks more effectively. In particular in the simulated home scenario which we propose in this paper, a robot has to perform a task which consists of cleaning a table assisted by an external trainer giving different degrees of guidance.

Manuscript received September 23, 2015; revised December 16, 2015 and February 10, 2016; accepted March 16, 2016. Date of publication April 22, 2016; date of current version December 7, 2016. This work was supported in part by the Universidad Central de Chile, in part by the Comisión Nacional de Investigación Científica y Tecnológica Scholarship under Grant 5043, in part by the German Research Foundation DFG under Project CML (TRR 169), and in part by the Hamburg Landesforschungsförderungsprojekt CROSS.

The authors are with the Knowledge Technology Group, Department of Informatics, University of Hamburg, Hamburg 22527, Germany (e-mail: cruz@informatik.uni-hamburg.de; magg@informatik.uni-hamburg.de; weber@informatik.uni-hamburg.de; wermter@informatik.uni-hamburg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2016.2543839

In developmental robotics [2] 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 (RL) [3], [4]. In our scenario, the simulated robot has no previous knowledge on how to perform the task and it can learn only through interaction with and reward from the environment. Therefore, the apprenticeship process is carried out with RL.

RL is a learning approach supported by behavioral psychology where an agent uses sequential decisions to interact with its environment trying to find an optimal policy to perform a particular task. In every time step, the agent selects an action to be performed reaching a new state and obtains either a reward or a punishment. It attempts to maximize the collected reward over time by choosing the best action in a given state. Therefore, the problem is reduced to finding a proper policy that allows to associate actions to states in order to get maximal future reward [5].

RL has demonstrated to be a very useful learning approach, but one problem for RL agents is often the excessive time spent during the learning process [6], mainly due to large and complex state spaces which lead to excessive computational costs to find a suitable policy [7]. There are different approaches that attempt to speed up RL. Among them, interactive RL (IRL) involves an external trainer who provides some instructions on how to improve the decision-making [8], [9].

A promising alternative method to improve convergence speed is the use of affordances [10], where 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 [11]. Affordances limit the number of meaningful actions in some states and can reduce the computational complexity of RL.

Contextual affordances are a generalization of Gibson's affordance concept which has recently been used successfully in robotics [12]. We implement them by a deep multilayer perceptron (DMLP) allowing us to estimate either the robot's next state or whether the affordance is temporally unavailable. In our approach, we integrate IRL and contextual affordances to enhance the performance of classic RL methods, demonstrating better results when introducing the affordance concept to classic RL as well as combining IRL with instructions coming from a previously trained agent.

We previously presented a method which allows improving the speed of convergence of an RL agent using affordances and interaction [13]. The trainer agent provided feedback with a probability of 30%. The results showed a reduction in the number of required episodes during the training as well as a reduction in the number of actions performed in each episode. However, in our previous work, affordances were given as prior knowledge and were not learned automatically, which will be addressed in this paper.

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 the external trainer interaction. Next, we provide a description of the concepts of affordance and contextual affordance and explain how they are used in our approach. We then define our robotic agent scenario for a domestic task and describe our experimental set-ups to speed up RL with both, interactive instructions and contextual affordances. Moreover, we show and compare our main results with both RL and IRL approaches. Finally, we present our main conclusions and describe future research.

II. Reinforcement Learning and Interactive Feedback

To autonomously explore the environment is one of the first developing behaviors for a human. An infant is constantly exploring its surroundings and learning from it most of the time without the need of a trainer to instruct it on how to perform a task. In a simplified similar manner, RL attempts to maximize received reward in a given scenario through interaction between an agent and its environment. Fig. 1 shows the basic elements and structure of RL. To transit from one state to another, the agent performs actions obtaining either a reward or a punishment from the environment which may not be delivered at each state transition. Such actions are selected according to a policy π, which in psychology is called a set of stimulus–response rules or associations [14]. Thus, the value of taking an action a in a state s under a policy π is denoted $q^{\pi}(s,a)$ which is also called the action-value function for a policy π.

In essence, to solve an RL problem means to find a policy that collects the highest reward possible over the long run. If there exists at least one policy which is better or equal than all others this is called an optimal policy. Optimal policies are denoted by π^* and share the same optimal action-value function which is denoted by q^* and defined as

$$q^*(s,a) = \max_{\pi} q^{\pi}(s,a). \tag{1}$$

This optimal action-value function can be solved through the Bellman optimality equation for q^* as follows:

$$q^*(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right] \tag{2}$$

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, and 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

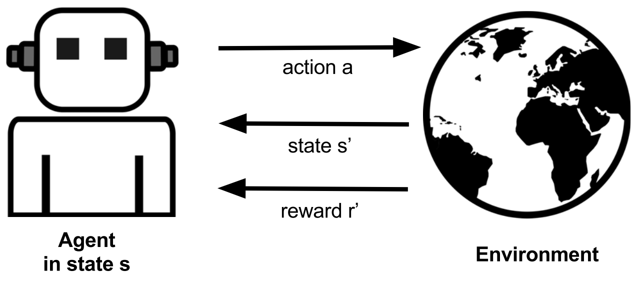


Fig. 1. RL between the agent and the environment. The agent performs action a in state s obtaining reward r' and reaching the next state s'.

reaching the state s'. The parameter γ is known as discount rate and represents how influential future rewards are [5].

Although autonomous learning is possible for an apprentice using RL, a prominent strategy to improve the speed of convergence is to use an external trainer to provide guidance in specific states during the learning process. Early research on this topic [15] shows that external guidance plays an important role in learning tasks, performed by both humans and robots, leading to a decrease of the time needed for learning. Furthermore, in large spaces where a complete search through the whole search space is not possible, the trainer may lead the apprentice to explore more promising areas at early stages as well as help to avoid getting stuck in suboptimal solutions.

In robotics there are different strategies of interaction between an agent and an external trainer for developing joint tasks, such as learning by imitation [16], demonstration [17]–[19], and feedback [20]–[22]. In particular in learning by feedback two main approaches are distinguished: 1) policy and 2) reward shaping. Whereas in reward shaping an external trainer is able to evaluate how good or bad performed actions by the RL agent are [21] and [23], in policy shaping the action proposed by the RL agent can be replaced by a more suitable action chosen by the external trainer before it is executed [20]. When the external trainer does not give feedback, acceptance of the action a or reward r is assumed. In both cases, an external trainer gives interactive feedback to the apprentice agent to encourage it to perform certain actions in certain states to reach a better policy leading to faster performance. Novel strategies can emerge from mixing both, namely, the decision on performing action a and manipulating the received reward r as well.

Fig. 2 shows the policy shaping 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 [24]. The reward shaping approach is shown in Fig. 3. 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 was. Examples of this approach were developed in [20] and [23].

In a domestic scenario a robotic agent is expected to work with humans as external trainers. There exist asymmetries when humans quantify a reward including sometimes feedback about the past and also about future intentions [20], [24].

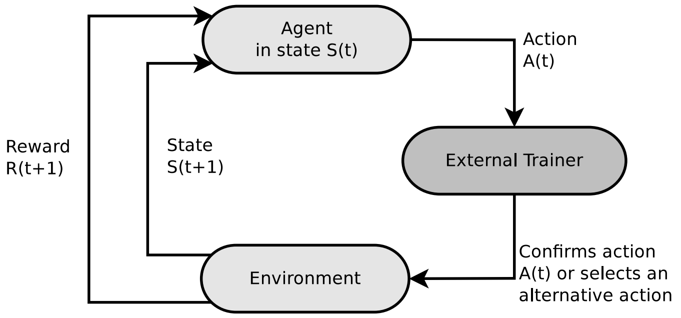


Fig. 2. Policy shaping feedback approach for interaction between a robotic agent and an external trainer. In this case, the external agent is able to change a selected action to be performed in the environment.

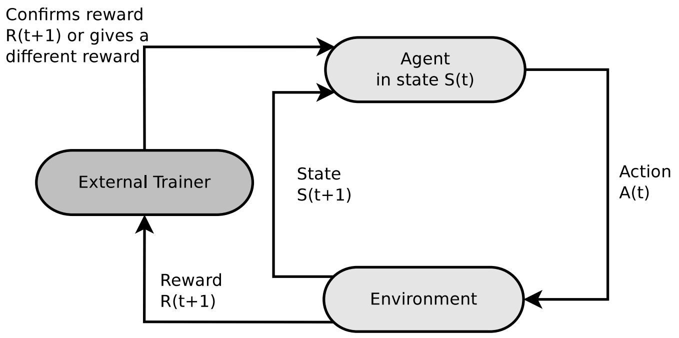


Fig. 3. Reward shaping feedback approach for interaction between a robotic agent and an external trainer. In this case, the external agent is able to modify the proposed reward.

Hence, we decided to use the policy shaping method in this paper, shown in Fig. 2. For this, we create a simulated environment where an agent learner has previously been trained using classic RL and then this learner becomes the external trainer. It therefore has full knowledge about all possible actions and delivers it to a second robot which is trained with IRL. The artificial trainer is used to have better control of the feedback compared with a human trainer. Nevertheless, diverse information sources can be employed to obtain feedback from, for instance, a person, another robot, or any other artificial system.

In our scenario it is desired to keep the rate of interaction with an external trainer as low as possible; otherwise, with a high rate of interaction, RL becomes supervised learning. Also, the consistency or quality of the feedback should be considered to determine whether learning is still improving given that the external trainer could also make mistakes [25].

III. Use of Contextual Affordances

Affordances are often seen as opportunities for action of an agent (a person, an animal, a robot, or an organism). The original concept comes from cognitive psychology and was proposed by Gibson [10]. For instance, a cup and a sofa afford different actions to a person who is able to grasp the cup and sit down on the sofa, but cannot do it the other way around. Thus, an agent is able to determine some object affordances, i.e., the caused effect of performing a specific action with an object.

Horton *et al.* [12] distinguished three essential characteristics of an affordance:

1) the existence of an affordance is associated with the capabilities of an agent;
2) an affordance exists regardless whether the agent is able to perceive it or not;
3) affordances do not change, unlike necessities or goals of an agent.

In Gibson's book many diverse examples are given but no concrete, formal definition is provided. Even nowadays, we find marked differences among cognitive psychologists about the formal definition of affordances [12], [26] and these discrepancies could even be stronger between them and artificial intelligence scientists [27], [28].

In developmental robotics, affordances are aligned with basic cognitive skills which are acquired on top of previous skills by interacting with the environment [29]. It is expected that domestic service robots learn, recognize, and apply some social norms in the same way as humans do. Commonly these social rules are learned by interaction and socialization with other agents of the group. In this regard, an object can be used in a restricted manner not considering all its action opportunities but only socially accepted actions. These constrains of use are usually shaped by the group norms and are called functional affordances [30] which also lead to a reduced action space. Such a human-like behavior is an important issue in developmental robotics [31].

In the literature we can find different approaches for learning affordances in robotics; for instance, Lopes *et al.* [32] addressed the imitation learning problem using affordance-based action sequences. Moldovan *et al.* [29] extended the affordance model allowing the robot to work with a second object using an enlarged Bayesian network to represent affordances. Other approaches have been also surveyed in [12] and [28]. Nevertheless, all aforementioned approaches do not consider the agent's state to anticipate the effect of an action.

In the following sections, we present a formal computational definition based on the original concept of Gibson and then propose an extension considering an additional context variable.

A. Affordances

Affordances have been particularly useful to establish relationships between actions performed by an agent with available objects. We use them in a way to represent object/action information. They represent neither agent nor object characteristics, but rather the characteristics of the relationship between them. Montesano *et al.* [33] defined an affordance as the relationship between an object, an action, and an effect as the triplet affordance := <object, action, effect> which encodes relationships between its components. Hence, it is possible to predict the effect using objects and actions as domain variables, that is

$$\text{effect} = f(\text{object}, \text{action}). \tag{3}$$

Fig. 4 shows the relationship between the previous components, where objects are entities which the agent is able to

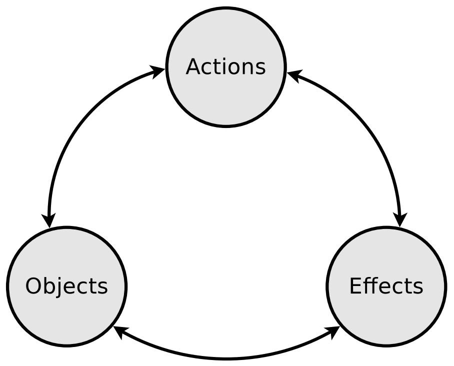


Fig. 4. Affordances as relations between objects, actions, and effects. Objects are entities which the agent is able to interact with, actions represent the behavior that can be performed with the objects, and effects are the results caused by applying an action [33].

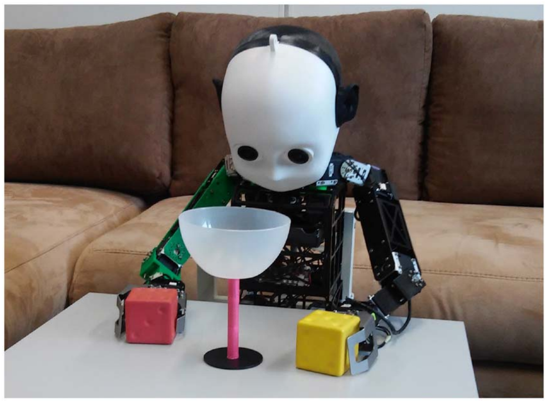


Fig. 5. Cup affords grasping as long as the agent's current state allows this action to be performed. In the shown scenario it is not feasible to use that affordance since both hands are already occupied, but once one hand is free the affordance can be utilized again.

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 involving an object [33], [34].

It is also important to note that the object in (3) can also be a place or a location, for instance, a hill affords climbing. From here onward, we employ the term object to refer to the affordance component but we consider also locations.

B. Contextual Affordances

If an affordance exists and the agent has knowledge and awareness of it, the actual, next step is to determine if it is possible to utilize it considering the agent's current state. For instance, let us consider the following scenario: a cup affords grasping, as does a die, but in case we have an agent with both hands occupied with dice (e.g., with one die in each hand), then the agent cannot grasp the cup anymore or in other words, the affordance is temporarily unavailable. This situation is depicted in Fig. 5. This does not mean that the affordance does not exist, to the contrary, the affordance is still present but it cannot be used by the agent in that particular situation due to its current state.

Kammer *et al.* [35] proposed to consider the dynamics in the environment in which the object was embedded rather

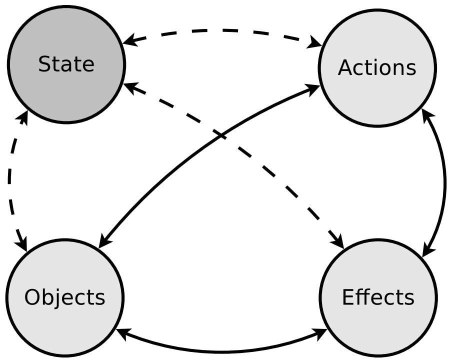


Fig. 6. Contextual affordances as relations between state, objects, actions, and effects. The state is the agent's current condition and different effects could be produced for different occasions.

than the agent's dynamic state. The awareness of this extra variable is called situated affordances. Even though a formal definition was provided, neither applications nor results are shown in their work. Nevertheless, we use the same concept to address the problem when the agent's state is dynamic. Thus, we propose a model where the current state of an agent is also considered for the effects of an action performed with an object, we call this contextual affordance. In this case, the previous triplet is now extended to contextualAffordance := <state, object, action, effect> and to predict the effect we consider the following function:

$$\text{effect} = f(\text{state}, \text{object}, \text{action}). \tag{4}$$

For instance, given an agent performing the same action a with the same object o, but from a different agent's state $s_1 \neq s_2$: when action a is performed, different effects $e_1 \neq e_2$ could be generated, since the initial states s_1 and s_2 are different. It is unfeasible to establish differences in the final effect when we utilize affordances to represent it, because $e_1 = (a, o)$ and $e_2 = (a, o)$. Hence, to deal with the current states $s_1 \neq s_2$, an agent must distinguish each case and learn them at the same time utilizing contextual affordances defined by $e_1 = (s_1, a, o)$ and $e_2 = (s_2, a, o)$, establishing clear differences between the final effects.

Fig. 6 shows the relationship between object, action, effect, and the agent's current state. We use contextual affordances to provide knowledge about actions that lead to undesirable or failed states from which it is not possible to reach the goal. Therefore, the action space is reduced by avoiding these states. In our approach, the set of possible actions is filtered for every state that the agent is in. This contextual affordance model allows us to determine beforehand when it is possible to apply an affordance using an artificial neural network (ANN) to learn the relationship with the state, the action, and the object as inputs and the effect as output. In Section V, we will describe in detail the neural network architecture used to anticipate the effect.

IV. Domestic Cleaning Scenario

We have designed a simulated domestic scenario focused on cleaning a table, which can be learned by a robot that is

TABLE I
LIST OF DEFINED OBJECTS, LOCATIONS, AND ACTIONS FOR CLEANING-TABLE SCENARIO

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

supported by an external agent to reduce the training time. For 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. On the table, there is an additional object like a cup.

Furthermore, three locations are defined, left, right, and home, representing the left and right side on the table within the reachable area for the robot's arm, and the location home which is the initial position of the robot's hand and also the storage place of the object sponge.

In this scenario four actions are possible: 1) get an object, which allows the robotic agent to pick up the object near the current robot hand location; 2) drop an object, which allows the robot to put down the object that is currently kept in its hand; 3) go to a location, which allows to move the robot hand to any defined position; and 4) clean, which cleans only the current location where the hand is positioned. Table I shows a summary of objects, locations, and actions defined for this domestic cleaning scenario, which has been developed in a simulated robot environment.

The table-cleaning task is carried out by a robot in a simulated environment using the V-REP simulator [36]. All actions are performed using only one arm and one effector. Fig. 7 shows an example of the scenario where one robot, which has already learned the task, teaches a second robot how to clean the table using the sponge. We did not focus on investigating grasping since the main aim in this paper is to learn the right sequence quickly. Nevertheless, for reaching the defined locations we employed direct planning [37] and for grasping inverse kinematics as a support for low-level control.

For instance, 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 above home with both sides of the table clean. In this context, an episode is defined as one attempt to reach the goal. The following example shows an episode to complete the task successfully: get sponge, go right, clean, go home, drop sponge, go left, get cup, go right, drop cup, go home, get sponge, go left, clean, go home, and drop sponge. The example shows, for one situation, the shortest sequence to reach the final state, therefore, the minimum number of actions to complete the task is $|A_{\min}| = 15$.

To implement the described scenario we developed a state machine with one final state; each state is obtained considering the combination of the following four variables:

1) the robot's hand position;
2) the object held in the hand, or free;
3) the position of the cup;

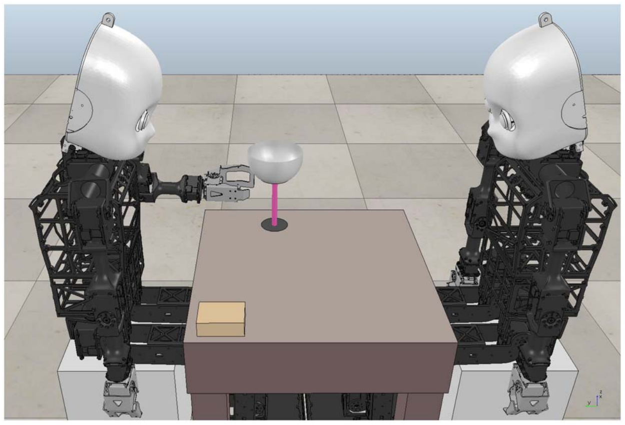


Fig. 7. Example of the simulated scenario where robots perform the actions in the environment which is created in the V-REP simulator. The cleaning scenario consists of three locations, two objects, and four actions.

4) the current condition of every side of the table surface, that is, whether the table has already been cleaned or not.

Nevertheless, from certain states the agent can perform actions which lead to a failed state from where it is not possible any more to complete the task. These actions include getting an object when the robot's hand is occupied, to lose either the cup or the sponge due to an incorrect drop, or cleaning a section of the table where the cup is also placed.

Let us consider the set of states S and the set of actions A. Given one sequence of states $\psi_s = \{s_t, s_{t+1}, s_{t+2}, \ldots, s_n\}$ with $s_i \in S$ and one sequence of actions $\psi_a = \{a_t, a_{t+1}, a_{t+2}, \ldots, a_n\}$ which leads to ψ_s with $a_i \in A$. Then $s_i = f(s_{i-1}, a_{i-1})$ for $0 < i \leq n$. Now, let $\Psi_A(s_t)$ be the set of all possible ψ_a from a state $s_t \in S$. If $\nexists\ \psi_a \in \Psi_A(s_t)$ which produces a $\psi_s | s_f \in \psi_s$ with s_f the final state $\implies s_t$ is a failed state.

For instance, let us assume that the robot has just performed the action get cup and the state is defined by $s_t =$ `<left, cup, left, (dirty, dirty)>` according to the four previous state variables; therefore the cup is held in its hand. If the robot then cleans the left section of the table with the cup in its hand instead of a sponge, it may shatter the cup; hence, it is not feasible to finish the cleaning task from the next state s_{t+1}.

At first, we ran a classic RL algorithm to confirm that failed states are not a problem as such; they can be handled and controlled by giving punishment (or negative reward). In this case, the agent is discouraged to perform that action from the same state in the future. Nevertheless, a more suitable strategy is to consider the use of affordances which have been shown to improve the convergence speed of learning algorithms [38], [39].

In this paper, we are interested in reducing the needed actions of an episode to reach a reasonable performance by using contextual 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 environment one

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 CLEANED

Side condition	**Cup position**	**Hand position**	**Held object**	**Nr.**
dirty-dirty	left	home	free	0
			sponge	1
		left	free	2
			sponge	3
			cup	4
		right	free	5
			sponge	6
	right	home	free	7
			sponge	8
		left	free	9
			sponge	10
		right	free	11
			sponge	12
			cup	13
...	...	...	...	...
clean clean-	left or right	left	sponge	42
		right	sponge	43
		home	sponge	44
			free	45

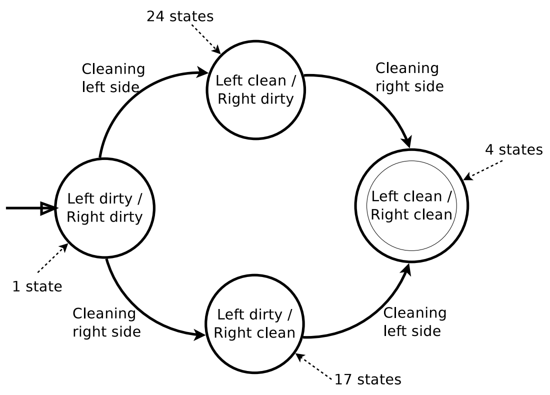


Fig. 8. Given that the cup is on the left side, there are two feasible paths for reaching the final state from the initial state. The lower path would include 17 possible different states and consists of cleaning first the empty side of the table, and then moving the cup in order to clean the second 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 each case, the final state is reached involving different numbers of intermediate states. The ending sequence of the task contains four states in which the robot returns its hand to the home location and drops the sponge.

cannot afford to run excessive episodes until reaching a suitable policy. Given the defined actions, objects and locations, we are able to set the presence of four different contextual affordances which allow us to determine whether objects are graspable, droppable, movable, or cleanable according to the robot's current state.

Considering contextual affordances, the scenario consists of 46 regular states, some of which are shown in Table II where the initial state is the number 0 and the final state is the number 45. The states between numbers 14 and 27, and the states between 28 and 41 are not shown since they are the same as the first 14 states apart from the side conditions which are clean-dirty and dirty-clean, respectively. Fig. 8 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, successful state. As we already stated above, the shortest path is composed of 15 actions for reaching the final state.

V. EXPERIMENTAL SET-UP

This section describes the experimental set-up considering aspects such as our IRL approach and how contextual affordances are implemented with an ANN architecture to estimate the robot's next state.

A. *Interactive Reinforcement Learning Approach*

Since RL is used, most of the time the robot performs actions autonomously by exploring the environment unless guidance is delivered by the previously trained robot which already has full knowledge on how to carry out the task. The apprentice robot takes advantage of this advice in these periods during a learning episode and performs the suggested actions to complete the task with fewer actions.

In the learning algorithm, to solve (2), we allow the robot to perform actions considering transitions from state–action pair to state–action pair rather than transitions from state to state only. Therefore, we implement the on-policy method state-action-reward-state-action (SARSA) [40] to update every state–action value according to

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\big] \tag{5}$$

where s_t and s_{t+1} are the current and next state, respectively, a_t and a_{t+1} are the current and next action, Q is the value of the action–state pair, r_{t+1} the collected reward, α is the learning rate, and γ the discount factor. The reward function delivers a positive reward equal to 1 to the agent every time that it reaches the final state, and a punishment or negative reward equal to -1 every time that a failed state is reached to discourage accessing this state in the future again. We also apply a small negative reward in all other states to encourage the agent to choose shorter paths toward the final state. Equation (6) shows the reward function

$$r(s) = \begin{cases} 1 & \text{if } s \text{ is the final state} \\ -1 & \text{if } s \text{ is a failed state} \\ -0.01 & \text{otherwise.} \end{cases} \tag{6}$$

The parameters used in (5) are empirically set to $\alpha = 0.3$ and $\gamma = 0.9$. Furthermore, we use the ϵ-greedy method for action selection with $\epsilon = 0.1$. Therefore, most of the time the next action a_t is determined as follows:

$$a_t = \underset{a \in A}{\operatorname{argmax}}\ Q(s_t, a) \tag{7}$$

where s_t is the current state at time t, a is an action, and A corresponds to the set of all actions. Algorithm 1 shows this action selection method, whereas Algorithm 2 presents the classic RL approach.

Algorithm 1 SELECTACTION Method Used in the Classic RL Approach

```
Input: Agent's current state $s_t$
Output: Next action $a_t$ to perform
1: function SELECTACTION($s_t$)
2:     if $rand(0,1) < \epsilon$ then
3:         $a_t \leftarrow$ choose any random action $a$ from $A$
4:     else
5:         $a_t \leftarrow \underset{a \in A}{\operatorname{argmax}}\, Q(s_t, a)$
6:     end if
7:     return $a_t$
8: end function
```

Algorithm 2 Classic RL Approach With the On-Policy Method SARSA

```
1: Initialize $Q(s,a)$ arbitrarily
2: for each episode do
3:     Choose an action using $a_t \leftarrow$ SELECTACTION($s_t$)
4:     repeat
5:         Take action $a_t$
6:         Observe reward $r_{t+1}$ and next state $s_{t+1}$
7:         Choose an action using $a_{t+1} \leftarrow$ SELECTACTION($s_{t+1}$)
8:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
9:         $s_t \leftarrow s_{t+1}$
10:        $a_t \leftarrow a_{t+1}$
11:    until $s$ is terminal
12: end for
```

Using contextual affordances we slightly modify the policy in the action selection shown in (7) as in the following expression:

$$a_t = \underset{a \in A_s}{\operatorname{argmax}}\, Q(s_t, a) \tag{8}$$

where s_t is the current state in time t, a is an action, and A_s corresponds to a subset of available actions in the current state s_t. The subset is determined based on the contextual affordances [see (4) and Fig. 6]. In this regard, it is possible to anticipate whether the action can be performed with an object in a particular state which is called effect. Algorithm 3 shows the action selection method used during RL with contextual affordances where the subset A_s is created by observing the output of the ANN and populated with those which return a valid state value.

We use the advise method parameters [25] for interaction, i.e., probability of feedback $\mathcal{L}$ and consistency of feedback $\mathcal{C}$. Algorithm 4 shows the method used when advice is required. The higher the values of $\mathcal{C}$, the more often a good advice is given. In this context, the best advice is obtained from the subset of actions A_s considering the highest state–action pair value whereas the worst action advised is also taken from A_s but considering the lowest state–action pair value. In this case, we also create the subset A_s from available actions which lead to valid state values.

Algorithm 3 SELECTACTIONWITHAFFORDANCES Method Used in the RL With Contextual Affordances Approach

```
Input: Agent's current state $s_t$
Output: Next action $a_t$ to perform
1: function SELECTACTIONWITHAFFORDANCES($s_t$)
2:     Create subset $A_s$
3:     if $rand(0,1) < \epsilon$ then
4:         $a_t \leftarrow$ choose any random action $a$ from $A_s$
5:     else
6:         $a_t \leftarrow \underset{a \in A_s}{\operatorname{argmax}}\, Q(s_t, a)$
7:     end if
8:     return $a_t$
9: end function
```

Algorithm 4 GETADVICE Method Used in the IRL Approach With Contextual Affordances

```
Input: Agent's current state $s_t$
Output: Next action $a_t$ to perform
1: function GETADVICE($s_t$)
2:     Create subset $A_s$
3:     if $rand(0,1) < \mathcal{C}$ then
4:         $a_t \leftarrow$ best advice from $A_s$
5:     else
6:         $a_t \leftarrow$ worst advice from $A_s$
7:     end if
8:     return $a_t$
9: end function
```

Algorithm 5 shows the IRL approach using contextual affordances and interaction. The conditional statement in line 8 represents the fact that the external trainer delivers advice and changes the next action a_{t+1} by calling the method GETADVICE shown in Algorithm 4 where contextual affordances are used.

Each set-up was carried out 100 times using the obtained average values for the subsequent analysis. The Q-values were initialized randomly using a uniform probability distribution between 0 and 1.

B. Contextual Affordances With Deep Neural Architecture

It has been shown that a multilayer feedforward neural network (MLP) with only one hidden layer and a sufficient number of neurons in this layer is able to approximate any continuous nonlinear function with arbitrary precision [41]–[43]. Nevertheless, MLPs with only one hidden layer may need an exponential number of neurons in order to reach a particular degree of precision [44]. Besides that, in the last years deep neural architectures have become a topic of interest within the research community due to their distributed and sparse representation [45].

Therefore, to learn the relationship between inputs and outputs in contextual affordances we implemented a DMLP which is a feedforward network with more than one hidden layer as proposed in [46]. As inputs we use the agent's current state, the action, and the object giving a code number to every state and every action plus object, the two latter

Algorithm 5 IRL Approach Using Contextual Affordances and Interaction

```
1:  Initialize $Q(s, a)$ arbitrarily
2:  for (each episode) do
3:      Choose an action using
          $a_t \leftarrow$ SELECTACTIONWITHAFFORDANCES($s_t$)
4:      repeat
5:          Take action $a_t$
6:          Observe reward $r_{t+1}$ and next state $s_{t+1}$
7:          $a_{t+1} \leftarrow$ SELECTACTIONWITHAFFORDANCES($s_{t+1}$)
8:          if $rand(0, 1) < \mathcal{L}$ then
9:              Change action $a_{t+1} \leftarrow$ GETADVICE($s_{t+1}$)
10:         end if
11:         $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
12:         $s_t \leftarrow s_{t+1}$
13:         $a_t \leftarrow a_{t+1}$
14:     until $s$ is terminal
15: end for
```

TABLE III
LIST OF CODES FOR ACTIONS AND OBJECTS TOGETHER

Action	Object	Code
get	sponge	000
	cup	001
drop	sponge	010
	cup	011
go	home	100
	left	101
	right	110
clean	sponge	111

together. Table III shows the encoded action numbers; as output we use the next state or -1 to indicate that it is not feasible to perform the action with the object in the current state.

To get data we initially coded all possible failed states and used a previous run of autonomous RL to collect actions that lead to such failed states. The DMLP is employed as an associative memory to map states, actions, and objects to the subsequent effect, as in (4). Therefore, the neural network is able to store not only failed states but also transitions when an action leads to another valid state. The neural network training is carried out in an offline fashion before the IRL execution with the previous collected data.

The final architecture consists of 46 neurons in the first hidden layer and eight neurons in the second one. Both hidden layers have sigmoid transfer functions and the output layer has one neuron with a linear transfer function, as shown in Fig. 9. The number of neurons selected in every hidden layer is empirically determined related to our scenario and representing the number of states and actions.

A general problem during the training process of a deep neural network is the vanishing gradient. For first order gradient-based methods, a second problem of getting stuck in local minima can arise due to the error surface possessing large plateaus [47]. To overcome these issues, we use Nguyen–Widrow weight initialization [48] and the second

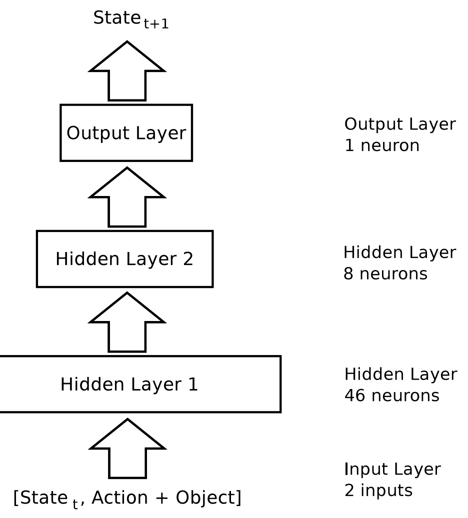


Fig. 9. DMLP utilized to determine contextual affordances. Hidden layers use sigmoid transfer functions and the output layer use a linear transfer function.

order training method Levenberg–Marquardt due to better performance shown in [49].

VI. SIMULATIONS AND RESULTS

To carry out this experiment, first, we used the classic RL approach to train a robot for reaching the final state. Afterward, we introduced contextual 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 a previously trained robot that sometimes showed which action to choose in a specific state. These three methods in detail will be explained and their results shown in the next sections.

A. Training the Agent Using Classic RL

In this first step, simulations are performed to train the first agent with the SARSA algorithm (5) using the reward function shown in (6).

Due to the large number of states, and since many of them are actually failed states, the agent needs more than 400 episodes of training until reaching at least once the final state in 100 attempts. Fig. 10 shows the average number of actions involved in every episode until reaching the final state with green crosses. Here, the number of actions is only shown when the final state is reached; hence, in the episodes where no cross is shown, only failed states were reached. In the first half of the training the final state is reached just a few times, but in the last part the agent becomes more successful and furthermore, in these cases close to 15 actions are being performed which in fact is the minimal number of possible actions as mentioned above. The dashed green line shown in Fig. 10 with a different scale on 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, although it only

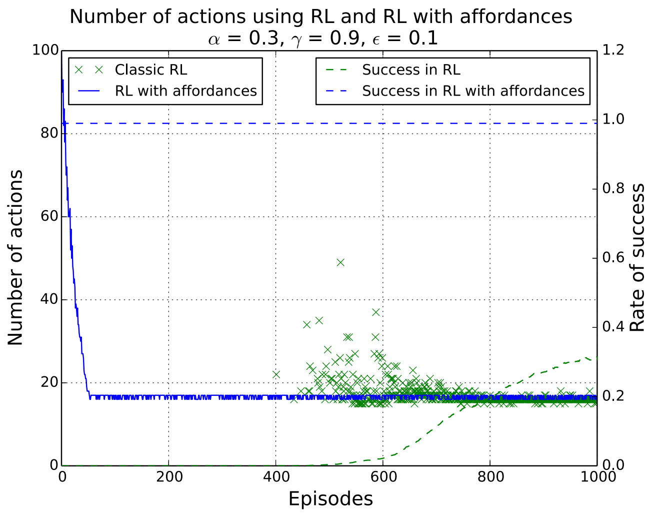


Fig. 10. Average number of actions needed for reaching the final state for classic RL (green) and RL with contextual affordances (blue) over 100 runs in 1000 episodes. For classic RL, the average number of actions is shown as a green cross only if at least one run was successful. Dashed lines show the rate of success of runs that have reached the final state, which is always 1 in case of RL with contextual affordances since failed states cannot be reached anymore. Rate of success was smoothed by a moving average with window size 50.

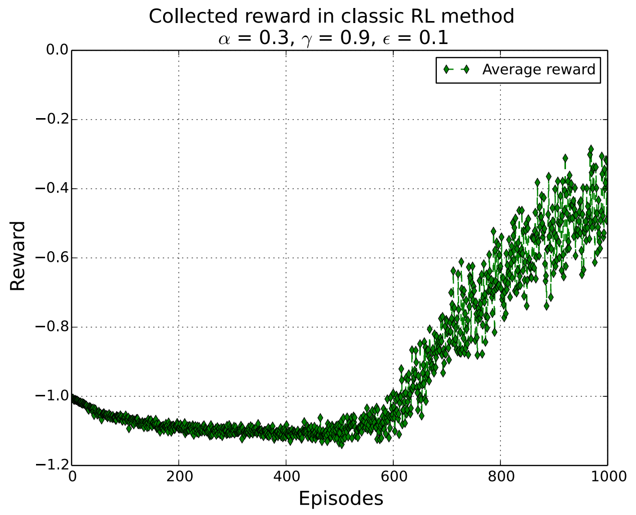


Fig. 11. Average collected reward over 100 runs using classic RL in 1000 episodes. The collected reward starts at -1 which means the robot failed on performing the cleaning task most of the time immediately. From there onward and until approximately 600 episodes the robot still mostly fails the task and then completes it more and more often.

reaches success rates of 35%. This clearly shows the difficulty to obtain a stable behavior by RL and the corresponding long training times.

The average collected reward over 100 runs in 1000 episodes is shown in Fig. 11. It is possible to see that the reward curve starts with values of -1 which means that in the beginning the robot fails the task immediately and up to 600 episodes later is still failing most of the time. However, after 600 episodes the robot is able to finish the task more regularly and thus increasing its collected reward.

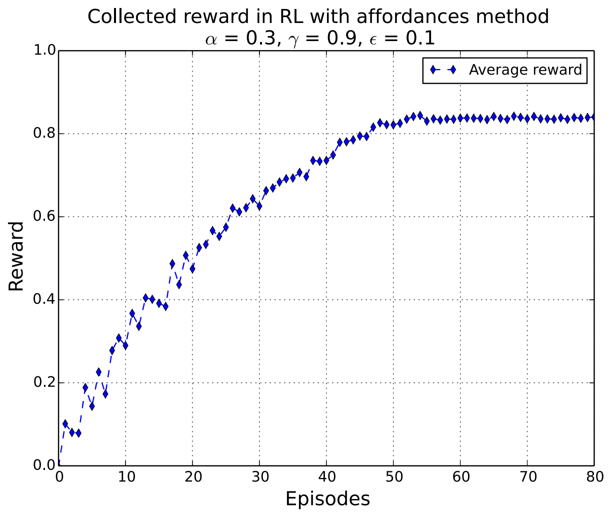


Fig. 12. Average collected reward over 100 runs using RL with contextual affordances in 80 episodes. The collected reward starts already near to 0 since in the beginning the robot does not have knowledge on how to perform the task but the final average reward is much bigger than in the previous case since no failed state is reached anymore.

B. Training the Agent Using RL With Contextual Affordances

As mentioned in the previous section, excessive episodes are 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 these quantities of episodes in a real scenario. Therefore, we decided to explore the benefit of contextual affordances which are implemented to reduce the valid action space for the agent by avoiding failed states. Using this approach, we manage to reduce the number of episodes considerably, i.e., we need less than 100 episodes to obtain a stable behavior and an average number of performed actions close to the minimum. Fig. 10 shows the number of actions in each episode with this set-up.

Whereas in the classic RL method the probability of success is still low in the first episodes of training, in this set-up no episode ends in a failed state because of the use of contextual affordances since an episode can only end when the agent reaches the final state (see dashed blue line in Fig. 10), which also produces considerably lower variation in comparison with the preceding method.

Fig. 12 shows the average collected reward over 100 runs in only 80 episodes to highlight the behavior in the first part of the training. It can be seen that the reward curve starts with values close to 0 (-0.0024 in the first episode) since in the beginning the robot needs many intermediate actions until completing the task but around 60 episodes later the robot is able to finish the task performing a number of actions near to the minimum and increasing the average collected reward.

C. Training the Second Agent Using IRL With Contextual Affordances

Once a first agent has been trained, a second agent is trained with an IRL approach that allows manipulating selected actions as shown in Fig. 2. In this method, the external trainer that provides feedback is the trained agent which already has

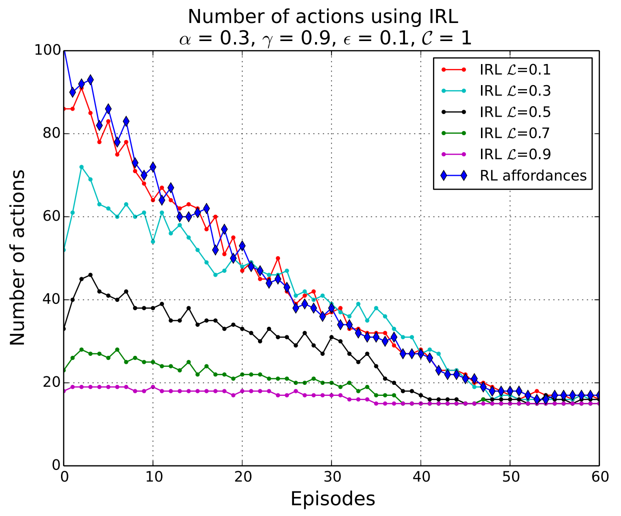


Fig. 13. Average number of actions needed for reaching the final state for RL with contextual affordances approach (blue diamonds) and IRL approach with different probabilities of interaction $\mathcal{L}$ and a fixed probability of consistency $\mathcal{C} = 1$ over 100 runs. The agent takes advantage of probabilities of interaction as small as $\mathcal{L} = 0.3$ by reducing the total number of performed actions.

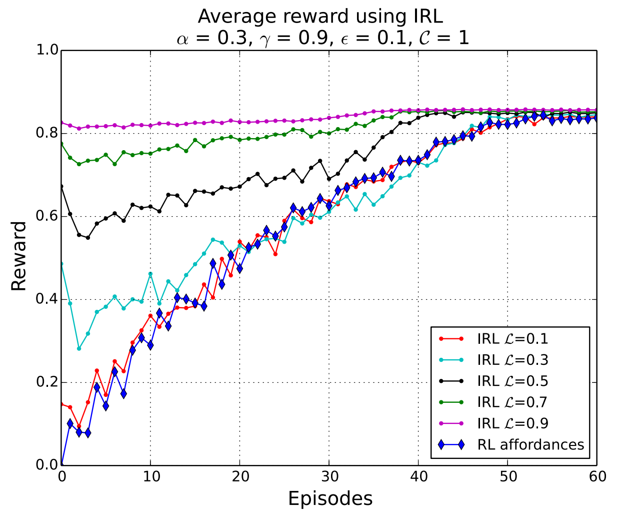


Fig. 14. Average collected reward over 100 runs for RL with contextual affordances approach (blue line) and IRL approach with different probabilities of interaction $\mathcal{L}$ and a fixed probability of consistency $\mathcal{C} = 1$. After 50 episodes all approaches reach a reward over 0.8.

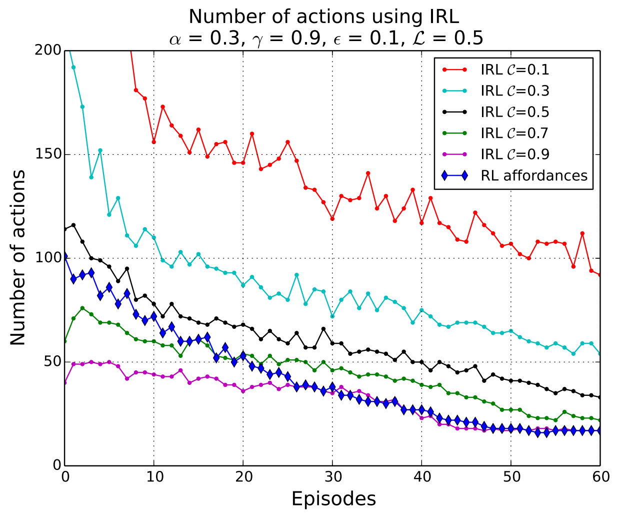


Fig. 15. Average number of actions needed for reaching the final state for RL with contextual affordances approach (blue diamonds) and IRL approach with different probabilities of consistency $\mathcal{C}$ and a fixed probability of interactions $\mathcal{L} = 0.5$ over 100 runs. The agent takes advantage of probabilities of interaction larger than $\mathcal{C} = 0.5$ by reducing the total number of performed actions as small as RL with contextual affordances approach.

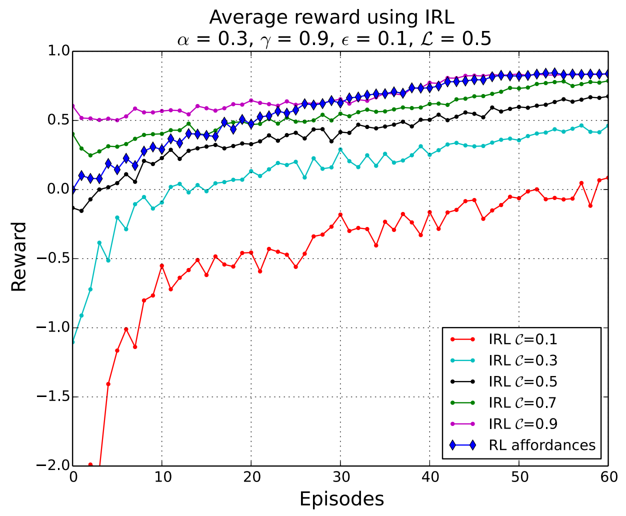


Fig. 16. Average collected reward over 100 runs for RL with contextual affordances approach (blue line) and IRL approach with different probabilities of consistency $\mathcal{C}$ and a fixed probability of interaction $\mathcal{L} = 0.5$. The final reward in all cases is less than RL with contextual affordances approach, nevertheless with probabilities of consistency over 0.5 is observed a similar behavior than a perfect trainer.

knowledge about the task to be performed. Furthermore, we base the interaction model on the advise method [25] which uses 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 this paper, we tested diverse values for $\mathcal{L}$ and $\mathcal{C}$ to investigate the influence within the learning process in terms of performed actions and collected reward. Fig. 13 shows the average number of performed actions with $\mathcal{L} \in [0.1, 0.9]$ and $\mathcal{C} = 1$. As a reference, the number of performed actions with RL using contextual affordances is shown with blue diamonds which is equivalent to have $\mathcal{L} = 0$. It is observed that even with a probability of feedback as small as $\mathcal{L} = 0.3$ the agent can improve its performance especially in the first episodes. Moreover, Fig. 14 shows the average collected reward by the agent over episodes for different values of $\mathcal{L}$.

Afterward, we explored the learning behavior with different values for the consistency of feedback $\mathcal{C}$. The higher the consistency, the more accurate the advice which means that fewer mistakes are made during the learning process. We also use contextual affordances but in this case the worst guidance is advised as shown in Algorithm 4 and as a result a bad advice is selected among the actions which still do not lead to a failed state.

To investigate the consistency of feedback $\mathcal{C}$, we fixed the average probability of feedback to $\mathcal{L} = 0.5$ and then perform experiments with different consistency $\mathcal{C} \in [0.1, 0.9]$.

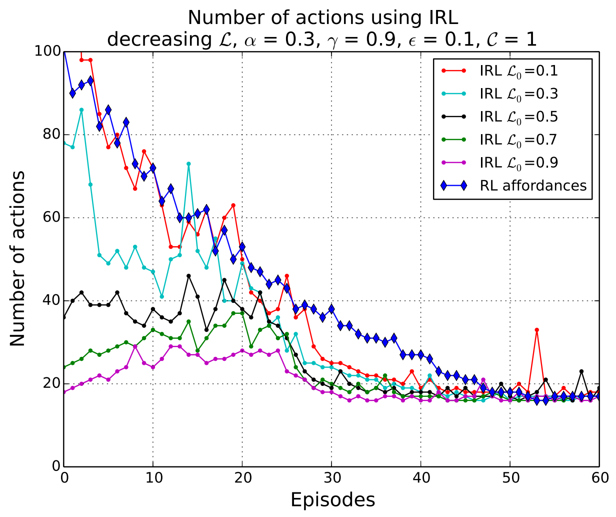


Fig. 17. Average number of actions needed for reaching the final state for RL with affordances approach (blue diamonds) and IRL approach with different initial probabilities of interaction $\mathcal{L}_0$ and decreasing over time.

Fig. 15 depicts the results and also shows the number of actions for RL with contextual affordances which is equivalent to have $\mathcal{L} = 0$ and $\mathcal{C} = 0$. It is clear that a more consistent trainer leads to better results or in this context to perform fewer actions. However, it is also observed that even with a consistency as small as $\mathcal{C} = 0.5$ the agent can improve its performance over time, especially in the beginning of the training. Additionally, Fig. 16 shows the average collected reward by the agent over episodes for different values of $\mathcal{C}$.

Finally, we ran an additional experiment where we decreased the probability of feedback $\mathcal{L}$ and therefore the contribution of advice over time to simulate the fatigue of an external trainer to provide feedback during the whole learning process. We made a reduction of the feedback after every episode as

$$\mathcal{L}_{t+1} = \eta\mathcal{L}_t \tag{9}$$

starting from different initial values of $\mathcal{L}$ and with $\eta = 0.95$ for all cases. Fig. 17 shows the average number of actions performed in every case. It is possible to observe that as interaction is decreasing, the number of performed actions increases after the first episodes where the agent explores nonoptimal actions in the absence of guidance. Nevertheless, after 25 episodes even with a very low amount of interaction the agent is able to reduce the number of performed actions due to its own knowledge on how to perform the cleaning task.

VII. Discussion

A. Summary

In this paper, we proposed an IRL simulated domestic scenario in which a robot should learn to clean a table. We defined objects, locations, and actions to represent the problem by means of a state machine.

Three particular learning methods were realized to test the performance. The first method consisted of a robot learning to execute the task in an autonomous fashion using classic RL. The robot was not able to learn the task before 400 episodes but still with a low success rate which increases slowly to 35% in episode number 1000. Furthermore, collected reward decreased in the first 600 episodes because of the low success rate and from there onward it increases to values around −0.4.

In the second method using RL with affordances, the robot also learned the task in an autonomous fashion but this time utilizing contextual affordances to avoid failed states. The agent mastered the task faster in comparison with the method used previously, reducing the number of actions needed to complete the task from 100 in the beginning to fewer than 20 within 100 episodes. Furthermore, with this method collected reward was always positive, reaching values over 0.8 because the success rate in this case is always 100% as a result of the usage of contextual affordances.

The third method consisted of IRL with affordances, and in this method the robot which had previously learned to execute the task became the trainer of a second robotic agent. In this scheme, the second robot was the apprentice which was advised in certain periods of the training process by the robot which had acquired knowledge on how to perform the cleaning task.

B. Conclusion

Through the three different learning methods explained in this paper, we can verify significant differences in terms of the particular learning performance of each of them. In the case of classic RL, we obtained a substantial improvement in comparison with our previous work [13]. Now we included a small negative reward after each performed action to encourage the robot to choose shorter paths toward the final state. This negative reward led to faster convergence and improved the success rate considerably from previously 4% to a level close to 35%.

Nevertheless, despite the improvements in the classic RL paradigm, this approach still led to a lower performance than RL utilizing contextual affordances. Certainly this was because the robot in this occasion did not reach failed states because the neural network architecture, using as inputs the current state, the action, and the object, anticipated the next state or the caused effect before the task execution, avoiding failed states when necessary. This effectively decreased the search space for the learning. Better performance was furthermore observed in the collected reward as described in Figs. 11 and 12. The maximal reward value reached by the robotic agent with classic RL was still less than the minimal reward value when RL with contextual affordances was used in all tested cases.

Results of the IRL approach showed that interaction provides advantages over RL with affordances in most of the tested levels of feedback where even a small amount of interactive feedback above 10% helped the robot to finish the cleaning task faster. This is illustrated by a smaller number of performed actions as well as a bigger amount of collected reward as shown in Figs. 13 and 14. When consistency of feedback was considered, it was observed that values under 50% make the learning process even slower. Nevertheless, the robotic agent was still able to learn in the long run since an important part of the time the robot performed actions with RL by autonomously exploring the environment. This suggests that the consistency of feedback has a different impact on learning according to the probability of feedback used.

Training robotic agents with interactive feedback and contextual affordances presented an advantage over classic RL in terms of number of performed actions and collected reward. Even low levels of interaction showed progress in comparison to RL working without an external trainer. Moreover, the agent was able to learn the proposed cleaning task even when being misadvised or receiving inconsistent feedback in some time steps during the learning process.

C. Future Work

As further improvements to this paper, we consider to investigate variations on the action selection method such as semi-uniform strategies like adaptive ϵ-greedy strategy based on value differences (VDBE) [50] where epsilon is reduced on the basis of the learning progress. On the one hand, high fluctuations in the estimates of value differences lead to a higher epsilon and further exploration and, on the other hand, low fluctuations lead to a lower epsilon and more exploitation. The method can also be combined with softmax-weighted action selection [51]. We hypothesize that a stronger classic RL approach which is the base for the other methods can lead to the necessity of less external advice allowing 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 impractical.

Furthermore, in this paper either the same or decreasing probability of feedback has been applied during the whole training process, i.e., we have not tested yet what the best time steps are to deliver interactive feedback. Evidence proves that there exist diverse factors which affect the ultimate performance of an apprentice agent using IRL methods such as the time period when the feedback is received [52], [53] as well as the magnitude of the problem where the method is applied [54]. Therefore, adjustments on the frequency of feedback in the implementation of the method GETADVICE can also be investigated.

The applicability of the proposed method in more complex scenarios is still an open question to be addressed, therefore, we also plan to transfer the present set-up to a human–robot interaction scenario with continuous state representation as in [55], acquiring advice from human trainers who must not necessarily be experts on developmental robotics or machine learning. In this regard, human advice can be interpreted as parental scaffolding [56] and therefore it is interesting to investigate diverse levels of scaffolding in terms of the number of given instructions and the frequency of these. The apprentice robot performance can be subsequently evaluated in relation to the number of performed actions by the agent once the scaffolding is removed.

In our previous work [57], we used spoken instructions to control the apprentice robot through speech recognition with the DOCKS system [58] to transfer this scenario to real environments in a more plausible manner. We are currently developing an advanced architecture considering also the vision modality using a depth sensor to make it more realistic and integrate it in a multimodal system to control the robot interactively. This is going to allow us to get much closer to real environments with which any human trainer even without background in robotics can teach a robot.

REFERENCES

[1] T. S. Tadele, T. de Vries, and S. Stramigioli, "The safety of domestic robotics: A survey of various safety-related publications," *IEEE Robot. Autom. Mag.*, vol. 21, no. 3, pp. 134–142, Sep. 2014.

[2] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Cambridge, MA, USA: MIT Press, 2015.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*, 2nd ed. New York, NY, USA: Springer, 2011.

[4] V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems*. Heidelberg, Germany: Springer, 2011.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: Bradford Book, 1998.

[6] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *Proc. 5th Int. Conf. Knowl. Capture*, Redondo Beach, CA, USA, 2009, pp. 9–16.

[7] H. B. Ammar, M. E. Taylor, K. Tuyls, and G. Weiss, "Reinforcement learning transfer using a sparse coded inter-task mapping," in *Multi-Agent Systems* (LNCS 7541). Heidelberg, Germany: Springer, 2012, pp. 1–16.

[8] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun. RO-MAN*, Atlanta, GA, USA, 2011, pp. 1–6.

[9] J. Grizou, M. Lopes, and P.-Y. Oudeyer, "Robot learning simultaneously a task and how to interpret human instructions," in *Proc. ICDL Epirob*, Osaka, Japan, 2013, pp. 1–8.

[10] J. J. Gibson, *The Ecological Approach to the Visual Perception of Pictures*. Boston, MA, USA: Houghton Mifflin, 1979.

[11] C. Wang, K. V. Hindriks, and R. Babuska, "Robot learning and use of affordances in goal-directed tasks," in *Proc. IROS Int. Conf.*, Tokyo, Japan, 2013, pp. 2288–2294.

[12] T. E. Horton, A. Chakraborty, and R. S. Amant, "Affordances for robots: A brief survey," *AVANT J. Philos. Interdiscipl. Vanguard*, vol. 3, no. 2, pp. 70–84, 2012.

[13] F. Cruz, S. Magg, C. Weber, and S. Wermter, "Improving reinforcement learning with interactive feedback and affordances," in *Proc. 4th Joint IEEE ICDL EpiRob*, Genoa, Italy, 2014, pp. 165–170.

[14] S. Kornblum, T. Hasbroucq, and A. Osman, "Dimensional overlap: Cognitive basis for stimulus-response compatibility—A model and taxonomy," *Psychol. Rev.*, vol. 97, no. 2, pp. 253–270, 1990.

[15] L. J. Lin, "Programming robots using reinforcement learning and teaching," in *Proc. AAAI*, 1991, pp. 781–786.

[16] J. P. Bandera, J. A. Rodríguez, L. Molina-Tanco, and A. Bandera, "A survey of vision-based architectures for robot learning by imitation," *Int. J. Humanoid Robot.*, vol. 9, no. 1, 2012, Art. no. 1250006.

[17] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 360–375, 2012.

[18] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intell. Service Robot.*, vol. 6, no. 1, pp. 33–51, 2013.

[19] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann, "Towards robot skill learning: From simple skills to table tennis," in *Machine Learning and Knowledge Discovery in Databases* (LNCS 8190), Heidelberg, Germany: Springer, 2013, pp. 627–631.

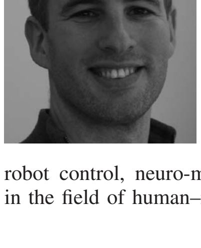
[20] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *Proc. 21st Nat. Conf. Artif. Intell.*, vol. 1. Boston, MA, USA, 2006, pp. 1000–1005.

[21] A. L. Thomaz, G. Hoffman, and C. Breazeal, "Real-time interactive reinforcement learning for robots," in *Proc. AAAI Workshop Human Comprehensible Mach. Learn.*, 2005, pp. 9–13.

[22] W. B. Knox, P. Stone, and C. Breazeal, "Teaching agents with human feedback: A demonstration of the TAMER framework," in *Proc. Int. Conf. Intell. User Interf. Companion*, Santa Monica, CA, USA, 2013, pp. 65–66.
[23] W. B. Knox and P. Stone, "Reinforcement learning from human reward: Discounting in episodic tasks," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun. RO-MAN*, Paris, France, 2012, pp. 878–885.
[24] A. L. Thomaz and C. Breazeal, "Asymmetric interpretations of positive and negative human feedback for a social learning agent," in *Proc. IEEE Int. Symp. Robot Human Interact. Commun. RO-MAN*, Jeju-do, South Korea, 2007, pp. 720–725.
[25] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2625–2633.
[26] A. Chemero, *Radical Embodied Cognitive Science*. Cambridge, MA, USA: MIT Press, 2011.
[27] E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adapt. Behav.*, vol. 15, no. 4, pp. 447–472, 2007.
[28] A. Chemero and M. T. Turvey, "Gibsonian affordances for roboticists," *Adapt. Behav.*, vol. 15, no. 4, pp. 473–480, 2007.
[29] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, 2012, pp. 4373–4378.
[30] I. Awaad, G. Kraetzschmar, and J. Hertzberg, "The role of functional affordances in socializing robots," *Int. J. Soc. Robot.*, vol. 7, no. 4, pp. 421–438, 2015.
[31] O. Sigaud and A. Droniou, "Towards deep developmental learning," *IEEE Trans. Cogn. Develop. Syst.*, to be published, doi: 10.1109/TAMD.2015.2496248.
[32] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, San Diego, CA, USA, 2007, pp. 1015–1021.
[33] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation," *IEEE Trans. Robot.*, vol. 24, no. 1, pp. 15–26, Feb. 2008.
[34] İ. Atıl, N. Dağ, S. Kalkan, and E. Şahin, "Affordances and emergence of concepts," in *Proc. Int. Conf. Epigenetic Robot. Model. Cogn. Develop. Robot. Syst.*, 2010, pp. 149–156.
[35] M. Kammer, T. Schack, M. Tscherepanow, and Y. Nagai, "From affordances to situated affordances in robotics—Why context is important," *Front. Comput. Neurosci.*, vol. 5, no. 30, 2011, doi: 10.3389/conf.fncom.2011.52.00030.
[36] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, 2013, pp. 1321–1326.
[37] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, U.K.: Springer, 2009.
[38] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from RGB-D videos," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 951–970, 2013.
[39] J. Kober and J. Peters, "Reinforcement learning in robotics: A survey," in *Reinforcement Learning*, vol. 12. Heidelberg, Germany: Springer, 2012, pp. 579–610.
[40] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge Univ. Eng. Dept., Cambridge, U.K., Tech. Rep. CUED/F-INFENG/TR166, 1994.
[41] G. Cybenko, "Approximation by superpositions of a sigmoid function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[42] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, no. 3, pp. 183–192, 1989.
[43] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
[44] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *Algorithmic Learning Theory* (LNCS 6925). Heidelberg, Germany: Springer, 2011, pp. 18–36.
[45] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[46] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2010, pp. 249–256.
[48] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, San Diego, CA, USA, 1990, pp. 21–26.
[49] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[50] M. Tokic, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences," in *KI 2010: Advances in Artificial Intelligence*, vol. 6359. Heidelberg, Germany: Springer, 2010, pp. 203–210.
[51] M. Tokic and G. Palm, "Value-difference based exploration: Adaptive control between ϵ-greedy and softmax," in *KI 2011: Advances in Artificial Intelligence*, vol. 7006. Heidelberg, Germany: Springer, 2011, pp. 335–346.
[52] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst. (AAMAS)*, St. Paul, MN, USA, 2013, pp. 1053–1060.
[53] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connection Sci.*, vol. 26, no. 1, pp. 45–63, 2014.
[54] C. Stahlhut, N. Navarro-Guerrero, C. Weber, and S. Wermter, "Interaction is more beneficial in complex reinforcement learning problems than in simple ones," in *Proc. Interdisziplinärer Workshop Kognitive Systeme (KogSys)*, 2015, pp. 142–150.
[55] I. Farkaš, T. Malík, and K. Rebrová, "Grounding the meanings in sensorimotor behavior using reinforcement learning," *Front. Neurorobot.*, vol. 6, no. 1, pp. 1–13, 2012.
[56] E. Ugur, Y. Nagai, H. Celikkanat, and E. Oztop, "Parental scaffolding as a bootstrapping mechanism for learning grasp affordances and imitation skills," *Robotica*, vol. 33, no. 5, pp. 1163–1180, 2015.
[57] F. Cruz, J. Twiefel, S. Magg, C. Weber, and S. Wermter, "Interactive reinforcement learning through speech guidance in a domestic scenario," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2015, pp. 1–8.
[58] J. Twiefel, T. Baumann, S. Heinrich, and S. Wermter, "Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing," in *Proc. 28th AAAI Conf. Artif. Intell. (AAAI)*, Quebec City, QC, Canada, 2014, pp. 1529–1535.

Francisco Cruz received the bachelor's degree in engineering and the master's degree in computer engineering from the University of Santiago, Santiago, Chile, in 2004 and 2006, respectively. He is currently working toward the Ph.D. degree in developmental robotics with the Knowledge Technology Group, University of Hamburg, Hamburg, Germany, with a particular focus on interactive reinforcement learning.

He joined the School of Informatics, Central University of Chile, Santiago, as a Research and Teaching Associate, in 2009. His current research interests include human–robot interaction, artificial neural networks, reinforcement learning, affordances, and psychological and bio-inspired models.



Sven Magg received the M.Sc. degree in intelligent systems from the University of Sussex, Brighton, U.K., and the Ph.D. degree in swarm intelligence from the University of Hertfordshire, Hatfield, U.K., in 2012.

Since 2011, he has been with the Knowledge Technology Group, University of Hamburg, as a Post-Doctoral Teaching Associate, where he is also the Coordinator of the M.Sc. Program "Intelligent Adaptive Systems." His current research interests include evolution of artificial neural networks for robot control, neuro-modulation, bio-inspired solutions for data processing in the field of human–robot interaction, and self-organizing systems.

Cornelius Weber received the Diploma degree in physics, from the University of Bielefeld, Bielefeld, Germany, and the Ph.D. degree in computer science with the Technische Universität Berlin, Berlin, Germany, in 2000.

He is a Laboratory Manager with the Knowledge Technology Group, University of Hamburg, Hamburg, Germany. He was a Post-Doctoral Fellow of Brain and Cognitive Sciences with the University of Rochester, Rochester, NY, USA. From 2002 to 2005, he was a Research Scientist of Hybrid Intelligent Systems with the University of Sunderland, Sunderland, U.K. He was a Junior Fellow with the Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany, until 2010. His current research interests include computational neuroscience with a focus on vision, unsupervised learning, and reinforcement learning.

Stefan Wermter received the Diploma degree from the University of Dortmund, Dortmund, Germany, the M.Sc. degree from the University of Massachusetts, Amherst, MA, USA, and the Ph.D. and Habilitation degrees from the University of Hamburg, Hamburg, Germany, all in computer science.

He has been a Visiting Research Scientist with the International Computer Science Institute, Berkeley, CA, USA, before leading the Chair of Intelligent Systems at the University of Sunderland, Sunderland, U.K. He is currently a Full Professor of Computer Science with the University of Hamburg, and the Director of the Knowledge Technology Institute. His current research interests include neural networks, hybrid systems, cognitive neuroscience, bio-inspired computing, cognitive robotics, and natural language processing.

Prof. Wermter was the General Chair for the International Conference on Artificial Neural Networks, in 2014. He is also on the Current Board of the European Neural Network Society and an Associate Editor of the *Transactions on Neural Networks and Learning Systems*, *Connection Science*, the *International Journal for Hybrid Intelligent Systems*, and *Knowledge and Information Systems*. He is on the Editorial Board of *Cognitive Systems Research* and the *Journal of Computational Intelligence*.