Towards Learning to Handle Deviations Using User Preferences in a Human Robot Collaboration Scenario

Sharath Chandra Akkaladevi[1,2(✉)], Matthias Plasch[1], Christian Eitzinger[1], Sriniwas Chowdhary Maddukuri[1], and Bernhard Rinner[2]

[1] Profactor GmbH, Im Stadtgut A2, Steyr-Gleink 4407, Austria
sharath.akkaladevi@profactor.at
[2] Institute of Networked and Embedded Systems, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria

Abstract. In a human robot collaboration scenario, where robot and human coordinate and cooperate to achieve a common task, the system could encounter with deviations. We propose an approach based on Interactive Reinforcement Learning that learns to handle deviations with the help of user interaction. The interactions with the user can be used to form the preferences of the user and help the robotic system to handle the deviations accordingly. Each user might have a different solution for the same deviation in the assembly process. The approach exploits the problem solving skills of each user and learns different solutions for deviations that could occur in an assembly process. The experimental evaluations show the ability of the robotic system to handle deviations in an assembly process, while taking different user preferences into consideration. In this way, the robotic system could both benefit from interaction with users by learning to handle deviations and operate in a fashion that is preferred by the user.

Keywords: Learning to handle deviations · User preferences · Human robot collaboration · Ontology based knowledge representation

1 Introduction

Human robot collaboration (HRC) allows to combine the cognitive strength of humans together with the physical strength of robots and can lead to numerous applications [5]. For a close collaboration with humans, the robotic systems should attribute meaning to beliefs, goals and desires, collectively called "mental models" of humans during a particular task [11]. This would not only allow the robotic system to understand the actions and expressions of humans within an intentional or goal-directed architecture [14], but also for the human operators to better understand the capabilities of the robotic system [11]. In context of HRC, user preferences communicate the human's desire to achieve a goal (based on the belief that the goal is possible), either by themselves or in collaboration with the robotic system. User preferences can be communicated implicitly (via the actions done by the human) or explicitly (with the help offline knowledge representation). User preferences aid the robotic system in

A. Basu et al. (Eds.): IHCI 2016, LNCS 10127, pp. 3–14, 2017.
DOI: 10.1007/978-3-319-52503-7_1

creating and understanding the perspective of the human and collectively forms the basis for building the “mental model” of the human.

A detailed survey on human robot interaction, emerging fields and applications is given in [3, 6]. In this paper we present an HRC approach capable of handling deviations, where the robot interacts with the human by considering the ‘preferences’ of the human. By considering user preferences, the robotic system can take actions from the perspective (mental model) of the human to facilitate a ‘natural’ interaction. HRC should deal with the non-deterministic factor of the environment with human at its center, but it could also benefit from the cognitive and problem solving skills of the human. In recent works [7, 15, 18], it was shown that given the presence of the human, the robotic system can take advantage of human feedback and learn a given task in short time using Interactive Reinforcement Learning (IRL) techniques [15]. Inspired by this success, we propose an extension to the Interactive Reinforcement Learning (eIRL), which would allow the robotic system to take the aid of the human (asking which action to take next, receiving feedback) to learn to cope with deviations/novel situations as they occur. The advantage of allowing the user to teach the robot in coping with deviations is that each user has an own way of solving a given problem (communicated via user preferences). This allows the robot to learn different possibilities (when they exist) to deal with a deviation and use this knowledge to better collaborate and cooperate with the human. This also helps the robotic system to better personalize its behavior to a given user and their corresponding preferences.

In order to showcase the deviation handling capability of the proposed approach, an assembly process use case that deals with assembling objects into a box in collaboration with the human is chosen. This use case consists of four real-world objects, namely *heater*, *tray*, *ring* and *base* as shown in Fig. 1a. The goal of the use case is to assemble these objects by placing each of them inside a box. An example execution of pick and place operation of the *heater* object is shown in Fig. 1b. It is important to note that the manipulation possibility of each object either by the robot or the human is dependent on the construction (e.g., heavy, objects that require complex grasp - *base*) and configuration of the objects in the workspace.

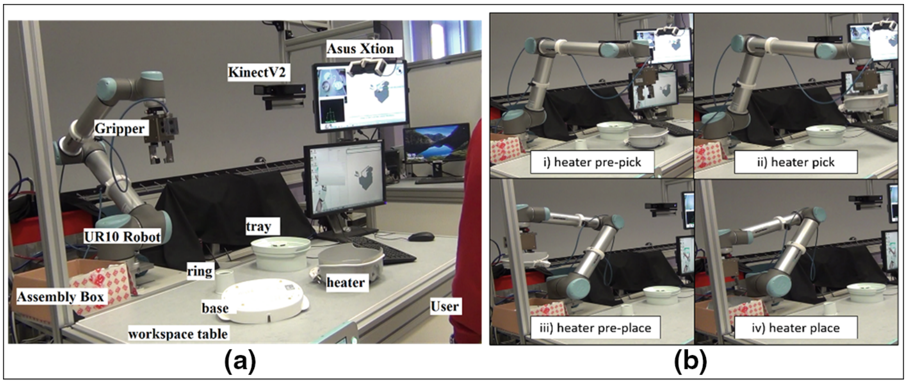


Fig. 1. (a) Use Case setup; (b) Pick and Place operation of the heater object

In this article, an HRC architecture that is capable of dealing with uncertainties and deviations of task executions using eIRL is presented. The architecture exploits the assistance of the human (who interacts with the robot to teach it) in the HRC loop, to learn and recover from deviations/novel situations that can occur during a task execution, using an interactive reinforcement learning algorithm. The approach where the robotic system:

- assesses the feasibility of executing the goal (to a certain level) before performing it
- **proactively suggests** possible course of action/s while considering the robot state (proprioception), the environment and the **preferences of the user**
- **enables technically unskilled users** to teach and customize the robot's behavior to their preferred manner
- **provides an interface** to communicate and learn from the human the necessary/alternative steps to take in case of deviations

The emphasis is on dealing with the deviations that occur in the action planning phase of the task. The low-level deviations that occur while carrying out the motion planning (path planning, collision avoidance) of the robot, while executing the task are not in the focus of this work.

The remaining part of the paper is organized as follows: In Sect. 2, a brief description of the existing state of the art approaches that deal with RL to learn a complete task are presented. Then in Sect. 3, the functional components and the architecture of the extended IRL is described. The theoretical background and the algorithmic description of eIRL is given in Sect. 4. Finally, the experimental setup and evaluation of the proposed eIRL is discussed in Sect. 5 followed by concluding remarks and possible future work described in Sect. 6.

2 Related Work

In order for the collaborative robots to work hand in hand with human operators the robotic system should be able to deal with complex and continuously changing environments. Thorough offline modeling of the environment and task conditions is not a feasible solution. Hence the robotic system should learn to respond to the environment by performing actions in order to reach a goal. The problem of agents learning in complex real-world environments is dealt with by numerous approaches [3, 4, 6]. Reinforcement learning (RL) [16] is one such popular approach that deals with learning from interaction, to teach the agent how to behave (which actions to perform when) in order to complete a task. The main aim in RL is to maximize a cumulative reward that is attained by taking some actions in the environment.

In RL, the agent learns over discrete time steps by interacting with its environment and gaining experience about the outcome. However, in order to reach an optimal policy (the set of actions that lead to the maximum reward), the RL approach requires substantial interaction with the environment. Depending on the nature of the task, RL results in a memory intensive storage of all state action pairs [8]. Another disadvantage of RL is its slow convergence towards a satisfactory solution. Despite such drawbacks, recent works on learning have shown considerable interest in using RL [7, 10, 15, 18].

The basic idea is to use the human teacher in the loop to provide feedback and hence speedup the convergence time in RL for reaching an optimal policy. In conventional RL, a reward is a positive or negative feedback for being in the current state (or for a particular state-action pair). Thomas et al. [18] introduced a run-time human feedback as a reward to the IRL approach and argue that this approach is beneficial for both the human teacher and the learning algorithm. Their study suggests that users (human teachers) employ the feedback as a single communication channel for various communicative intents-feedback, guidance, and motivation. Inspired by this approach, Suay et al. [15] study how IRL can be made more efficient for real-world robotic systems. In this approach, the user provides rewards for preceding actions of the robot and additionally provides guidance for subsequent actions. They further show that this guidance reduces the learning time of the robot and that it is more evident when large state-space size (number of interactions in the environment for the robot) is considered.

Knox et al. [9] propose a framework to train an agent manually via evaluative reinforcement, using real-valued feedback on its behavior from a human trainer. This allows the human trainer to interactively shape the agent's policy (interactive shaping) and thereby, directly modify the action selection (policy) mechanism of the IRL algorithm. Rather than in influencing the policy indirectly through a reward, Griffith et al. [7] use the user feedback in making a direct statement about the policy itself. Knox et al. [10] extended their previous work in [9] by applying the TAMER framework to real-world robotic system. Rozo et al. [12, 13] propose an approach for a human robot interactive task, where the robot learns both the desired path and the required amount of force to apply on an object during the interaction.

In this article, the following contributions are made to the state of the art:

- an extension to the Interactive Reinforcement learning algorithm (eIRL), that enables the robotic system to learn from the human, to deal with deviations.
- at each instant the robotic system encounters a deviation, it proactively suggests a list of actions possible by the robot in it's current state. This is achieved through efficient representation of the task knowledge using web ontology web language (OWL), see Sect. 3 for more details. The advantage Learning to Handle Deviations for Human Robot Collaboration 5 is that even untrained users can effortlessly teach the robot (how to deal with deviations) by choosing an appropriate action from the list provided.
- unlike the other approaches, the human feedback is divided into two: (a) feedback to choose an action policy from the list of options available (b) like/dislike feedback to function as a reward for an action executed by the robot, when a deviation is encountered.

3 eIRL Architecture

In order to collaborate and cooperate with the human in HRC scenario, the following components are developed within the extended IRL (eIRL) architecture as shown Fig. 2: (a) **perception and interpretation capabilities**: to understand and interpret the current state of the environment from sensor data (b) **knowledge representation**: to represent different aspects of the task carried out in the HRC environment that include representing the abilities and activities of the human, interplay between human activities and object configurations, robot's own self with respect to the task, etc. (c) **learning and reasoning**: the robotic system is equipped with learning capabilities that allows the accumulation of knowledge over time to enhance in particular its abilities to perceive the environment, to make decisions, to behave intelligently, and to interact naturally with humans. The robotic system also reasons about its current state to plan its own actions accordingly to aid in completing the task (d) **action planning**: to generate plans for future actions to achieve the given task (goals). This includes task planning, scheduling and observation planning, as well as planning under uncertainty for an efficient human robot collaboration. These plans generated need to be carried out in real world where the robot plans its path (path/navigation planning) and manipulates the environment accordingly. Note that the learning, reasoning and the action planning components form the heart of the eIRL architecture. (e) **functional component layer**: provides the necessary low-level sensory information like real-time 3D tracking of objects (implemented similar to [2]), current action of the human [1] together interfaces to robotic manipulation and a GUI-interface for human robot interaction.

Knowledge Representation in the proposed architecture is based on KnowRob [17]. The main reason for choosing KnowRob is that it provides the following knowledge processing features: (a) mechanisms and tools for action centric representation (b) automated acquisition of grounded concepts through observation and

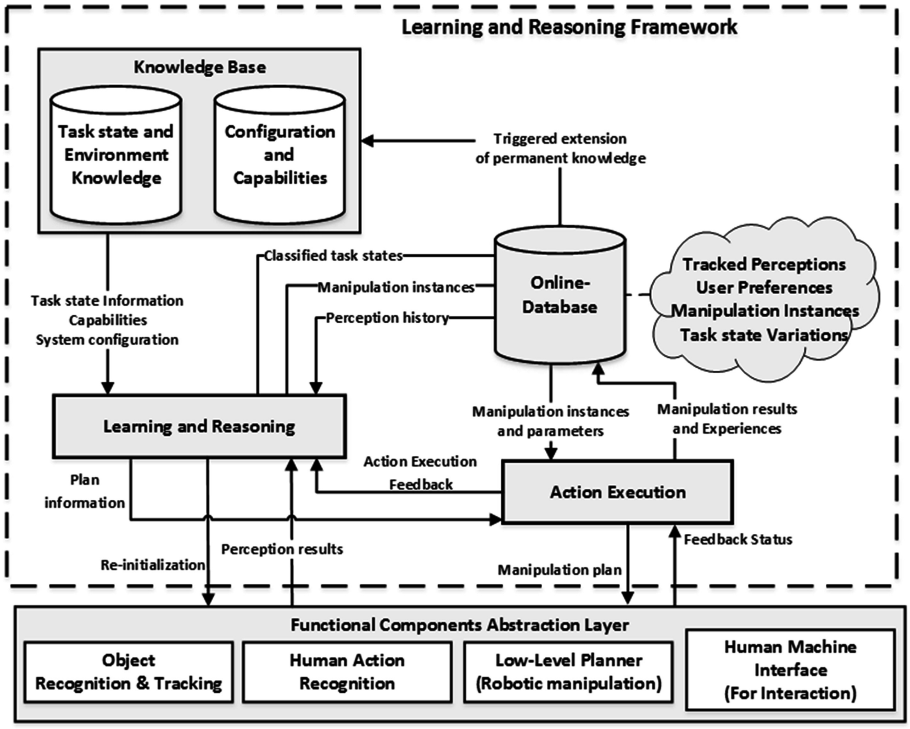

Learning and Reasoning Framework
Knowledge Base
Task state and Environment Knowledge
Configuration and Capabilities
Triggered extension of permanent knowledge
Classified task states
Manipulation instances
Perception history
Online-Database
Tracked Perceptions
User Preferences
Manipulation Instances
Task state Variations
Task state Information
Capabilities
System configuration
Learning and Reasoning
Manipulation instances and parameters
Manipulation results and Experiences
Action Execution Feedback
Plan information
Action Execution
Re-initialization
Perception results
Manipulation plan
Feedback Status
Functional Components Abstraction Layer
Object Recognition & Tracking
Human Action Recognition
Low-Level Planner (Robotic manipulation)
Human Machine Interface (For Interaction)

Fig. 2. Architecture diagram of the extended Interactive Reinforcement Learning approach

experience (c) reasoning about and managing uncertainty, and fast inference. The knowledge is represented using ontology (description logics) in the Web Ontology Language (OWL) and SWI Prolog is used for loading, accessing and querying the ontology. The representation consists of two levels: *Classes* that contain abstract terminological knowledge (type of objects, events and actions - taxonomic fashion) and *Instances* which represent the actual physical objects or the actions that are actually performed. The link between *Classes* and *Instances* is given in *Properties*, which defines if an *Agent* $\in$ {*Human*, *Robot*} can perform a particular action (defined in *Classes*) on/with a *Target* $\in$ {*Objects*, *Robot*, *Human*}. A Relation denoted by a triple <*Agent*, *Property*, *Target*> defines a particular aspect of the assembly task. For example, <*Robot*, *pick*, *ring*> conveys that the robot (is capable and) should pick the ring object. A detailed description of KnowRob and its features are given in [17].

4 eIRL Algorithm Description

Reinforcement Learning (RL) is an area of machine learning that defines a class of algorithms that enable a robot to learn from its experience. Reinforcement learning in our case is defined using the standard notation of the Markov Decision Process (MDP). In a MDP, any state s_{t+1} occupied by the robot is a function of its previous state s_t and the action taken at, in other words $s_{t+1} = f(s_t, a_t)$. A MDP is denoted by the 5-tuple $<S, A, P, R, \gamma>$, where the set of possible world states is defined by S, and A denotes the set of actions available to the agent in each state. The probability function $P : S \times A \rightarrow Pr[S]$, describes the transition probability of State s_t to State s_{t+1}, when an action a_t is performed on State s_t. The reward function $R : S \times A \rightarrow R$, and a discount factor γ, where $0 \leq \gamma \leq 1$. Together P and R describe the dynamics of the system. The goal of a RL algorithm is to find an approximation function $Q : S \times A \rightarrow R$, that defines an optimal policy π. Where, Q maps the state-action pairs to the expected reward and the optimal policy : $S \rightarrow A$ maximizes the expected reward. In other words, π specifies the best possible action to perform in a given state in order to gain a maximum reward.

In this paper, Q-learning RL [20] is used as a basis for the learning algorithm. As mentioned earlier, IRL approaches [15] [18] showed the success of learning a complete task using interactive RL algorithm, Q-learning. Using this as basis, we extend IRL to be used in handling deviations in an assembly process as they occur during execution as shown in Fig. 3. The main idea is to integrate the human operator as advisor into the learning process. In this way the user can teach the system, how to deal with deviations during assembly process execution. It is assumed that the optimal action selection policy, say π^*, to perform the assembly task, is already known (using [15, 18]). In this section we describe the algorithm that modifies Q-learning, to handle deviation using user preferences. In this context, two types of user feedback are considered (a) the optimal action policy selection by the user during the deviation - treated as the user preference (b) the reward provided by the user for the action taken by the robotic system during the deviation - the like/dislike option.

During a normal assembly process, the robotic system follows the already learned optimal policy π^* and performs actions a_{ti} accordingly based on its current state st to

```
1: Initialize Q table $Q^*$ containing the action values $Q^*_{t,i}$, where $1 \leq t \leq |S|$ and
   $1 \leq i \leq |A_t|$ are the indices of states $s_t$ and actions $a_{ti}$ possible at the given state. $Q^*$
   leads to an optimal assembly process execution policy $\pi^*$, maximizing the reward
2: for each execution episode of the assembly process do
3:     Set assembly process to initial state $s_t \,|\, t = 1$
4:     for each execution step do
5:         Execute action $a^*_{ti}$ in state $s_t$ following policy $\pi^*$
6:         if no deviation occurred while executing $a^*_{ti}$ then
7:             Observe resulting state $s_{t+1}$
8:             Q table $Q^*$ is unchanged
9:         else
10:            Propose deviation handling actions $a^{\Delta}_{ti,j} \in A^{\Delta}_{ti} \,|\, j \in \mathbb{N}$
11:            if Action value $Q^*_{t,i} \not\sim \vec{Q}^{\Delta}_{ti}$ then
12:                Initialize deviation Q vector $\vec{Q}^{\Delta}_{ti}$ with $Q^{\Delta}_{ti,j} = 0$
13:                Associate $Q^*_{t,i}$ with $\vec{Q}^{\Delta}_{ti}$: $Q^*_{t,i} \sim \vec{Q}^{\Delta}_{ti}$
14:            if $\vec{Q}^{\Delta}_{ti}$ holds the initial values, i.e. $\max Q^{\Delta}_{ti,j} = 0$ then
15:                User shall select a deviation handling action $a^{\Delta}_{ti,k} \,|\, 1 \leq k \leq |A^{\Delta}_{ti}|$
16:            else
17:                Take handling action $a^{\Delta}_{ti,k} \leftarrow \arg\max_{a^{\Delta} \in A^{\Delta}_{ti}} Q^{\Delta}_{ti,j}$
18:            Observe resulting state $s_{t+1}$, if $a^{\Delta}_{ti,k}$ was unsuccessful then go to ln.14
19:            Receive user feedback (like/dislike), determining the reward $r^{\Delta}_{t,k}$
20:            for each $Q^{\Delta}_{ti,j} \in \vec{Q}^{\Delta}_{ti}$ do
21:                $Q^{\Delta}_{ti,j} \leftarrow Q^{\Delta}_{ti,j} + \alpha \left[ r^{\Delta}_{t,j} + \gamma \max_{1 \leq l \leq |A_{t+1}|} Q_{t+1,l} - Q^{\Delta}_{ti,j} \right] \,|\, \alpha = 1, \gamma = 0$
22:                if $j = k \implies a^{\Delta}_{ti,j} = a^{\Delta}_{ti,k}$ then
23:                    if $userFeedback = like$ then
24:                        $r^{\Delta}_{t,j} \leftarrow r_{max}$
25:                    else
26:                        $r^{\Delta}_{t,j} \leftarrow 0$
27:                else
28:                    $r^{\Delta}_{t,j} \leftarrow 0$
29:        $s_t \leftarrow s_{t+1}$
```

Comments
Ln 10: j is the index of possible deviation handling actions $a^{\Delta}_{ti,j}$
Ln 11: Each action value $Q^*_{t,i}$ can be associated with a deviation Q vector $\vec{Q}^{\Delta}_{ti}$
Ln 11: The symbol $\not\sim$ means "not associated with"
Ln 12: $Q^{\Delta}_{ti,j}$ is the action value for applying the deviation handling action $a^{\Delta}_{ti,j}$ to s_t, after encountering a deviation in action a^*_{ti}
Ln 18: It is assumed that the resulting state s_{t+1} is a known state of the assembly process
Ln 23: The reward $r_{max} \in \mathbb{R}^+$ is given, if the deviation handling result is *liked* by the user

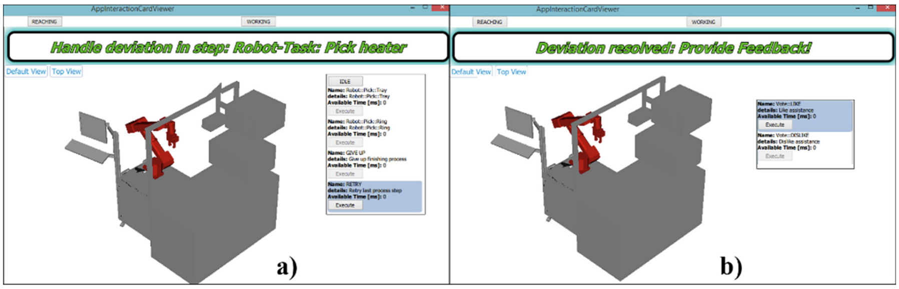
Fig. 3. Algorithm to Handle deviations based on Q-Learning

progress to the next state s_{t+1}. As described earlier, the robotic system is enabled with state of the art action recognition and object tracking perception capabilities coupled with knowledge representation framework to comprehend the current status of the assembly task. After completing an action a_{ti} the robotic system observes the resulting state, if the resulting state is the expected next state s_{t+1}, the next optimal action policy in π^* is carried out and so on until the end. However, due to the dynamic nature of the environment, though an action is chosen by the robotic system, the task execution might not be successful and is called a deviation. A deviation could occur due to (a) the chosen action cannot be performed - e.g., object out of reach, object not available (b) expected resulting state s_{t+1} was not observed after the action a_{ti} is performed. A typical example could be an object was not graspable due to sensor error or the object configuration. It could also happen that the object might be out of reach of the robot. In such cases, it is not possible for the robotic system to continue with the assembly task as the learned optimal policy action cannot be completed.

Let us say action a_{ti} is an optimal policy action (given π^*) taken in state s_t and a deviation Δ_{ti} has occurred. In this case, the robotic system (depending on the current state, its capabilities, objects, knowledge base) proactively suggests a set of deviation handling actions $a^{\Delta}_{ti,j}$ possible in the current state s_t. Since deviations are special cases that occur during the assembly process, the original optimal policy should not be directly affected. Therefore, a deviation Q vector $\vec{Q}^{\Delta}_{ti}$ is associated with this deviation Δ_{ti} that occurred for this state-action pair $\{s_t, a_{ti}\}$ and its corresponding entry $Q^*_{t,i}$ in the Q table. If no vector $\vec{Q}^{\Delta}_{ti}$ existed previously for Δ_{ti}, then it is created and its values $Q^{\Delta}_{ti,j}$ are uniformly initialized to zero. This is generally the case, if this deviation was occurring for the first time. The robotic system then waits for the user to select an action policy ($a_{ti,j} \in A^{\Delta}_{ti}$) suitable for the deviation, from the list of suggested actions. In order to suggest the list of possible deviation handling actions A^{Δ}_{ti}, the robotic system observes the current status in the assembly process, available objects and robot's capabilities and then proposes a list of actions to progress the assembly process. Once the user selects an action policy, the action is carried out and the resulting state s_{tnext} is observed. Whenever, a deviation occurs the robotic system learns the optimal action policy in case of that deviation myopically i.e., $\gamma = 0$ [10]. In other words, it is assumed that no matter what deviation action policy is chosen by the user, it will progress the assembly process from the deviation to a known state s_{known}, from which an optimal state-action pair exists. This kind of learning from human feedback eliminates any exploration and results in a deviation handling policy that is only as good as the decision made by the user.

Upon successful completion of the suggested action policy ($s_{tnext} = s_{known}$), the robotic system asks for a *reward feedback*. The user at this stage can either *like* or *dislike* the robot's performance, as shown in Fig. 4. If the action is *liked* then the respective Q value for that action in the deviation $\vec{Q}^{\Delta}_{ti}$ vector is set to a maximum reward value and the other action entries are set to zero. In case of *dislike*, all the Q values in $\vec{Q}^{\Delta}_{ti}$ are set to 0. The execution is then continued with the optimal policy. If the deviation handling action was not successfully executed ($s_{tnext} \neq s_{known}$), the robotic system goes back and suggests possible deviation actions again. In this way the robotic system learns how to deal with the deviation that occurred for that state-action pair.

At a later point in time, if the robotic system encounters a deviation Δ_{ti}, there already exists a $\vec{Q}^{\Delta}_{ti}$ with initialized values. The robotic system then directly chooses the action policy $a^{\Delta}_{ti,j}$ in the $\vec{Q}^{\Delta}_{ti}$ that leads to a maximum reward value. After performing the deviation handling action, the robotic system asks the user for reward feedback. If the



Name: Robot::Pick::Tray
details: Robot::Pick::Tray
Available Time [ms]: 0

Name: Robot::Pick::Ring
details: Robot::Pick::Ring
Available Time [ms]: 0

Name: GIVE UP
details: Give up finishing process
Available Time [ms]: 0

Name: RETRY
details: Retry last process step
Available Time [ms]: 0

Name: Vote::LIKE
details: Like assistance
Available Time [ms]: 0

Name: Vote::DISLIKE
details: Dislike assistance
Available Time [ms]: 0

Fig. 4. Example of interaction cards provided while handling deviations in the assembly process (a) shows the list of deviation handling actions possible when robot encountered a deviation while Pick&Place heater action and (b) shows the like/dislike feedback interaction card presented to the user after successful handling of the deviation

user then *likes* the action performed by the robot, the Q values of the $\vec{Q}^{\Delta}_{ti}$ are updated accordingly and the assembly process proceeds as usual. Since the occurrence of these deviations are separated in time (a deviation can occur today and is repeated after an year), it is possible that user might like to teach the robot a different action policy $a^{\Delta}_{ti,diff}$ for handling that deviation Δ_{ti}. In such cases, the user *dislikes* the action performed by the robot which allows the robot to learn a new deviation action policy $a^{\Delta}_{ti,diff}$ for that deviation Δ_{ti}.

5 Experimental Setup

As shown in Fig. 1a, the experimental setup consists of a UR10 [19] robotic arm with 6 degrees of freedom, a SCHUNK 2 finger electric parallel gripper and two commercially available RGB-D sensors each equipped with object tracking and action recognition functionalities respectively. As mentioned in Sect. 3, the functional component layer with the help of sensors and actuators as shown in Fig. 1a provides the reasoning and learning module with necessary perception (object tracking and action recognition) and actuation (robot and gripper state combined to deduce if an action execution was successful) information.

The assembly process use case presented in this paper concerns itself with assembling the given objects (*heater*, *base*, *tray*, *ring*) in to a box. Depending on the capabilities of the *Agent* ϵ {*Human*, *Robot*} and the object properties (configuration, construction) defined in the knowledge representation an optimal policy of a normal assembly process is defined as given in Fig. 5 and is assumed to be known. Figure 5 shows the optimal policy sequence (Step 1, Step 2, Step 3, Step 4 respectively) that expresses ‘what’ action is required to be performed by ‘which’ *Agent* and on ‘which’ *Target*. In this section, the ability of the system to learn, how to deal with deviations that occur during online execution (which were not encountered during training) of the assembly process is evaluated. During a normal assembly, Step 1 to 4 are executed in a sequence following the optimal policy and thereby gaining a maximum reward.

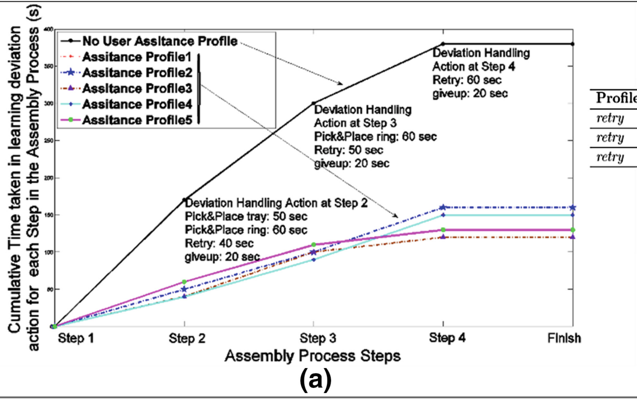
The *Pick&Place* action includes localizing the object, reaching for the object, grasping and lifting it and then placing it in the pre-assigned assembly box as shown in Fig. 1b. Note that, as we deal with a human robot collaboration scenario, some steps needs to be carried out by the user, in this case Step 1. The robotic system waits for the user to complete Step 1, observed with the help of embedded object tracking and action recognition functionalities. Once Step 1 is complete the robot proceeds with the next steps.

Assembly Process Step $< Agent, Property, Target >$	**Possible Deviation Handling Actions**
Step 1: User *Pick&Place base*	Note: only user can perform this action
Step 2: Robot *Pick&Place heater*	*retry*; *Pick&Place tray*; *Pick&Place ring*; *giveup*
Step 3: Robot *Pick&Place tray*	*retry*; *Pick&Place ring*; *giveup*
Step 4: Robot *Pick&Place ring*	*retry*; *giveup*

Fig. 5. Optimal action policy of the assembly process and the respective deviation handling actions available at each step

A deviation can occur at any step in the assembly process and the possible deviation handling actions at each step are given in Fig. 5. These deviation handling actions are presented as interaction cards to the user as shown in Fig. 4, who can then select an action by simply clicking on the 'Execute' button. The deviation handling action *retry* entails that the same action step has to be retried, e.g., if in Step 2 a deviation occurs and the user selects *retry*, then Step 2 will be executed again. The deviation handling action *give-up* communicates to the robot that assembly process should be stopped and the robot should move to an initialized position. A total of 25 executions of the assembly process by 5 users (roboticists) with 5 executions per user were carried out. Figure 6 shows five different profiles created during online execution. Each profile has three entries, which consists the chosen deviation handling action by that user for Step 2, 3 and 4 respectively. During the online execution of the 5 assembly process per user, different deviations possible were introduced at random such that, overall the user teaches a deviation handling action to the robot for each step of the assembly process.

Learning to handle deviation in this fashion has the following advantages: (a) the robot is not required to explore all possible deviation handling actions for each step to find a solution. This reduces the amount of time exponentially as shown in Fig. 6 depending on the number of deviation handling actions possible at a given step where a deviation occurred (b) the robotic system is capable of handling novel situations (situation not previously encountered) as they occur online during the assembly process with the help of the user interaction. Also, each user has their own profile which will enable the robot to interact with different users according to their chosen fashion. In the case of a new user, the robot learns deviation handling actions specific to that user and hence collaborates more closely using the user specific preference.



Profile1	Profile2	Profile3
retry	*Pick&Place tray*	*retry*
retry	*retry*	*Pick&Place ring*
retry	*retry*	*giveup*

Profile4	Profile5
retry	*Pick&Place ring*
retry	*retry*
retry	*giveup*

(b)

Fig. 6. (a) Shows the time taken for the robot to handle deviations in presence and absence of the user assitance. (b) Deviation Handling Profiles of 5 users, learned over a course of 5 assembly process executions per user (total 25 Executions). The three entries in each profile column describe the deviation handling action chosen by that user for the assembly process steps (as given in Fig. 5) 2, 3 and 4 respectively.

6 Conclusion and Future Work

In this paper we have extended the IRL to enable the robotic system to handle deviations in an assembly process that occur during real-time execution. The eIRL exploits the presence of the human user in the human robot collaboration scenario by interacting with the user for assistance and feedback. The robotic system proposes a set of possible solutions to the user, given a deviation at a particular step in the assembly process. The proposed set of solutions is derived from the knowledge of the assembly process, already known optimal policy, robot capabilities and the set of objects currently present in the workspace. The user assistance is divided into a direct deviation action policy choice and a reward feedback. This enables the system to keep track of user preferences and interact with them more intuitively. An interesting future work would be to combine the already existing preferences of users (the choice of the existing users) and use it to interact with a new untrained user. Also, the evaluation was carried out with roboticists and we would like to extend this evaluation by including users with varying levels of experience with robots.

Acknowledgment. This research is funded by the projects KoMoProd (Austrian Ministry for Transport, Innovation and Technology), and CompleteMe (FFG, 849441).

References

1. Akkaladevi, S.C., Heindl, C.: Action recognition for human robot interaction in industrial applications. In: CGVIS 2015, pp. 94–99. IEEE (2015)
2. Akkaladevi, S., et al.: Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In: ICRA 2016, pp. 5644–5649 (2016)

3. Bauer, A., Wollherr, D., Buss, M.: Human-robot collaboration: a survey. Int. J. Humanoid Robot. **5**, 47–66 (2008)
4. Dautenhahn, K.: Methodology and themes of human-robot interaction: a growing research field. Int. J. Adv. Robot. Sys. **4**, 103–108 (2007)
5. euRobotics: Robitcs 2020 Strategic Research Agenda for Robotics in Europe (2013)
6. Goodrich, M.A., Schultz, A.C.: Human-robot interaction: a survey. Found. Trends Hum. Comput. Interact. **1**(3), 203–275 (2007)
7. Griffith, S., et al.: Policy shaping: integrating human feedback with reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 2625–2633 (2013)
8. Kartoun, U., et al.: A human-robot collaborative reinforcement learning algorithm. J. Intell. Rob. Syst. **60**(2), 217–239 (2010)
9. Knox, W.B., et al.: Interactively shaping agents via human reinforcement: the TAMER framework. In: Proceedings of International Conference on Knowledge Capture, pp. 9–16 (2009)
10. Knox, W.B., Stone, P., Breazeal, C.: Training a robot via human feedback: a case study. In: Herrmann, G., Pearson, M.J., Lenz, A., Bremner, P., Spiers, A., Leonards, U. (eds.) ICSR 2013. LNCS (LNAI), vol. 8239, pp. 460–470. Springer, Heidelberg (2013). doi:10.1007/978-3-319-02675-6_46
11. Lee, S., et al.: Human mental models of humanoid robots. In: IEEE International Conference on Robotics and Automation, pp. 2767–2772 (2005)
12. Rozo, L., et al.: Learning collaborative impedance-based robot behaviors. In: Twenty-Seventh AAAI Conference on Artificial Intelligence, pp. 1422–1428 (2013)
13. Rozo, L., et al.: Learning force and position constraints in human-robot cooperative transportation. In: Robot and Human Interactive Communication, pp. 619–624 (2014)
14. Scassellati, B.: Theory of mind for a humanoid robot. Autonomous Robots **12**(1), 13–24 (2002)
15. Suay, H.B., Chernova, S.: Effect of human guidance and state space size on interactive reinforcement learning. In: 2011 Ro-Man, pp. 1–6. IEEE (2011)
16. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction, vol. 1, no. 1. MIT Press, Cambridge (1998)
17. Tenorth, M., et al.: KnowRob: a knowledge processing infrastructure for cognition enabled robots. Int. J. Robot. Res. **32**, 566–590 (2013)
18. Thomaz, A.L., et al.: Reinforcement learning with human teachers: understanding how people want to teach robots. In: ROMAN 2006, pp. 352–357. IEEE (2006)
19. Universal Robots, UR10 robot - a collaborative industrial robot. http://www.universal-robots.com/products/ur10-robot/. Accessed 09 Aug 2016
20. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)