2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Adaptive Indirect Control through Communication in Collaborative Human-Robot Interaction

Rui Silva[1], Miguel Faria[2], Francisco S. Melo[2], Manuela Veloso[3]

Abstract— **This paper addresses the problem of human-robot collaboration in scenarios where a robot assists a human by executing a complex motion involving the manipulation of an object. We focus on tasks in which success in the task depends on reaching a target pose that is controlled by the human. We contribute a reinforcement learning-based approach that allows the robot to reason about its own ability to successfully complete the task given the current target pose and indirectly adjust that pose by prompting the human user. Our approach allows the robot both to trade-off the benefits of adjusting the target position against the cost of bothering the human user while, at the same time, adapting to each user's responses. Our approach was tested in a real-world human-robot collaboration scenario involving the Baxter robot.**

I. Introduction

In this paper we consider the human-robot collaboration problem, where a human and a robot jointly execute a predefined task. We focus on scenarios where the robot assists the human by executing a complex motion involving the manipulation of an object, which is successfully completed only upon reaching a target pose that is controlled by the human user. Examples of tasks that fit these characteristics include the joint preparation of a drink [1] or helping a human user dressing a piece of clothing [2], [3], [4]. In the former, the robot executes a pouring motion towards a target (cup), the pose of which is controlled by the human user. In the latter, the robot performs a dressing motion involving the manipulation of a piece of clothing, while the human controls the target pose by moving his/her body.

However, the ability of the robot to successfully execute the desired motion critically depends on the target pose. Certain poses will lie outside of the robot's reach; others may impose trajectories that are unsafe for the robot to execute or altogether impossible, given the configuration of the environment and obstacles that may exist. Or, in scenarios where the motion of the robot is learned from demonstration, the target may lie in a region where the robot is unable to properly generalize. Successfully completing the task will require the target to *move* to a more convenient pose.

The topic of cooperative object manipulation has been extensively investigated in the literature of human-robot interaction. Typical approaches consider that the human user leads the interaction by guiding the execution of the task, whereas the robot is expected to adapt its execution to that of the human. In this interaction paradigm, several approaches rely on having the robot predicting the human motion and intentions, and act accordingly [5], [6], [7], [8].

Recent years have seen a shift in this paradigm, with new approaches starting to place more responsibility and initiative on the side of the robot. An example of this is the recent effort on improving the legibility of the trajectories executed by the robot, in order to make it easier for human users to interpret the intentions of the robot [9], [10], [1].

More recent approaches to address robot limitations take inspiration from human-human interactions, where humans typically enroll the assistance from other humans to ease their own execution. For example, when assisting someone to dress a jacket, we will naturally ask the other to adjust his/her position so that we may more easily assist him/her. Such approaches rely on the typical compliance of human users to requests from others. For example, Rosenthal, Biswas and Veloso pioneered the concept of *symbiotic autonomy*, where a robot explicitly considers its own limitations and the existence of humans in the environment, and *plans* to enroll human assistance in the execution of its task [11]. In another work involving a dressing task, the robot determines whether or not it is able to execute the desired motion given the position of the human user [3]. When that is not the case, the robot informs the user, asking him/her to move to a more convenient position. In a more recent work, the robot trades off the benefits arising from enrolling the human assistance versus the cost of disturbing the user with a request for positioning [4].

In this paper, we follow on this line of work and propose a novel approach that allows a robot to proactively voice assistance requests to the human teammate. Instead of a trade-off between direct task execution and a request to reposition, we propose a reinforcement learning approach in which the robot effectively engages in a process of *indirect control* over the target pose, voicing successive requests that direct the user towards a pose in which task execution can be safely completed. As in previous works, our approach also trades-off the costs of disturbing the user against the benefits

*This work was partially supported by national funds through the Portuguese Fundação para a Ciência e a Tecnologia under project UID/CEC/50021/2013 (INESC-ID multi annual funding) and the Carnegie Mellon Portugal Program and its Information and Communications Technologies Institute, under project CMUP-ERI/HCI/0051/2013. Rui Silva acknowledges the PhD grant SFRH/BD/113695/2015.

[1] Rui Silva is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; and with Instituto Superior Técnico, University of Lisbon, Porto Salvo, Portugal. E-mail: ruisilva@cmu.edu

[2] Miguel Faria and Francisco S. Melo are with INESC-ID and with Instituto Superior Técnico, University of Lisbon, Porto Salvo, Portugal. E-mail: miguel.faria@tecnico.ulisboa.pt, and fmelo@inesc-id.pt

[3] Manuela Veloso is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA E-mail: mmv@cs.cmu.edu

978-1-5386-2682-5/17/$31.00 ©2017 IEEE

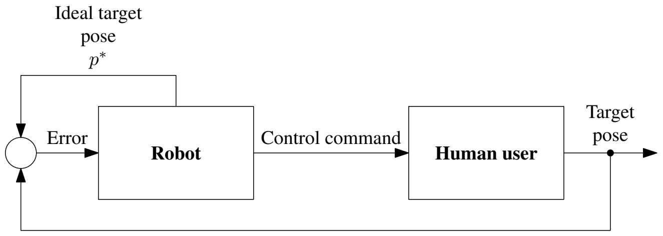


Fig. 1. The problem of indirectly adjusting the target pose depicted from a classical control perspective.

of its repositioning. However, this is done implicitly as part of the indirect control process. Additionally, by observing the response of human users to the robot's requests, our approach effectively adjusts its actions so as to adapt to possible limitations of the user. This feature is particularly useful when dealing with users with disabilities or in cluttered scenarios, where the user may not be able to comply to all the requests from the robot.

We model the robot's decision process as a Markov Decision Process [12] and use reinforcement learning to learn the best indirect control policy. We show that our approach is able to successfully direct human users to more convenient poses while trading off the benefits of human assistance with the cost of disturbing the user. We test our approach in a real-world collaboration scenario involving the Baxter robot, where the latter assists a human user to put on a backpack.

II. Problem Statement

As discussed in the previous section, we are interested in problems where the robot must execute some complex motion with respect to a target pose. The target pose is controlled by a human user and the robot can indirectly control the target's pose by communicating with the human—essentially, by voicing specific requests. The human may or may not comply with the robot's requests, and there is a cost associated with each request. Additionally, there is a cost associated with the execution of each motion, which greatly depends on whether the motion succeeds or not. The goal of the robot is to determine which command to execute at each time, so as to minimize the incurred cost and maximize the probability of executing a successful motion.

We can look at this class of problems from a traditional control perspective, as illustrated in Fig. 1. Roughly speaking, the robot internally assesses whether the current pose of the target corresponds to the "ideal pose" for execution. Depending on how much that pose deviates from the ideal pose, the robot will determine which control command to issue in order to balance the cost associated with each request and the cost associated with moving to a difficult pose.

From a control perspective, there are several challenges involved in the problem described above. First of all, there is a significant uncertainty/variability regarding the user's response to the robot's commands. Although that uncertainty may be captured probabilistically, the robot should take it into consideration when selecting the commands to issue. Secondly, even ignoring the variability in the user's response, there is no *a priori* model on how the user responds.

To address the first challenge, we propose the use of a decision-theoretic model, a *Markov decision process* (MDP), to represent the control problem summarized in Fig. 1. To address the second challenge, we propose the use of a reinforcement learning approach. We describe the MDP model in this section and postpone the description of the reinforcement learning approach until the next section.

An MDP is a tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, c, \gamma)$, where:

- $\mathcal{X}$ is the set of states. In the class of problems that we focus on, the state will typically contain information regarding the target pose for the task at hand.
- $\mathcal{A}$ is the action repertoire for the robot. In our case, we can write $\mathcal{A} = \mathcal{A}_{\text{comm}} \cup \mathcal{A}_{\text{exec}}$, where $\mathcal{A}_{\text{comm}}$ is the set of the *communication actions* (the requests voiced by the robot) and $\mathcal{A}_{\text{exec}}$ is the set of *execution actions*, corresponding to the different motions that the robot can perform.
- P represents the transition probability function. If $X(t)$ denotes the state of the system at time-step t,
$$\mathsf{P}(y \mid x, a) = \mathbb{P}\left[X(t+1) = y \mid X(t) = x, A(t) = a\right].$$
In other words, the transition probabilities $\mathsf{P}(y \mid x, a)$ provide a model of how the state changes depending on the actions of the robot. In our goal tasks, the transition probabilities encode the *response of the user to the robot's requests*.
- $c : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the cost, which in our case encodes both the cost incurred by communicating with the user and the execution cost.
- Finally, γ is a discount factor, indicating that rewards arriving earlier are preferable to rewards arriving later.

Solving the MDP consists in finding an *optimal policy* π^*, i.e., a mapping from states to actions which ensures that the robot, if selecting the actions as prescribed by π^*, incurs in as little cost as possible. In particular, the optimal policy π^* is such that

$$\pi^*(x) = \operatorname*{argmin}_{\pi} \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t c(X(t), \pi(X(t))) \mid X(0) = x\right],$$

for all $x \in \mathcal{X}$. Such policy can be found from the *optimal Q-function*, which can be defined recursively for every state and action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ as

$$Q^*(x, a) = c(x, a) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}(y \mid x, a) \min_{a' \in \mathcal{A}} Q^*(y, a').$$

The function Q^* can be computed using dynamic programming, if the MDP model is fully known, or via reinforcement learning, for example using the Q-learning algorithm [13]. In our specific scenario, and since the model for the human response is unknown, we adopt the latter approach, as described in the next section.

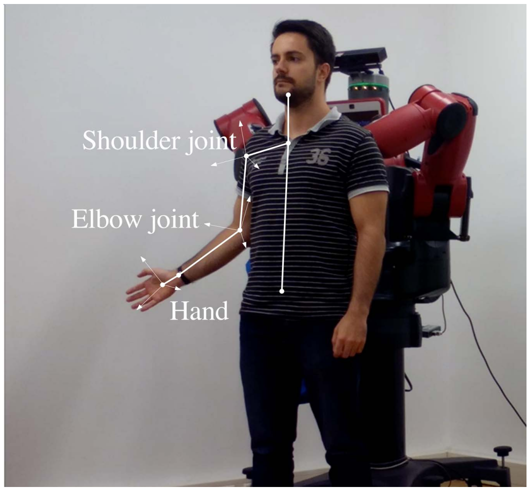


Fig. 2. The 3D positions of shoulder, elbow and arm provide the target pose for the execution of the robot's motion.

III. Reinforcement Learning Approach to Putting a Backpack

In this section we instantiate the indirect control problem defined in the previous section to the specific scenario of assisting a human to put a backpack using a 2-arm manipulator (the Baxter robot). We describe how the problem is modeled as an MDP as well as the reinforcement learning approach adopted to tackle that MDP.

A. The MDP Model

As discussed in the previous section, we propose to model indirect control problems using the MDP formalism. That requires specifying the elements in the tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, c, \gamma)$.

a) The states.: For the task of putting on a backpack, the motion of the robot will involve passing a strap through the arm of the user up to the shoulder, and drop it there. The target pose, in this case, will depend on the position of the human—specifically the shoulder, arm and hand up which the robot will pass the strap. We thus consider as our state-space the 3D position of these 3 joints (see Fig. 2), where each state $x \in \mathcal{X}$ is thus a 9-dimensional real valued vector.

b) The actions.: As discussed in the previous section, we consider $\mathcal{A} = \mathcal{A}_{\text{comm}} \cup \mathcal{A}_{\text{exec}}$, where $\mathcal{A}_{\text{comm}}$ is the set of the *communication actions* and $\mathcal{A}_{\text{exec}}$ is the set of *execution actions*. In our case we have a single execution action, EXECUTE, corresponding to the motion of putting the strap up the user's arm. This motion was taught to the robot by demonstration (more on this ahead).

The communication actions correspond to the voice requests that the robot will direct to the human user. In our case, we consider a total of 5 actions which provide the robot with the ability to indirectly control the target pose, as intended: MOVE FORWARD, MOVE BACKWARD, MOVE LEFT, MOVE RIGHT, RAISE RIGHT ARM. Each of these actions has a voice command associated.

c) The cost function.: The cost function c specifies, for each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ a scalar value that measures the immediate utility of executing action a in state x. We design our cost function taking into consideration the cost-benefit trade-off involved in requesting human-assistance. In particular, when designing the cost function used in this task, we followed several rules of thumb which are relevant for the type of problems considered in this paper:

- *We penalize unsuccessful executions.* An unsuccessful execution corresponds to a failure in the target task, which should be penalized.
- *We reward successful executions.* Conversely, a successful execution corresponds to a task successfully completed, which should be rewarded.
- *We penalize a large number of interactions*. In order to prevent the robot from engaging in successive requests to fine tune the target's pose, we penalize each request posed by the robot.
- *We penalize physically demanding requests*. Depending on their individual circumstances, different human users will find it easier or harder to oblige a request from the robot. In those situations where it is known beforehand that the user has certain limitations (for example, a broken arm), we penalize some requests more heavily (for example, a request to raise the arm), to indicate that the robot should take those limitations into consideration.

Guided by the general principles above, we use

$$c(x, a) = \begin{cases} 5C & \text{if } a = \text{EXECUTE and failed;} \\ C & \text{if } a \in \mathcal{A}_{\text{comm}}; \\ -C & \text{if } a = \text{EXECUTE and succeeded,} \end{cases}$$

where C is a constant adjusted empirically. Roughly speaking, the cost function establishes 6 as an acceptable number of human assistance requests. After 6 assistance requests, even if the robot decides to execute the task and succeeds, the total cost will be equivalent to an immediate failed attempt. We note, however, that the failure or success of the EXECUTE action are difficult to estimate from the pair (x, a) alone, for which reason the cost $c(x, a)$ is hard to specify as a deterministic function of (x, a).

d) The transition probabilities.: Since the transition probabilities depend on how the human user responds to the robot's requests, we do not have, beforehand, a reasonable model of how the actions of the robot impact the target pose. In fact, for most tasks fitting into the general setting considered in this paper, it may be hard or even impossible to define such a transition probability function beforehand. And, while such transition function could be learned from data, this would involve actual interactions with human users—that we rather use to directly learn the policy for the robot.

B. Reinforcement Learning Approach

Reinforcement learning (RL) allows an agent (typically modeled as an MDP) to learn, by trial-and-error, an optimal mapping from the situations it encounters to the actions it has available. During this trial-and-error, the agent must balance between exploring untried actions and exploiting whatever knowledge it already collected to decide what to do in each situation. And, depending on the initial policy, RL methods may require a lot of experimentation/data before a reasonable policy is obtained.

In our case, data must be obtained from interactions with human users and is thus expensive to acquire. To make the most of the data, we adapt the Dyna-Q approach to our scenario, breaking the learning problem in two stages [14]:

- In a first stage, we collect some data from real interactions with human users. This data is used to build an initial estimate for P and c.
- The estimate for P and c is then used in a second stage to generate additional (simulated) data, which is used to build an initial estimate of the optimal Q-function for the MDP using standard Q-learning.
- Finally, the estimate for Q^* is refined during additional interactions with real users.

Q-learning updates the estimate for Q^* from observed transitions (x_t, a_t, c_t, x_{t+1}) using the rule

$$\begin{aligned} Q^{(t+1)}(x,a) = Q^{(t)}(x,a) + \alpha_t(x,a)\big(c_t \\ + \gamma \min_{a'} Q^{(t)}(x_{t+1},a') - Q^{(t)}(x,a)\big), \end{aligned} \tag{1}$$

where $Q^{(t)}$ is the t-th estimate and α_t is a step-size sequence. However, the standard update in (1) requires a tabular representation for Q^*. In our case, due to the continuous nature of the state-space, this is not possible. We instead combine Q-learning with *linear function approximation*: the estimates $Q^{(t)}$ are represented in the form

$$Q^{(t)}(x,a) = \sum_{i=1}^{p} \phi_i(x)\theta_{i,a}^{(t)} = \boldsymbol{\phi}^\top(x)\boldsymbol{\theta}_a^{(t)},$$

where each $\phi(x)$ is a feature vector describing the state x, and $\boldsymbol{\theta}_a^{(t)}$ is a vector of parameters for action a. Using this representation, the update in (1) becomes

$$\begin{aligned} \boldsymbol{\theta}_a^{(t+1)} = \boldsymbol{\theta}_a^{(t)} + \alpha_t(x,a)\boldsymbol{\phi}(x)\big(c_t \\ + \gamma \min_{a'} Q^{(t)}(x_{t+1},a') - Q^{(t)}(x,a)\big). \end{aligned} \tag{2}$$

Using our approach, the robot learns how to *reason* about its ability to successfully complete the target task, *request* the assistance of the human user and *adapt* to that user, as new interaction episodes occur. At each episode, the robot faces different intermediate and goal target poses, and successively voices human assistance requests before trying to execute the task. At the end of each episode, the robot either succeeds or fails, but always incorporates this new experience by taking into account the rewards obtained throughout the episode.

IV. Experimental Evaluation

We now describe an evaluation of our proposed approach using the Baxter robot in the backpack task. In this task, the robot holds a backpack and assists the user to put in the right strap. We describe our experimental setup, deployment details and main results obtained with our approach.

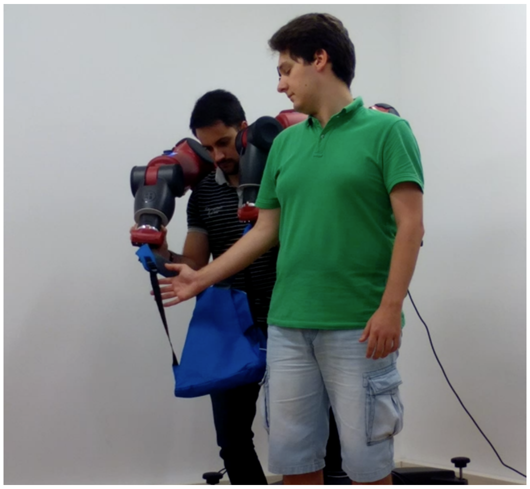


Fig. 3. Kinesthetic teaching of the intended motion: the human teacher shows the robot how to put the right strap of the backpack on a user.

A. Experimental Setup

The Baxter robot is a two-manipulator robotic platform designed to work alongside human users. Each manipulator has 7 degrees of freedom and an actuated end gripper that is used to hold the backpack. To determine the position of the human user, we use a Microsoft Kinect pointing towards the Baxter robot, and OpenNI tracker to compute the human joint positions (see Fig. 2).

B. Learning from Demonstration

The motion of putting the backpack was taught to the robot using kinesthetic teaching (see Fig. 3). The robot was provided with 23 demonstrations of the intended motion, with the user placed in different positions in front of the robot. For each position of the user, the Kinect provided the target pose, consisting of the 3D position of the right shoulder, elbow and hand. The demonstrated trajectories were stored in joint space.

The demonstrated trajectories and the target pose were used to build a *cooperative probabilistic motion primitive* (CoProMP) [15], a parameterized distribution over the space of trajectories. Tasks such as modulating the trajectory to a particular goal can be achieved by standard probability operations such as conditioning.

C. MDP Learning and Simulation

As described in the previous section, we use simulated data to build an initial estimate of the optimal Q-function. In order to build the simulation model, we first built estimates of the transition probabilities P for each of the request actions of the user. We adopt a simple dynamic model that can be summarized in the expression

$$x_{t+1} = x_t + d(a_t),$$

where x_t is the target pose at time-step t, a_t was the request issued by the robot at that time, and $d(a_t)$ was the observed displacement. We built a distribution $p_d(\cdot \mid a)$ as a function of a, and adopted, as transition model,

$$\mathsf{P}(y \mid x, a) = p_d(y - x \mid a).$$

Specifically, for each action $a \in \mathcal{A}_{\mathrm{comm}}$ we collected a total of 50 samples from two human users responding to the robot's requests. We used Gaussian kernel density estimation to represent $p_d(\cdot \mid a)$, with the bandwidths selected through grid search with 5-fold cross validation.

To estimate the average cost $c(x, a)$ for each pose x and execution action a, we set the constant $C = 1$ in the definition of the cost function, and used a Support Vector Machine (SVM) classifier, trained to predict the success or failure of an execution, given a target pose. The training set contained a total of 48 samples, 23 of which from the successful examples demonstrated to build the CoProMP and the remaining collected from further experiments. In these experiments, the robot always tries to execute the motion and the target pose is randomly selected by two human users. The final dataset contained a total of 28 positive samples and 20 negative samples. The kernel coefficient and error term penalty parameters were tuned using a grid search with 2-fold cross validation.

D. Function approximation

We used an approximation of radial kernel to represent Q^* in our reinforcement learning algorithm. In particular, we adopted the *random kitchen sinks* method of Rahimi and Recht [16], [17], and used the implementation in the *scikit-learn*[1] machine learning library for Python.

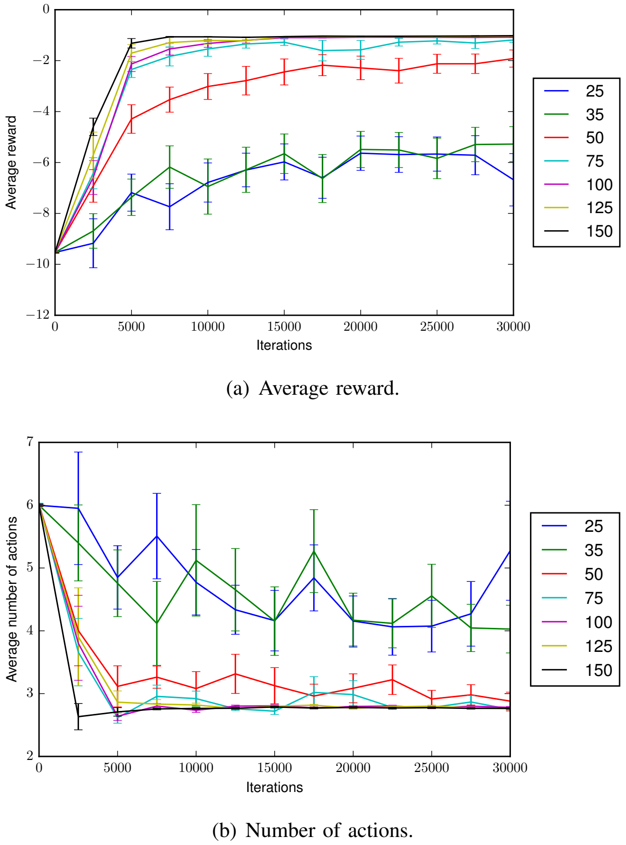
In order to determine an appropriate dimensionality for the feature transformation, we ran multiple simulations and observed the effects in the expected reward and number of actions. We ran our RL algorithm for $30,000$ iterations, measuring the average reward obtained every $2,500$ iterations across $2,500$ episodes. Results were averaged over 40 runs starting from different seeds, and are reported in Fig. 4.

As expected, higher feature dimensions result in better performances both in terms of average reward and number of actions executed. Note that, with 150 basis functions, the reward and number of actions executed converge to values close to -1 and 2.5, respectively. This indicates that, on average, an episode will consist in one to two communication actions followed by a successful execution action.

Based on the results obtained in simulation, we use a total of $p = 150$ basis functions in our feature representation.

E. Experiments

We tested our system on the real Baxter robot, using the parameters θ learned after $30,000$ iterations in the simulation model, with $p = 150$ basis functions and the seed that led to the best results. Specifically, we performed 10 trials with a real user who was was given the freedom to choose the starting position. The robot communicated its requests through voice commands such as *"Please raise your right arm"*. Table I summarizes the results obtained in terms of the average reward and the number of actions executed by the robot in each episode.



(a) Average reward.

(b) Number of actions.

Fig. 4. Performance of the Q-learning algorithm for different feature transformation dimensions. We ran the algorithm for $30,000$ iterations, measuring the average reward obtained every $2,500$ iterations across $2,500$ episodes. Results were averaged over 40 runs and the bars represent the standard error. Note that 4(b) includes both the communication and execution actions.

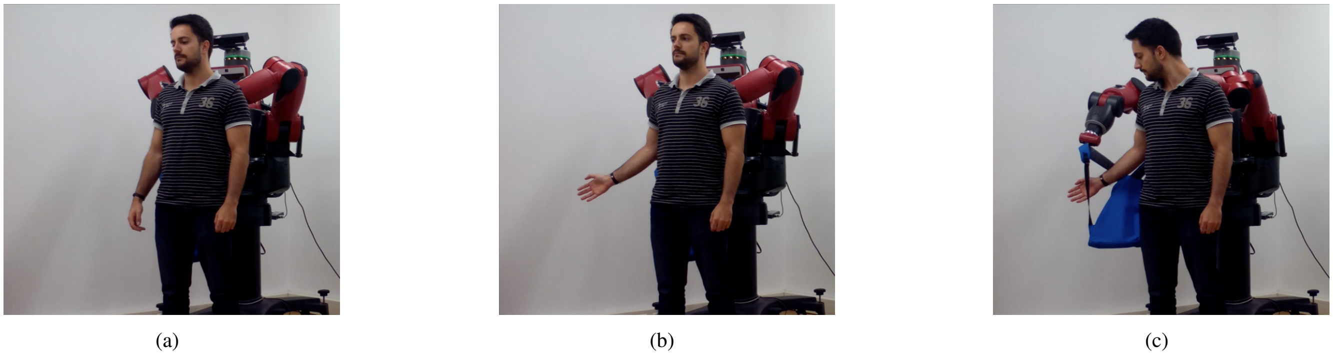
Two observations are in order. First of all, the robot is able to execute successfully in all trials, even though the initial pose of the human user varied significantly. This shows that our approach is, indeed, able to indirectly control the target pose by means of vocal commands. Second, the numerical results (concerning reward and number of actions) are very similar to those obtained in the simulation. This suggests that, even though simple, the simulation model adopted was capable of modeling the real task adequately.

TABLE I

RESULTS ACHIEVED WITH THE REAL ROBOT. PARAMETERS WERE INITIALIZED AFTER 30,000-STEP SIMULATION RUN. RESULTS ARE AVERAGED OVER 10 TRIALS.

	Mean	Std
Total reward	-1.5	0.67
Number of actions	3.5	0.67
% Successful executions	100	–

[1] http://scikit-learn.org/



(a) (b) (c)

Fig. 5. Sequence of steps in the task where the robot helps a human user putting on the right strap of a backpack. 5(a) The human user starts with his right arm in a lowered position. 5(b) The human user raises his arm by request of the robot. 5(c) The robot putting the backpack on the human user.

V. Conclusions

In this paper we presented a novel approach that explores the concept of adaptive *indirect control* through communication, in the context of human-robot collaborative tasks, where the robot assists the human by executing a complex motion that is successfully completed only upon reaching a target pose that is controlled by the user.

Our approach allows the robot to engage in a process of *indirect control* over the target pose, by voicing successive requests to the human user. These requests direct the user towards a target pose that allows for a successful execution of the task. Moreover, this indirect control process also considers the cost-benefit trade-off of the assistance requests.

We tested our approach in a real-world human-robot collaboration scenario using the Baxter robot, where the latter helps a human user putting on a backpack. Results show that our approach effectively allows a robot to indirectly control the target pose through of vocal commands.

The approach proposed can be generalized to more complex scenarios. In scenarios where $\mathcal{A}_{\text{exec}}$ includes multiple *execution actions*, the reinforcement learning approach computes the optimal indirect control policy that allows the robot to select the best execution motion. In scenarios with multi-step tasks, the state-space must include information on the current state of the task, and the reward function should penalize out-of-order state sequences. For example, the approach generalizes to the task where the robot helps a human dressing both straps of the backpack, and where the robot is taught multiple strategies on how to put each strap.

This work also opens interesting directions for future work. We believe the approach proposed is not limited to human-robot interaction scenarios, and that it can be used with generic agents. We envision applications in robot-robot interaction scenarios where, due to integration issues, the robots must communicate through a coarse set of commands. There is also recent research in *ad hoc* teamwork that seeks to develop strategies to enable a generic agent to quickly infer the strategies of its teammates and act accordingly, towards the joint completion of a common task [18]. The challenges faced in this area bare a close resemblance to those in the human-robot collaboration problems we focused on.

References

[1] A. Dragan, S. Bauman, J. Forlizzi, and S. Srinivasa, "Effects of robot motion on human-robot collaboration," in *Proc. 10th ACM/IEEE Int. Conf. Human-Robot Interaction*, 2015, pp. 51–58.

[2] Y. Gao, H. Chang, and Y. Demiris, "User modelling for personalised dressing assistance by humanoid robots," in *Proc. 2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015, pp. 1840–1845.

[3] S. Klee, B. Ferreira, R. Silva, J. Costeira, F. Melo, and M. Veloso, "Personalized assistance for dressing users," in *Proc. 7th Int. Conf. Social Robots*, 2015, pp. 359–369.

[4] R. Silva, F. Melo, and M. Veloso, "Adaptive symbiotic collaboration for targeted complex manipulation tasks," in *Proc. 22nd Eur. Conf. Artificial Intelligence*, 2016, pp. 863–870.

[5] Y. Maeda, T. Hara, and T. Arai, "Human-robot cooperative manipulation with motion estimation," in *Proc. 2001 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2001, pp. 2240–2245.

[6] B. Corteville, E. Aertbelien, H. Bruyninckx, J. De Schutter, and H. Van Brussel, "Human-inspired robot assistant for fast point-to-point movements," in *Proc. 2007 IEEE Int. Conf. Robotics and Automation*, 2007, pp. 3639–3644.

[7] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion," in *Proc. 2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2013.

[8] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," in *Proc. 2015 IEEE Int. Conf. Robotics and Automation*, 2015, pp. 885–892.

[9] A. Dragan and S. Srinivasa, "Generating legible motion," in *Robotics: Science and Systems*, 2013.

[10] M. Gielniak, C. Liu, and A. Thomaz, "Generating human-like motion for robots," *Int. J. Robotics Research*, vol. 32, no. 11, 2013.

[11] S. Rosenthal, J. Biswas, and M. Veloso, "An effective personal mobile robot agent through symbiotic human-robot interaction," in *Proc. 9th Int. Joint Conf. Autonomous Agents and Multiagent Systems*, 2010.

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

[13] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, 1989.

[14] R. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. 7th Int. Conf. Machine Learning*, 1990, pp. 216–224.

[15] G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 527–534.

[16] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20*, 2007, pp. 1177–1184.

[17] ——, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1313–1320.

[18] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, "Ad hoc autonomous agent teams: Collaboration without pre-coordination," in *Proc. 24th AAAI Conf. Artificial Intelligence*, 2010, pp. 1504–1509.