Proceedings Paper:
Sun, C., Cohn, A.G. orcid.org/0000-0002-7652-8907 and Leonetti, M. (2023) Online Human Capability Estimation Through Reinforcement Learning and Interaction. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 01-05 Oct 2023, Detroit, MI, USA. IEEE , pp. 7984-7991. ISBN 9781665491907

https://doi.org/10.1109/iros55552.2023.10341868

Online Human Capability Estimation through Reinforcement Learning and Interaction

Chengke Sun[1], Anthony G. Cohn[1] and Matteo Leonetti[2]

***Abstract*— Service robots are expected to assist users in a constantly growing range of environments and tasks. People may be unique in many ways, and online adaptation of robots is central to personalized assistance. We focus on collaborative tasks in which the human collaborator may not be fully able-bodied, with the aim for the robot to automatically determine the best level of support. We propose a methodology for online adaptation based on Reinforcement Learning and Bayesian inference. As the Reinforcement Learning process continuously adjusts the robot's behavior, the actions that become part of the improved policy are used by the Bayesian inference module as local evidence of human capability, which can be generalized across the state space. The estimated capabilities are then used as pre-conditions to collaborative actions, so that the robot can quickly disable actions that the person seems unable to perform. We demonstrate and validate our approach on two simulated tasks and one real-world collaborative task across a range of motion and sensing capabilities.**

I. INTRODUCTION

Assistive and service robots are expected to help people in a wide variety of settings, from factories to care homes. The range of people that robots will support is large and unpredictable. We consider robots performing collaborative tasks with the goal of providing personalized support at the right level for each person. People may be differently able for a number of reasons, such as chronic conditions, injury, or simply age. We want the robot to determine how to best assist automatically and online.

We propose a framework for human capability estimation based on Bayesian inference and online Reinforcement Learning (RL). We focus our modeling effort on the robot, rather than the person, and determine what human capability each collaborative action that the robot can execute may require. For instance, consider a robot giving a guided tour. The robot can drive fast, which would require the person to move fast. A person may not be able to move fast for a number of reasons (old age, injury, handicap...), but the robot will not attempt to estimate the underlying condition. It focuses on the pre-condition of its own actions and if it acquires enough evidence that the person cannot benefit from that action it disables the action.

The capability estimation provides another layer of abstraction, orthogonal to the state space, and not observable by the robot. Capabilities are a property of the person, and as such are latent variables. We propose a Bayesian framework for capability estimation that gathers evidence through RL without deploying the more general and computationally intensive model of Partially Observable Markov Decision Processes (POMDPs). In the previous example of the guided tour, the RL agent may receive a low reward because by driving fast it is leaving the person behind. The agent ties this information to the observable state space, and learns that it cannot drive fast with that person *at that particular location*. It will have to receive low rewards across a range of locations to generalize that driving fast is just never appropriate for this person. With the additional capability estimation, on the other hand, the latent capability to move fast can be detected early on, and the action can be disabled everywhere when enough evidence has been accumulated.

A plethora of approaches have been developed for adaptation and personalization using demonstrations, preference learning, statistical inference, and POMDP planning, to mention a few. To the best of our knowledge, this is the first method using the behavior learned by an RL agent as evidence for human capability. We introduce an online learning component that requires no special calibration, demonstrations, question answering, or user-specific tuning. While we propose to gather evidence through RL, the framework is more general, and can be integrated with other existing approaches to provide further information to the Bayesian estimation, for example through computer vision. In this work, we focus on adaptation through interaction and learn exclusively from the robot's actions.

We implemented our framework in a PAL Robotics TIAGo[1], and demonstrated it in three collaborative tasks, two in simulation, and one in the real robot. We demonstrate that learning capabilities indirectly through RL allows the robot to adapt much faster than by RL only. Furthermore, the capability estimation enables online personalized support, contributing to the democratization of collaborative robotics.

II. RELATED WORK

User adaptability of service robots has been widely researched in the robotic community. Users prefer robots that respect human conditions and preferences [1]. Preferences have been organized in a taxonomy for physically assistive robots involving human capabilities as parameters that affect how robots' actions are performed [2]. Fischinger et al. present an adaptive system where the robot actively asks the user for preferences to keep related parameters as individual configurations [3]. In our work, capabilities are not global

[1] School of Computing, University of Leeds. {sc17cs, a.g.cohn}@leeds.ac.uk

[2] Department of Informatics, King's College London. matteo.leonetti@kcl.ac.uk

[1] https://pal-robotics.com/robots/tiago/

but task-related (as a consequence of model-free RL), and we learn them through acting rather than by asking questions.

Much research on user-adaptable robots is rooted in the framework based on stochastic dynamic programming, chiefly among all, POMDP approaches. Broz et al. introduced a time-indexed POMDP model that contains hidden user intentions [4]. Another approach is based on Mixed Observability Markov Decision Process (MOMDP), which uses hierarchical states to simplify the POMDP computation [5]. A POMDP model encoding multiple Human-robot Interaction (HRI) variables has also been applied to an intelligent wheelchair [6]. Our approach is also based on latent variables, but does not require the fully-fledged POMDP modeling. The state space is treated just as an MDP, and the capabilities are estimated separately. Model-free learning is used online, circumventing complex POMDP planning.

Another category of approaches is those based on statistical inference. Abdo et al. present a system that learns user preferences from pre-collected ratings and makes predictions using collaborative filtering [7]. The pre-trained user model can adapt to new users rapidly by matching known characteristics. Bayesian inference has been applied to estimate user intention [8]. Martins et al. introduce a user-adaptive service-selecting model using Bayesian programming [9]. In our work, Bayesian inference does not produce the action that the robot executes, which is instead obtained through RL. It is used as a side-effect of RL, to estimate human capabilities as action pre-conditions. The outputs of an independent user labeling module can be generalized to diverse control flows. For example, planners based on symbolic reasoning can adjust social interaction parameters using capability labels. Martins et al. present a distributed learning system to infer the characteristics of users [10]. Our work focuses on human capabilities that are prerequisites for collaborative actions in task planning. We further show how the estimated capability labels make the learning process user-adaptable.

Lastly, existing work aims to learn adaptive behaviors from guidance and human feedback explicitly. Senft et al. present a supervised Reinforcement Learning framework to learn personalized behaviors from expert guidance [11]. The robot can also learn from user feedback [12] [13] [14]. Such approaches require an expert or user to keep track of the robot's behaviors and rely on providing proper human advice. Our work, instead, aims at adaptation without human demonstration.

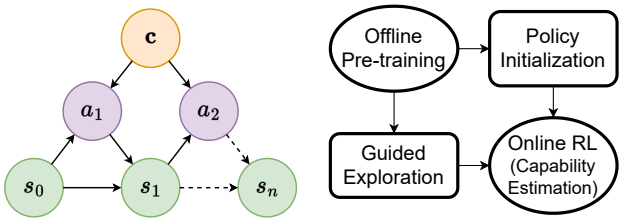
III. Problem Definition

We build on the common task model of Markov Decision Processes (MDPs) $D = \langle S, A, P_a, R_a, \gamma \rangle$, where S is a set of states, A is a finite set of actions, P_a is the transition function, R_a is the reward function, and $0 \leq \gamma \leq 1$ is the discount factor.

We augment this model with the capabilities of the human collaborator as preconditions to robot actions. For instance, in the tour guide example, the robot's action `drive_fast` depends on the person's capability to `move_fast`. We represent the capabilities as a vector of binary random variables $\mathbf{c}$, and denote with $2^{\mathbf{c}}$ the set of all possible binary vectors of capabilities $\mathbf{c}$. We define for each action the subset of the capabilities that are required for that action, and denote it with $C(a)$. In the example above, $C(\texttt{drive\_fast}) = \{\texttt{move\_fast}\}$. The set $C(a)$ must be specified by the designer along with the set of actions. An action a is *enabled* at time step i, if the probability of all the capabilities in $C(a)$ is above a given threshold d. The set of enabled actions at time i is, therefore, $A_i = \{a \in A \mid \forall c \in C(a); P(c = 1) >= d\}$.

The goal of the agent is to compute an optimal policy $\pi^*(a \mid s)$ that maximizes the expected return $G = E_\pi \left[\sum_{i=1}^{T} \gamma^{i-1} r_i\right]$, where r_i is the reward at time step i, extracted from R_a. The policy is only allowed to choose among enabled actions, that is, the domain of the random variable $\mathcal{A}$ in $\pi(\mathcal{A} = a \mid s)$ is A_i.

The explicit estimate of the probability of human capabilities allows the robot to accumulate evidence and, when the presence of a capability is sufficiently unlikely, to disable the corresponding action, regardless of the current state. A critical aspect of this formulation, therefore, is the estimation of human capabilities, for which we describe our approach in the next section.



(a) (b)

Fig. 1: (a) We model capabilities as random variables on which actions are conditioned. (b) The optimal policy under known capabilities must be available for the online Bayesian update, and is pre-trained. Since the pre-trained policy is available, it is also used to initialize the agent and to guide its exploration.

IV. Approach

We propose to estimate the probability of human capabilities indirectly, through the underlying continuous RL process. In this section, we introduce a Bayesian framework for capability estimation, while in the next section we discuss the implementation details and offline training.

A. Capability Estimation

We define a capability-conditioned policy $\pi(a \mid s, \mathbf{c})$ as shown in the Bayesian network in Figure 1(a). In our model, capabilities influence the effectiveness of actions, and therefore the actions are conditioned on them. The agent is trained offline under known capabilities, converging to the optimal policy $\pi^*(a \mid s, \mathbf{c})$ (cf. Section V-B).

The robot is deployed with a new user and expected to adapt to the user's actual capabilities as quickly as possible. The underlying RL process selects actions while improving

the policy and thereby generating a sequence of states and actions $\tau = \{(s_0, a_1), (s_1, a_2), ..., (s_{l-1}, a_l)\}$. The approach does not require an episodic task, and the sequence is possibly infinite, but we consider it in batches of length l. Given such a trajectory, the pre-trained policy can be used to accumulate evidence on human capabilities as follows:

$$P(\mathbf{c} \mid \tau) = \frac{P(\tau \mid \mathbf{c})P(\mathbf{c})}{P(\tau)}. \tag{1}$$

The likelihood in Equation 1, for a given policy π, unfolds to:

$$\begin{aligned} P(\tau \mid \mathbf{c}) =& P(s_0)\pi(a_1 \mid s_0, \mathbf{c})P_a(s_1 \mid s_0, a_1)\pi(a_2 \mid s_1, \mathbf{c}) \cdots \\ & P_a(s_{l-1} \mid s_{l-2}, a_{l-1})\pi(a_l \mid s_{l-1}, \mathbf{c}). \end{aligned} \tag{2}$$

The probability of the trajectory can be factored into two parts, one that depends on the capabilities, and one that does not:

$$P(\tau \mid \mathbf{c}) = P(s_0) \prod_{i=1}^{l-1} P_a(s_i \mid s_{i-1}, a_i) \prod_{i=1}^{l} \pi(a_i \mid s_{i-1}, \mathbf{c}). \tag{3}$$

The first two terms in the right-hand side of Equation 3, which do not depend on the capabilities, appear in both the numerator and the denominator of Equation 1 (that is, in both $P(\tau \mid \mathbf{c})$ and $P(\tau)$) and therefore cancel each other out. The estimate of the capabilities can then be obtained from the optimal capability-conditioned policy and a trajectory through the Bayesian update:

$$P(\mathbf{c} \mid \tau) = \frac{\prod_{i=1}^{l} \pi^*(a_i \mid s_{i-1}, \mathbf{c})P(\mathbf{c})}{\sum_{\mathbf{k} \in 2^{\mathbf{c}}} \prod_{i=1}^{l} \pi^*(a_i \mid s_{i-1}, \mathbf{k})P(\mathbf{k})}. \tag{4}$$

The resulting multi-label classification can be simplified in case of dependency between capabilities, but for this paper we consider the full set of $|2^{\mathbf{c}}|$ combinations, and therefore work with a small number of capabilities.

V. Implementation and Deployment

The proposed method is composed of the following steps as shown in Figure 1(b): (i) pre-training of the capability-conditioned policy $\pi^*(a \mid s, \mathbf{c})$ under known capability combinations; (ii) initialization of the value function of the RL agent using the pre-trained value function; (iii) online RL with a new user, while continually estimating capabilities using Equation 4 and enabling or disabling actions; (iv) taking advantage of the pre-learned policy for exploration. In the rest of this section, we discuss each step. We will use the first task of our experimental evaluation as a running example, and therefore begin with its description.

A. Robot Navigation Task

In this task, the robot has to lead a person to a desired location. The Gazebo[2] simulation environment is shown in Figure 3 (a). The robot starts from a fixed initial location, marked in green, and is requested to lead the person to the goal location, in blue. Two red doors provide shortcuts in the environment.

The state space is composed of the robot's position, the observed distance between the robot and the human, and whether each door is open or closed. The action space consists of nine actions: four navigation actions in the four cardinal directions both driving fast and slow, and the action `open_door` to ask the collaborator to open doors and take the shortcut. The dimensions of the navigation map are 7×6 units. Slow-driving actions make the robot move 1 unit per step, while fast-driving actions make the robot move by 2 units. Two human capabilities, $\mathbf{c} = \{\texttt{c\_fast}, \texttt{c\_open}\}$, are used, to enable the fast navigation actions and the `open_door` action. The human model always catches up to the robot if $\texttt{c\_fast} = 1$, otherwise it moves by 1 unit per step. Significant time may be saved by passing doors and taking shortcuts. Every step will result in a negative reward of -1. Reaching the goal will bring a reward of 50. When the distance between the robot and the collaborator exceeds 1 unit, the agent will be assigned a negative reward of -50. To make the environment stochastic and more realistic, we introduce the following features: The robot's perceived distance is affected by Gaussian noise ($\mu = 0, \sigma = 0.05$); Each door may be open at the start of every episode with probability $P = 0.1$; The human model makes slow steps even if $\texttt{c\_fast} = 1$ with $P = 0.05$; the collaborator may refuse to open doors even if $\texttt{c\_open} = 1$ with $P = 0.05$.

B. Pre-training

The online capability estimation hinges on having the optimal policy $\pi^*(a \mid s, \mathbf{c})$, which must be trained offline under a sufficiently representative, known, set of capabilities. This training can be entirely model-free, executing the task in the real world with a group of people with different known capabilities, but such an effort would be considerable and, in many cases, unrealistic. In this paper, we rely on simulations modeling the behavior of people with different capabilities, and on which the training to learn the optimal policy $\pi^*(a \mid s, \mathbf{c})$ can be carried out offline. The simulation will inevitably be different from real users, but the capability estimation can be made tolerant to uncertainty by selecting a sufficiently low threshold d, so that an action is disabled only when the system is highly confident that the capability is not present.

In the navigation task, training results in four policies, one for each combination of $\texttt{c\_open} = \{0, 1\}$ and $\texttt{c\_fast} = \{0, 1\}$. Having the pre-trained policies $\pi^*(a \mid s, \mathbf{c})$ and their value functions, the robot can take advantage of them both as initialization and to guide online learning, as described in the following sections.

C. Policy Initialization

The underlying online learning is generalized policy iteration, and its convergence is not affected by the initial value function (in MDPs). However, learning can be sped up significantly with a favorable initialization, and the pre-training phase may provide such further benefit.

[2]https://gazebosim.org/

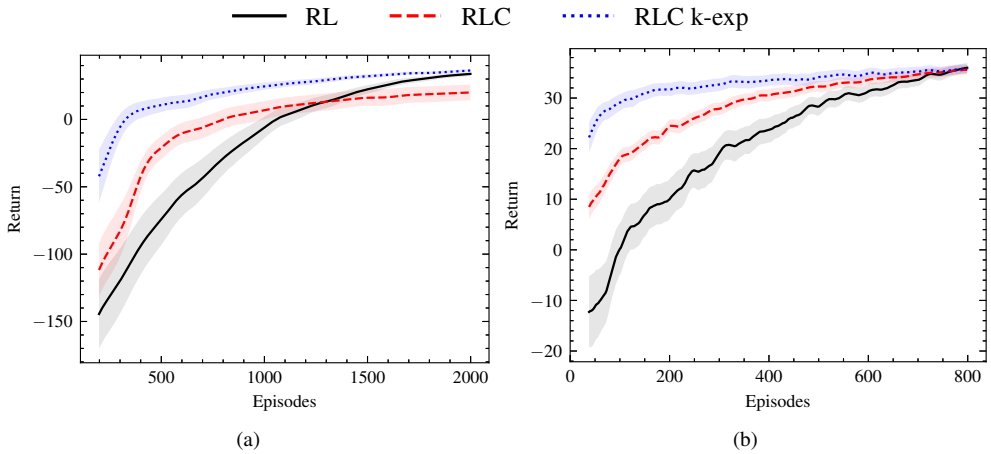


Fig. 2: The average return of the navigation task: (a) Agents are initialized with the pre-trained policy over fully available capabilities. (b) Agents are initialized with the pre-trained policy over random capabilities.

The robot has a set of $|2^{\mathbf{c}}|$ value functions to choose from and use while adapting to the particular user. The policy trained with a user with all capabilities, that is, $\pi^*(a \mid s, \{1, 1, \ldots, 1\})$ may seem like a natural choice: since all actions are enabled in this policy it is an optimistic initialization, and then exploration may determine which actions to disable. However, the value function of this policy relies on all actions, and if some actions are disabled during learning it becomes invalid, and must be relearned almost entirely. Committing to any initial capability combination has a risk of similar overfitting. Therefore, a better option for initialization is a value function pre-trained over random capabilities.

We demonstrate this phenomenon in the navigation task. Figure 2 shows the return of the pure RL agent, the RL agent using capability estimates (RLC) and the same agent using both initialization and exploration (RLC k-exp) from pre-training (cf. Section V-E). All three agents start with the same initial value function, which for Figure 2(a) is the value function of $\pi^*(a \mid s, \{1, 1\})$ and in Figure 2(b) is pre-trained across uniformly random capabilities. The agents are then deployed separately with collaborators of all 4 combinations, and the plot shows the average reward across all. All three agents perform better when initialized with the policy pre-trained using random capabilities. Note that the initialization of the value function is distinct from the prior over capabilities. The agent starts with the value function pre-trained with random capabilities, but always a prior over all capabilities being present, so that actions are not disabled from the beginning. The parameters used of this domain are presented in Section VI-B.

D. Online Capability Estimation

Online capability estimation is a direct application of Equation 4, as shown in Algorithm 1. The estimation starts with a prior $P(\mathbf{c})$. The robot accumulates l state-action pairs, and then updates the probabilities of the capabilities, which will result in actions being enabled or disabled. The `step()` function at Line 4 returns, in addition to the latest state and action, also whether the last action was the result of exploration, so that it is disregarded for capability estimation. Exploratory actions are not the expression of the learned policy, and would taint the capabilities.

Algorithm 1: Human Capability Estimation

Input: $P(\mathbf{c})$: The prior probabilities of capabilities.
l: The batch length.

```
1  τ ⟵ [];
2  j ⟵ 0;
3  while learning is running do
4      s, a, e ⟵ step();
5      if e is false then
6          Append the state-action pair (s, a) to τ;
7          j ⟵ j + 1;
8      end
9      if j mod l = 0 then
10         Update capability probabilities according to
             Equation 4;
11         Replace prior with current estimate;
12         τ ⟵ [] ;
13     end
14 end
```

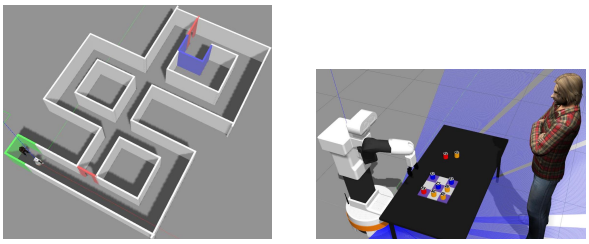
E. Capability-guided exploration

The online capability estimation can be used in conjunction with pre-training to further improve sample complexity. In principle, the robot starts with the optimal policy for every capability combination, therefore when sufficiently confident in the presence of the capability, it could switch to the corresponding optimal policy. However, the capability-conditioned optimal policy was learned in simulation, and it may not be optimal in the real world and with the specific user at hand. Nonetheless, we propose to use it for exploration advice, limiting purely random exploration. The exploration strategy we used is shown in Algorithm 2. The principle is the same as ϵ-greedy: when exploring, the agent selects with probability k the optimal action of the pre-

Algorithm 2: Capability-guided Exploration

Data: A: The set of actions.
$q^*(s, a \mid \mathbf{c})$: The pre-trained value function.
s: The current state.
$\mathbf{c}$: The current capability estimate.
$d(n)$: The optional liner decay function.

$n \longleftarrow 0$;
Function `Exploration(`n`)`**:**
  $\lambda \longleftarrow$ uniform random value between 0 and 1;
  $k \longleftarrow d(n)$;
  if $\lambda < k$ **then**
    $a \longleftarrow \arg\max q^*(s, a \mid \mathbf{c})$;
    $n \longleftarrow n + 1$;
  else
    $a \longleftarrow random(A)$;
  end
  return a;



(a) (b)

Fig. 3: The Gazebo worlds for simulation tasks: (a) The navigation task. (b) The manipulation task.

trained policy corresponding to the current estimate of the capabilities, and a random action with probability $1 - k$. If executed on top of ϵ-greedy this algorithm results in choosing the current optimal action with probability $1 - \epsilon$, the optimal action of the pre-trained policy with probability ϵk and a random action with probability $\epsilon(1 - k)$. We decay k over time, so that the advice is gradually removed.

The navigation task is a prototypical case in which a capability like `c_fast` can be estimated locally in a few steps and generalized globally regardless of the state. Figure 2 shows how the RLC agent outperforms the RL agent from early on. Furthermore, this domain highlights how the additional use of pre-training for initialization and exploration carries additional benefits on the agent convergence speed.

VI. Experimental Evaluation

A. Tasks

For all our tasks we used a PAL Robotics TIAGo, both in a Gazebo simulation and in the real world. The first task is the robot navigation task introduced in Section V-A.

Robot Manipulation The second task is a simulated collaborative manipulation task whereby the collaborator and robot tidy up items together. The Gazebo world is shown in Figure 3 (b). A board with three rows and three columns on the table provides nine slots for placing items. There are nine objects to be sorted: two are red, four are orange, and three are blue. At the beginning of the task, some objects are spawned randomly on the board, leaving at most two empty slots. The remaining items are placed outside the board. The goal is to sort all objects on the board so that objects of the same color are next to each other, like in the examples in Figure 4, resulting in different possible solutions. Once all items are grouped based on color, the task ends.

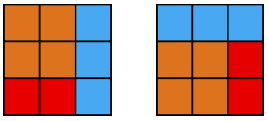


Fig. 4: Examples of acceptable solutions in the manipulation task.

The state space consists of observations of the content of nine slots (3×3) on the board. The action space contains nine actions: `place_{red, orange, blue}`, `swap_red_orange`, `swap_red_blue`, `swap_orange_blue`, `ask_swap_red_orange`, `ask_swap_red_blue`, `ask_swap_orange_blue`.

The `place_`$\star$ actions indicate that the robot puts an item of the corresponding color into the board. The `swap_`$\star$ actions indicate the robot swaps two items on the board by itself. The `ask_swap_`$\star$ actions indicate that the robot asks the collaborator to swap items. The robot and collaborator operate alternately, but the latter can only rearrange items within the board. The RL agent serves as the high-level decision maker and only issues instructions that indicate the action type and target color. A separate motion planner computes the manipulation movement for the robot.

This experiment considers two human capabilities `c_color` and `c_swap`. Lack of `c_color` means that the collaborator cannot distinguish between red and orange (Deuteranomaly) [15]. The capability `c_swap` implies being able to swap two objects in one move, impacted, for instance, if the person has a mobility impairment, or is one-armed. In the simulation, the human model does not distinguish red items from orange items if it does not have the `c_color` capability, and will not execute swaps if it does not have the `c_swap` capability. The reaction of a person requested by the robot to swap two objects is also affected by `c_color`, because a correct exchange depends on the accurate perception of colors. To consider that the person can make mistakes, the human model improves, with its actions, the correct grouping of the objects (e.g., place a blue object near other blue objects) with probability $P = 0.7$, and make the configuration worse with $P = 0.3$. The initial object layout is generated randomly at the beginning of each run.

The reward function is as follows. The cost of any placing action the robot performs is -1. The cost of a robot asking a human to swap items is -0.1. The cost of the robot performing the swap itself is -5. If the robot's strategy does not result in layout variations, the additional reward is -10 as a penalty. An additional reward of 80 would be assigned to the agent if the robot's strategy resulted in correct goal

configurations.

Scavenger Hunt Game The experiment we run in the real world is a collaborative version of the Scavenger Hunt game [16], a benchmark for autonomous robots. The task consists in finding a certain number of objects, which, in our version, the robot and the person do together. At the beginning of each episode, the referee program generates the layout of objects based on a predefined distribution. Then the operator places the objects accordingly. Next, the robot and the collaborator will set off to find the items. During the execution, the referee program also automatically rewards the RL agent. The person does not need to know the environment. The robot, which has access to the object distribution, but does not know the exact locations, can lead the way and ask whether the person can find a specific object in the current room. The person can help with object recognition, which involves both vision and hearing. However, should the person be partially impaired, the robot will fill in the required capabilities. With a fully capable person, the pair acts most effectively with the robot leading the way and the person identifying the objects. However, the robot can slow its pace as in the navigation task, if necessary, and also use its own vision and sound detection if it collects enough evidence that the person is unable to. The robot can explicitly ask the collaborator if an object is around during the game and decide whether to trust the human's answer.

The state space consists of the position of the robot base, the current navigation goal, the target object, the latest answer from the person, the robot's distance from the person and the position of the objects found up to that point. The action space consists of `drive_slow`, `drive_fast`, `ask`, `inform`, `accept`, and `reject`. `drive_slow` represents the robot's mobile base moving to a waypoint at low speed, while `drive_fast` is at high speed. `ask` represents the robot asking the person if he can find the target object around the current location. Then, the robot has two actions: `accept` and `reject`. `accept` represents that the robot trusts and takes the human's answer. `reject` means that the answer is rejected, and the robot plans to scan the area by itself (the person may not see the object, even though the object is there). Besides asking for human responses, the robot has an alternative action `inform`. This action represents the robot expecting the person's answer to be wrong and sticking to its observations. Rather than explicitly asking the person, rejecting the answer and then scanning around, it is considerably time-efficient to scan the area directly and tell the person the result.

Four human capabilities `c_fast`, `c_color`, `c_sight`, and `c_sound` are considered in this experiment. The `c_fast` capability represents that the person can walk fast; `c_color` represents that the person can discriminate red and orange; `c_sight` represents that the person can recognize objects from far away; `c_sound` represents that the person can locate hidden objects that emit a sound. There are a total of six rooms to be searched. Four target objects `red_ball`, `chickpeas`, `breadsticks` and `speaker` are shown in Figure 7. A blue ball or orange ball may also be placed where the `red_ball` may be, requiring `c_color` to be discriminated by the collaborator. The canned `chickpeas` can be found anywhere. The `breadsticks` will be placed far away from the center of the room. Thus the collaborator requires `c_sight` to see it. The speaker will be turned on to play music and placed in a position invisible to the robot and the collaborator, requiring `c_sound` to be located.

The reward function is as follows. Denoting with d the human-robot distance, and with $d_{max} = 4$ the desired maximum distance, the rewards are assigned to the agent based on the rules listed below, with a default reward of 0. Since this task aims to encourage human-robot collaboration, the reward function favors actions that involve discussing the location of objects with the person over providing answers directly by the robot, even if the latter takes less time:

- If $d \leq d_{max}$: $R_a(\texttt{drive\_fast}) = -1$, $R_a(\texttt{drive\_slow}) = -2$.
- If $d > d_{max}$: $R_a(\texttt{drive\_fast}) = -20$, $R_a(\texttt{drive\_slow}) = -10$.
- If the robot correctly identifies the location of an object that is in the room: $R_a(\texttt{accept}) = 80$, $R_a(\texttt{inform}) = 60$, $R_a(\texttt{reject}) = 40$.
- If the robot does not return a false positive, for an object that is not, in fact, in the room (otherwise the robot may just guess in every room): $R_a(\texttt{accept}) = 30$, $R_a(\texttt{inform}) = 20$, $R_a(\texttt{reject}) = 10$.

The collaborator carries a smartphone with software to answer the robot's questions, providing two options `Yes` and `No`. This software also calculates the human-robot distance according to the signal strength of Bluetooth beacons on the torso of TIAGo, and reports the distance to the robot. The robot also carries a tablet reporting the text of what the robot says, allowing hearing-impaired people to answer and interact.

B. RL Performance

We experimentally evaluate two aspects: the performance of the RL agent in terms of cumulative reward, and the accuracy of the capability estimate over time. The RL performance is shown in Figure 2 for the navigation task and Figure 8 for the manipulation task.

The pure RL agent and our full agent (estimating capabilities, initializing from the policy pre-trained over random capabilities, and performing capability-guided exploration) use the same Q-learning implementation, with the same parameters. In the navigation task of Figure 2(a) the learning rate $\alpha = 0.3$, the discount factor $\gamma = 0.99$, $\epsilon = 0.8$ for the ϵ-greedy exploration, and $k = 0.5$. In Figure 2(b) the learning rate $\alpha = 0.1$, the discount factor $\gamma = 0.95$, $\epsilon = 0.4$ for the ϵ-greedy exploration, and $k = 0.75$. In the manipulation task $\alpha = 0.1$, $\gamma = 0.95$, $\epsilon = 0.6$, and $k = 0.7$. In both tasks, the agent starts with a prior of $P(\mathbf{c}) = 0.8$ for all capabilities, and is evaluated on all four combinations for 30 trials in the navigation task, and 20 trials in the manipulation task (the simulation is significantly more time-consuming). Figure 2 and Figure 8 show the average return and the 95% confidence intervals over the 30×4 and 20×4 trials. Given

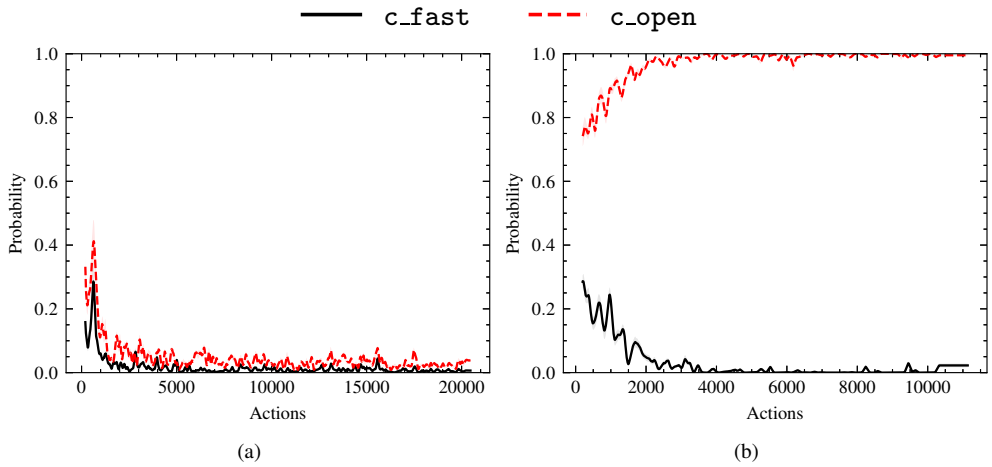


Fig. 5: The average probabilities of capabilities in the navigation task: (a) The collaborator has no available capabilities. (b) The collaborator's capability set is {c_open}.

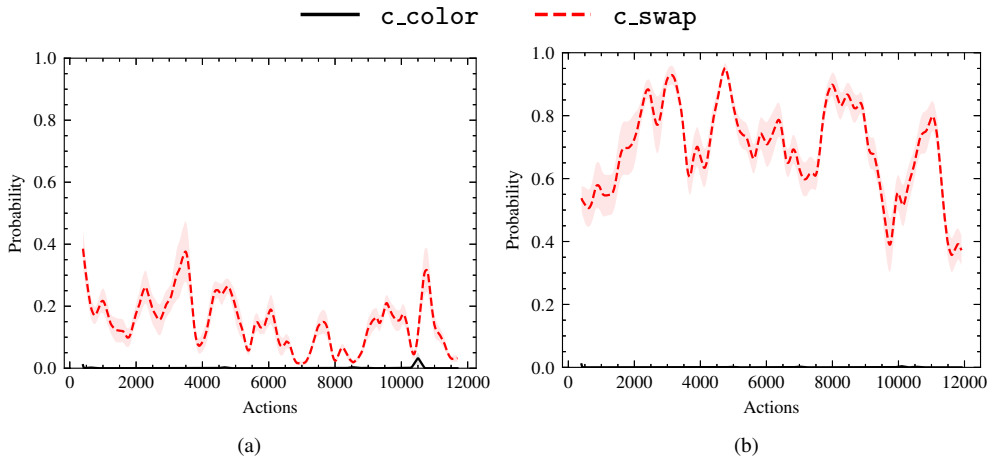


Fig. 6: The average probabilities of capabilities in the manipulation task: (a) The collaborator has no available capabilities. (b) The collaborator's capability set is {c_swap}.

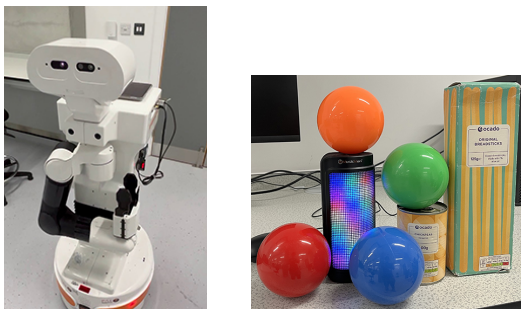


Fig. 7: TIAGo Robot and Objects

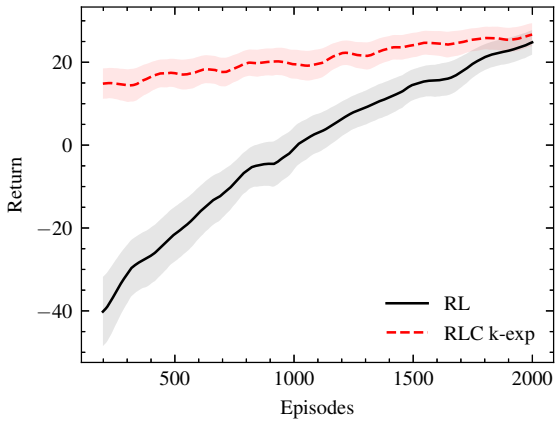


Fig. 8: The average return of the manipulation task.

the thousands of episodes required for the RL baseline to converge, this experiment was infeasible on the real robot. The plots show how the agent starts with an average return already close to optimal, and fine-tunes over time.

C. Capability Estimation

The second experiment answers the main question behind the presented methodology: can capabilities be estimated as a side-effect of the RL process? This question is particularly of interest in the real-world experiment, where the model and the world are different, and therefore we also aim to show that the estimate of the capabilities does converge to the correct values, in addition to how fast. There are 4 combinations of capabilities for each simulated task, and 16 for the real-world task. In all cases, the prior is on all capabilities being present, so we selected 2 combinations of capabilities for the simulated domains, and 3 for the real-

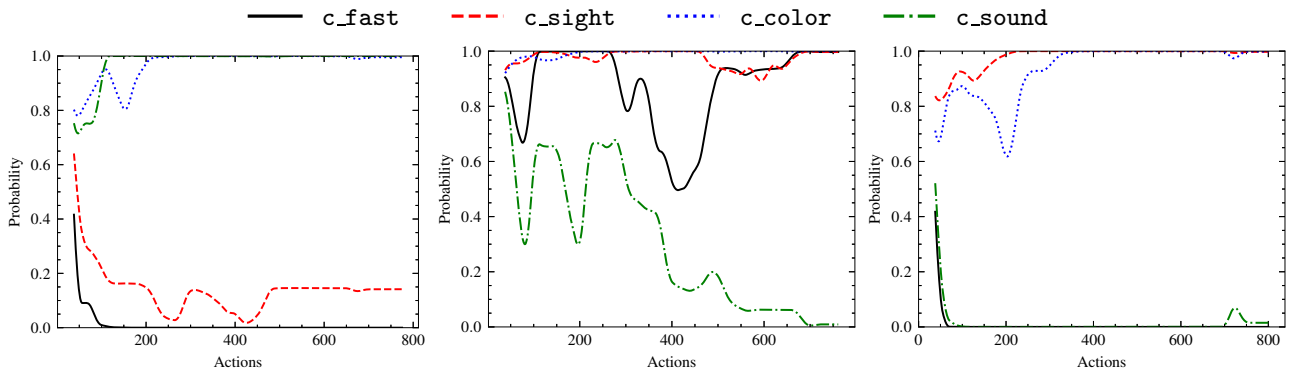


Fig. 9: The average probabilities of capabilities in the Scavenger Hunt Game: (a) The actual collaborator's capability set is {c_color,c_sound}. (b) The actual collaborator's capability set is {c_fast,c_sight,c_color}. (c) The actual collaborator's capability set is {c_sight,c_color}.

world domain, in which some capabilities are not present and therefore their estimates need to change, to show how they vary across learning. In the navigation task, we show in Figure 5, the estimation averaged over 30 trials. For the manipulation task, we show in Figure 6, the estimation averaged over 20 trials. Lastly, in Figure 9 we show the estimation for the Scavenger Hunt task, averaged over 5 trials. In all cases, the x-axis is the number of actions, and the estimates are updated for every $l = 10$ state-action pair. The estimates may fluctuate, but are eventually correct in all cases, and almost always fluctuate on the correct side of $d = 0.5$.

VII. CONCLUSIONS

We presented an approach for human capability estimation based on Reinforcement Learning and Bayesian inference. The adaptation based purely on interaction complements existing approaches and contributes to the personalization of robotics to people with different capabilities. We experimentally showed that estimating capabilities is indeed feasible with a real-world experiment, and using them as pre-conditions for actions significantly improves the agent's online performance. Since our approach is independent of the RL algorithm, we used simple Q-learning, resulting in a relatively slow learning speed. However, we expect that performance can be further improved with more efficient algorithms and representations. Due to the model-free nature of the approach, learning must be undertaken for each task. Generalization of capability estimation across tasks is a promising direction for future work.

REFERENCES

[1] C. Ray, F. Mondada, and R. Siegwart, "What do people expect from robots?" in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3816–3821.

[2] G. Canal Camprodon, G. Alenyà Ribas, and C. Torras, "A taxonomy of preferences for physically assistive robots," 2017.

[3] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, and M. Vincze, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and autonomous systems*, vol. 75, pp. 60–78, 2016.

[4] F. Broz, I. Nourbakhsh, and R. Simmons, "Planning for human–robot interaction in socially situated tasks: The impact of representing time and intention," *International journal of social robotics*, vol. 5, no. 2, pp. 193–214, 2013.

[5] M. Fiore, H. Khambhaita, G. Milliez, and R. Alami, "An adaptive and proactive human-aware robot guide," in *Social Robotics*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 194–203.

[6] T. Taha, J. V. Miro, and G. Dissanayake, "A pomdp framework for modelling human interaction with assistive robots," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 544–549.

[7] N. Abdo, C. Stachniss, L. Spinello, and W. Burgard, "Robot, organize my shelves! tidying up objects by predicting user preferences," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1557–1564.

[8] T. Matsubara, J. V. Miro, D. Tanaka, J. Poon, and K. Sugimoto, "Sequential intention estimation of a mobility aid user for intelligent navigational assistance," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2015, pp. 444–449.

[9] G. S. Martins, P. Ferreira, L. Santos, and J. Dias, "A context-aware adaptability model for service robots," in *IJCAI-2016 Workshop on Autonomous Mobile Service Robots*, 2016.

[10] G. S. Martins, L. Santos, and J. Dias, "Bum: Bayesian user model for distributed learning of user characteristics from heterogeneous information," *IEEE transactions on cognitive and developmental systems*, vol. 11, no. 3, pp. 425–434, 2019.

[11] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, "Supervised autonomy for online learning in human-robot interaction," *Pattern recognition letters*, vol. 99, pp. 77–86, 2017.

[12] K. Tsiakas, M. Abujelala, and F. Makedon, "Task engagement as personalization feedback for socially-assistive robots and cognitive training," *Technologies (Basel)*, vol. 6, no. 2, pp. 49–, 2018.

[13] A. B. Karami, K. Sehaba, and B. Encelle, "Adaptive artificial companions learning from users' feedback," *Adaptive behavior*, vol. 24, no. 2, pp. 69–86, 2016.

[14] N. Wilde, D. Kulić, and S. L. Smith, "Learning user preferences in robot motion planning through interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 619–626.

[15] D. R. Keene, "A review of color blindness for microscopists: Guidelines and tools for accommodating and coping with color vision deficiency," *Microscopy and microanalysis*, vol. 21, no. 2, pp. 279–289, 2015.

[16] H. Yedidsion, J. Suriadinata, Z. Xu, S. Debruyn, and P. Stone, "A scavenger hunt for service robots," 2021.