International Journal of Social Robotics
https://doi.org/10.1007/s12369-023-01044-7

Deep Reinforcement Learning for Robotic Approaching Behavior Influenced by User Activity and Disengagement

Luca Raggioli[1,2] · **Fabio Aurelio D'Asaro**[1,3] · **Silvia Rossi**[1]

Accepted: 15 August 2023

Abstract
A robot intended to monitor human behavior must account for the user's reactions to minimize his/her perceived discomfort. The possibility of learning user interaction preferences and changing the robot's behavior accordingly may positively impact the perceived quality of the interaction with the robot. The robot should approach the user without causing any discomfort or interference. In this work, we contribute and implement a novel Reinforcement Learning (RL) approach for robot navigation toward a human user. Our implementation is a proof-of-concept that uses data gathered from real-world experiments to show that our algorithm works on the kind of data that it would run on in a realistic scenario. To the best of our knowledge, our work is one of the first attempts to provide an adaptive navigation algorithm that uses RL to account for non-deterministic phenomena.

Keywords Activity recognition · Human–robot interaction · Reinforcement learning · User disengagement

1 Introduction

This work is rooted in the field of *Socially Assistive Robotics* (SAR), and more specifically in the UPA4SAR project.[1] The project focused on the design of adaptive robotic systems to monitor people with dementia. In the context of this project, robots need to monitor older people in their households to ensure they are following a healthy routine (e.g., get enough sleep, take their medicines, do not skip meals, etc.) by keeping track of their *Instrumental Activities of Daily Living* (IADLs) while possibly not interfering with them. Motivated by these requirements, we define and implement a *Reinforcement Learning* (RL) strategy for a robot's approaching behavior to adapt to the user's preferences. The main idea is that our robots should not negatively interfere with the users, for instance by scaring them or altering their behavior. Our algorithm learns a strategy to monitor the user, mainly considering the user's activity and awareness of the robot. This is because a high disengagement level implies that the user is not being distracted by the presence of the robot, thus allowing for effective monitoring.

Our implementation is a proof of concept, in which we simulate a 2D world consisting of a robot and a human that carries out an activity. To show that our approach would in principle work in a realistic scenario, we trained our RL algorithm using data from real experiments. The robot is assumed to be under the control of the RL algorithm at all times. We also simulate the user's reaction to the approaching robot in terms of his/her disengagement. To make the simulation of such complex phenomena as realistic as possible, we set our simulation parameters according to previous real-world HRI experiments. We also simulate the process of detecting user activity from the wearable stream and the camera stream alternatively and estimate the user's disengagement level to fine-tune the stopping distance from the user without causing him/her distress. We implemented a second specialized policy (alongside the one presented in [21]) that is explicitly trained to decide how and whether to take a final step

[1] http://www.upa4sar.unina.it.

✉ Silvia Rossi
silvia.rossi@unina.it

Luca Raggioli
luca.raggioli@manchester.ac.uk

Fabio Aurelio D'Asaro
fabioaurelio.dasaro@univr.it

[1] Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy

[2] Department of Computer Science, University of Manchester, Manchester, UK

[3] Department of Human Sciences, University of Verona, Verona, Italy

Published online: 07 September 2023

toward the user according to his/her disengagement level. The importance of considering the disengagement level towards the robot lies in the fact that it is often desirable not to distract the user being monitored. The consequences of taking an extra step toward the user could result in the user stopping his/her current activity, which is unwanted in most scenarios.

It is worth noting here that in our work we have two policies that tackle the navigation and disengagement maximization tasks separately. Obviously, one could have trained a single police do the whole job. However, the idea behind the double-policy approach, which partly constitutes the novelty of our contribution, is to have a *modular* approach, in the sense that one can concatenate the second policy to any navigation algorithm (including classical non-RL navigation) and still achieve the desired result (i.e., maximize disengagement).

Results show that the approach is promising for deployment, although further investigations in a real setting must be made to validate it.

2 Background and Related Work

In this section, we briefly overview some recent approaches dealing with socially aware navigation that use RL techniques to incorporate user feedback and tackle problems involving activity recognition. It is crucial to emphasize that our work focuses on non-interactive tasks [23], and, in particular, on monitoring elderly individuals in their home environment when they are affected by cognitive impairments such as Alzheimer's disease or dementia [5]. In these situations, the person may be unable to actively engage with the robot, or they might even be averse to the robot's presence. Indeed, the literature reveals mixed responses from the elderly, particularly those with cognitive impairments. In [2] it is pointed out that while some SARs have been successful, others have elicited unfavorable responses from the elderly, suggesting the need for careful design considerations. Studies such as [31] further reinforce this argument by documenting that elderly participants often did not associate positive connotations with using robots in elder care, considering it stigmatizing. Discussion in [19] suggests that certain factors might dissuade caregivers from utilizing SARs with elderly individuals who have cognitive impairments. These factors are implied by the robot's presence, include the robot's size, privacy concerns, fear of robots replacing humans, and overall negative attitudes towards technology. Taken together, these findings underscore the importance of our approach, focusing on non-interactive tasks and minimizing the robot's intrusive presence to promote positive interactions between SARs and cognitively impaired elderly individuals. It is worth noting that when the task at hand is not interactive, it is important to model non-intrusive behavior. Consequently, the robot must monitor the user while minimizing the impact of its presence as the user goes about their daily activities [24]. In other words, it must maximize the *disengagement* level.

2.1 Socially Aware Approaching Behavior

A fundamental aspect of a robot performing monitoring tasks is the capability of deploying approach behaviors and socially aware navigation taking into account human–robot proxemics. When a robot is introduced in a social environment with humans, its behavior has to be carefully crafted to preserve human comfort in terms of distances, speed, direction, and social rules (see [14, 22]). Path planning approaches can be divided into two categories: those where people are walking freely in the room, and those where the robot navigates in a space that is statically populated by people. Some state-of-the-art works, such as [7, 10, 11, 26], belong to the former class. They model an approaching behavior where a robot initiates conversations with people who are walking, predicting their walking behavior, planning the approaching path, and showing its intention to start a conversation using non-verbal cues. They predict the target's position using an anticipation model, based on clusters of walking trajectories, to approach him/her in an appropriate, polite way. Different from our approach, which is mainly *reactive*, they propose a system based on the *anticipation* of the targets' future behavior.

The idea of influencing the robotic approaching behavior with the user's positioning preferences and comfort has been considered also in works focused on the use of path and motion planners.

Sisbot et al. [27] propose a motion planner that explicitly considers a human partner's accessibility, visual field, and preferences in terms of relative human–robot placement and motions, to obtain socially acceptable paths. In this case, the robot path planning is conducted by considering sophisticated cost functions or potential fields that take into account the person's comfort (see [3]). In our work, we also model user activity (such as watching TV, sitting on the couch, etc.). However, the identification of the activity is not determined a priori as it depends on the recognition capabilities of classifiers working on different types of data.

Particularly interesting for the scope of our approach are also some works that explored the role of indirect social cues to provide meaningful information about the appropriateness of the path and approaching behavior of a robot. For example, [17] implements a "roboceptionist" (i.e., a robot-receptionist) that detects the engagement of people around it using a camera and a laser tracker. Engagement is modeled using four categories that correspond to different macro-behaviors of the users towards the robot. According to the specific class of engagement, the roboceptionist decides whether to rotate its head toward the user(s) or not.

In that work, differently from our approach, four behaviors are designed to be applied to each engagement class. Hüttenrauch et al. [9] use a WoZ strategy to investigate the impact of spatial distances and orientation of a robotic agent with respect to a human target during HRI. This is particularly relevant for our work, although we intend to provide an autonomous behavior capable of approaching within comfortable distances without the external intervention of an operator.

Psychological and social cues may impact the human user's perception of the robot [4, 18]. These aspects can be essential to understand how to design robot behaviors that are appropriate in different contexts.

2.2 Reinforcement Learning with User Feedback

In an interactive setting, human reactions to the robot's actions must be taken into account when shaping the behavior of an agent. In RL approaches, these provide immediate online feedback that can be naturally incorporated within the reward function. Tsitsimis et al. [29] describe a hybrid RL algorithm that uses human engagement perturbations for online adaptation in HRI. Contrary to our work, they employ the user's current engagement in the task and variations of this engagement to obtain a positive reward when it is low but increasing.

Akalin et al. [1] present a Deep RL algorithm to shape the behavior of a social robot interacting with an elderly human. They use negative valence, diminishing engagement, and difficulties in carrying out an exercise or activity to modulate the difficulty level of the interaction and stop the ongoing task.

The effectiveness of RL in designing adaptive systems is also shown in [12], where a NAO robot interacts with children suffering from autistic spectrum disorders. A parameterized RL model adapts the behavior of the robot to the current level of human engagement, estimated through measures of human gaze and body posture. Qureshi et al. [20] employ a Multimodal Deep Q-Learning model to characterize the social intelligence of the robot to greet a human interlocutor, rewarding a successful handshake, and penalizing an unsuccessful one. Contrary to our approach, these works maximize *engagement* exhibited by the target towards the task s/he is currently performing. In a sense, the aim of our work is the opposite: we want to minimize the user's awareness of the robot, thus maximizing his/her *disengagement*.

Also, in our setting, the navigation task is carried out using only data retrieved from the camera of the robot and does not require other sensors located across the environment to track the targets [6]. Once a user is identified, range finders evaluate the relative distance and the approach begins.

2.3 Underlying Ideas

The design of the present system builds upon the findings of previous work that we describe in this section.

The impact of the presence of a robot operating in the surroundings of a human is the object of [24], where, using a WoZ robot control strategy, we assessed and evaluated user disengagement and distraction, and produced the dataset that we also use in the present work (described more in detail in Sect. 3.1). We focused particularly on cases where the robot is performing monitoring tasks, potentially distracting the users from what they are doing. Our analysis was based on the identification of non-verbal disengagement cues, such as gaze and pose variation of the user as s/he is approached by the robot.

In [6], we discussed and tested a reactive algorithm for performing human-like approaching based on the coordinates of the user's shoulders. However, that work lacked a characterization of the activity performed. Raggioli et al. [21] describes a robotic agent navigating towards a human companion, based on RL. With respect to that work, we use an additional policy and a new data source (skeleton data). We build upon considerations made in [24, 25] for what regards the relationship between the activity performed by the user and the appropriate stopping distance. In these works, we recorded real-world experiments with elderly participants and a Pepper robot. Recordings gathered during these experimental sessions were not sufficient to build a dataset of approaching trajectories. In the present work, we overcome this issue by simulating user coordinates.

3 Proposed Approach

Our proposed method for approaching a human user as s/he is performing an activity is essentially a *Reinforcement Learning* (RL) approach. RL is a Machine Learning technique that learns a mapping from states to actions, usually called a *policy*, that is then employed by an *agent* to maximize the long-term *reward* resulting from an interaction with the surrounding environment. The *training phase* consists of running several *episodes* in which the agent is free to try different sequences of movements and use the collected reward to update its behavior.

In this work, a robotic agent learns to navigate towards the user using a RL technique known as *Policy Gradient*. We employ this method as formalized through the *REINFORCE* (short for *REward Increment = Nonnegative Factor* $\times$ *Offset Reinforcement* $\times$ *Characteristic Eligibility*) algorithm [30] to approximate the policy function with a *Deep Neural Network* (DNN). The core idea behind Policy Gradients is to parametrize the policy function π using parameters θ, with respect to which, stochastic gradient ascent is performed

on the cumulative reward. The method can be formalized through a DNN, with weights θ, that directly models the action probabilities. As the interactions between agent and environment go by, the parameters, or weights, θ of the DNN are adjusted, so that actions that produced better outcomes, will be more likely sampled in the future.

The objective of Policy Gradients is to maximize the performance measure $J(\theta)$, which is the total future expected rewards:

$$J(\theta) = \mathbb{E}[R(\tau)] \tag{1}$$

where τ is the model approximating the policy function. To better perform, parameters θ are updated approximating gradient ascent in J:

$$\theta_{t+1} = \theta_t + \nabla J(\theta_t) \tag{2}$$

where $\nabla J(\theta_t)$ is a stochastic estimate approximating the gradient of the performance measure to the parameters θ. This method allows one to learn stochastic policies.

The REINFORCE algorithm approximates the gradient $\nabla J(\theta_t)$ according to the following equation:

$$\nabla J(\theta_t) = \nabla_\theta \mathbb{E}[R_t] = \mathbb{E}[\nabla_\theta \log \pi(a_t|s_t, \theta_t) R_t] \tag{3}$$

where the reward R_t is used as a scaling factor for the policy function π (whose output is the probability of the action given the state). If the action chosen brings profit, $\pi(a_t|s_t, \theta_t)$ will be updated by a large magnitude, whereas if the action results to be poorly convenient or inconvenient, $\pi(a_t|s_t, \theta_t)$ will be discouraged. R_t can be obtained as the immediate reward or the discounted reward produced at the end of the episode.

In the case in which the policy function approximation is derived through a DNN, the problem of training an unsupervised model must be tackled. The objective is to maximize the performance of the model and output the correct probability distribution over the set of possible actions to be taken. A possible solution is to use the classification log-probabilities, labeled by gradients. The reward is used as a scaling factor: if the selected actions provided good results, and therefore the reward is positive, the probabilities are increased, while they are decreased otherwise. To affect actions that were not as bad as the outcome of the episode (or that were not good enough), the reward can be smoothed accordingly.

The resulting loss function is as follows:

$$\left[\sum \log p(y_t|x_t; \theta)\right] R_t \tag{4}$$

where y_t is the action probabilities returned in output by the model and x_t is the input.

REINFORCE uses the complete return, consisting of the sum of all the rewards until the end of the episode. For this reason, REINFORCE is considered to be a Monte Carlo algorithm and is therefore employed for cases where all the updates are made in retrospect after the episode is completed (see [28]).

3.1 PRISCA Dataset

The dataset employed in this work, dubbed *PRISCA Dataset*, consists of data obtained at the PRISCA Laboratory during experimental sessions with 20 elderly people (11 males, 9 females). Participants were aged between 53 and 82 years old (average 61, standard deviation 7.6) and involved in activities of daily living. They were recorded as they were being monitored by a robot (Softbank Pepper, see [24]). Specifically, data were gathered using a smart band worn on the dominant wrist of the participant (Empatica E4[2]), and a real-time video shot from the 2D camera mounted on the robot's head. The six activities of daily living (and corresponding postures) are the following: (i) "*PC*" (working on PC), (ii) "*Phone*" (talking on the phone), (iii) "*Coffee*" (making a coffee), (iv) "*TV*" (watching TV), (v) "*Iron*" (ironing), (vi) "*Couch*" (talking on the couch with another person).

During the experimentation, users were asked to perform all the activities and, for each activity, the robot performed two monitoring actions, one from afar (approximately 2.5m from the user), and one from a closer distance (approximately 1.5m from the user).

We processed this dataset using the Affectiva SDK, see [15]. For each video, the Affectiva SDK produces a *disengagement value*, i.e. an integer from 0 to 100 measuring the human's awareness of the robot. We normalized these values in the real interval [0, 1] and took the maximum and minimum for each activity and distance (*near* and *far*) to get the ranges in Table 1. We use these ranges to simulate user disengagement in our 2D world. For instance, if the activity is "Coffee", and the robot is standing at 1.5 m from the user (corresponding to the "Near Range"), we will generate the user disengagement as a random number from the uniform distribution in the real interval [0.22, 0.79].

3.2 RL Agent

In our setting, the *Agent* is a mobile robot that has to navigate towards a human target, ultimately stopping at a distance that does not produce discomfort in the user. As shown in Fig. 1, the agent consists of two three-layer DNNs that approximate the two policy functions. Both DNNs receive input from the Environment in the form of a *state*. A state encodes the position of the user and the activity s/he is carrying out. *Policy function 1* (*P1*) is solely responsible for the navigation task and can decide to perform one of the actions *FWD*, *L*, or

[2] https://www.empatica.com/research/e4/.

Table 1 Disengagement intervals and threshold for each considered human activity

Activity	Threshold (m)	Near range	Far range
PC	0.9	0.3–0.92	0.22–0.99
Phone	1.8	0.14–0.29	0.51–0.65
TV	1.2	0.31–0.91	0.05–0.9
Coffee	2.1	0.22–0.79	0.51–1
Iron	1.5	0.13–0.36	0.06–0.44
Couch	2.4	0.31–0.88	0.28–0.9

R, corresponding to taking one step forward, left and right respectively. *Policy function 2* (*P2*) is in charge of deciding the size of the last step forward, which can be one of 0.0 m, 0.3 m, 0.6 m, or 0.9 m. The XOR symbol $\bigoplus$ in Fig. 1 stands to mean that the output from the User Position and Activity Recognition modules is received either by *P1* or *P2* but not both. The environment consists of information about the user's position, activity, and disengagement level. The user disengagement level and activity are generated according to Sect. 3.3.1. These parameters are all used in the reward function during the training phase of the DNNs.

The two DNNs share the same state space. Their input is the state of the environment, i.e. a tuple $(d, \alpha, Rs, Ls, activity)$ where (Rs, Ls) are the coordinates of the human's shoulders joints, d is the human–robot distance, α is the angle the robot should rotate to face the user, and $activity = (p_1, \ldots, p_6)$ is a 6-tuple of probabilities provided by the Activity Recognition module for each one of the six activities. More details about Activity Recognition are provided in Sect. 3.3. The system outputs the probability of performing each one of the actions *FWD*, *L*, and *R* described below.

P1 was first introduced and studied in [21], and it makes the robot navigate towards the user, although other navigation algorithms could be employed to approach the user. At each step, *P1* chooses one of three actions: *FWD*, *L*, and *R*, corresponding to the robot moving forward by 0.4m, rotating 30 degrees to the left, and rotating 30 degrees to the right respectively. Once *P1* has brought the agent within a reasonable stopping range from the user, which depends on the performed activity (see Table 2), *P2* takes control. This has to choose whether or not to take another step towards the target, and how close the robot should get. As mentioned above, *P2* has four actions at its disposal: 0m, 0.3m, 0.6m, and 0.9m, each corresponding to the size of the last step forward.

P1 and *P2* use the Policy Gradient method (described above), and therefore it is not necessary to implement the value function as well. Their DNN architecture is as follows:

- *Structure*: Three fully-connected layers; The hidden layers of the model consist, respectively, of 64 and 32 neurons. In the last layer, we have 3 neurons (which is the number of actions available).
- *Activation function*: Hyperbolic tangent for the first layer, sigmoid functions for the second layer, soft-max function for the third layer;
- *Loss function*: Categorical cross-entropy;
- *Optimizer*: Adam [13].

The number of neurons in each layer was chosen with a grid search in [16, 32, 64, 128] subject to the constraint that the

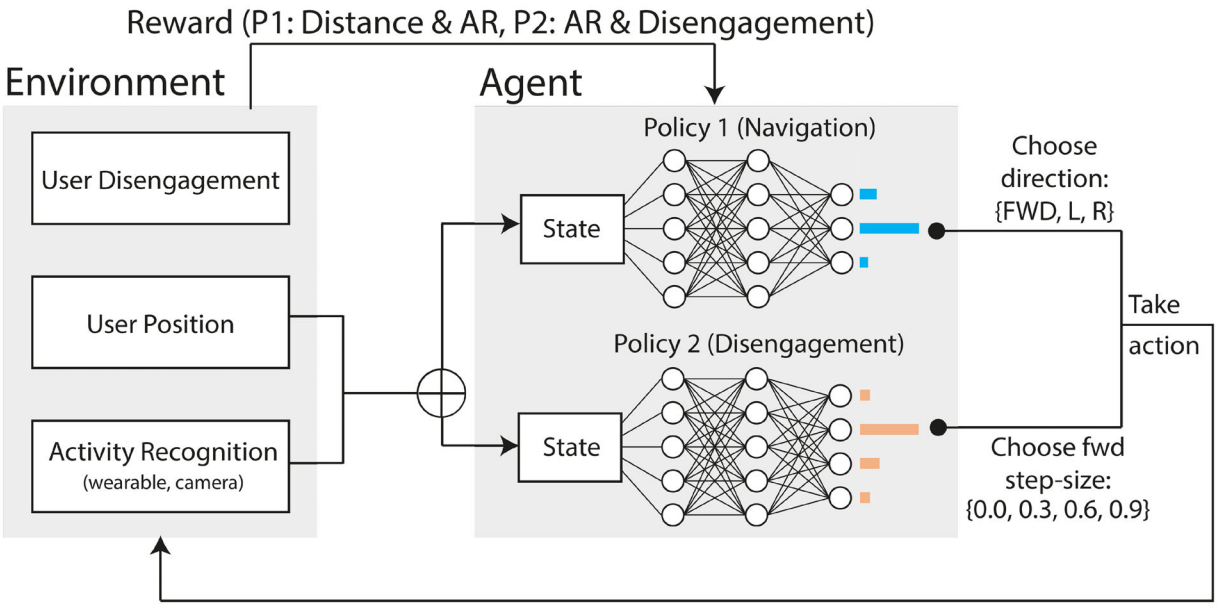

Fig. 1 A diagram of our proposed RL system

Table 2 Stopping ranges associated with each activity

Activity	Cognitive Load	Posture	Stopping Range (m)
PC	High	Sitting	0.3–1.5
Phone	High	Standing	1.2–2.4
TV	Medium	Sitting	0.6–1.8
Coffee	Medium	Standing	1.5–2.7
Couch	Low	Sitting	0.9–2.1
Iron	Low	Standing	1.8–3.0

For each activity, we specify below the cognitive load and the posture of the human

first layer should have at least as many neurons as the second layer. Similarly, the activation function was chosen with a grid search among Relu, Hyperbolic Tangent, and Sigmoid.

Recall that the DNNs output the probability of choosing each one of the actions. Given these probabilities, the agent will choose which action to take according to the ϵ-decreasing strategy (see e.g. [28, Chapter 2]). The ϵ-decreasing strategy chooses a "greedy" action (the one with the highest value) with probability equal to or greater than ϵ, otherwise, it picks another action at random.

3.3 RL Environment

The *Environment* models the characteristics of the robot and its human interlocutor such as their position, disengagement, and current activity. In this work, we simulate these characteristics and then pass them on to the RL algorithm that decides which action to take. The four modules that are responsible for the simulation are:

- Two *Activity Recognition* modules that share the same internal structure (in terms of neural networks hyperparameters), one that handles *wearable* readings (which from now on we will refer to as the *wearable AR module*), and the other *skeleton joints* data acquired from the camera (which from now on we will refer to as the *camera AR module*).
- A *User Position* that simulates the position of the user.
- A *User Disengagement* that simulates user disengagement.

These modules should not be confused with the *policies* that we discuss later on, which are trained and used to model the robotic agent's behavior. Below we provide a detailed description of these modules:

3.3.1 Activity Recognition Modules

Activity Recognition (AR) relies on data acquired from a wearable device (that is worn on the dominant wrist of the user providing inertial movement readings), and on video footage recorded with a camera.

The wearable AR module provides a less precise estimate, as it relies only on tri-axial inertial readings of a single point moving in space. The camera AR module allows for the retrieval of skeleton readings that provide the position and movements of a wider set of points in space. In this case, the camera AR module offers a more informative and precise estimation. This latter solution, however, requires a closer interaction with the user if compared to the wearable AR module that does not need the robot to be in relative proximity to him/her to acquire data. In this work, we employ both these strategies: we use the wearable AR module when the user is too distant from the robot to use the camera reliably, otherwise, we use the camera AR module. Both AR modules are implemented with a DNN, share the same structure, and differ only by the form of input and output.

The wearable dataset consists of $\sim$ 80k instances. The camera dataset consists of $\sim$ 60k instances. For both AR modules, we used the early stopping strategy with respect to the error rate. Convergence was reached for the wearable AR module at $\sim$ 50 epochs, while for the camera AR module, it was reached at $\sim$ 120 epochs. The input size of the networks is 3072 floats for the wearable AR module DNN and 3600 floats for the camera AR module DNN.

We use Long Short-Term Memory (LSTM) Networks (refer to [8]). More specifically, each Neural network consists of three layers. Two recurrent layers with LSTM units detect the temporal dependencies between the readings forming the input of the net. A dense, fully connected layer with a softmax activation function serves as the classifier and outputs the probability for each activity. The loss function of the net is set to categorical cross-entropy. The number of units and the optimizer were chosen by performing a grid search in the range 40–100 (step 6) and then evaluating the resulting model with fivefold cross-validation: 52 and 46 units are employed for the first and second levels respectively. For this pair, the accuracy was 97.40% (standard deviation 0.36) for the wearable module and 93.22% (standard deviation 0.75) for the camera module when using the *RMSprop* optimizer,[3] which gave the best results.

The wearable AR module has been trained with data consisting of accelerometer readings provided by the Empatica E4 smart band (Sect. 3.1) worn by the participants on their dominant wrist. For the camera AR module, we used the 2D-camera footage presented in Sect. 3.1. The joints of the

[3] https://www.coursera.org/learn/neural-networks/home/welcome.

user are retrieved using OpenPose[4] on the videos captured by the robot. We used OpenPose's own representation for the skeleton joints and face key points, i.e. coordinates of the form (x, y) for non-negative floats x and y. Once the skeleton joints and the face key points are retrieved, they are fed to the Neural Network.

One of the two AR modules is selected depending on the distance between the target and the robot. In [24], we found the wearable AR module to be more reliable when the user is at least 2.5 m from the robot. Up to 2.5 m, the camera AR module is more precise. Therefore, we employ the wearable AR module for distances greater than 2.5m and use the camera AR module otherwise. A more in-depth description of *how* the data are processed and used as input for the AR modules is provided n Sect. 4.1.

3.3.2 User Position Module

The *User Position module* extends [21] and builds upon considerations made in [6, 24, 25]. Similar to these works, the module simulates the user position with respect to the robot and updates these estimates at each step. We assume that the robot is always located at the origin of the axes in a two-dimensional Cartesian system. The position of the user is given by the coordinates of his/her shoulders seen from above.

3.3.3 User Disengagement Module

The user reaction to the approaching behavior constitutes a defining element of this work, as we employ below (see Sect. 3.4.2) to evaluate the effect of the actions chosen by the Agent when using *P2*. Similar to the approaching distances, a range is defined for each activity (see Table 1). These ranges were obtained from the camera footage gathered with the Pepper robot, during the experimentation sessions of [24]. For training and simulation purposes, we draw random values from these intervals.

Rossi et al. [24] employ two fixed stopping distances (1.5 m and 2.5 m), while in this work the stopping distances are customized for each activity. For this reason, we defined two disengagement ranges for each activity. One is based on the videos shot from 1.5 m, the *near* range. The other one is based on videos taken from 2.5 m, the *far* range.

In Table 1 we show disengagement ranges and thresholds for each activity and distance. These values were calculated from videos in the PRISCA Dataset (see Sect. 3.1), which were shot from fixed distances of 1.5 m and 2.5 m. We grouped videos by activity and distance, and for each group calculated minimum and maximum disengagement by processing all videos in the group with the Affectiva SDK. In the table, we refer to the group of videos shot from 1.5 m as "Near", and those shot from 2.5 m as "Far". Whenever an estimate of disengagement must be generated, if the current distance of the robot from the target is smaller than a threshold, which may change according to the activity, then a random value is drawn from the "near" range, otherwise, the disengagement is extracted from the "far" range. For instance, if the current activity is *Coffee* and the distance from the robot is greater than 2.1 m, disengagement is simulated as a value uniformly drawn from the interval [0.51, 1].

[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose.

3.4 Reward Functions

The *reward function* is a crucial element of RL, that evaluates the effect of the Agent's actions. Recall that this work uses two policy functions *P1* and *P2* and, therefore, it has two reward functions. *P1* (with corresponding reward function r_1) is used to make the robot approach the user. *P2* (with corresponding reward function r_2) is a single-step policy whose purpose is to decide whether the robot should take another final step towards the target. Note that these two functions are never used at the same time, since the second reward function is only employed when the first one has concluded the navigation task successfully. Both policies can make the robot navigation task result in a *successful* episode or a *failure*. This depends on satisfying a series of commonsensical constraints for assistive scenarios. More formally, if (i) the robot goes too far away from the user, (ii) it gets too close to the user, (iii) the user disappears from the robot's view, or (iv) the robot positions itself behind the user, then the episode *fails*. If none of these conditions is true at the end of the navigation, the robot *successfully* terminates the episode.

3.4.1 Reward Function for *P1*

We employed a shaped reward: in the intermediate steps, zero is returned if the action chosen by the agent decreases the distance (d) from the target, otherwise, a penalty is returned. In the last step of the episode, 10 or -10 is returned according to whether the distance from the target is acceptable or not. The episode terminates *successfully* if it is within an acceptable *stopping range* from the user, otherwise it *fails* (see Equation (5)).

Let $A(t)$ be the probability of the activity predicted by the AR module with the highest value. Formally, the reward $r_1(t)$ (at step t) has the following form:

$$r_1(t) = \begin{cases} 10 & t = T \text{ and } success \\ -10 & t = T \text{ and } failure \\ 0 & t \neq T \text{ and } d(t) \leq d(t-1) \\ -A(t) - D(t) & t \neq T \text{ and } d(t) > d(t-1) \end{cases} \quad (5)$$

where T is the last step of the episode, $D(t) = (d(t-1) - d(t))$ is a difference in distance between the previous timestep and the current timestep. It is worth noting here that both $A(t)$ and $D(t)$ take value in the range [0, 1] and therefore have a comparable impact on the reward function.

3.4.2 Reward Function for *P2*

Another important feature of our work is the evaluation of the user reaction to the ongoing approaching behavior of the robot that is used to influence the final stopping distance. The underlying idea is that once the robot is close to the user, it can decide whether to move further toward the user to improve the monitoring performance (always remaining within the stopping distance range) but without causing too much distress and distraction from what the user is doing. Therefore, we want to maximize the disengagement of the user towards the robot (obtained as described in Sect. 3.3.3). The problem is formalized as the maximization of the product between disengagement and prediction probability, which preserves the scale of the two terms.

To this aim, we define the reward of *P2*, which is a measure of goodness of the action that has to rely solely on the terms we want to maximize. The negative reward aims at punishing the agent in an inversely proportional way with respect to how good the effect of the action was, as shown in Eq. 6.

$$r_2(t) = -(1 - A(t) \cdot disengagement(t)) \tag{6}$$

where $disengagement(t)$ is defined following the procedure outlined in Sect. 3.3.3.

4 Simulation

In this section, we demonstrate the ability of the proposed RL system to approach the user and decide the proper stopping distance taking into account the activity as it is recognized from wearable and camera data.

To do so, we trained our system by running a fixed number of episodes in a *simulated* 2D environment, whereas the user activity data is extracted from the real data collected (see [24]). We then tentatively evaluate our system via the number of episodes concluded successfully.

We hypothesize that an agent employing both $P1$ and $P2$ manages to get closer to the user, if compared to $P1$ alone, without causing the number of successful episodes to decrease significantly. Indeed, $P2$ is defined to potentially improve monitoring performance while respecting user comfort. In other words, we aim to prove that our implementation can account for disengagement levels without degrading the performance of the system. An episode terminates *successfully* if the agent manages to stop in the appropriate stopping range for the activity.

In the following sections, we provide a more detailed description of this workflow and discuss the results.

4.1 Experimental Procedure

At the beginning of each episode, the coordinates of the target's shoulders are randomly generated. The environment produces the current state of the user, consisting of the coordinates of the shoulders, the human–robot distance, the angle the robot should rotate to face the user, and the probability predictions of the six considered activities. Specifically, the activity estimation is obtained as follows: for each episode, a target activity is selected and, at each step of that episode, an instance corresponding to the selected target activity is randomly drawn from the dataset (from the wearable data or the skeleton data respectively, depending on the distance from the user). The instance is used as input for the AR Neural Network which then outputs the probability distribution.

The predictions obtained with the two Activity Recognition modules are handled as follows:

- The prediction obtained using the camera-based module is weighted according to the user-robot distance in the current step, as also suggested in [16]. The probability of the predicted activity ($p_i(t)$, where i is an index representing the activity) is attenuated in an inversely proportional way to the current human–robot distance. Specifically, the prediction that will be employed in the reward function ($A(t)$) is obtained with the equation:

$$A(t) = \max_i [p_i(t)] * (1 - (d(t)/10)) \tag{7}$$

  The camera module is employed for distances equal to or less than 2.5m, therefore, in the case of maximum prediction probability (1, in a range from 0 to 1), the minimum scaled probability would be obtained at a distance of 2.5m, with a corresponding value of 0.75.
- The predictions returned from the Empatica-based module are instead scaled differently. The wearable data are not sensitive to distance but are generally less reliable. For this reason, they are mapped from the range [0, 1] to the range [0, 0.75]: the upper bound is changed to simulate the lower precision of a wearable Activity Recognition module, compared to a camera Activity Recognition. The upper bound is set to 0.75 for the considerations made above for the camera module.

The agent decides which action to take based on the state of the environment. If the current distance from the subject is outside the stopping range, it chooses to move forward or

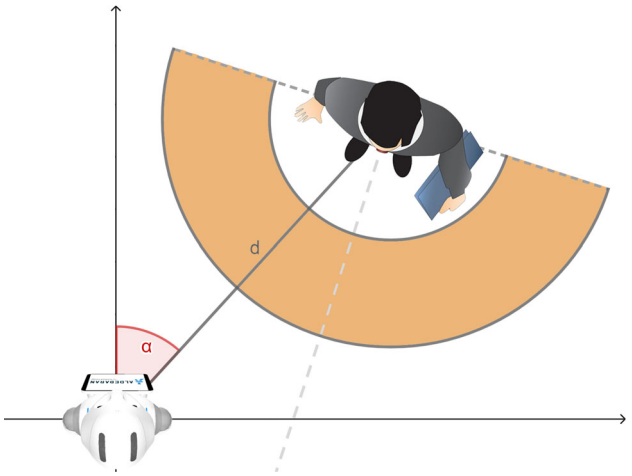

Fig. 2 Example of stopping range: the robot has to stop in the orange area to win the episode

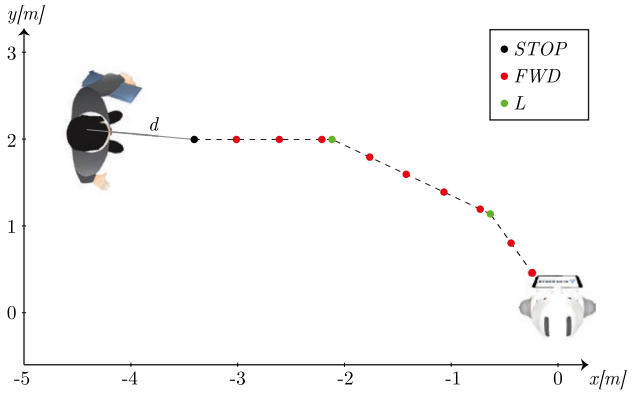

Fig. 3 An illustration of $P1$ in action

rotate left or right. If the navigation task has already concluded successfully (i.e., the robot is within the stopping range), $P2$ will decide whether to move another step toward the target.

The environment propagates the effects of the action of the agent on the user position and produces an evaluation (the *reward*) that is returned to the agent together with the new state describing the user. If the new current distance from the target is within the stopping range, the environment will consider the user's reaction.

We use the human–robot relative position and distance to evaluate the final step of the episode: if the robot is too distant or too close to the target, if the robot cannot see the user in its visual field, or if the robot is approaching the user from behind, then the episode concludes with a negative outcome. Otherwise, the episode concludes successfully.

The stopping distances result from an analysis conducted in previous work (see [25]), and define what distances are acceptable for the robot to stop in order to have a good monitoring performance while not distracting the user. For standing poses (i.e., "Coffee", "Phone", "Iron") the distances are higher than those employed for the static poses (i.e., "Couch", "TV", "PC"). The ranges also vary within the two types of poses, with respect to the level of attention required, as defined in [24]: for poses requiring higher cognitive load, closer stopping distance resulted to cause less distraction, as compared to activities requiring lower cognitive load. The ranges are defined in Table 2. In Fig. 2, we show an example of stopping range (in orange), where the agent would terminate the episode successfully.

The procedure laid out above is instrumental in the training of the models that approximate the policy functions, describing what happens in each episode. The training of the model is obtained by executing a certain number of episodes, updating the model approximating the policy function (as described in the Proposed Approach) with respect to the (s_{t-1}, a_t, r_t) triplets produced during the episode: for each step of the episode a new triplet is stored, and after the final step, these are used to update the weights of the models. The number of episodes executed is not fixed, and depends on the trend of the success rate: when it does not change significantly anymore, the learning phase is considered concluded.

4.2 Performance of *P1*

Regarding $P1$, 180,000 episodes were executed. The final success rate (the number of episodes concluded successfully over the total amount executed) was 82.11%. From the extended results in Table 3, we can notice that performances progressively decrease as the considered stopping range gets closer to the target. In particular, the highest success rates were observed for the "coffee" and the "ironing" activities, with a success rate of 84.06% and 84.05%, while the lowest success rate was obtained for the "PC" activity, with a success rate of 76.55%. Moreover, the columns *Mean Dist.* and *Succ. Dist.*, show the average final distance, for every activity, respectively if we consider all the episodes executed or only the ones that concluded with a success. In Fig. 3, we show an example of a trajectory followed by the robot to navigate toward the target. At the beginning of each episode, Pepper is conventionally located at (0, 0), whereas the coordinates of the user's shoulders are generated at random. In the example shown in the figure, the user is initially located to the left of Pepper. $P1$ makes Pepper execute the actions FWD and L. As a result, Pepper moves along the dashed trajectory. Eventually, it stops and finds itself located at a distance d from the user.

Table 3 Comparison of performance between the system using *P1* only, and the system using *P1* in conjunction with *P2* (*P1+P2* for short)

Act.	Range (m)	*P1* Succ. Rate (%)	*P1* Mean (m)	*P1* Succ. Dist. (m)	*P1* + *P2* Succ. Rate (%)	*P1* + *P2* Mean (m)	*P1* + *P2* Succ. Dist. (m)	Imp (%)
All		82.11	2.63	2.05	82.46	2.39	1.80	64.59
PC	0.3–1.5	76.55	2.15	1.49	78.67	1.92	1.22	69.37
Phone	1.2–2.4	83.35	2.81	2.26	84.67	2.54	1.97	66.30
TV	0.6–1.8	81.65	2.45	1.78	83.69	2.16	1.55	67.43
Coffee	1.5–2.7	84.06	2.84	2.33	83.13	2.63	2.09	60.86
Iron	1.8–3.0	84.05	2.97	2.46	80.13	2.74	2.27	56.51
Couch	0.9–2.1	83.00	2.57	2.00	84.49	2.31	1.70	66.96

4.3 Performance of *P2*

In the second presented setting, we integrate the user disengagement, obtained as described in Sect. 3.3.3, in the approaching process.

The performances of this approach are shown in Table 3. In particular, the contribution of $P2$ to the final result can be observed in columns *P1 succ. dist.* and *P1* + *P2 succ. dist.*, by looking at the distance from the user of the last state processed by *P1*+*P2*, along with the percentage of successful episodes where $P2$ chose to move closer to the target, in column *Impact*.

The results show that $P2$ succeeds in influencing consistently the outcome of the algorithm. As shown in Table 3, for each activity, $P2$ ends up choosing an action that moves the robot closer to the user in, at least, 50% of the episodes. Note that the PC has the closest ranges to the target. For this reason, slight movements lead to a failure condition. Regarding Iron, this activity corresponds instead to the far-away range. Particularly, this means that in a greater number of cases, wearable Activity Recognition will be used instead of a Camera-based one. As explained in the Experimental Procedure, this estimation is scaled down in the interval [0, 0.75] to simulate less precision of the wearable AR. Because of this, the reward gathered in this stopping range are smaller in absolute value than those in the other ranges, and ultimately this may affect the training process by updating the weights in a less pronounced way.

4.4 Comparison with Other Methods

To further demonstrate the effectiveness of our proof-of-concept, it is useful to compare it to other methods. Since the novelty of our proposal lies especially in a second policy function that adapts to user disengagement and activity, it is useful to replace our second policy function with another module that either (i) picks the final action at random, or (ii) always picks one of the actions. The resulting algorithms, therefore, plan to navigate towards the user using the first policy function and then perform a step that completely discards information about user disengagement and activity. We use these algorithms as our baselines for a comparison with the fully-fledged framework. The results are given in Table 4 and refer to 9000 test episodes. Note that the best mean disengagement and success rate are achieved by always performing action 0 as the last step, at the cost of a bad performance in terms of mean distance. On the other hand, always performing action 2 as the last step results in the best mean distance and worst mean disengagement and success rate. Letting our method $P2$ choose the last step results in a favorable compromise, whose performance sits somewhere in the middle of methods 1 and 2. $P2$ has the advantage of being able to learn this behavior automatically on the basis of our user model. Unlike other hard-coded methods, it can be simply re-trained if the user model changes.

Table 4 Comparison of policy 2 ($P2$) with baseline methods to perform the last step

Last step	Mean dist. (m)	Mean diseng	Succ. rate (%)
Random	1.58	0.44	72.76
0	*1.79*	**0.52**	**84.19**
1	1.60	0.47	79.71
2	1.48	0.40	71.02
3	**1.45**	*0.34*	*58.04*
P2	1.54	0.45	77.41

The *random* method uniformly draws one of the actions {0, 1, 2, 3} and performs it. Method 0 (resp. 1, 2, 3) always performs action 0 (resp. action 1, 2, 3) as the last step. Best performances are in bold, and worst performances are in italic

4.5 Discussion

Table 3 observes that, in our simulated environment, *P1* and *P2* together manage to get closer to the user when compared to *P1* alone. This could improve monitoring accuracy. Furthermore, note that in Table 3, the success rates for $P1$ and $P1 + P2$ are very similar, meaning that the improvement

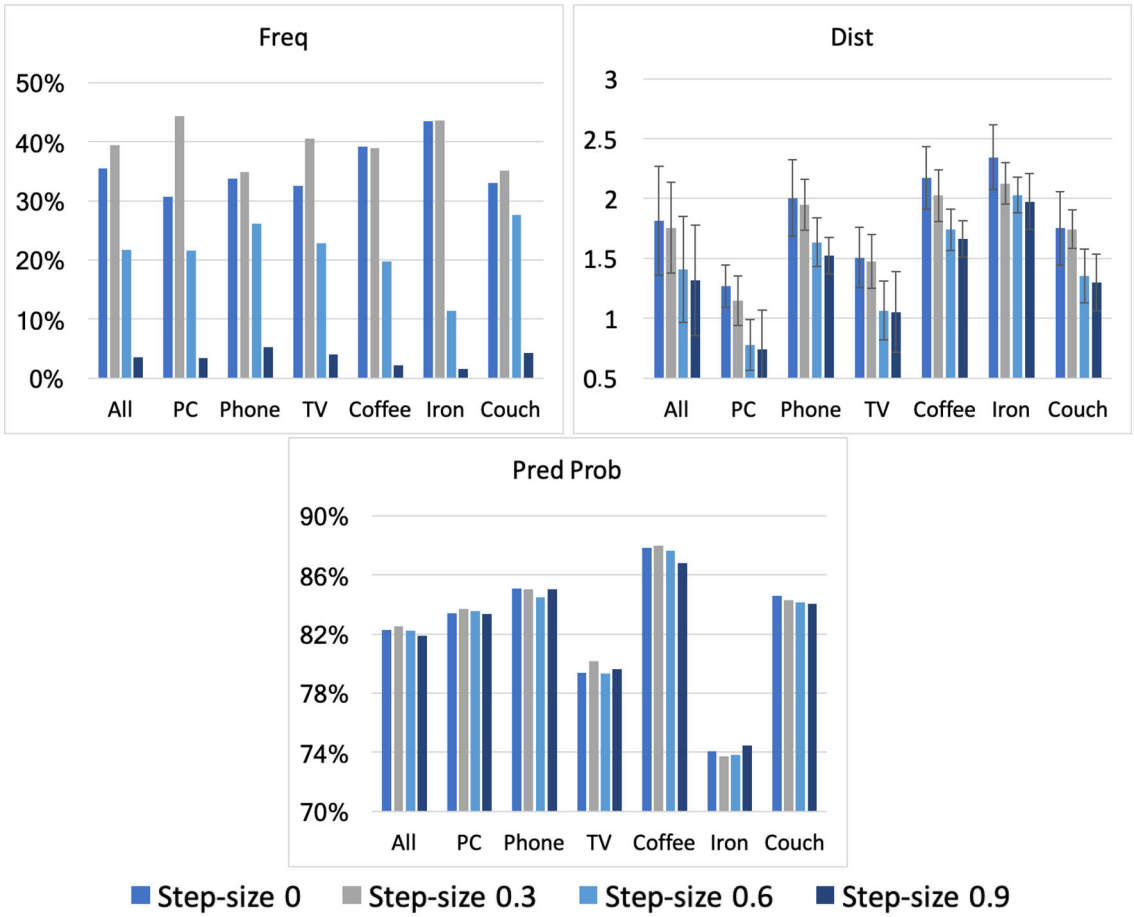

Fig. 4 RL performance for each activity and action. In particular, the histogram on the left shows the frequency of choosing each of the actions (*Freq*), and the central histogram shows the average distance from the user (*Dist*) with error bars corresponding to ±standard deviation, and the histogram on the right shows the average prediction probability (*Pred Prob*)

brought by $P2$ does not come at the expense of degrading the performance of the system. In Fig. 4, we show more details about the performances of $P2$ (considering only successful episodes), reporting for each action respectively the frequency of selection (Freq), the average final distance (Dist), and the prediction probability (Pred Prob).

Note that prediction probabilities do not differ within each activity, as it is evident from Fig. 4 by looking at the *Pred Prob* subfigure. Therefore, the average distance improvement of 0.26 m (see Table 3) brought by $P2$ is to be attributed almost entirely to disengagement levels. Getting closer to the user by 0.26 m improves the monitoring performance of the robot.

The prediction probabilities for the activity "Iron" are significantly lower than that of the other activities. The stopping range starts from 3.0 m, and therefore the prediction probabilities used could be obtained more likely with the wearable-based activity recognition module. As described in Sect. 4.1, the activity predictions of the wearable classifier are scaled to the interval [0, 0.75].

Table 4 shows that when compared to baseline methods, our second policy function P2 finds a good compromise between distance from the user, disengagement, and success rate, without being explicitly programmed.

Finally, let us reflect on the success rate of $P1 + P2$ of around 82%. We can consider this as a positive result, especially since what we are dealing with is a non-critical, harmless task: in a worst-case scenario, the user notices the robot and gets distracted. We definitely expect such a success rate to be lower in the real scenario. Nonetheless, since training was (partially) performed using data from real experiments, we do not expect this value to be dramatically lower. Ways of improving it include the use of more sensors and acquiring more data to improve the disengagement model.

5 Conclusion

The main goal of this work is to program a system that, in addition to navigating towards a human target, can optimize its stopping distance to other parameters depending on contextual information (e.g., the current activity) and subjective and personal preferences (e.g., disengagement). To this aim, we provide an algorithm that exploits RL to perform such optimization.

In our proof-of-concept, which constitutes the key contribution of this work, we chose to optimize the trade-off between the recognition probability (that depends on distance) and disengagement level, as encoded in the reward function of *P2*. To this aim, we use the RL paradigm, which is known to be highly effective as a device to perform this optimization. The effectiveness of our approach was tested and proved by showing that an agent that performs this optimization can get closer to the user with respect to a system that does not account for these parameters (i.e., the system consisting of *P1* only).

The behavior of the algorithm is further characterized by taking into account the activity performed by the target, which could affect the perception and impact of the presence of the robot on the user. Two Activity Recognition deep learning models (employing Long Short-Term Memory units) are employed to process wearable data and/or camera recordings, retrieved during experimental HRI sessions conducted with a Pepper robot [24]. The two modules are activated alternatively depending on the distance from the target.

The novel aspect of this work is the simulation of the user's reaction to the ongoing interaction. Specifically, we employ the simulated disengagement of the target to train *P2* to decide if (and eventually with which step-size) the agent has to move any closer to the user. This feature is of great interest, especially in the study of interaction with users with low levels of technological literacy or affected with mild impairments. To the best of our knowledge, the application of solutions similar to the one presented in this work has not been widely investigated in the literature yet.

This work leaves room for further improvement and extensions that may be addressed in the future. Extensions to the present approach should consider moving targets by e.g., integrating an obstacle avoidance strategy. Since the robot velocity has an impact on both proxemics as well as disengagement, the investigation of how velocity affects the disengagement of the user is planned for future work. Experiments within a real environment are planned to evaluate how the second policy has an impact on the user's disengagement. One of the most important limitations of the present works is that we did not carry out a comparison of disengagement metrics before and after the execution of P2, as we only tested our algorithms in a simulated environment. To validate the effectiveness of our approach, we plan to perform this comparison when we will perform these tests in an experimental setting with real users. Moreover, another extension could explore an ad-hoc model for user reaction estimation, able to consider other parameters and contextual information.

Funding Open access funding provided by Università degli Studi di Napoli Federico II within the CRUI-CARE Agreement.

Data Availability The datasets analyzed during the current study are available on request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Akalin N, Kiselev A, Kristoffersson A, Loutfi A (2018) Enhancing social human–robot interaction with deep reinforcement learning. In: FAIM/ISCA workshop on artificial intelligence for multimodal human robot interaction (AI-MHRI). MHRI, pp 48–50
2. Broadbent E, Stafford R, MacDonald B (2009) Acceptance of healthcare robots for the older population: review and future directions. Int J Soc Robot 1:319–330
3. Chen W, Zhang T, Zou Y (2018) Mobile robot path planning based on social interaction space in social environment. Int J Adv Robot Syst 15(3):1729881418776183
4. Dautenhahn K, Walters M, Woods S, Koay KL, Nehaniv CL, Sisbot A, Alami R, Siméon T (2006) How may i serve you? A robot companion approaching a seated person in a helping context. In: Proceedings of the 1st HRI, pp 172–179
5. Di Napoli C, Ercolano G, Rossi S (2022) Personalized home-care support for the elderly: a field experience with a social robot at home. User Model User Adapt Interact 1–36
6. Ercolano G, Raggioli L, Leone E, Ruocco M, Savino E, Ross S (2018) Seeking and approaching users in domestic environments: testing a reactive approach on two commercial robots. In: 27th IEEE RO-MAN. IEEE, pp 808–813
7. Hayashi K, Shiomi M, Kanda T, Hagita N (2012) Friendly patrolling: a model of natural encounters. In: Proceedings of the RSS. p 121
8. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
9. Hüttenrauch H, Eklundh KS, Green A, Topp EA (2006) Investigating spatial relationships in human–robot interaction. In: 2006 IEEE/RSJ IROS. IEEE, pp 5052–5059

10. Kanda T, Glas DF, Shiomi M, Hagita N (2009) Abstracting people's trajectories for social robots to proactively approach customers. IEEE Trans Robot 25(6):1382–1396
11. Kato Y, Kanda T, Ishiguro H (2015) May i help you? Design of human-like polite approaching behavior. In: 10th ACM/IEEE HRI. IEEE, pp 35–42
12. Khamassi M, Velentzas G, Tsitsimis T, Tzafestas C (2018) Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning. IEEE Trans Cognit Develop Syst 10(4):881–893
13. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
14. Kruse T, Pandey AK, Alami R, Kirsch A (2013) Human-aware robot navigation: a survey. Robot Auton Syst 61(12):1726–1743
15. McDuff D, Mahmoud A, Mavadati M, Amr M, Turcot J, El Kaliouby R (2016) Affdex SDK: a cross-platform real-time multi-face expression recognition toolkit. In: Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems. ACM, pp 3723–3726
16. Mead R, Mataric MJ (2015) Robots have needs too: people adapt their proxemic preferences to improve autonomous robot recognition of human social signals. New Front Hum Robot Interact 100:100–107
17. Michalowski MP, Sabanovic S, Simmons R (2006) A spatial model of engagement for a social robot. In: 9th IEEE international workshop on advanced motion control. IEEE, pp 762–767
18. Mumm J, Mutlu B (2011) Human–robot proxemics: physical and psychological distancing in human–robot interaction. In: Proceedings of HRI, pp 331–338
19. Pino M, Boulay M, Jouen F, Rigaud A-S (2015) "Are we ready for robots that care for us?" attitudes and opinions of older adults toward socially assistive robots. Front Aging Neurosci 7:141
20. Qureshi AH, Nakamura Y, Yoshikawa Y, Ishiguro H (2016) Robot gains social intelligence through multimodal deep reinforcement learning. In: 2016 IEEE-RAS 16th International Conference on Humanoid robots (humanoids). IEEE, pp 745–751
21. Raggioli L, Rossi S (2019) A reinforcement-learning approach for adaptive and comfortable assistive robot monitoring behavior. In: 2019 28th IEEE international conference on robot and human interactive communication (RO-MAN). IEEE, pp 1–6
22. Rios-Martinez J, Spalanzani A, Laugier C (2015) From proxemics theory to socially-aware navigation: a survey. Int J Soc Robot 7(2):137–153
23. Rossi S, Rossi A, Dautenhahn K (2020) The secret life of robots: perspectives and challenges for robot's behaviours during non-interactive tasks. Int J Soc Robot 12(6):1265–1278
24. Rossi S, Ercolano G, Raggioli L, Savino E, Ruocco M (2018) The disappearing robot: an analysis of disengagement and distraction during non-interactive tasks. In: 27th IEEE RO-MAN. IEEE, pp 522–527
25. Rossi S, Staffa M, Bove L, Capasso R, Ercolano G (2017) User's personality and activity influence on HRI comfortable distances. In: International conference on social robotics. Springer, pp 167–177
26. Satake S, Kanda T, Glas DF, Imai M, Ishiguro H, Hagita N (2009) How to approach humans? Strategies for social robots to initiate interaction. In: Proceedings of the 4th ACM/IEEE international conference on human robot interaction, pp 109–116
27. Sisbot EA, Marin-Urias LF, Alami R, Simeon T (2007) A human aware mobile robot motion planner. IEEE Trans Robot 23(5):874–883
28. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, Cambridge
29. Tsitsimis T, Velentzas G, Khamassi M, Tzafestas C (2017) Online adaptation to human engagement perturbations in simulated human-robot interaction using hybrid reinforcement learning. MultiLearn workshop at the 25th European Signal Processing Conference (EUSIPCO 2017)
30. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3–4):229–256
31. Ya-Huei W, Cristancho-Lacroix V, Fassert C, Faucounau V, de Rotrou J, Rigaud A-S (2016) The attitudes and perceptions of older adults with mild cognitive impairment toward an assistive robot. J Appl Gerontol 35(1):3–17

Luca Raggioli is a Ph.D. student in Artificial Intelligence and Cognitive Robotics at the University of Manchester funded by a Marie Skłodowska-Curie Actions fellowship. He got a BSc and a MSc in Computer Science from the University of Naples Federico II. His main interests include applications of Artificial Intelligence in Robotics, Cognitive and Developmental Robotics, and Human-Robot Interaction.

Fabio Aurelio D'Asaro is an Assistant Professor at the University of Verona. Prior to this, he was a postdoctoral researcher at the University of Milan and at the University of Naples Federico II and got his PhD in Artificial Intelligence from University College London. His primary research interests revolve around logical models of reasoning and their applications to Transparent and Trustworthy AI.

Silvia Rossi received the M.Sc. degree in Physics and the Ph.D. in Information and Communication Technologies. She is an Associate Professor in Computer Science at the Department of Electrical Engineering and Information Technologies, University of Naples Federico II, where she is currently co-chief of the PRISCA (Intelligent Robotics and Advanced Cognitive System Projects) Laboratory. Her research interests include Multi-agent Systems, Human-Robot Interaction, Socially Assistive Robotics, and Recommender Systems. She is an Associate Editor for IEEE Robotics and Automation Letters (RA-L), International Journal of Social Robotics, Pattern Recognition Letters, and Intelligent Service Robotics journal.