Adapting robot task planning to user preferences: An assistive shoe dressing example

Gerard Canal · Guillem Alenyà · Carme Torras

Abstract Healthcare robots will be the next big advance in humans' domestic welfare, with robots able to assist elderly people and users with disabilities. However, each user has his/her own preferences, needs and abilities. Therefore, robotic assistants will need to adapt to them, behaving accordingly. Towards this goal, we propose a method to perform behavior adaptation to the user preferences, using symbolic task planning. A user model is built from the user's answers to simple questions with a Fuzzy Inference System, and it is then integrated into the planning domain. We describe an adaptation method based on both the user satisfaction and the execution outcome, depending on which penalizations are applied to the planner's rules. We demonstrate the application of the adaptation method in a simple shoe-fitting scenario, with experiments performed in a simulated user environment. The results show quick behavior adaptation, even when the user behavior changes, as well as robustness to wrong inference of the initial user model. Finally, some insights in a non-simulated world shoe-fitting setup are also provided.

Keywords Planning with preferences · Behavior adaptation · Task personalization · Shoe fitting

This work has been partially supported by the CHIST-ERA Project I-DRESS PCIN-2015-147, by the Spanish Ministry of Science and Innovation HuMoUR TIN2017-90086-R and the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656). Gerard Canal is also supported by the Spanish Ministry of Education, Culture and Sport via a doctoral grant FPU15/00504.

G. Canal · G. Alenyà · C. Torras
Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens i Artigas 4-6, 08028 Barcelona, Spain
E-mail: {gcanal, galenya, torras}@iri.upc.edu

1 Introduction

The Assistive Robotics field is an area of growing interest in which robots are used as a *tool* to help caregivers and nurses to better assist human users with special needs. Such robotic systems can allow these people to perform some Activities of Daily Living (ADLs) by themselves, hence empowering them (Chen et al, 2013).

However, the potential users of these systems, caregivers or disabled and elderly people themselves, may find it difficult to manipulate or configure the system. Thus, natural interfaces as well as suitable adaptation mechanisms must be developed to ease robot instruction and involve users in the task.

Furthermore, the robot should not try to perform the task in a general way, but to take each individual as a unique person who has special capabilities, tastes and feelings, therefore providing a personalized assistance. For example, a robot should not treat a person that trusts and fully accepts the robot in the same manner as a user that expresses some concerns.

In this paper, we propose a method to obtain the actions preferred by the user to then drive an off-the-shelf planner towards the plan that best suits him/her. To do so, we follow the FUTE (Factory setting, User Tailoring, Execution tuning) framework approach for assistive robotic applications (Canal et al, 2016) in which there is an initial phase, called Factory setting, where the robot is configured for a general user. Then, once the robot is to assist a particular person, the User Tailoring process is performed in order to adapt the robot behavior to that specific user. Finally, the robot assists the user by performing an Execution tuning using the adapted parameters to perform the task in the user preferred manner. The framework has been used to develop a behavior adaptation method based on stochastic planning, which has been exemplified with a shoe-fitting domain, such

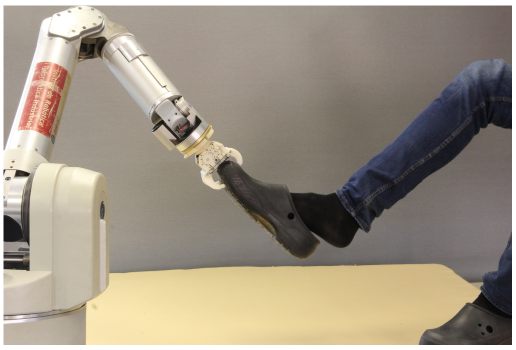

Fig. 1 Shoe-fitting example.

as the one shown in Figure 1, in which the user is able to determine the interaction level as well as the speed of the actions. The preferences are obtained from the answers to indirectly-related questions and the system evolves to the final user model based on the outcome of the actions while they are performed. We will consider preferences of the "information obtaining" type and the robot motion speed type, as defined in the taxonomy of user preferences in assistive scenarios from Canal et al (2017). The method has been tested with simulated users to provide a constant behavior to which the method adapts, and its deployment on a real robot has been assessed.

The remainder of the paper is as follows: Section 2 reviews similar works in the field. Section 3 explains the proposed methodology, while Section 4 is devoted to its evaluation. Finally, the main conclusions are depicted in Section 5.

2 Related work

Robot behavior personalization and adaptation is an interesting topic which is gaining lots of attention from the research community. And personalization can make a difference for the users, specially in the case of assistive scenarios, in which robots help people with disabilities or age-related issues (Robinson et al, 2014). In close-contact applications such as feeding or dressing, taking into account the needs and abilities of the user is essential for the success of the task.

There are different works that tackle dressing scenarios similar to the one we propose. Gao et al (2015) tackle the problem of assisting a user to put on a sleeveless jacket with the help of a Baxter robot. They model the user's movement space using Gaussian Mixture Models, the user pose being obtained by means of a depth camera. The model is used to dress the user taking into account their movement capabilities. Later on, they propose an online iterative path optimisation method (Gao et al, 2016). By means of vision and force estimation, they find the optimal personalized path to help a user to put on a jacket.

Similarly, Chance et al (2016) use a Baxter robot to put on a sleeve of a jacket to a wooden mannequin. They explore dressing error detection using force sensors and an Inertial Measurement Unit (IMU) installed in the end-effector of the robot. Moreover, speech recognition is employed to enable the user to correct the end-effector trajectory, which is planned for different arm positions.

Yamazaki et al (2014) develop a procedure to help disabled users to put on trousers. Visual information is used to recognize the trousers state, while force sensing is also employed to detect failures. The system is able to adapt to leg differences by using different trajectory segments and fitting them to the current user.

Tamei et al (2011) use reinforcement learning to dress a mannequin with a shirt by means of a dual-arm robot. They adapt to different person postures, and represent the state using the topological relation between the garment and the user. The system is able to modify the arms motion to insert the shirt in the mannequin's head.

Another dressing example is the one by Klee et al (2015), in which a robot assists the user to put on a hat. This is performed in a collaborative manner by taking turns when moving. The robot learns the user's limitations as constraints, which are used to personalize the repositioning requests to the user. The dressing task is represented as a sequence of robot goal poses with respect to the user. The robot tries to fulfill the goals, asking the user to reposition him/herself when the motion planning fails.

In our approach, we are interested in viewing the dressing task from a higher-level perspective, in which there are different actions available to fulfill the task, and the user's preferences are taken into account to choose one action instead of another, while in the case of Gao et al (2015) and Klee et al (2015), they model the user capabilities to adapt the robot's movement rather than using preferences.

Our approach uses a planner to choose the most suitable action for the user. Planning with preferences has been slightly explored in different scenarios. The Human Aware Task Planner (HATP), by Alili et al (2009), is able to define plans in environments in which other agents, such as humans, are present. It performs plans that take into account the state and capacities of the other agents and anticipates their actions. The plans should also satisfy social rules, which are implemented as penalties to the agent's behavior. The Hierarchical Agent based Task Planner (Lallement et al, 2014; de Silva et al, 2015) is a Hierarchical Task Network (HTN) planner that treats the different agents in the environment as first-class entities in the domain representation language. Moreover, it is able to split the final solution into different subsolutions for the different agents. Fiore et al (2016) propose a system able to execute collaborative tasks with a user taking into account the preferences of the human partner by providing three different operation modalities: one in which the human plans and asks the robot for single tasks, another where the robot computes a plan to fulfill the joint goal with the human, and one in which the robot is able to adapt the plans to the human actions by proactively executing actions towards the goal. The method is evaluated in an object manipulation scenario. However, these approaches are designed to work in a collaborative task solving scenario which does not suit the assistive tasks we are planning to tackle. Moreover, their notion of user preference is related to allowing the human to take the lead or leave the reasoning to the robot, while in our case the preferences are related to how the user prefers the task to be done in terms of the chosen actions.

Castellano et al (2016) explore how the task context, the social context and their interdependencies can be used to predict the affective state of the user and the quality of the interaction. They show that the task context along with social context-based features are better than turn-based features to predict social engagement and affective states of the user. In our work, we distinguish between interaction actions, which may relate to the social context, and task actions, related to their game context, as we believe the addition of the interaction can improve the task actions' performance.

We propose an adaptation mechanism that takes into account user feedback to tune the system. Other similar examples are mainly related to reinforcement learning, such as Thomaz and Breazeal (2006), in which reward signals from users are used to provide feedback about past actions as well as to guide the future ones. In the TAMER framework Knox and Stone (2009), the human trainer interactively shapes the agent's policy by providing reinforcement signals. A different approach is the one by Griffith et al (2013), in which the policy is shaped directly by human feedback rather than using such feedback as a shaping reward.

3 User-oriented task planning

Assistive tasks, such as shoe-fitting, tend to be complex. Apart from the usual uncertainties that are found in all kinds of robotic applications, such as noisy perceptions and inaccurate actions, these tasks usually involve physical contact with a human who will probably be unfamiliar with the robot and, therefore, may have some difficulties due to mobility, age or cognitive impairments. Therefore, simple reactive techniques are not enough to handle all the involved uncertainties in a safe manner. Note that the user behavior cannot be accurately predicted and this may introduce multiple sources of error in the interaction. That is why symbolic planning techniques are useful in this kind of environments as they provide an appropriate abstraction of the task. A planner is used to obtain a sequence of actions to drive the system from an *initial* state to a *goal* state in which the task is completed or some criterion is satisfied. For instance, there will be applications in which the running time needs to be minimized, others will rather use the minimum number of actions or will try to maximize a target function.

In the case we present, the goal of the planner is to balance the **satisfaction of the user** and obtain the shortest possible plan, with maximum acquired reward. We aim to obtain a plan that selects the action that will best suit the person, and takes into account their needs and preferences.

A planning problem can be formulated as a Markov Decision Process (MDP), which is defined by a five tuple $\langle S, A, P, R, \gamma\rangle$ where

- S: set of discrete states.
- A: set of actions that can be performed.
- $P(s'|s,a)$: transition function computing the probability of obtaining a new state s' when action a is executed in state s.
- $R : S \times A \to \mathbb{R}$: the reward function.
- $\gamma \in [0,1)$: discount factor for future rewards degradation.

With this representation, the planner finds a policy $\pi : S \to A$ that maximizes a value function (sum of expected rewards) for a given state.

There are two main families of symbolic planners: deterministic planners, in which the actions can yield one unique outcome; and stochastic planners, able to handle non-deterministic actions where different outcomes can happen with a certain probability. In this work, we use a probabilistic planner because we consider that each action can lead to different results. The probability associated with each one of the different outcomes encodes naturally the uncertainty of each action (see a formal example below).

More precisely, we will define the problem domain using a set of Noisy Indeterministic Deictic rules (NID) (Pasula et al, 2007). Briefly, each NID rule models one action execution in a given state, and can lead to different next states, each one with a different associated probability P_o^r. Each NID rule is defined by its preconditions, which are the predicates that must be satisfied in the state in order to apply the rule, and its effects, which are the changes that are applied to the state, each of them with an associated probability. An example of NID rule is:

```
Action: approachFoot(F - foot)
Preconditions:
  - not(reachableFoot(F))
  - shoeInGripper(S - shoe)
  - not footMoving(F)
  - inWorkingSpace(F)
Effects:
  - reachableFoot(F)        ($P_{o_s} = 0.80$)
      -- (Successful outcome)
  - footMoving(F)           ($P_{o_2} = 0.15$)
  - not(inWorkingSpace(F))  ($P_{o_3} = 0.05$)
```

Note that an action of the domain can be represented by many NID rules while each rule can only represent one action. For instance, the action `approachFoot` may be defined by different NID rules with different outcomes, although each one of the rules is only linked to a single action - the `approachFoot` one.

We want to clearly separate the action outcomes and the user model. As explained before, action outcomes are modeled by NID rules representing the probabilities of the different expected outcomes of each action. Complementarily, we propose to include the user preferences as a part of the planning domain in the form of expected behavior of the robot (see Sec. 3.1), for example, the ones we have used in the experiments: maximum expected velocity and degree of verbal interaction (Sec. 4). Note that, although we will focus on speed and verbal feedback preferences, other preferences such as maximum force, preferred approach direction or non-verbal communications could also be included. However, the acquisition of such preferences by the robot should be easy and natural for the user. Thus, rather than trying to obtain the actual preferences directly, we propose to ask simple and apparently unrelated questions to the user. Taking inspiration from the Numerical Pain Rating Scale (NPRS), which is one of the most common pain assessment scales used in nursery (McLafferty and Farley, 2008), we believe it is easier for the user to express preferences by means of a numerical score. Therefore, we also use numerical scores to assess the user state and preferences:

– From 0 to 10, how confident do you feel with the robot?
– From 0 to 5, how comfortable are you now?

The answers are used to infer preferences such as the speed of the robot and the interaction level. This is achieved by the addition of a Fuzzy Inference System (FIS) to transform the user answers to planning domain predicates, as shown in Fig. 4. Moreover, a similar method is used to obtain a feedback value after each robot interaction, which is used as a scoring method employed to refine the preferences and adapt to possible biases. In this case, we ask the user about their satisfaction score (from 0 to 10) with the overall interaction, and use it as the input to another FIS that provides the feedback value.

Algorithm 1 Preference-based task personalization

```
    – Initialization –
 1: userInfo ← getUserInfo()
 2: userPreferencesPredicates ← FIS(userInfo)   ▷ Section 3.2
 3: planningDomain ← add(userPreferencesPredicates)
 4: for all $r \in \{ruleset\}$ do
 5:     if satisfiesUserModel($r$) then
 6:         updateSatisfyingRule($r$)           ▷ Use Eqs. 1 and 3
 7:     else
 8:         updateNonSatisfyingRule($r$)        ▷ Use Eqs. 2 and 4
    – Task execution –
 9: repeat
10:     nextRule ← getSuitableRule(ruleset)     ▷ Planning step
11:     success ← executeAction(nextRule)
12:     usedRules ← append(nextRule)
13:     removeUnsuccessfulRules(ruleset)        ▷ Force exploration
14: until taskIsComplete()
    – Update outcome probabilities based on experience –
15: for all $r \in \{usedRules\}$ do
16:     updateRuleProbabilities($r$)            ▷ Use Eq. 5
    – Update the executed rules based on user feedback –
17: userSatisfaction ← getUserSatisfaction()
18: userFeedback ← FIS(userSatisfaction)
19: for all $r \in \{usedRules\}$ do
20:     if ruleWasSuccessful($r$) then
21:         updatePenalizations($r$, userFeedback) ▷ Use Eq. 6
```

Although it has been shown that performance rating, similar to the one we are using with the satisfaction, is influenced by the user's empathy and trust Kühnlenz et al (2013); Muir (1987), in this paper we are using the FIS only as an example of the inputs that can be fed to the adaptation system. However, a more sophisticated FIS could also be employed, as well as other methods that provide a numerical feedback measure. For instance, the satisfaction value obtained from the user could be weighted by the perceived confidence and trust of the user in the system to overcome this confidence and trust bias. User acceptance, measured using methods such as Heerink et al (2009), would be another useful metric to balance the user feedback.

A system representation is depicted in Figure 2. The process is also shown in Algorithm 1, in which lines 1-8 consist in the initial refinement of the planning operators, lines 9-14 are the execution of the task, and lines 15-21 are the update of the planning operators based on the outcome of the task and the user's satisfaction.

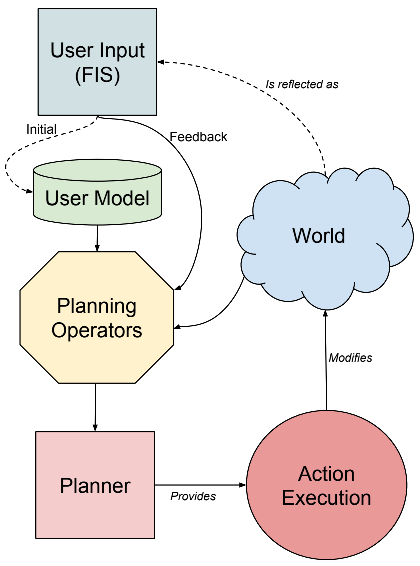

Fig. 2 System flow representation. Notice the action execution loop from *Planning Operators - Planner - Action Execution - World*, which goes to *User Input* once the task is completed to adapt the *Planning Operators* based on the user feedback.

The following sections describe the details of these methods. For illustration and better understanding, we will use a simple shoe-fitting task to exemplify the used methodology.

3.1 Domain definition

We propose to add preference-related predicates directly into the planning domain in order to guide the planner towards the user's preferred sequence of actions.

Then, the actions are defined so that those actions not complying with the user model are penalized and thus are less likely to be chosen. To achieve this behavior, each NID rule has an extra cost associated to the compliance of the rule with the current user model. For this reason, each action has an associated rule for each combination of user preference predicates, all of them including its own execution cost (fixed penalization for the execution of the action), user model penalizations and stochastic outcomes when needed.

Note that with this definition, there are multiple paths to transform the world from the initial state to the goal state. However, as the planner is set up to minimize the cost (which could also be seen as maximizing the negative reward), those actions complying with the user model will result in a lower penalization and will be favored by the planner.

We use a simple shoe-fitting scenario to explain and test the method. For this shoe-fitting domain, we have defined three **movement actions**:

- `approachFoot`: The robot approaches the person's foot with the shoe in the gripper. Possible failures are that the user moves away if he/she is disturbed by the sudden robot motion, or moving the foot aside in a hard-to-reach position if he/she gets tired because the robot takes too long. The corresponding NID rule has been shown in the example in Section 3.
- `insertShoeInFoot`: The robot inserts the shoe in the foot. The action will fail if the foot is moving, the foot is in an incorrect pose or the person has put the foot aside. In the latter case, the robot will have to approach the foot again. An example of NID rule for this action is:

```
Action: insertShoeInFoot(F - foot, S - shoe)
Preconditions:
  - reachableFoot(F)
  - shoeInGripper(S)
  - not(footMoving(F))
  - bareFoot(F)
  - correctPose(F)
Effects:
  - shoeInFoot(S, F) & not(bareFoot(F))
                                    ($P_{o_s} = 0.850$)
       -- (Successful outcome)
  - not(footInCorrectPose(F)) ($P_{o_2} = 0.0625$)
  - footMoving(F)             ($P_{o_3} = 0.0625$)
  - not(inWorkingSpace(F)) &
    not(reachableFoot(F))     ($P_{o_4} = 0.025$)
```

- `releaseShoe`: The robot releases the shoe, which has already been placed in the foot. We assume this action does not fail (though it may result more or less pleasant to the user depending on its execution). A NID rule that represents the release action is:

```
Action: releaseShoe(S - shoe)
Preconditions:
  - shoeInGripper(S)
  - not(footMoving(F))
  - shoeInFoot(S - shoe, F)
Effects:
  -  not(shoeInHand(S)) ($P_{o_s} = 1.0$)
       -- (Successful outcome)
```

We have also defined two **interaction actions**:

- `informUser`: The robot informs the user about the next action that will be performed. We expect less failures if the user knows in advance the robot intentions, but the overall task will last longer.
- `askUser`: The robot asks the user to do something when the current state is not the expected one. For instance, the robot can ask the user to stop moving the foot or to set the foot in the working area.

Examples of wrong action outcomes are depicted in Figure 3. All the actions can be executed either in a *quick*, *intermediate* or *slow* speed, and information may have

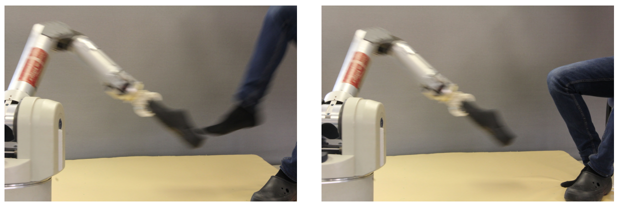
(a) The user moves the foot away from the robot. (b) The user has put the foot aside.

Fig. 3 Example of shoe-fitting action failures.

been given to the user or not before every action execution. The user model predicates are the speed modifier $sm \in \{quick, slow, intermediate\}$ and the informer mode is defined as $im \in \{informer, not\ informer\}$. So, there are six rules per action, one for each combination of sm and im. In case of an action failure, the robot uses the `askUser` action to obtain the missing condition, so it may ask the user to reposition or reorient the foot if the action failed for this reason. Other task-related predicates are used to define the state of the environment. Examples of these predicates are: *reachableFoot(F)*, *footMoving(F)*, *inWorkingSpace(F)*, *shoeInGripper(S)* and *shoeInFoot(S, F)*.

In this paper, we focus on planning with high-level symbolic actions, similar to the ones defined in other frameworks such as the high-level operations in ARMAR-X Vahrenkamp et al (2015); or similar to the non-primitive tasks of the HTN planning framework Erol et al (1994). Therefore, we will assume that the robot already knows how to perform such actions. These low-level smart actions are learned beforehand (in the Factory setting phase) using a learning framework such as the one presented by Rozo et al (2016); Pignat and Calinon (2017). Note that these smart low-level actions are able to interact with the environment, for instance using the foot as reference to modify the learned trajectory.

Once the planner issues an action, the low-level controller executes it, handling elements such as perception and robot motion. The trajectories are taught kinesthetically, as in Canal et al (2016). The perception is implemented using an RGB-D sensor such as a Kinect$^{\text{TM}}$ sensor, from which a 3D point cloud is obtained and processed to obtain the foot's location and build the symbolic state.

A typical execution scenario would start with the robot holding the shoe and the user seated in front, as shown in Figure 1. Then, if the user was defined as $im = informer$ the robot will tell the user that it is going to approach the shoe to the foot. Then, it will start the `approachFoot` action. If the user model specifies so, an utterance informing the insertion will follow, and the `insertShoeInFoot` action will be executed afterwards. Finally, the `releaseShoe` action will be performed, having informed the user beforehand if required. Therefore, movement actions are interleaved with interactive actions in the plan. The resulting plan sequence is

```
1: approachFoot(F)
2: insertShoeInFoot(F)
3: releaseShoe(F)
```

in the non-informative case, and

```
1: informUser
2: approachFoot(F)
3: informUser
4: insertShoeInFoot(F)
5: informUser
6: releaseShoe(F)
```

when the user is to be informed. Note that, after adaptation, the system may use informing actions only before one conflicting action, avoiding the utterance prior to the execution of the rest of actions. In case of failure, the `askUser` action is performed to interact with the user and return the system to a known state, suitable to continue with the plan execution:

```
1: informUser
2: approachFoot(F)
   -- Failure: User moves the foot away
3: askUser(approach)
4: informUser
5: approachFoot(F)
6: informUser
7: insertShoeInFoot(F)
8: informUser
9: releaseShoe(F)
```

3.2 Fuzzy user model extraction

As already introduced, we use two simple questions in order to obtain the user traits. This step corresponds to the arrow that goes from *User Input* to *User Model* in Figure 2, and is shown in lines 1-3 in Alg. 1. The answer to the questions is fed to a Mamdani-like Fuzzy Inference System (FIS) (Mamdani and Assilian, 1975) built using a simple fuzzy library (Rada-Vilela, 2014), which consists of a rule block that outputs the predicates relative to the inferred preferred speed of the actions as well as whether the robot should inform the user before every action execution or not.

Another FIS is used to obtain the feedback value, corresponding to the arrow from *User Input* to *Planning Operators* in Figure 2. The feedback computation is also shown in lines 15 and 18 in Alg. 1. In this case, the user is asked about his/her satisfaction with the executed task, also in a value between 0 and 10. The satisfaction, along with the initial confidence value, provides the feedback score which is used to update the penalizations of the rules.

The linguistic variables have been defined as follows. The ranges of the variables can be seen in Figure 4 along with the fuzzy inference systems.

- Confidence (*Input*, $[0, 10]$): Includes the terms "very unconfident", "unconfident", "confident", and "very confident".
- Comfortability (*Input*, $[0, 5]$): Includes the terms "none", "low" and "high".
- Satisfaction (*Input*, $[0, 10]$): Includes the terms "very unsatisfied", "unsatisfied", "slightly satisfied", "satisfied" and "very satisfied".
- Speed (*Output*, $[0, 15]$): Includes the terms "slow", "intermediate" and "quick".
- Informer (*Output*, $[0, 1]$): Includes the terms "yes" and "no".
- Feedback (*Output*, $[-5, 5]$): Includes the terms "worst", "bad", "neutral", "good" and "best".

3.3 Initial model refinement

Once the predicates conforming the initial user model are defined, all the rules' specifications are refined to favor more those complying with the user model. Here we are in the step from *User Model* to *Planning Operators* in Figure 2. The *Planning Operators* from the figure correspond to the NID rules defined in Section 3.1. The initial refinement corresponds to the block comprising lines 4 to 8 in Alg. 1 Building on the intuition that the rules satisfying the user preferences will be more likely to succeed, we increase the probability of the successful outcome of those rules, at the same time that we decrease the one of the rest of rules. For each rule r, the probability of the successful outcome $P^r_{o_s}$ (the one that allows the planner to advance towards the goal[1]) is updated as follows. For the rules that satisfy the user model predicates, we increment it as

$$P^r_{o_s} = P^r_{o_s} + \frac{1 - P^r_{o_s}}{K}, \tag{1}$$

while probabilities of the rules not complying with the user model are decreased as

$$P^r_{o_s} = P^r_{o_s} - \frac{P^r_{o_s}}{K}. \tag{2}$$

This update sets the value of $P^r_{o_s}$ between $[\frac{1}{K}, 1]$ when increasing the probability and between $[0, \frac{(K-1)}{K}]$ when it is decreased. The idea is to increase more the lower probabilities that satisfy the user model, while only slightly increasing those which were already high. As a counterpart, the same principle is applied when the probabilities are decreased. We have used a value of $K = 3$ in all the performed experiments. After the update of $P^r_{o_s}$, the probabilities of the rest of the outcomes are also tuned so they sum up to one. This is achieved by applying the opposite equation to the other outcomes. That is, in the cases in which we applied Equation 1 to $P^r_{o_s}$, we apply Equation 2 to the other rule's outcomes, and viceversa.

Similarly, we update the penalizations applied to all the rules based on the user model. Each rule has, apart from the fixed execution cost, a speed penalization and an interaction penalization. These penalizations are applied when a NID rule not satisfying the user model is executed. Thus, we aim to increase the cost of the rules whose penalization condition is satisfied by the current user model , and lower it in the other cases. For instance, in the shoe-fitting scenario, the `approachFoot` with the $[quick,\ informer]$ modifiers will imply a penalization when the user model is defined as $im = (not\ informer)$, and another one when the user model is either $sm = slow$ or $sm = intermediate$. Be R^r_c the penalization of type c of the rule r, we update it as follows. When the rule is not adequate for the current user model, we update the costs[2] that satisfy the user model as

$$R^r_c = min(R_{max}, max(R_{min}, R^r_c - C)). \tag{3}$$

To keep the cost balanced, the opposite is applied for the rules satisfying the user model (note that in this case, penalizations will not be applied since the user model is satisfied):

$$R^r_c = min(R_{max}, max(R_{min}, R^r_c + C)). \tag{4}$$

R_{min} and R_{max} are used to maintain the costs in a reasonable range, avoiding the degeneration of the system in the long term. C is a fixed constant value used as update factor.

3.4 Improvement based on user feedback

With the proposed problem definition, the system is able to provide plans that comply with the specified

[1] For simplicity, we consider only one successful outcome, though it can be easily extended to several successful outcomes.

[2] We define costs as negative rewards, thus subtracting to the previous cost we are worsening it.

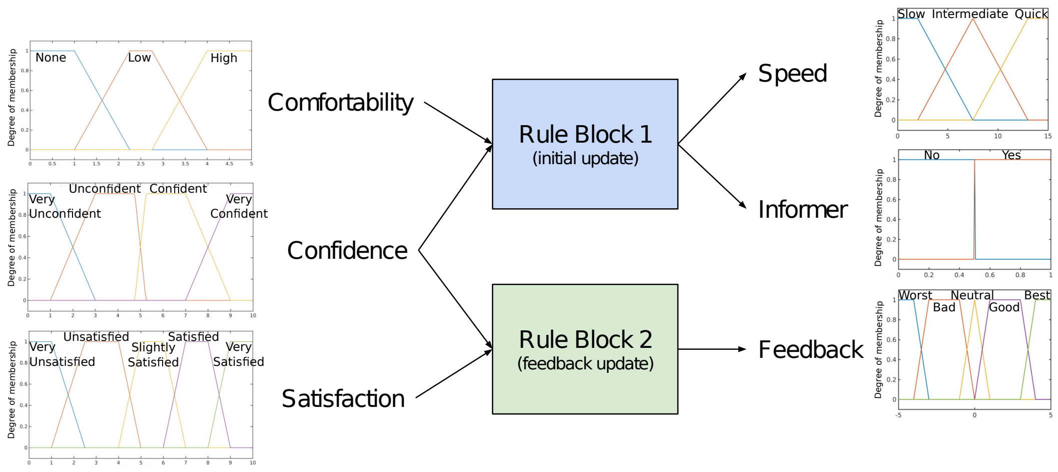

Fig. 4 Fuzzy inference system used to obtain the initial user model, as well as its improvement using feedback.

user model. Nevertheless, the user model may not be properly determined. Given that the preferences are specified by the user him/herself, they may not be accurate. For instance, imagine fitting a shoe to a user who specifies that the robot should move quickly, but he/she is not confident enough with the robot, so when the robot moves quickly he/she gets scared and puts his/her foot aside. Moreover, the user may change his/her behavior with respect to the robot with the use, as his/her confidence will increase when he/she is accustomed to the robot. Thus, adaptation is needed to cope with these user model deviations.

The adaptation is performed in a similar manner to the initial refinement, but it is carried out each time the task is fulfilled. However, some distinctions are taken into account at this point. We believe that the users' satisfaction can not be measured only by the outcome of the actions. Thus, a failed action does not imply that the action was not suitable for the user in the same way a successful action does not indicate that it was the best for that user. For this reason, we will update the probabilities based on the expected outcome, while the penalizations will be updated based on the feedback obtained from the user, them being related to the user preferences. This favors actions that follow the user model even though they have low probability of success, allowing for more exploration towards the user model. In case the user preferences were not correctly established, the adaptation procedure will modify the penalizations, along with the probabilities, towards the correct behavior. This rule update based on user feedback corresponds to the lines 15 to 21 in Alg. 1.

3.4.1 Outcome probability update based on executed actions

For the probability update, we use the decreasing m-estimate as defined by Martínez et al (2015). We update the probability of the i_{th} outcome of rule r, P_i^r as

$$P_i^r = \frac{p + \frac{m}{\sqrt{p+n}} P_{i,0}^r}{p + n + \frac{m}{\sqrt{p+n}}}, \tag{5}$$

where p is the number of positive examples (number of times i was the execution outcome), n the number of negative examples (number of times i was not the execution outcome) and $P_{i,0}^r$ is the prior or initial probability (defined in the Factory setting phase).

3.4.2 Rule penalization update based on user feedback

The feedback is then used to update every rule r that was successfully applied during the task execution. We only use the rules that were successfully applied because the feedback value is the score of the whole execution rather than that of individual actions. Therefore, a positive score would diminish the cost of those rules that failed, which would not lead the system to the user preferred behavior.

The penalization update for each cost c of the rule r is computed as

$$R_c^r = min(R_{max}, max(R_{min}, R_c^r + C\frac{1}{f})), \tag{6}$$

where f is the feedback value represented as the user's task score in the range $[-5, 5]$. In case the user feedback is negative, the costs will be worsened and this will lead the system to explore beyond the current user model. Otherwise, the current rule definition will be updated as it satisfies the user, and successful rules will have more chances to be applied.

After the feedback update step, all the costs are normalized in order to keep balance and avoid the degenerated case in which all the rules of the system have the minimum cost[3]. Therefore, for each action we normalize each kind of cost of its rules so they always sum up to the same. Thus, decreasing a cost for one rule increases those of the other rules of the same action.

While the task is being carried out, the planner is used after each action execution to build a new plan to go from the current state to the goal one, which will compute a recovery plan if the action failed. Therefore, when an action fails and the system recovers to a previous state, the planner will suggest the same action rule again. And, if the action failed due to the user not being satisfied with it, it is likely to fail again. This is solved by removing the failed rule from the set of available rules after 3 failed execution attempts. The removal of the rule forces the system to explore other options. The rules are removed only during the task execution, and are re-added to the set after the task has been finished, just before the update of the probabilities and penalizations.

In this paper we use 3 attempts arbitrarily, following the next rationale: one failed attempt might have been caused by the robot or some other element. Two failed attempts are more suspicious, but can still be due to the robot. After a third attempt it may simply be concluded that the action is not adequate for the user and it is time to explore other options. In the general case, the number of attempts can be selected differently, or computed on-line depending on the past experiences. However, note that with 3 attempts, 18 failures in the same action would be necessary in order to completely remove one action, which would lead to an unsolvable planning problem. In such case we would consider that the task cannot be completed.

4 Experimental evaluation

The proposed adaptation method has been evaluated through simulated experiments using the same shoe-fitting scenario, and qualitatively assessed in real robot experiments. In them, we define a simulated user with an associated ground-truth user model as well as an inferred user model (which may differ from the real one). The simulated user's behavior consists in accepting only those actions that coincide with the ground-truth model. Otherwise the action fails. Obviously, in a real scenario the action may not fail even if it is not exactly the one the user was expecting. However, for the sake of clarity, we use this simulated user behavior because the adaptation mechanism is better observed. Therefore, a simulator inferred as $[quick,\ informer]$, but whose ground-truth real model is $[slow,\ informer]$, will only allow slow actions that have informed the user beforehand. With this, we can easily test how the behavior of the robot adapts to match the user.

To avoid bias due to the randomness in the plan executions, we have executed each simulated experiment 15 times, and the results shown in this section are the average of all the executions.

In order to assess the effect of the different steps of the method, shown in Figure 2 and explained in the previous section, each experiment has been executed with different combinations of them. Therefore, we start with single methods, the first being to use only the decreasing m-estimate (Martínez et al, 2015) to adapt the probabilities based only on the outcomes of the actions. Similarly, the second one uses the feedback update from Section 3.4.2, adapting the user model by means of decreasing the penalization of the successful rules. Then we combine two steps, applying the initial refinement from Section 3.3 along with the decreasing m-estimate, and the decreasing m-estimate with the feedback update. Finally, we evaluate the full method consisting in the combination of the initial refinement, the decreasing m-estimate and the feedback update.

When the user model is correct, all the actions succeed and the robot finishes the task with the minimum number of actions. To show how the method successfully adapts to the user behavior, we will show the degenerated case in which the simulator's ground-truth user model is the opposite to the initial inferred one. Figure 5 shows the results of a user who behaves as $[slow,\ informer]$, but whose inferred model from the initial questions was $[quick,\ not\ informer]$. The comparison between different method combinations is also displayed. The figure shows the rewards obtained and the length of the plan at each iteration. As it can be seen, all the methods start with a low reward and a long plan, as the system is behaving to suit a different type of user. However, they quickly improve with few iterations, drastically reducing the plan length. Note that this is a degenerated case, and all the actions fail if they are not exactly those of the ground-truth user model. In a normal house set-up, the user would accept some actions even if they are not exactly the ones matching their exact preferences. The methods combining the decreasing m-estimate and the feedback update are the ones that converge faster and reach the optimum number of actions in the plan, as well as being more stable. Given that the inferred model is not the

[3] This would happen if the user changes his/her behavior from the adapted model, always providing a positive feedback score.

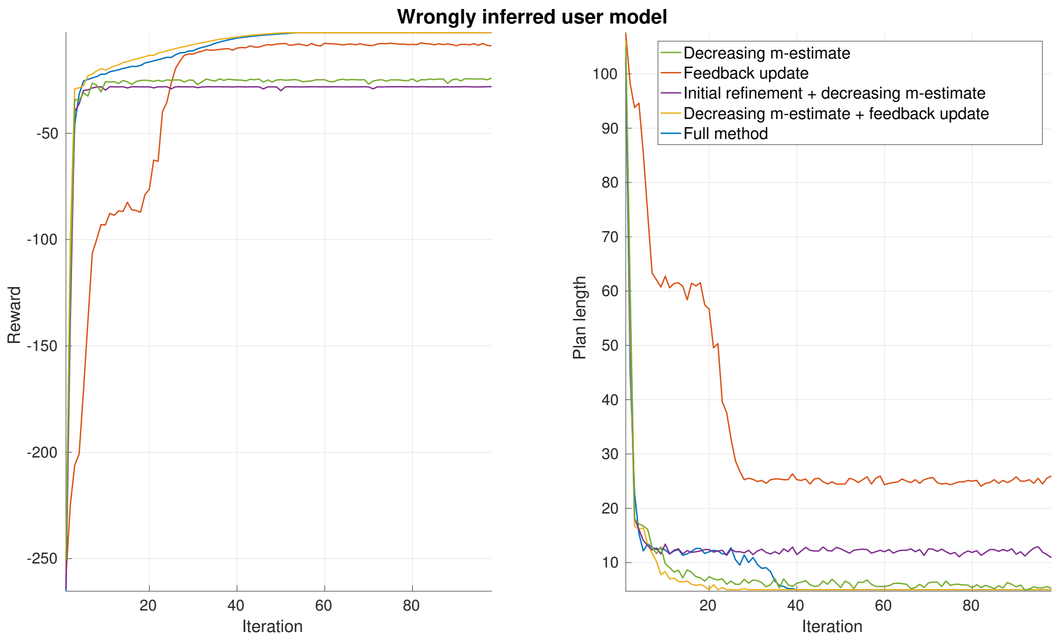

Fig. 5 Evolution of the rewards and plan lengths using different method combinations. The inferred user model is $[quick,\ not\ informer]$, but the simulator behaves as a $[slow,\ informer]$ user.

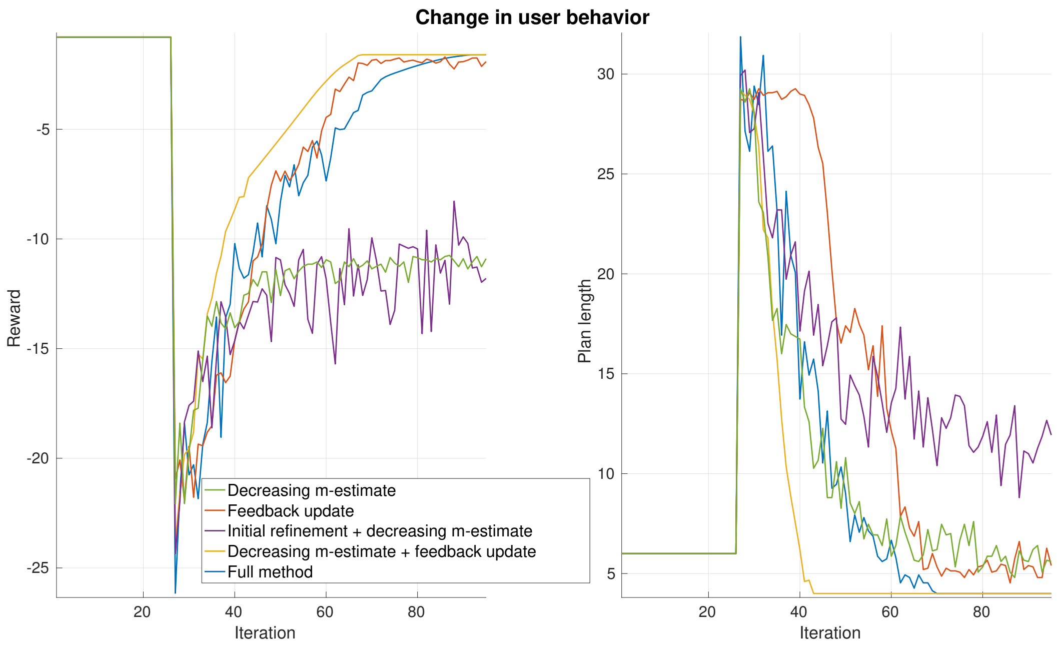

Fig. 6 Evolution of the rewards and plan lengths using different method combinations. The inferred user model and behavior is $[quick,\ informer]$, but in iteration 27 the behavior changes to be $[quick,\ not\ informer]$.

correct one, the method not including the initial refinement performs slightly better, as it can be also seen in the plan length plot. But, if only the feedback update is used, the method gets stuck. Given that the probabilities are not modified, the planner's best option is to go on with the actions that comply with the inferred user model, as they lead to less penalizations. Nevertheless, when the success probabilities of those actions are decreased, the planner has better gain when choosing actions that do not satisfy the inferred model. When only using the decreasing m-estimate (with and without the initial refinement)[4], the method is slower to converge to the preferred solution, and is less stable, oscillating around the preferred solution. Therefore, the combination of the feedback update with the decreasing

[4] Note that the feedback update modifies the reward, thus the reward plots of the methods using it keep improving its reward because of this.

m-estimate are appropriate to converge to the preferred solution, with few iterations and reaching a constant minimum number of actions in the plan. The most similar cases are the full method and the one not using the initial update. In the reward plot, it can be seen how they perform almost equally well, although when checking the plan length, it is clear that the method not using the initial update converges faster than the full method. Again, this is due to the initial method making the algorithm keep the initial inferred model, leading to a slower convergence to the correct model, but making it more robust when the user model has been correctly inferred.

Figure 6 shows an example of adaptation when the user suddenly changes his/her behavior. In this case, the user was behaving as $[quick, informer]$, and his/her model was correctly inferred. Thus, the planner by domain definition is able to find the preferred plan since the beginning. However, around iteration 27, the user starts to behave as $[quick,\ not\ informer]$ as it gets used to the robot, and thus every action fails. When this happens the system needs to readapt, as there is a drop in the reward and the plan length increases. In this case, the methods involving the feedback update and the decreasing m-estimate are the only ones able to cope with the change and converge to the new solution. Note that after the change, the preferred plan is shortened as there are no informative actions.

Although our proposed method shows correct adaptation, the initial refinement may be counterproductive in cases in which the user model was not correctly inferred, slowing the convergence to the real user model. However, in the cases in which the user model was properly inferred, the initial refinement helps to lead the planner towards the preferred goal and avoids the system to wrongly adapt to an erroneous new model due to occasional action failures.

4.1 Experimental feasibility assessment

In this section we show the feasibility of the proposed approach in a real assistive robotics scenario. To do so, we present an experimental setup with a real robot. The setup can be seen in Figure 7. The robot is a 7 degrees of freedom Barrett® WAM Arm and a Kinect™ camera mounted on the ceiling is used for the perception. The software has been developed using the Robot Operating System (ROS) Quigley et al (2009). For the informative actions, the text-to-speech is performed using the `hmi_robin` ROS node[5] from the Institute for Robotics at Johannes Kepler University.

[5] `http://wiki.ros.org/hmi_robin`

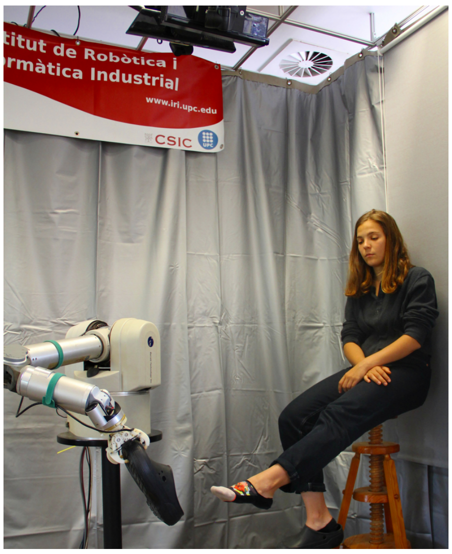

Fig. 7 Experimental setup with the robot, a user and the ceiling camera.

The environment state is obtained by processing the point cloud retrieved from the RGB-D camera. Given the presented setup (see Fig. 7), the point-cloud P is first splitted to remove the ground points, thus obtaining a new point-cloud $P' = \{p \in P | p_y \leq t_g\}$ for a threshold t_g representing the distance from the camera to the ground. The resulting point-cloud P' is further divided in two smaller clouds, $P'_h = \{p \in P' | p_y \geq t_h\}$, corresponding to the human space, and $P'_r = \{p \in P' | p_y < t_h\}$ corresponding to the region where the robot moves, for a threshold t_h representing the distance between the corner of the image and the end of the human space. Then, we perform a simple blob segmentation to obtain the user's leg point-cloud and define the extreme region of the blob as the foottip. The shoe position, as assumed to be grasped by the robot, is defined by the Tool Center Point (TCP) pose of the robot's end effector and is used to filter out the shoe detection.

The user is seated near the robot and lifts the foot as a signal to start the shoe-fitting task. The robot has already the shoe in its end-effector[6] and has been taught the task via kinesthetic reproduction. In the first execution, the user is asked about his/her confidence and comfortability (Section 3) and the initial refinement is performed. Then, the planner is called to obtain the next action to execute based on the perceived state, and the execution of the action is carried out. Each action can be either a robot motion, including the foot perception to accomplish a part of the task, or a verbal interaction, to make request or to inform the user. Once finished, the state is recom-

[6] Shoe grasping is out of the scope of the paper.

puted and the planner is called again, until the shoe has been fit. When the shoe-fitting has been completed, the user's satisfaction is asked and the feedback modification is then performed in order to refine the domain for the next executions. Figure 8 shows the robot executing the three shoe-fitting task actions. A video demonstration can be found at `www.iri.upc.edu/groups/perception/plannedBehaviorAdaptation`.

As seen in the video, the proposed method shows a robust behavior of the robot in which the planner is able to adapt to the changes in user preferences and to unexpected situations. We show how the robot asks the user to put the foot forward when it is not in sight, and how it speaks only when needed, e.g. when the informative behavior is not specified in the user state. These decisions are made by the planner. Moreover, the video also shows how the robot changes its behavior in the short term in order to fulfill the task, by exploring different speed alternatives, when a failure occurs in the current situation. However, the video cannot show the long-term adaptation of the rewards. Moreover, the robot moves slowly to ensure safety as well as due to nonoptimized computations (vision, trajectory generation and planning). More robust and safe movements are planned as future work to come up with a more efficient and less tiring fitting scenario.

5 Conclusions

In this paper, we have defined a method to guide a planner to choose the preferred actions by the user. The user model is included in the planning domain as predicates, and the actions' associated costs depend on them, the most costly actions being those that do not satisfy the user model. Moreover, we use an stochastic planner with NID rules that contemplate the possibility of different action outcomes and failures. The initial user model is inferred by asking two simple questions to the user, related to his/her confidence and comfortability. A Fuzzy Inference System (FIS) is then used to translate the answers to planning predicates.

In order to make the planner adapt to user behavior change and to cope with wrongly inferred user models, each rule's probabilities and costs are updated. First, an initial refinement is performed to favor the inferred user model. Then, after each task completion, the satisfaction of the user is used to refine each rule cost, and the outcome of each action is used to refine the success' probabilities. This defines a separation between the user model and the action outcomes, as the user delight should not be measured only by the success of the actions, which may fail due to events unrelated to the users' preferences.

Moreover, the system is able to plan with task related actions as well as with interaction actions, asking the user to move when needed and informing him/her regarding the next action when this increases the success rate of the action.

We show how the system is able to adapt to user behavior changes, as well as how the use of feedback to update the action costs with the decreasing m-estimate produces a more stable behavior and faster convergence to the preferred solution.

Although the system keeps adapting to new changes, long-term adaptation should be analysed more thoroughly, as well as the inclusion of more actions and preferences, with the possibility of automatically learning the actions along with the preferences, which is out of the scope of this paper and therefore is left as future work.

Acknowledgements The authors thank Clea Parcerisas, Sergi Foix and Adrià Colomé for their help in the realization of the experiments, pictures and video.

References

Alili S, Warnier M, Ali M, Alami R (2009) Planning and plan-execution for human-robot cooperative task achievement. In: 19th International Conference on Automated Planning and Scheduling, pp 19–23

Canal G, Alenyà G, Torras C (2016) Personalization framework for adaptive robotic feeding assistance. In: 8th International Conference on Social Robotics (ICSR), pp 22–31

Canal G, Alenyà G, Torras C (2017) A taxonomy of preferences for physically assistive robots. In: IEEE Intl. Symp. on Robot and Human Interactive Communication (RO-MAN), pp 292–297

Castellano G, Leite I, Paiva A (2016) Detecting perceived quality of interaction with a robot using contextual features. Autonomous Robots pp 1–17

Chance G, Camilleri A, Winstone B, Caleb-Solly P, Dogramadzi S (2016) An assistive robot to support dressing–strategies for planning and error handling. In: Proceedings of the 6th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics, IEEE

Chen TL, Ciocarlie M, Cousins S, Grice PM, Hawkins K, Hsiao K, Kemp CC, King CH, Lazewatsky DA, Leeper AE, Nguyen H, Paepcke A, Pantofaru C, Smart WD, Takayama L (2013) Robots for humanity: using assistive robotics to empower people with disabilities. IEEE Robotics Automation Magazine 20(1):30–39

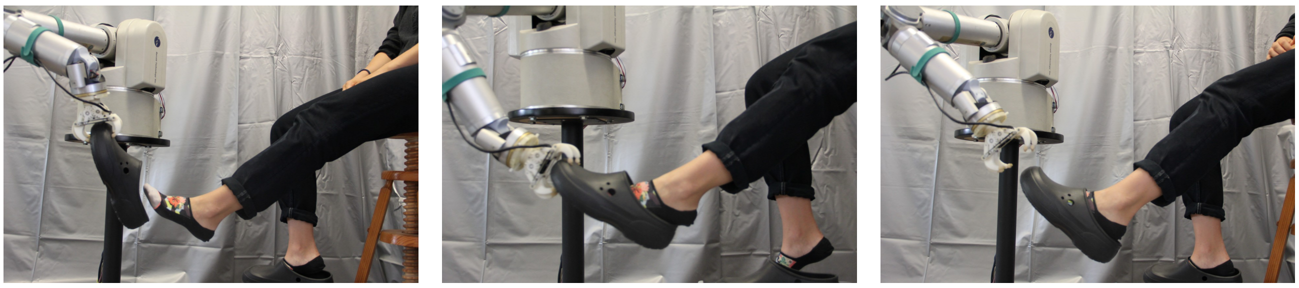
(a) *Approach* action, to get closer to the foot. (b) *Insert shoe* action, using an elaborated wrist rotation (see supplementary video). (c) *Release* action, to move away.

Fig. 8 Example of the execution of the three movement actions. The current location of the foot is obtained with a ceiling-mounted RGB-D sensor (see Fig. 7).

Erol K, Hendler J, Nau DS (1994) HTN planning: Complexity and expressivity. In: AAAI, vol 94, pp 1123–1128

Fiore M, Clodic A, Alami R (2016) On Planning and Task Achievement Modalities for Human-Robot Collaboration. In: Springer Tracts in Advanced Robotics, vol 109, pp 293–306

Gao Y, Chang HJ, Demiris Y (2015) User modelling for personalised dressing assistance by humanoid robots. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 1840–1845

Gao Y, Chang HJ, Demiris Y (2016) Iterative path optimisation for personalised dressing assistance using vision and force information. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4398–4403

Griffith S, Subramanian K, Scholz J, Isbell C, Thomaz AL (2013) Policy shaping: Integrating human feedback with reinforcement learning. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in Neural Information Processing Systems 26, Curran Associates, Inc., pp 2625–2633

Heerink M, Krose B, Evers V, Wielinga B (2009) Measuring acceptance of an assistive social robot: a suggested toolkit. In: RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication, pp 528–533

Klee SD, Ferreira BQ, Silva R, Costeira JP, Melo FS, Veloso M (2015) Personalized assistance for dressing users. In: Social Robotics: 7th International Conference, ICSR 2015, Springer, pp 359–369

Knox WB, Stone P (2009) Interactively shaping agents via human reinforcement: The tamer framework. In: The Fifth International Conference on Knowledge Capture

Kühnlenz B, Sosnowski S, Buß M, Wollherr D, Kühnlenz K, Buss M (2013) Increasing Helpfulness towards a Robot by Emotional Adaption to the User. International Journal of Social Robotics 5(4):457–476

Lallement R, De Silva L, Alami R (2014) Hatp: An htn planner for robotics. In: 2nd ICAPS Workshop on Planning and Robotics

Mamdani E, Assilian S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies 7(1):1 – 13, DOI 10.1016/S0020-7373(75)80002-2

Martínez D, Alenyà G, Torras C (2015) Planning robot manipulation to clean planar surfaces. Engineering Applications of Artificial Intelligence 39:23 – 32

McLafferty E, Farley A (2008) Assessing pain in patients. Nursing Standard 22(25):42–46

Muir BM (1987) Trust between humans and machines, and the design of decision aids. International Journal of Man-Machine Studies 27(5):527 – 539

Pasula HM, Zettlemoyer LS, Kaelbling LP (2007) Learning symbolic models of stochastic domains. Journal of Artificial Intelligence Research 29:309–352

Pignat E, Calinon S (2017) Learning adaptive dressing assistance from human demonstration. Robotics and Autonomous Systems 93:61 – 75

Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol 3, p 5

Rada-Vilela J (2014) fuzzylite: a fuzzy logic control library. URL `http://www.fuzzylite.com`, accessed 26 Oct 2016

Robinson H, MacDonald B, Broadbent E (2014) The role of healthcare robots for older people at home: A review. International Journal of Social Robotics 6(4):575–591

Rozo L, Calinon S, Caldwell DG, Jimenez P, Torras C (2016) Learning physical collaborative robot behaviors from human demonstrations. IEEE Transactions on Robotics 32(3):513–527 de Silva L, Lallement R, Alami R (2015) The hatp hierarchical planner: Formalisation and an initial study of its usability and practicality. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 6465–6472

Tamei T, Matsubara T, Rai A, Shibata T (2011) Reinforcement learning of clothing assistance with a dual-arm robot. In: 2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids), IEEE, pp 733–738

Thomaz AL, Breazeal C (2006) Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI Press, AAAI'06, pp 1000–1005

Vahrenkamp N, Wächter M, Kröhnert M, Welke K, Asfour T (2015) The robot software framework armarx. Information Technology 57(2):99–111

Yamazaki K, Oya R, Nagahama K, Okada K, Inaba M (2014) Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions. In: 2014 IEEE/SICE International Symposium on System Integration, pp 564–570