UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Adapting robot behavior to user preferences in assistive scenarios

Gerard Canal Camprodon

Universitat Politècnica de Catalunya

Doctoral Programme

Automatic Control, Robotics and Computer Vision

Ph.D. Thesis

Adapting robot behavior to user preferences in assistive scenarios

Gerard Canal Camprodon

Advisors:
Guillem Alenyà and Carme Torras

Barcelona, January 2020

Adapting robot behavior to user preferences in assistive scenarios

A thesis submitted to the Universitat Politècnica de Catalunya
to obtain the degree of Doctor of Philosophy

Doctoral programme:
Automatic Control, Robotics and Computer Vision

This thesis was completed at:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Thesis advisors:
Guillem Alenyà and Carme Torras

Dissertation Committee:
Cecilio Angulo (Universitat Politècnica de Catalunya)
Yiannis Demiris (Imperial College London)
Dongheui Lee (Technische Universität München)

To my grandma, l’àvia Carme

“Emotions wouldn’t be much of an asset for a bathtub-cleaning robot. But if the robot is reminding me to take my meds or helping me put the groceries away, I will want a little more personal interaction, with the sort of feedback that lets me know not just whether it’s understanding me but how it’s understanding me.”

– Rodney Brooks

ADAPTING ROBOT BEHAVIOR TO USER PREFERENCES
IN ASSISTIVE SCENARIOS

Gerard Canal Camprodon

Abstract

Robotic assistants have inspired numerous books and science fiction movies. In the real world, these kinds of devices are a growing need amongst the elderly, who will continue requiring more assistance. While life expectancy is increasing, life quality is not necessarily doing so. Thus, we may find ourselves and our loved ones being dependent and needing another person to perform the most basic tasks, which has a strong psychological impact. Accordingly, assistive robots may be the definitive tool to give more quality of life by empowering dependent people and extending their independent living.

Assisting users to perform daily activities requires adapting to them and their needs, as they might not be able to adapt to the robot. This thesis tackles adaptation and personalization issues through user preferences. We focus on physical tasks that involve close contact, as these present interesting challenges, and are of great importance for the user. Therefore, three tasks are mainly used throughout the thesis: assistive feeding, shoe fitting, and jacket dressing. We first describe a framework for robot behavior adaptation that illustrates how robots should be personalized for and by end-users or their assistants. Using this framework, non-technical users determine how the robot should behave. Experimental evaluation on the framework demonstrates its usefulness. Therefore, we build the behavior adaptation upon it. Then, we define the concept of preference for assistive robotics scenarios and establish a taxonomy, which includes hierarchies and groups of preferences, grounding definitions and concepts. We then show how the preferences in the taxonomy are used with AI planning systems to adapt the robot behavior to the preferences of the user obtained from simple questions. Our algorithms allow for long-term adaptations as well as to cope with misinferred user models, as demonstrated in the experimental evaluation. We further integrate the methods with low-level motion primitives that provide a more robust adaptation and behavior while lowering the number of needed actions and demonstrations. Moreover, we perform a deeper analysis of planning and preferences with the introduction of new algorithms to provide preference suggestions in planning domains. We show how the suggestions maximize the plan reward, improving the chances of task success. The thesis then concludes with a user study that evaluates the use of the preferences in the three real assistive robotics scenarios. The experiments show a clear understanding of the preferences by users, who were able to assess the impact of their preferences on the robot's behavior.

In summary, we provide tools and algorithms to design the robotic assistants of the future. Assistants that should be able to adapt to the assisted user needs and preferences, just as human assistants do nowadays.

Keywords: Robot behavior adaptation, Physically Assistive Robots, Robot personalization, Planning for robot adaptation, Robotics.

Resum

Els assistents robòtics han inspirat nombrosos llibres i pel·lícules de ciència-ficció al llarg de la història. Però tornant al món real, aquest tipus de dispositius s'estan tornant una necessitat per a una societat que envelleix a un ritme ràpid i que, per tant, requerirà més i més assistència. Mentre l'esperança de vida augmenta, la qualitat de vida no necessàriament ho fa. Per tant, ens podem trobar a nosaltres mateixos i als nostres estimats en una situació de dependència, necessitant una altra persona per poder fer les tasques més bàsiques, cosa que té un gran impacte psicològic. En conseqüència, els robots assistencials poden ser l'eina definitiva per proporcionar una millor qualitat de vida empoderant els usuaris i allargant la seva capacitat de viure independentment.

L'assistència a persones per realitzar tasques diàries requereix adaptar-se a elles i les seves necessitats, donat que aquests usuaris no poden adaptar-se al robot. En aquesta tesi, abordem el problema de l'adaptació i la personalització d'un robot mitjançant preferències de l'usuari. Ens centrem en tasques físiques, que involucren contacte amb la persona, per les seves dificultats i importància per a l'usuari. Per aquest motiu, la tesi utilitzarà principalment tres tasques com a exemple: donar menjar, posar una sabata i vestir una jaqueta. Comencem definint un marc (*framework*) per a la personalització del comportament del robot que defineix com s'han de personalitzar els robots per usuaris i pels seus assistents. Amb aquest marc, usuaris sense coneixements tècnics són capaços de definir com s'ha de comportar el robot. Una avaluació experimental del *framework* en demostra la seva utilitat. Per tant, l'adaptació del comportament presentada en la tesi es construeix sobre aquest *framework*. Posteriorment definim el concepte de preferència per a robots assistencials i establim una taxonomia que inclou jerarquies i grups de preferències, els quals fonamenten les definicions i conceptes. Després mostrem com les preferències de la taxonomia s'utilitzen amb sistemes planificadors amb IA per adaptar el comportament del robot a les preferències de l'usuari, que s'obtenen mitjançant preguntes simples. Els nostres algorismes permeten l'adaptació a llarg termini, així com fer front a models d'usuari mal inferits, tal com es mostra en l'avaluació experimental. Aquests mètodes són integrats amb primitives a baix nivell que proporcionen una adaptació i comportament més robusts a la mateixa vegada que disminueixen el nombre d'accions i demostracions necessàries. També fem una anàlisi més profunda de l'ús de les preferències amb planificadors amb la introducció de nous algorismes per fer suggeriments de preferències en dominis de planificació. Aquí mostrem com els suggeriments maximitzen la recompensa del pla, millorant les probabilitats d'èxit de la tasca. La tesi conclou amb un estudi amb usuaris que avalua l'ús de les preferències en les tres tasques assistencials. Els experiments demostren un clar enteniment de les preferències per part dels usuaris, que van ser capaços de discernir quan les seves preferències eren utilitzades.

En resum, proporcionem eines i algorismes per dissenyar els assistents robòtics del futur. Uns assistents que haurien de ser capaços d'adaptar-se a les preferències i necessitats de l'usuari que assisteixen, tal com els assistents humans fan avui en dia.

Acknowledgements

I would like to start by expressing my heartfelt gratitude and appreciation to my advisors, Dr. Guillem Alenyà and Prof. Carme Torras. For granting me the opportunity to work with them in this thesis about the most exciting and breakthrough topics. Also for their constant support, patience, understanding, dedication, and knowledge. I believe PhD. advisors are key in the success of any thesis, and mines have been the best advisors I could have ever wished for.

They also gave me the chance of visiting King's College London for three months. There, I was supervised by Dr. Michael Cashmore and Dr. Daniele Magazzeni, to whom I express my appreciation. I felt at home even when I was far, being one of the team. They also contributed to my learning and helped me grow as a researcher, for which I am very thankful.

Likewise, I thank my parents, Maria Àngels and Josep Maria, for all their love and support. For everything they've done over all these years to make me reach any goal I wanted. But also the rest of the family, to my grandmas Carme and Conxita, to all my cousins, aunts and uncles, and to those who are not around but are still present.

But apart from academic advisors and family, I've been very fortunate to be surrounded by lots of valuable people who supported me. Starting with all the people at IRI, especially my colleagues from D19. We spent many hours there (a lot!) with many different feelings. Albert, Aleks, Antonio, Víctor, Alejandros, and the passing ones. You've been the best rubber ducks (for code-debugging) I could have had! Thanks for listening to me and my complaints. I also owe a huge thanks to Sergi and Adrià, without your help and patience this would have been extremely difficult, if not impossible. Further, acknowledge the administrative staff who were essential to sort out the lots of academic bureaucracy. My most special thanks are for Víctor Vílchez. He is not only one of the best friends anyone can have but has also been a pillar for me during the thesis. I don't think I would have made it without his help in preparing documentation, asking for grants, proofreading my texts and papers, listening to me for many hours and feeding me breadsticks when I was stressed. I can't be thankful enough for being there in the past, the present and probably the future. Besides, I thank Laia for being a really good friend, Maria for being the first awesome student I supervised, Kiko and the indoor lunch crew for the great moments, and also Anaís Garrell for all her help and support (plus the Australian adventures!).

Furthermore, I want to thank all the people I met in London in late 2018. Starting with my friends at KCL: Diego, Parisa, Senka, Xavi, Yani, and the others, and to the people from the nice Catalan community in London. You all made it very easy to stay abroad, with many experiences I will never forget. Back to Barcelona, I want to thank my friend Daniel Pérez for always having a good advice, and to Cristina García for helping me keep my sanity, as well as to the many PhD-related Twitter accounts who always gave me a smile with their realistic irony and online support (@CientificoenEsp, @PHDcomics, @AcademicsSay, @ithinkwellHugh...).

Doing a PhD. is quite time-consuming, but I found the best escape in dancing. Thus, I thank with all my heart my Salsa and Bachata buddies, whom I met by pure chance but became the greatest friends. Especially to Diana, but also to Mel, Santi, Cèlia, Elena, Daria, and many others. Thanks for all the dances, happiness, and the healthy and much-needed disconnection.

Last, but not least, I would like to thank my good and lovely friends Roger and Cristina for everything they have done for many years, and for many more to come. In summary, to all of you with whom I crossed paths over the years, thank you too for leading me to this and helping me learn, grow and become a better person every day.

A totes i a tots: infinites gràcies!

This work has been partially supported by the following doctoral grants:

2016FI_B 00660 FI-DGR (Ministry of Economy and Knowledge of the Government of Catalonia).

FPU15/00504 FPU: Formación de Profesorado Universitario (Spanish Ministry of Education, Culture, and Sport).

EST17/00371 Estancias Breves (FPU): Mobility grant (Spanish Ministry of Education, Culture, and Sport).

This thesis has also been partially supported by the research projects:

PCIN-2015-147 I-DRESS: Assistive interactive robotic system for support in dressing (ERA-Net CHIST-ERA project).

201350E102 MANIPlus: Manipulación robotizada de objetos deformables (Consejo Superior de Investigaciones Científicas - CSIC).

H2020-ICT-2016-1-731761 IMAGINE: Robots understanding their actions by imagining their effects (European Commission).

ERC-2016-ADG-741930 CLOTHILDE: Cloth manipulation learning from demonstration (European Research Council).

Contents

Figures

Tables

1 Introduction

In the current aging world, the lack of care professionals such as nurses and caregivers [1, 2] will deem assistive devices necessary for many individuals to live a dignified life. Therefore, assistive robots will be key in the next health care revolution. However, research on assistive robotics poses several challenges, both technical and ethical, given the closeness of the robots to the human users and the many safety concerns this can raise [3].

People with reduced mobility tend to find themselves needing the help of another in order to do the most basic tasks. Hence, performing Activities of Daily Living (ADLs) such as eating, dressing, grooming or cleaning up can become very challenging. Intelligent robotic systems have proven useful in these situations by performing the helping task and, so, removing the constraint of constant attention from another person.

Nowadays, these assistive devices and technologies are getting common and some commercial products are starting to be available. However, the deployment of robots able to physically interact with a person in an assistive manner is still challenging. Apart from aspects such as design and control, an assistive robot should be able to *adapt* to the specific user *needs* and *preferences* in order to effectively assist a human user. And, rather than performing a generic action suitable for anyone, forcing the user to adapt to the robot, it is the robot who should modify its behavior taking into account the user and the situation; just as a human carer would do. This *empowering* of disabled people is crucial [4], and can be attained by providing more autonomy, intimacy and better quality of life. Nevertheless, this does not imply the substitution of the caregiver, as personal contact is also very important. Contrarily, we think the robots should rely on the caregiver to personalize their behavior to the disabled person's preferences and needs. Thus, this robot behavior adaptation comes through the definition of user preferences for the task such that the robot can act in the user's desired way.

While rehabilitation robotics has received more attention from the research community in the past, the field is going towards the development of autonomous assistive systems to be employed by non-technical users. The Assistive Robotics field is an area of growing interest where robots are used as a *tool* to help caregivers and nurses to improve the assistance. Socially Assistive Robots (SAR) [5] are robots that provide assistance by means of social interaction to guide processes of rehabilitation, learning and convalescence. On the other hand, Physically Assistive Robots (PAR) are the ones providing assistance by means of physical interaction, helping users to perform activities such as eating, dressing and grooming [6, 7].

However, the potential users of these systems, caregivers or disabled and older adults 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.

This thesis analyzes this adaptation of assistive robots to user preferences. Given that health care assistance is often related to close-contact interactions, the thesis will focus on the case of the Physically Assistive Robot. Firstly, an Assistive Robot Personalization methodology will be presented, to then define the possible preferences to be applied in such kind of robotic tasks. We will then propose some algorithms for robot behavior adaptation through symbolic task planning techniques, combining them with low-level adaptive movements. All the proposed methods have been evaluated experimentally involving real robots, and a final user study will conclude the thesis.

1.1 Motivation

The main motivation of this Ph.D. thesis is to provide methodologies to adapt the behavior of an assistive robot to the user it is helping. For example, there is expected a 34% increase in the number of stroke events in Europe [8], which may lead to many people in direct need of assistance to live. Such loss of independence to perform ADLs [9] results in a great impact on the patient's psychological wellbeing, with feelings of burden and guilt to those close to them [10, 11]. Thus, robots capable of physically assisting users in order to enable them to perform such tasks without the need of another human may have a considerable impact on the modern society [4]. Still, to successfully empower the users and ease the employment of robotic systems, such mechanical assistants should be able to cope with user disabilities and preferences, just as human caregivers do.

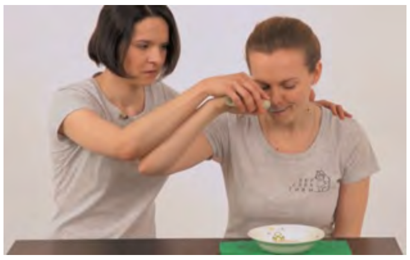
There are hundreds of ways of assisting a person to perform Activities of Daily Living (ADLs), along with other subtleties that human caregivers take into account. Caregivers, knowing the users and interacting with them, adapt the assistance to suit every individual's needs and preferences effectively making the task more pleasant for the patients. A clear example of this can be seen in Figure 1.1. As it can be observed in the pictures, extracted from Certified Nursing

(a) Feeding a laying person (Arizona Medical Training Institute[1])

(b) Feeding a sitting person (EK Medical Learning Center[2])

(c) Feeding reduced mobility (BR Nursing School[3])

(d) Feeding laterally (American Red Cross[4])


(e) Assisting handover self-feed (Perspektivy[5])

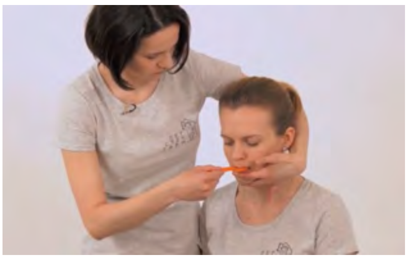

(f) Assistive feeding when mouth can't be closed (Perspektivy[5])

Figure 1.1: Examples of different feeding strategies from CNA training videos.

Assistant (CNA) training videos, there are many different ways of doing a simple task such as feeding a person. The strategy will depend on the condition and capabilities of the users, but their preferences will also determine the strategy and play an important role in the success of the task. Accordingly, a 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 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.

While we believe these assistive robots cannot supply the same as human contact does, we look at them as smart appliances that must act in a highly autonomous manner. But, as no person is identical and each of us has our own needs, what may work for some user may be disliking for another one. Thus, an increase on task variability implies the need for robot adaptation. However, the fact of considering physical interactions between the human and the robot introduces a new set of requirements in terms of robot action execution, and thus, also in terms of user preferences. Therefore, we strongly believe this robot personalization is essential in order to prevent rejection and foster the use of such devices. Ultimately, we hope the approaches proposed in this thesis will help users with reduced autonomy to empower themselves, having the robot as a tool for their own independence.

[1]Extracted from: youtu.be/aCIKmu4jIWg
[2]Extracted from: youtu.be/E1vd1O-LgWI
[3]Extracted from: youtu.be/j_D8g2ngVWs
[4]Extracted from: youtu.be/XvMX76BvAoM
[5]Extracted from: youtu.be/m4o5u7x4qys

1.2 Contributions

The main goal of this thesis is to advance in the field of autonomous robot personalization and behavior adaptation, making emphasis on physically assistive scenarios. We intend to develop methods to autonomously modify the robot's performance to match the expectations, preferences, and needs of the assisted user. To achieve it, this thesis has combined methods of robotics and AI to adapt the behavior of the robot to user preferences. Below the list of contributions:

1. We have proposed a framework for robot personalization along with some trajectory personalization strategies [6]. The framework defines the personalization steps and its relations, and it is intended to allow non-technical users such as nurses and caregivers to re-adapt the robot behavior in terms of robot positions and trajectories. We demonstrate its use in a feeding scenario, but its applicability is shown over all the thesis.

2. A methodological characterization of preferences for assistive robotics tasks. For this, we define a taxonomy [12] with different kinds of preferences and their uses. We show how this taxonomy is wide enough to consider user limitations, and how those can be expressed as preferences over the task.

3. We have proposed an algorithm for robot behavior adaptation using probabilistic symbolic task planning [13]. The method defines a user model by means of simple user answers to task-unrelated questions that are fed to a Fuzzy Inference System to be translated into specific preferences. Then the user feedback and the task performance are used to update the costs and probabilities of the task model, resulting in a long-term adaptation of the robot behavior to the user, able to cope with wrongly inferred models.

4. We have further improved the adaptation methods by adding low-level controllers [14] able to adapt online to the user movements. We demonstrate how joining such motion primitives with task planners is beneficial for both, as less physical demonstrations are needed to teach the task (the planner manages the errors at the logical level), while less symbolic actions need to be defined (as the low-level control can cope with small movements).

5. We have extended the ROSPlan framework for symbolic task planning in robotics [15]. Our extension makes the framework able to handle probabilistic planning solvers and domains. This allows the users to use the PPDDL and RDDL languages for probabilistic planning. ROSPlan has a considerable user base, so we believe that this is a valuable

contribution for the AI planning and robotics communities, together with the previous contribution.

6. We developed a new algorithm for making suggestions in planning [16]. We define what we call *suggestible* predicates, which we ground to preferences in the thesis. The method allows suggesting preference values that would improve the performance of the task. The algorithm uses the already known preferences, and even allows for suggestions to change some preference values.

7. An analysis towards adding more safety in Physically Assistive Robot [17], where we considered unavoidable impacts. Impact forces between the robot and the user were analyzed to assess whether they were harmful, as well as proposing some safety strategies based on compliant robot controllers and force limitation by monitoring.

8. A Human-Robot Interaction study on the use of preferences for Physically Assistive Robotics tasks [18]. In it, we evaluate whether users can discern when their preferences are used in a real assistive robotics task without any prior knowledge of the task or the robot behavior. This contributes to demonstrating that preferences are a successful manner for guiding the robot action selection.

1.3 Considered scenarios

This thesis mainly considers the addition of preferences to guide the behavior of Physically Assistive Robots (PAR) while providing help to users in need. Although many different Activities of Daily Living (ADLs) could be considered, we have focused on three main scenarios, whose challenges and particularities will be analyzed in this section.

Two of the considered scenarios have been taken from the research projects I-DRESS[6] and CLOTHILDE[7]. These are shoe fitting and assisted dressing of a coat or jacket. A third scenario we have considered is the one of assistive feeding, which is a very basic need of the target users. This scenario is used in the HuMoUR project[8].

Although many other scenarios could also be considered, we believe the selected ones are crucial for the needs of people and are realistic enough to avoid any potential harm to human users. These tasks are complex to solve and, to concentrate on the use of preferences to adapt the robot behavior, some assumptions will be made to simplify the execution. Other scenarios,

[6]I-DRESS: Assistive interactive robotic system for support in dressing (i-dress-project.eu)

[7]CLOTHILDE: Cloth manipulation learning from demonstration (clothilde.iri.upc.edu)

[8]HuMoUR: Markerless 3D human motion understanding for adaptive robot behavior (iri.upc.edu/project/show/193)

albeit still very important, may pose serious issues in terms of safety. An example of this kind of task would be shaving, which involves the use of sharp tools against the user's skin. Combing is another example of a potentially harmful task, where the comb could tear the user's scalp or be uncomfortable.

1.3.1 Assistive feeding

One of the most basic activities any human or animal needs to do is eating, which is at the base of Maslow's hierarchy of needs [19].The inability to eat or consume nutrients would result in certain death. Therefore, feeding assistance is essential for those unable to do it by themselves.

Even though humans can assist others to eat without many concerns, feeding is a repetitive task that can last for some time. Moreover, eating usually has other social implications for humans, being a social act of sharing, speaking and enjoying food together as it happens in different cultures. Thus, when assisting to feed a user, it stops being a shared social act to become a task of assistance, exposing such dependence on another person. This tends to impact the psychological wellbeing of the assisted individuals, who can sometimes feel that they bother their caregivers and may feel useless, which can lead to depressive symptoms [11].

Accordingly, feeding assistance may have a great impact on the lives of dependent people, providing a tool for empowering them by making them able to feed themselves autonomously without the need of a second person. A user being fed by a robot is depicted in Figure 1.2.

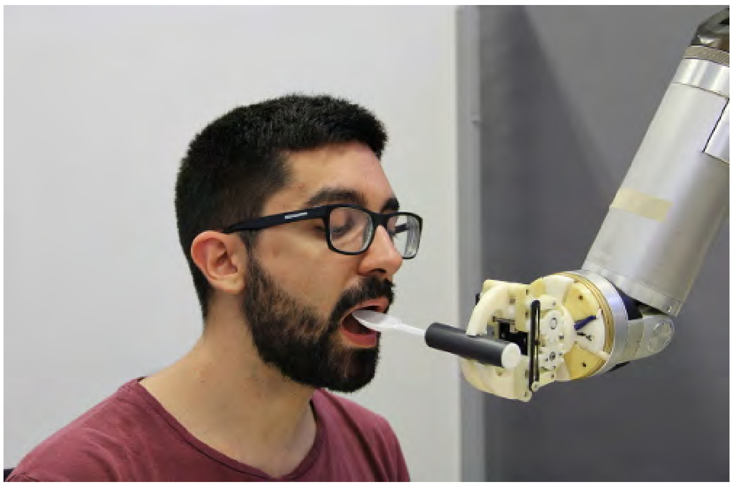


Figure 1.2: An example of robotic assistive feeding.

Challenges

Assistive feeding with a robot involves many elements that could result in potential dangers for the user. Therefore, the following challenges can be found:

- **Cutlery insertion:** Feeding requires to insert the food, which is usually attached to some piece of cutlery, inside of the user's mouth.
- **Potentially harmful tools:** The act of eating requires many sharp tools such as knives or forks, which could tear the skin of the user and cause serious injuries. And, as they need to be inserted in the user's mouth, this makes it easier to have potential impacts with the tool.
- **Sudden movements:** Due to the items above, a sudden movement, be it voluntary or spasmodic, may hinder the task.
- **Precision:** The food must be inserted in the user's mouth. Failure to do so in a precise manner may cause food spilling or harm the user.
- **Forces:** Potential unavoidable impacts should exert the least possible forces while keeping the required precision. Similarly, forces should not be applied when the feeding utensil is retained by the user.
- **Food perception:** Needed to detect the food to get and how much food is left.
- **Food manipulation:** Getting food from a plate requires some dexterity, as well as semantic knowledge on cutlery and how to use it. It also requires strategies on how to group and scoop different kinds of food, and strategies to insert the food in the user's mouth.

Considerations and assumptions

Given the extreme complexity of the assistive feeding tasks, some simplifications will be considered in order to research the application of preferences. Firstly, we will usually limit the used cutlery to a single spoon[9]. This choice is motivated by the spoon being the least harmful tool, although it requires a more complex feeding strategy as spoons are usually scooped inside the mouth to release the food. The use of a spoon also limits the kind of food to be used, which will be yogurt and cream-like foods. This kind of food makes for an easier strategy for spoon filling but is more prone to have food spilled so orientations must be taken into account.

[9]Some of the experiments will use a fork to show utensil adaptation, but otherwise a spoon will be the assumed setup over all the thesis.

Moreover, we will not consider the possibility of having sudden or spasmodic movements, so we will assume a fixed head position. This option would require the ability to predict such movements and reacting accordingly. Although interesting, it falls out of the scope of this thesis.

Still, these considerations will not hinder the addition of preferences to the task and adapt the behavior of the robot to the specific user needs in such an important task as feeding oneself.

1.3.2 Shoe fitting

Some older adults and people with reduced mobility may find very difficult to put their shoes on by themselves. This happens because self-fitting a shoe requires some mobility as well as flexibility in order to reach the foot with the hands.

Moreover, shoe fitting is essential to provide movement autonomy for people who can walk by themselves (with or without the help of other support tools), but need to wear shoes to do so. Therefore, the inability to self-fit a shoe may provoke the user to wait for an assistant to help them perform the task before they can walk and move autonomously. However, shoe fitting is not only useful for people able to walk but also to protect the feet against cold and other potential harms.

Accordingly, helping a dependent user to fit a shoe without the need of an external person would greatly improve their autonomy, and an autonomous robot could be the perfect tool to do so. An example of assistive shoe fitting by a robot is shown in Figure 1.3.

Challenges

Shoe fitting is a challenging task *per se*. The main challenges being:

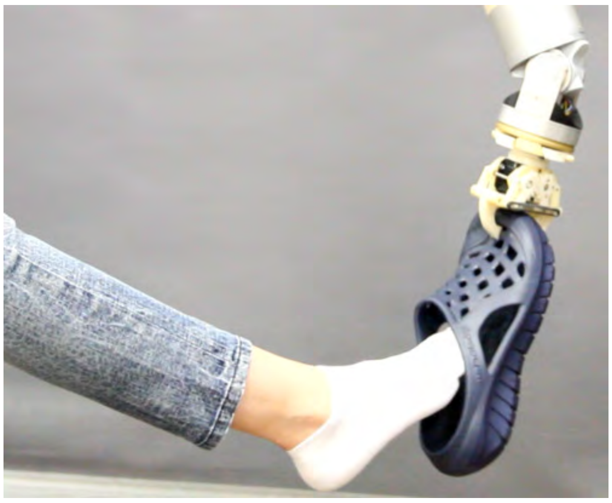


Figure 1.3: An example of robotic shoe fitting.

- **Shoe type:** A key factor for shoe fitting is the shoe type. Shoes may be of many different kinds and shapes, with different levels of flexibility and different fitting strategies.
- **Precision:** An imprecise fitting may leave some toes outside of the shoe, resulting in a harmful and uncomfortable situation.
- **Forces:** Shoe fitting requires some amount of force to successfully push the shoe until the end and provide a complete fit. However, too much force could be dangerous for the user or make him/her feel uncomfortable. It could also provoke an unwanted movement in the user's foot.
- **User movement:** The user may move the foot while the robot is performing the task. This means that the robot movement may fail to meet the foot. Even worse, the robot may collide with the user due to these unexpected movements.

Considerations and assumptions

Given the presented challenges, some considerations need to be made in order to safely tackle the scenario. First of all, we have fixed the shoe type. We have selected to fit Crocs^{TM}-like shoes, which are easy to fit. They are a kind of slippers, which do not have the problem of needing to fit the heel of the shoe, easing the task and the robot movement. In this manner, the task can be performed with a single-arm robot. Moreover, this kind of shoes are lace-less. Tying the shoelaces would require a level of dexterity and precision that poses a big research topic, and is out of the scope of this thesis.

This shoe type also reduces the needed amount of force to fit the shoe, as the heel does not have to be fixed. This also affects the precision, as the design of this shoe allows for easier insertion.

This scenario is a crucial one, and it is a good setting to develop different preferences to adapt the robot behavior. With the considerations defined above along with the use of compliant controllers, the task can be performed in a safe manner.

1.3.3 Assisted jacket dressing

Dressing capacity is another crucial need of any human being. Dressing protects us from the weather's harshness and also used for modesty, in order to comply with society's code of decency. Therefore, a need for assistance for dressing may imply not only a loss of independence when users need to wait for help to get dressed and leave the house but also as a loss of privacy, as they may need to get dressed and undressed in front of other people.

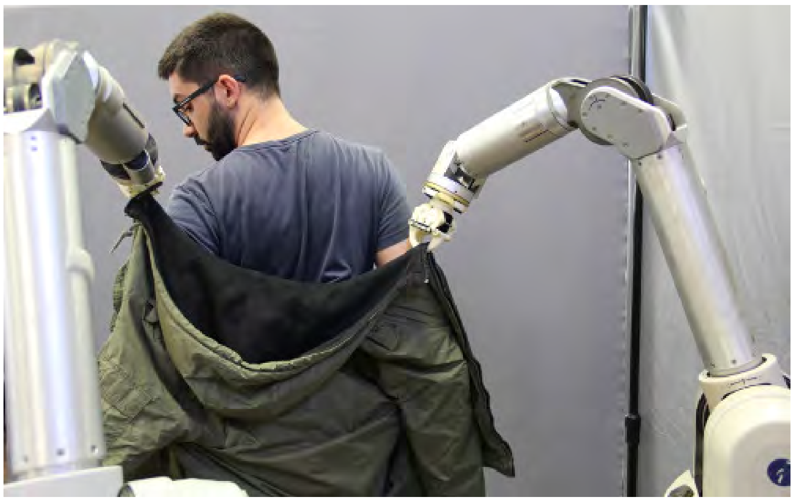


Figure 1.4: An example of robotic jacket dressing.

People with upper-limb mobility issues tend to find difficult to dress by themselves. Dressing, in general, requires at least some arm mobility in order to handle the clothes and fit them. But in the specific case of dressing the upper-body, this is even more clear as arms need to be moved in some specific poses to successfully fit a particular garment.

There are two main kinds of upper-body garments. One consists of T-shirt like clothes, which only have openings for the head and the arms. This kind of clothes is more limited in the ways they can be fit, as there are not many possibilities. The others are open shirts or jackets that close at the front with buttons or zippers. In this second case, there are more options to fit the garment. One could start with one arm or the other, or both together. The needed movements and abilities of the user are also different depending on how has the task started.

Therefore, we will focus on the case of jackets and shirts that provide many different ways of dressing the user, being these ways constrained by his/her own abilities and preferences. Figure 1.4 shows an example of jacket dressing with the help of a robot.

Challenges

Similarly to the shoe fitting case, many challenges are observed in the jacket dressing task.

- **Deformable garments:** Clothes are flexible objects and highly deformable. For this reason, it is difficult to perform state estimation, grasping, and manipulation; being those big open research topics.
- **Need for bi-manual manipulation:** While some kind of shoes can be fitted with a single hand, fitting a jacket requires at least two hands to correctly hold and fit the garment to the user.

- **Need for user collaboration:** It is essential to have a small amount of collaboration in terms of small movements in order to dress the jacket. Otherwise, the user's arms would need to be moved externally, making the task more complex.

- **Related limbs:** Fitting one arm depends on how has the other been fitted. In a dependency scenario like this one, the insertion order of the sleeves may matter, as one arm can have more mobility than another, or moving one arm can be harmful or difficult for the user. Therefore, there are different strategies based on the user's abilities and limitations.

- **Forces:** Although assisted jacket dressing does not require pushing forces against the user, the garment may get stuck while fitting, which may cause indirect forces applied to the user or undesired movements of the user's hands. Given that the limbs are related, this can cause dangerous situations when fitting one sleeve having the other one stuck.

- **Precision:** The sleeves opening must be precisely approached to the hands of the user to start the fitting. Failure to do so could cause the sleeve to get stuck in the arm, although this situation should not be as severe as in the shoe case.

Considerations and assumptions

Due to the number of research challenges present in this scenario, some assumptions have been made. To start with, we will consider the initial state to have the garment correctly grasped. This allows an abstraction from the grasping and garment detection tasks, which are research challenges on their own, and out of the scope of the thesis.

The garment to be fitted has been selected to be a jacket, against other clothes such as T-shirts. This is mainly due to jacket dressing being less intrusive, and therefore less dangerous for test subjects. A T-shirt dressing scenario would include movements that sometimes are close to the head or other vital parts. Moreover, T-shirts are more complex to fit and easier to misfit or get stuck. Lower limb clothes, such as trousers, have not been considered due to the extra challenges they involve. Some examples are the need for lifting the person when seated or resting on a bed, and the possibility for the user to lose balance if putting on the trousers while standing. Therefore, jacket dressing can be performed more safely and will serve as a good testbed for the proposed algorithms.

Finally, dressing a jacket allows for many different preferences to be added, as well as different strategies and constraints between the actions to be performed by the robot. For instance, fitting a sleeve constraints how can the other sleeve be fitted, as moving the garment would force the already fitted hand.

1.4 Outline

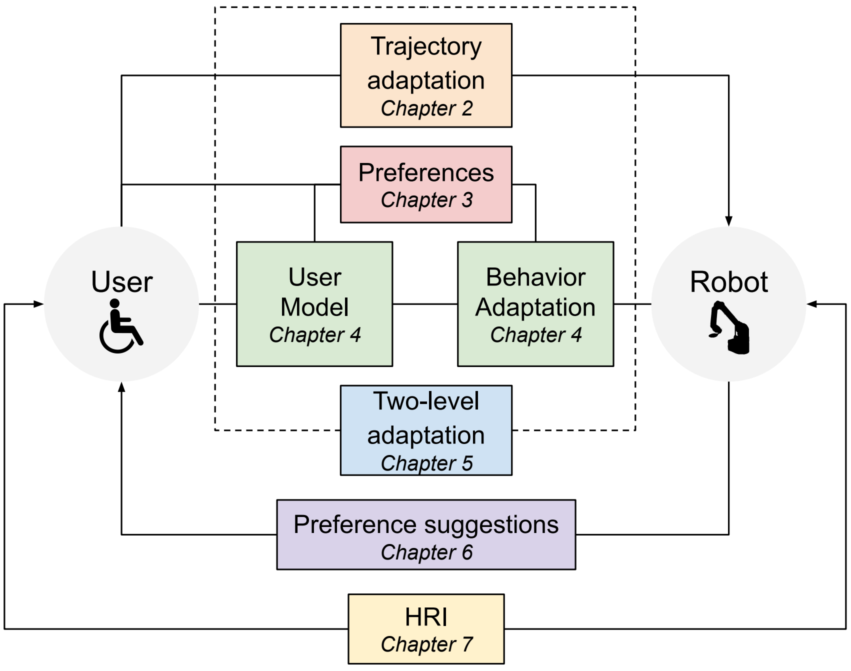


Figure 1.5: Graphical outline of the thesis.

The thesis has been organized as follows. Figure 1.5 shows a graphical outline.

- Chapter 2 presents the FUTE framework for at-home robot personalization and show its applicability in adapting the robot trajectories in a feeding task.

- Chapter 3 is devoted to define what can be considered as preferences for Physically Assistive Robots. It introduces a taxonomy of preferences that provides a wide definition and classification of them and shows how can those be integrated with the personalization framework. Some examples of its applicability are described with a use-case and puts them in the context of the FUTE framework.

- Chapter 4 proposes algorithms for behavior adaptation of the robot in a planning environment. The method creates a user model with a Fuzzy Inference System and uses it to guide the planner to select the best actions for the user. Such actions are chosen by the planner depending on some preferences extracted from the taxonomy defined in Chapter 3. After each task execution, the outcome of the task is used along with the feedback from the user to re-balance the task. For it, the action-outcome probabilities and the preference-related costs are updated to favor the observations of the execution. We evaluate the methods in a shoe-fitting task, both with simulated and real experiments.

- Chapter 5 demonstrates how the use of the planning techniques can be joined with "smart" low-level controllers for more efficient and robust task design. We show its applicability to a shoe-fitting scenario and demonstrate how some robustness arises when this approach is used, which can solve untaught situations. We also argue how this union of concepts allows for easier demonstrations and domain description, lowering the system's complexity. This is linked with all the steps of the FUTE framework.

- Chapter 6 introduces a novel algorithm for providing suggestions in planning. To do so, an extension to the ROSPlan framework has been proposed to ease the use of more expressive language like RDDL. This allows us to express a richer reward function that involves the preferences and the actions to provide the suggestions. Such suggestions are predicates that improve the total task performance when available. This task performance is measured in terms of the total reward of the plan. We show how this method can be used to provide preference suggestions even when some preferences are already grounded, and even provide change suggestions to the user. The system is evaluated in simulated experiments in three Physically Assistive Robotics tasks.

- Chapter 7 analyzes the use of the preferences proposed in Chapter 3 and the other adaptation methods through task planning in a user study. We evaluate the user's ability to determine in which executions their own chosen preferences are used, and whether they can distinguish the changes in the robot's behavior produced by such preferences. The study gives insights on the impact of said preferences in the assistive robotics tasks, and show promising results on the use of behavior adaptation for effective assistance.

- Chapter 8 provides the conclusions to this thesis with some of the future work and directions of the research proposed in the thesis.

Appendices

- Appendix A presents the list of academic publications resulting of this thesis.

- Appendix B analyzes some safety strategies for physical Human-Robot Interaction.

- Appendix C shows some experiments to support the use of probabilistic planning in robotics using the ROSPlan extensions described in Chapter 6.

- Appendix D transcribes the questionnaire used to evaluate the user study performed in Chapter 7.

2 Personal assistive robots for non-technical users

This chapter presents a robot behavior personalization framework focused on assistive tasks. The framework defines a three-step methodology to guide the development of adaptive and personalized assistive tasks, and more specifically the physical ones. A demonstration of the framework's use is provided in the context of feeding assistance, showing how physical adaptations can be performed by untrained users.

This work has been published in [6].

2.1 Introduction

Robot adaptation is especially useful in cases of users in need of assistance. Such users, with their own limitations, are usually unable of controlling or adapting a robot to suit their needs. And this vulnerability may hinder the use of such assistive devices, making them unusable. However, it is not clear how the process of personalization should be performed. We envisage the robot acquisition process as the robot being built, programmed, and shipped to a hypothetical user's home. However, personalizing the robot at building time or programming time is hard and costly, and the users may not still know their actual needs. But doing it at home may not be viable for dependent and possibly non-technical users.

In this chapter, we propose a novel Robot Personalization framework named FUTE (detailed in Section 2.3), that takes into account the user and allows concrete adaptation of generic pre-trained skills. In our framework, the robot is pre-trained at the factory with a set of abilities. Afterward, when it arrives at the user's home, a non-expert teacher (the user itself or a caregiver) must have the freedom to adapt such skills to his/her preferences, or even teach the robot new ones.

Second, we explore how to perform this training by using Learning-by-Demonstration techniques combined with a compliant robot controller [20]. We propose two interaction strategies:

the teacher intervening in the robot motion, and the demonstration of a completely new trajectory.

In the third place, we test the applicability of the proposed FUTE framework in an assistive task consisting of feeding a person. As feeding can be very complex, we focus on a specific aspect: how the robot approaches the cutlery to feed the person (see Figure 2.1). We will show how our system can extract the relevant aspects of the feeding task. Observe that, depending on the mobility and preferences of the user, the robot must wait with the food at some distance or introduce the food inside the mouth. Moreover, the feeding motion has to be adapted to the kind of food, for example, yogurt or fries as seen in Figure 2.1b.

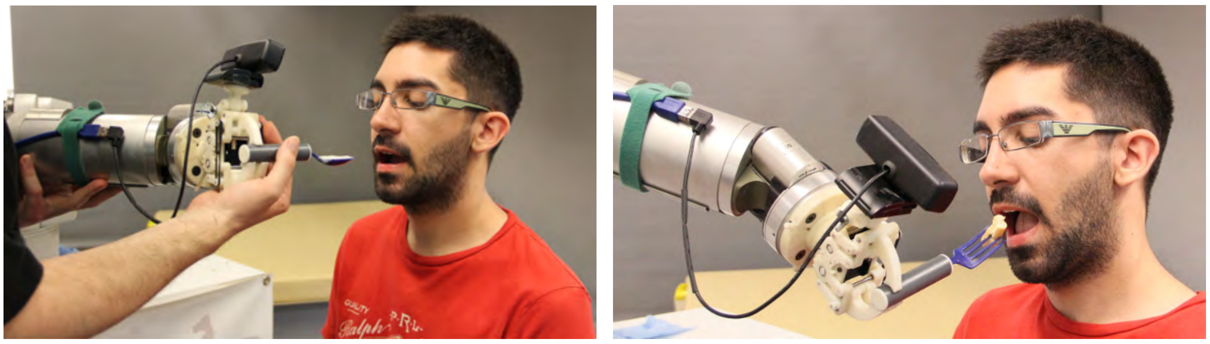

(a) Caregiver personalizing a spoon feeding skill. (b) A user eating from a fork.

Figure 2.1: Assistive personalized feeding application example.

2.2 Related work

Personalized Human-Robot Interaction has been studied in different works and fields. In education, it has been applied to Socially Assistive Robot (SAR) tutors that support the teaching task [21–23]. Baraka and Veloso [24] define three user models to adapt the luminous interactions between a robot and the user over time, learning the model parameters from user feedback. Personalized collaboration is shown in Fiore *et al.* [25], where an object manipulation task is performed jointly by the robot and the user whose preferences are taken into account. Abdo *et al.* [26] predict user preferences to tidy up objects in containers using collaborative filtering based on crowdsourced data and the observations of current dispositions or by querying the user. Although this strategy seems good for the tidying up task, it would not suit to capture the user preferences in an interaction context such as ours. Chernova and Veloso [27] present the Confidence-Based Autonomy (CBA) algorithm, which enables the agent to request demonstrations from a human teacher, and allows him to correct further mistakes with additional demonstrations. The idea is similar to the User Tailoring one, though they apply it to improve the policy rather than to adapt a well-learned task to a specific user. A framework to learn and generalize complex tasks from unstructured demonstrations is proposed in Niekum *et al.* [28]. The method is able to recognize repeated instances of skills and generalize them to new settings. Similarly, learning from demonstration has been used by Lawitzky *et al.* [29] to provide physical robotic assistance such as object maneuvering.

In addition, more in the scope of this thesis, personalized dressing assistance is performed by Gao *et al.* [30], where a user's movement space is modelled and used to put on a sleeveless jacket. Similarly, Klee [31] assist a user to place a hat in a collaborative way by means of asking the user to reposition itself when some user specific constraints do not hold. However, the personalization they propose consists in adapting to the user state or pose, but do not allow the user to modify the way in which the assistance will be carried out.

Moreover, we will apply the personalized interaction to the feeding scenario. Assistive feeding devices have been around for a while, mainly due to the evident need that some individuals have. Devices such as SECOM's MySpoon [32, 33] or the Handy 1 [34], among others, can provide significant help to allow people with upper limb disabilities to eat in a more autonomous manner.

Nonetheless, these systems lack the ability to adapt to the needs of each specific user. And, in cases of people with disabilities, this is a key factor for the system to be actually helpful in different kinds of environment, in which there is a handful of ways of assisting in the eating task, as often pointed out by long-term care nurses.

2.3 The FUTE Personalization Framework

We present a three-phase framework, the "FUTE framework", to design and develop, among others, the kind of adaptive assistive applications described in Section 1.3. The three phases are called "Factory setting", "User Tailoring" and "Execution tuning". They are depicted in Figure 2.2 and described as:

1. **Factory setting**: the robot is provided with the skills needed to perform the assistive task in a generic way. This would suit either the design of a new robot or the enhancement of an existing platform to carry out a new task.

2. **User Tailoring** (the focus of this thesis): This second phase takes place in the user's home. The robot performs a nominal skill, but personalization is encouraged in order to adapt its behavior to the user needs. In this phase, the robot should acquire, as automatically as possible, information about how the task has to be done for the user at hand while it performs the task in the generic way. This personalization may be done by the user or

by an external agent (such as a carer), and it could be either explicit or implicit. In the feeding example, this could consist in the selection of the feeding point, it being either inside or outside the mouth. The data in our implementation include the proprioceptive robot perception as well as 3D images from a camera located at the hand of the robot (Figure 2.1).

3. **Execution tuning**: In this last phase, the robot performs the task designed in the first phase but taking into account the personalization introduced in the second one. In the feeding example, the 6D pose of the user can be computed using an RGBD camera and a face detection algorithm, and the robot trajectories adapted to the current pose of the user. If the user is not satisfied with the robot behavior, the User Tailoring phase can be repeated to further adapt the robot's behavior.

2.4 Experimental assessment: User-Centered Feeding Assistance

To build intuition, we illustrate the different aspects of our framework using the robot feeding application. Eating is one of the most basic physiological needs all human beings have, appearing at the base of Maslow's hierarchy of needs [19]. However, some people with disabilities may not be able to do it by themselves, requiring the help of an external agent (usually a human carer), who will feed them taking into account their needs and capacities. An example of the complete robotic feeding process is shown in Figure 2.3. To illustrate this, in the following experiments we tackle two example use-cases in which different personalizations can be applied:

- **U1**: a person with very limited upper body mobility will require the caregiver to do all the feeding action. Figure 2.3 exemplifies this case, where the user does not move the neck.

- **U2**: a different patient with upper limb disabilities may be able to move and eat the food by himself when it is close enough.

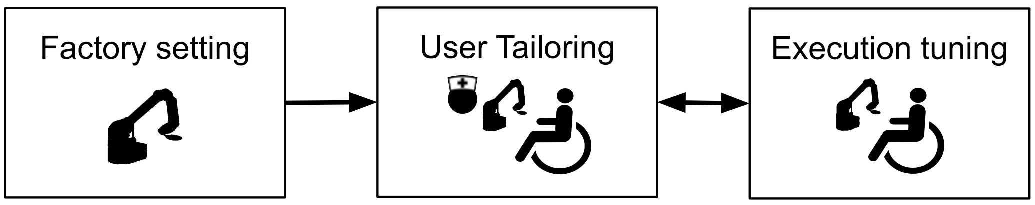


Figure 2.2: The FUTE process.

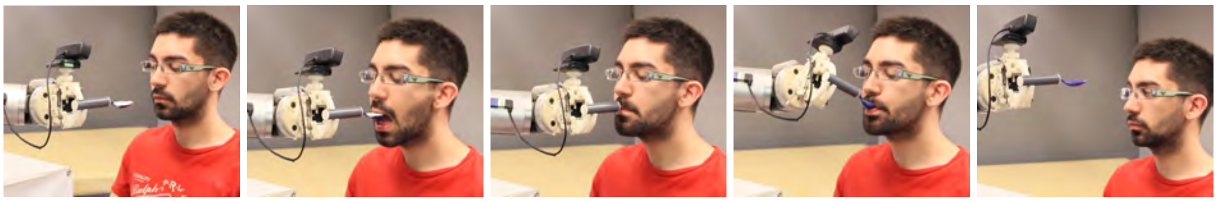

(a) Initial position (b) Feeding pose (c) Food ingestion (d) Move away (e) End of feeding

Figure 2.3: Example of a feeding execution for the case of a user with reduced mobility.

The following experiments focus mainly on the evaluation of the robotic system. Therefore, the user was just instructed to act as described in the two use cases and was there only to obtain a more realistic situation to analyze the robot trajectories. For an evaluation with real users refer to Chapter 7.

2.4.1 The Robot Feeding Process

Algorithm 2.1: Feeding execution

```
1 graspFeedingUtensil()                                  // Grasp a spoon or a fork.
2 repeat
3     pickUpFoodFromPlate()
4     userPose := getHumanPoseFromPerception()
5     moveToInitialPosition(userPose, initialPose)
6     moveUtensilToFeedingPose(userPose, feedingPoint)      // Approach the food
7     waitForFoodConsumption()
8     moveAwayFromUser(userPose)
9 until feedingIsComplete()        // User has had enough food or plate is empty
```

Five steps can be identified for the adaptive feeding application (see Algorithm 2.1). In the context of the proposed framework, steps between lines 1 and 3 would be provided to the robot during the factory training phase, while steps between lines 6 and 8 would be personalized at home. Thus, the complete execution is the outcome of joining the already known steps (at the factory phase) with the personalized ones, resulting in a successful feeding action for a specific person. The "`initialPose`" (line 5) and "`feedingPoint`" (feeding moment of the trajectory, line line 6) parameters are obtained during the User Tailoring phase, as seen in Algorithm 2.2. Note that in execution, the user can move freely. A vision system comprised of a low range RGBD sensor is used to compute their pose, and the robot motion is updated accordingly to obtain the desired feeding movement. The vision system is also used to detect the moment in which the user bites the food in the "`waitForFoodConsumption`" step (line 7).

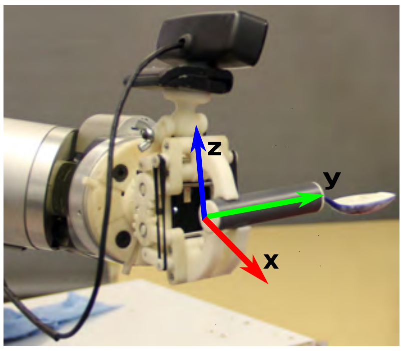


Figure 2.4: Representation of the feeding setup with the trajectory Cartesian coordinate axes.

In this chapter, we will just focus on the steps involving the user (lines 6, 7 and 8 from Algorithm 2.1), and how they can be personalized to different users[1].

The feeding setup used in the experiments can be seen in Figure 2.1a and Figure 2.1b, and the coordinate axes at the robot's end-effector are shown in Figure 2.4. In it, the y Cartesian axis represents the frontal distance to the user, the x axis corresponds to the horizontal displacement and the z to the vertical one (the feeding height).

2.4.2 Feeding personalization

The User Tailoring strategy for feeding is shown in Algorithm 2.2. It comprises the recording of N sample trajectories (line 4) including the approaching motion, waiting for the user to start the consumption, and a receding motion. The N trajectories are then used to learn a Probabilistic Movement Primitive (ProMP) [35,36] of the feeding movement (line 19). ProMPs are movement primitives that encode the time-varying variance of a set of trajectories. The state vector $\mathbf{y}_t$ is defined as

$$\mathbf{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \mathbf{\Phi}_t^T \mathbf{w} + \epsilon_y, \tag{2.1}$$

where $\mathbf{\Phi}_t = [\phi_t, \dot{\phi}_t]$ is the time-dependent basis matrix, $\mathbf{w}$ is the weight vector and $\epsilon_y \sim N(0, \Sigma_y)$ is Gaussian noise. The trajectories can then be represented as a mean trajectory and its variance, each time point being represented as $\mu_t \pm \sigma_t$. New trajectories can be sampled from the distribution, and via points are defined using the conditioning operator. We have used the ProMP formalism because, apart from the trajectory itself, as will be seen in Section 2.4.4, it also provides insights of the particularities of the task by means of the variance along the trajectory.

We would like to assess the impact of variations in the demonstrated trajectories, to provide

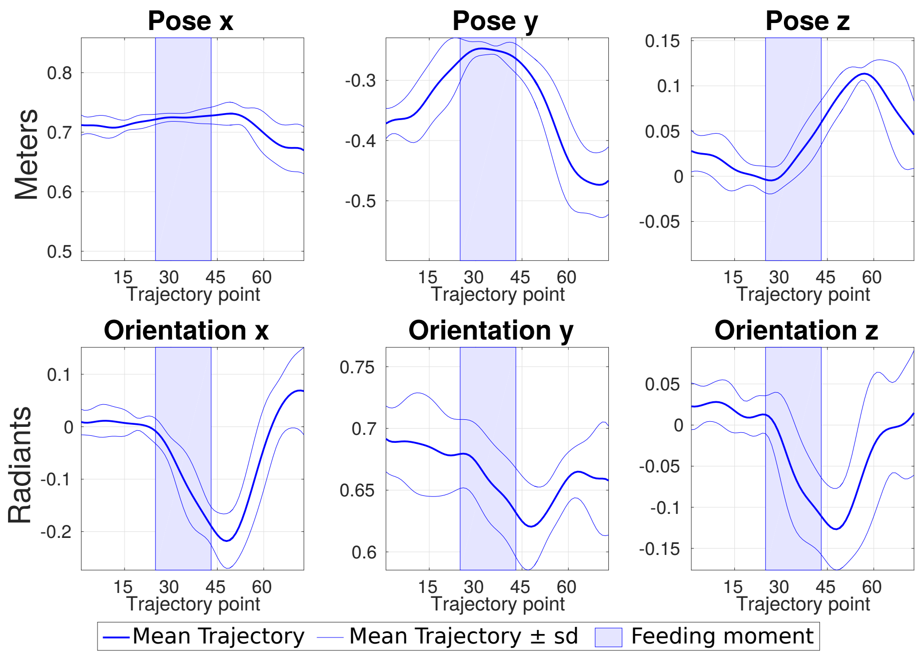
[1]A video showing the process regarding the personalized feeding task can be found at www.iri.upc.edu/groups/perception/frameworkFUTE.

hints to the caregiver demonstrating the task about how similar the N demonstrations should be. The next experiment tackles use-case **U1**: introducing the food inside the mouth of the user. It involves demonstrations using two different feeding paths with a mannequin as user: the first one in which the carer tried to perform the same trajectory 5 times, and the second set in which the 5 trajectories had different approaching movements (but with the same feeding point). The results are shown in Figure 2.5.

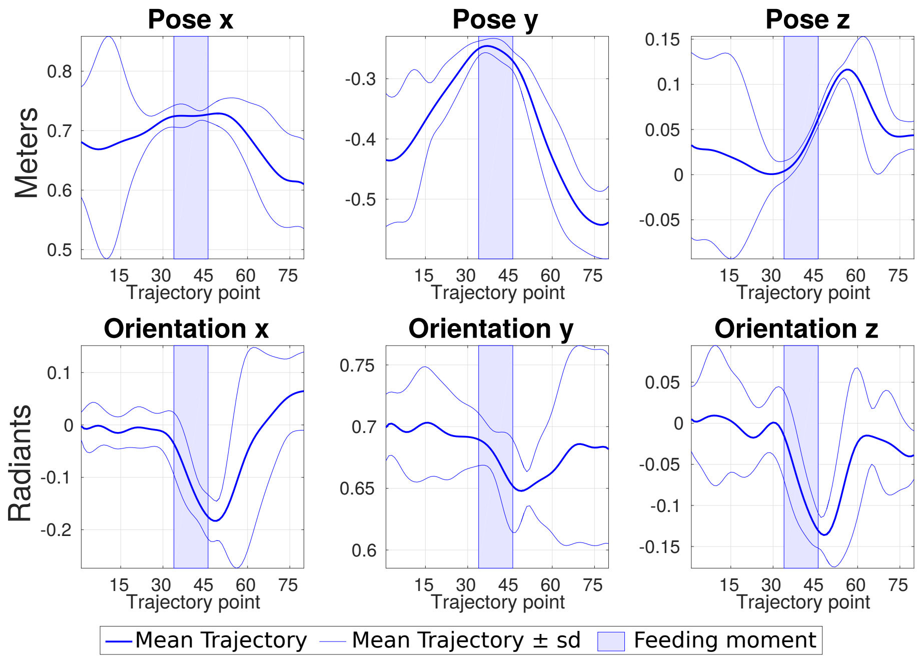
Comparing Figures 2.5a and 2.5b it can be seen that the shape of both mean trajectories is quite alike, both reaching the same feeding position (shaded area, corresponding to the steps from Figures 2.3b to 2.3d). As a consequence, apparently there is no need to have several similar trajectories in order to have a good average feeding movement. However, we observe different variances. In Figure 2.5a variance is almost constant during the whole trajectory, while in Figure 2.5b variances in the approaching and receding movements are larger, but smaller in the feeding point. Observe that obtaining this information is crucial, as the robot should act carefully while feeding the user (lower variance) whereas approaching and receding can exhibit a more careless behavior (larger variance). Thus, we conclude that showing some variability in the demonstrated trajectories is important.

Algorithm 2.2: User tailoring strategy

```
demonstrations := $\emptyset$                        // Will store the new recorded trajectories
feedingPoints := $\emptyset$                           // Time points of each feeding trajectory
initialPoses := $\emptyset$                          // Face pose at the start of each trajectory
forall $i \in \{1..N\}$ do
    if unassistedTraining then          // Set one of the two personalization modes
        SetRobot(gravityCompensationMode)
    else
        SetRobot(ReproduceFactoryTrajectory, stiffness)
    initialPoses := append(getUserFacialPose())
    newTrajectory := $\emptyset$
    while robotMoving do                             // Store approaching trajectory
        addPoints(newTrajectory)
    waitForFoodConsumption()                        // Wait until user starts eating
    feedingPoints := append(currentTrajectoryPoint)
    while robotMoving do                                // Store receding trajectory
        addPoints(newTrajectory)
    demonstrations := append(newTrajectory)
referenceFeedingPoint := alignToFeedingPoint(demonstrations, feedingPoints)
personalizedTrajectory := RecomputeProMP(demonstrations)
return <personalizedTrajectory, referenceFeedingPoint, avg(initialPoses)>
```



(a) Similar trajectories.



(b) Different trajectories.

Figure 2.5: Comparison between similar and different example trajectories. The thicker line is the mean trajectory, and the surrounding lines are the mean ± standard deviation. The shaded regions denote the part of the trajectory in which the food is consumed, corresponding to the steps shown in Figures 2.3b-2.3d.

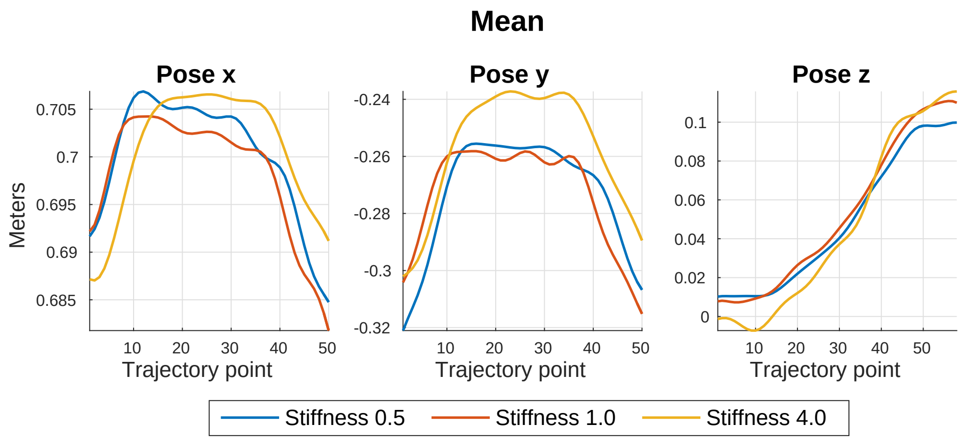


Figure 2.6: Mean trajectories generated from a default trajectory with different stiffness values. The lower the stiffness, the most docile the robot behavior is. Observe the oscillations introduced when the trajectory is perturbed.

2.4.3 Teaching Modes: Unassisted vs. Compliant Reproduction

Two teaching modes have been defined (Algorithm 2.2 lines 5–8). The first one is unassisted, the robot only compensates gravity and the caregiver has to start from scratch each demonstration handling the robot and freely performing a feeding trajectory. This allows the user to discard the factory settings and re-teach the whole movement. In the second one, the robot executes a generic feeding trajectory –which was recorded in the factory setting phase– using a compliant controller [20] that uses a *stiffness factor* to determine the arm's stiffness degree.

The next experiment is designed to assess the effect of the stiffness factor. Hence, we repeated the executions with different stiffness values for the same trajectory where the caregiver personalized the motion so that the feeding occurred further away from the person (a mannequin was used in this experiment to avoid noise induced by involuntary movements and ease the comparison). Here we tackle use-case **U2**: the trajectory is modified to end outside of the mouth, for instance for patients some mobility. The results are shown in Figure 2.6.

The intuition says that starting from scratch at every demonstration is harder, whereas if the robot reproduces the movement in a docile manner the user only has to physically perturb the execution in some parts and teaching becomes easy. However, as it can be seen in Figure 2.6, this second approach introduces oscillations of about half a centimeter in the resulting trajectory, not only in the y axis (the approaching direction) but also in x and z. With low stiffness values the oscillations tend to be higher as the robot reacts to slighter perturbations as when it tries to go on with the trajectory and return to the original path and the user holds it again. In contrast, higher stiffness makes it harder for the user to modify the trajectory, resulting in less oscillations but more physical effort for the user.

2.4.4 Parameter extraction from the learned trajectories

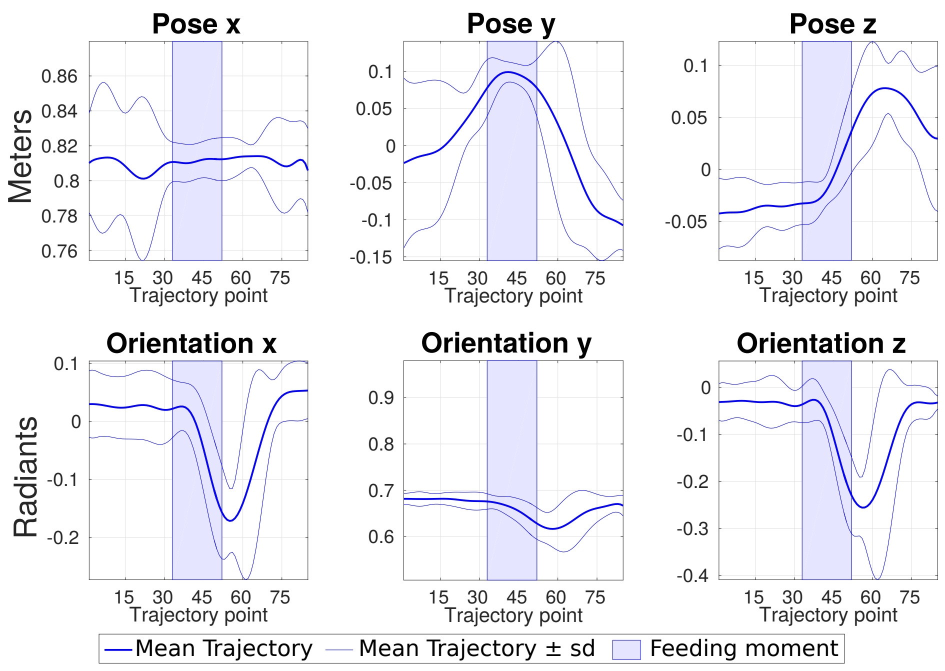
In the next experiment, the modifications that the caregiver can introduce to personalize the feeding process are (see Algorithm 2.2, lines 9, 17 and 18): the initial pose, the motion shape, and the feeding point (inside the mouth for use-case **U1** or just approaching the food for use-case **U2**).

We show how these parameters can be extracted during the User Tailoring phase. First, the feeding point is computed by recording the distance to the face in which the movement is stopped to feed the person. Second, the motion learning process captures the particularities of the task. We exemplify this fact by observing variances of the ProMP trajectory related to two different utensils: when a spoon is used the orientation is more restricted, while a fork allows for more flexibility.

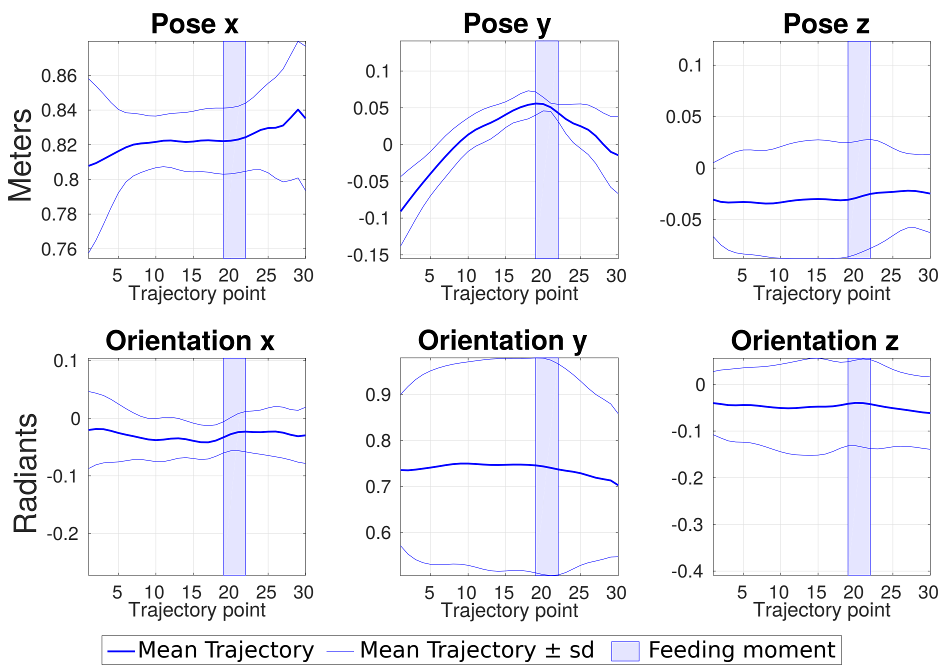
In this experiment, the re-teaching has been carried out with the robot holding a spoon with yogurt and also with a fork pinching a french fry. Five trajectories were recorded in order to generate the ProMP for each case. A human user was used here as test subject (not a mannequin) because the insertion orientation was relevant (see Figure 2.1). With this experiment, we can observe how the particularities of the task are integrated into the ProMP. Figures 2.7a and 2.7b show the trained trajectories for each Cartesian coordinate and the rotations around each axis, displaying the mean trajectory and its variance. The figures clearly show the moment in which the utensil is near the mouth (as seen in the shaded regions), because the variance of the movement narrows at that stage. This is, in fact, a representation of the flexibility of the movement, since the critical parts that need more precision are less flexible.

In addition, this variance effect can also be seen in the orientation plots, in which the spoon's sample orientation variances are narrower at the beginning of the trajectory to avoid spilling the content, while the move away part has wider variances as the food has already been taken. The fork trajectory has less restrictive orientations because there is less danger of dropping food, as clearly seen in the orientation around the y axis.

Moreover, this gives us insights on how the variance in the trajectory points provided by the ProMP could also be used to control the compliance (stiffness degree) of the robot during the trajectory execution phase. This way, the robot would be more docile to external forces in moments of high variance, corresponding to points of the path that have been taught in non precise ways, and more rigid in low variance points. Thus, the robot would not react to external forces while introducing the spoon in the mouth, avoiding any possible harm to the user due to accidental robot perturbations. Note this should not be applied in the joints interacting with the user, allowing for docile movement with the mouth but being stiff in external joints such as the elbow.



(a) Spoon feeding re-teached ProMP.



(b) Fork feeding re-teached Cartesian ProMP.

Figure 2.7: Learned trajectories for the spoon and fork experiments, where the gravity compensation mode was used for re-teaching the trajectories.

2.5 Summary

In this chapter, we presented the FUTE robot personalization framework consisting of three phases: Factory setting, User Tailoring, and Execution tuning. This framework has been devised to help the implementation of assistive applications by allowing easy adaptation of the assistive robot performance to specific users, given the fact that all of them are different and have their own special needs. Furthermore, it allows non-expert users to conduct the robot adaptation just by guiding the robot behavior.

Then, we tested this framework in a feeding application where a human caregiver can re-teach the feeding movement the robot has to perform, by physically modifying an already learned trajectory or by teaching it from scratch. This allows the person to teach the feeding point and distance so it can be either inside or near the mouth. Moreover, we demonstrate how the use of kinesthetic teaching to learn Movement Primitives, such as the Probabilistic Movement Primitives (ProMPs), is an appropriate choice for these kinds of assistive applications. These primitives are able to learn particularities of the task such as the feeding moment as well as the flexibility of each part of the trajectory.

In this chapter, we have explored the personalization of trajectories, which can be considered low-level adaptations. In the following chapters the focus will be set in the semantic adaptation to specific preferences of the user, which we can consider to be high-level adaptations of the robot.

3 Defining preferences for assistive scenarios

The FUTE framework, presented in the previous chapter, defines a methodology to create personalized robotic assistants at home and demonstrated its application using the feeding task as a case study, where the robot is adapted in the low-level trajectories. Although effective, further adaptation is essential for successfully assisting dependent users. We believe these adaptations must also consider more abstract concepts such as the ones of user preferences during the personalization phases. However, this concept of user preferences can be quite broad and thus it needs to be narrowed down to the case of robots that assists people in the performance of their ADLs.

In this chapter, we define the concept of preferences for assistive robotics tasks. We do so by defining a taxonomy of user preferences for assistive scenarios, including physical interactions. The preferences we consider here are those that may be used to improve robot decision-making algorithms. The taxonomy categorizes the preferences based on their semantics and possible uses. We propose the categorization in two levels of application (global and specific) as well as two types (primary and modifier). Examples of real preference classifications are presented in the three assistive tasks defined in Section 1.3: assisted feeding, shoe fitting, and jacket dressing.

This work has been published in [12].

3.1 Introduction

Given the number of manners in which assistance can be provided, there is a need for defining what can determine how the robot performs the task. We believe that user preferences are a good tool to do so, as knowing what does a person wants or likes allows humans to behave in accordance and in an acceptable way for the patient. Hence, robots should be able to use preferences to adapt to their actions. However, the idea of preference may be too fuzzy and wide for being useful to drive the robot's behavior, and some grounding of the concepts is needed. Thus, in order for a robot to successfully use preferences, they should be appropriately defined, structured and categorized.

In this chapter we answer the question of "how can we define and classify preferences?" in the context of Physically Assistive Robots by defining a taxonomy of user preferences for Human-Robot Interaction applications[1], putting special emphasis on *physically assistive* scenarios in which the inclusion of these preferences will make a difference. The taxonomy will ease the definition and classification of the preferences which, written in non-technical language, facilitate the inclusion of caregivers in the loop of assistive application design. Moreover, the taxonomy will also be useful to implement preference-based applications that take into account the different categories. This customization of the applications will allow the adaptation of the robot's autonomy from a simple tool to a shared-autonomy system or a fully autonomous robotic assistant.

When taking into account possible contacts between the person and the robot, we have identified two loops in the execution of actions: a higher-level decision-making in terms of finding a sequence of symbolic actions to be performed, and a lower-level one to execute these actions of the task. We observe that preference specification in the former has received more attention from the community, while the latter is less explored as it requires the grounding of the involved symbols. In the presented taxonomy, this has been translated into preferences that permit guiding action selection (named *decision-making preferences*) and those that define how the selected operators are executed (named *configuration preferences*).

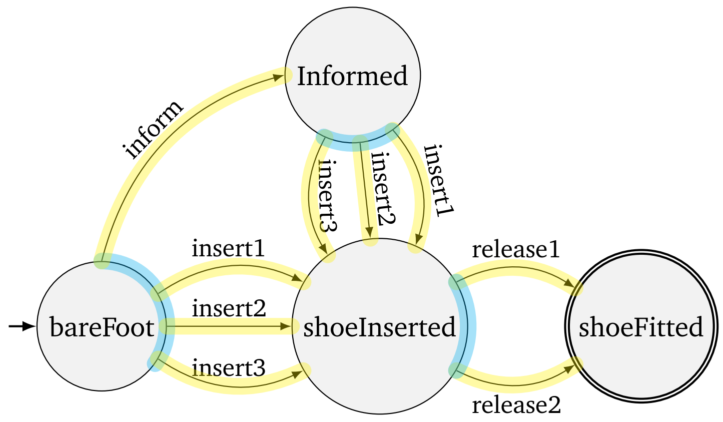


Figure 3.1: Graphical example of *decision-making* (blue) and *configuration preferences* (yellow): in this action-sequence flow for the shoe-fitting task, represented as a FSA where the arrows represent the actions executed to change the state, decision-making preferences aid to choose among alternative paths, while the configuration ones help to tune action parameters.

[1]The taxonomy could also be used to define preferences in a more generic assistance scenario with a human caregiver and a patient, but in this thesis we are mainly focused in the HRI scenario.

In Figure 3.1 we exemplify different states in the shoe fitting task along with some state transition actions. For the sake of simplicity of the example, we can imagine that the robot has only three shoe insertion actions and two shoe release operations available, and it can inform the user before inserting the shoe using any of the three available actions. The selection of the action to be performed (whether it has to inform or which insertion should it use) is based on the decision-making preferences (marked in blue), while the configuration preferences define how –with which parameters– the selected action is to be performed (depicted in yellow). Therefore, *decision-making* happens before the action execution, while *configuration* affects the action while it is being executed. A resulting action sequence example in this scenario is `[insert1, release1]`, but the robot may also inform before doing the same execution, thus resulting in an action sequence of the form `[inform, insert1, release1]`.

3.2 Related work

Preferences are a central problem in decision making. As an example, a comprehensive survey that reviews the different alternatives for modeling and using preferences in Artificial Intelligence was published by Pigozzi *et al.* [37].

Preferences in planning

The planning community has focused on the use of preferences in different manners. For instance, Preference-Based Planning (PBP) [38] is an extension of classical planning where a criterion is provided to select one plan among other valid plans based on user preferences. Hierarchical Task Networks (HTNs) have been also used to encode user preferences [39]. In HTNs, a hierarchy of non-primitive actions is provided along with a set of methods to decompose them into primitive actions. The manual construction of HTNs indirectly encodes the user preferences, but is complex, error prone, and preferences are not always explicitly stated. Unfortunately, these works do not consider particular problems that appear in robotics and physical interaction.

The Planning Domain Description Language (PDDL) is often used to describe planning domains. PDDL3 [40] was the first version to define the preference construct, which allows to describe three types of preferences. The temporally extended preferences consist in desirable temporal relationships, the precondition preferences are atemporal formulae that should hold true in the state in which an action is to be performed, and the simple –also called goal– preferences are conditions that should hold in the final state. Sohrabi *et al.* [41] address the generation of preferred plans by extending the PDDL3 language to handle preferences over

HTN constructs, supporting desires on how the tasks are decomposed.

Son and Pontelli [42] divide the preferences in different categories: preferences about a state define the preferred properties to hold in a state; preferences about an action describe actions that are preferred; preferences about a trajectory define preferred properties over sequences of actions; finally, multi-dimensional preferences consist in a set of preferences and an ordering among them. The authors introduce the language $\mathcal{PP}$ for planning preferences specification and subdivide the preferences in basic desires, atomic preferences and general preferences. Although this categorization is suitable for planning and other problem solving tasks, we find it is not sufficient to define a set of preferences for physical interactions [43]. In our case, we propose a hierarchical taxonomy in which preferences are categorized by function and type.

Taxonomies in Human-Robot Interaction

Taxonomies allow the description and classification of concepts involved in a domain, organized in a structured manner. In robotics, their usefulness has been demonstrated by the different taxonomies that have been proposed in the literature, with some of them related to the interaction and its social aspects.

A taxonomy for Human-Robot Interaction was proposed by Yanco and Drury [44] that allows to express elements such as: the social nature of the task, its type, the robot morphology and the interaction roles between teams of humans and robots. The taxonomy, however, does not include elements related to the preferences of the user but rather focuses on the interaction scenario.

Krauss and Arbanowski [45] build a social preference ontology to tackle typical issues of recommender systems, such as the cold start and the sparsity problems. The ontology represents topics the user is interested in along with a numerical score, and is filled up with information mined from social networks. Being task-specific, these ontologies do not suit our assistive robotics scenario as they lack the semantics specific to the personal satisfaction domain.

Bastemeijer *et al.* [46] define a taxonomy of the concepts patients value in health care based on a thorough literature review of several studies. They define three top-level categories: patient and personal context, the characteristics of the professional and the interaction between the patient and the professional. The key elements inside these categories are: uniqueness, autonomy, compassion, professionalism, responsiveness, partnership and empowerment. Although the elements they define could well suit our scenario, their concepts relate to general health care and patient's feelings, while our proposal is focused on defining key aspects of the behavior of the (robotic) assistant in the physical assistance environment.

A framework for levels of autonomy (LoA) is proposed alongside with a 10-point taxonomy in [47]. The taxonomy specifies each level of autonomy from the perspective of the human-robot interactions and the roles they play, and divides the HRI variables in robot-related, social, and human-related. We can link this definition of LoA to our proposal of preference categorization, as the set of values of the preferences can be used to determine the resulting LoA: from "shared control with human initiative" or "shared control with robot initiative" to "full autonomy".

Regarding social aspects of interaction, Peng *et al.* [48] propose a hierarchical taxonomy for robotic dance. Shim and Arkin [49] define a taxonomy of robot deception for HRI contexts. Wiltshire *et al.* [50] propose a taxonomy of social signals from an interdisciplinary point of view. They categorize five social cues that can be extracted to predict social signals. However, although they may look similar in some aspects, their taxonomy is presented to categorize the human behavior's rather than the robot's as we intend in our proposal.

Fong *et al.* [51] present a taxonomy of design methods, system components and applications for socially interactive robots, but preferences were not yet included. There exist other general robotics ontologies, such as KnowRob [52], which provide robots with knowledge of the environment, the actions, the tasks and mathematical concepts, and may be extended with information about preferences.

The concept of preference tends to be quite application-specific, as the reviewed works show. Though there are taxonomies for social robotics, they are still not enough to categorize the user preferences regarding the robot's behavior, which we are dealing with in this work. More specifically, we define the preferences for assistive tasks in robotics, taking inspiration from the commented works such as [51], and directly including the Big Five personality traits ontology [53] where personality is described based on five traits: extraversion, agreeableness, conscientiousness, neuroticism and openness.

3.3 A Taxonomy of preferences for Assistive Human-Robot Interaction

In this section, we present a hierarchical taxonomy of user preference categories designed for Human-Robot Interaction applications, with emphasis on assistive scenarios where the robot aids users facing difficulties to perform Activities of Daily Living (ADLs). The taxonomy has been developed based on previous experiences, intuition and comments from health care providers and potential users, and motivated by the need to classify preferences in order to use them. To maintain generality, we do not distinguish between preferences and user constraints (such as mobility issues), as the latter can be expressed as a preference. For instance, a user who cannot move the right arm will "prefer" not to use this arm. Encoding impairments as preferences allows us to present a less limited taxonomy as there is no need to separate similar preferences and impairments in different categories. The proposed taxonomy has been designed with the main goal of describing preferences that are useful for an autonomous system to make decisions in an assistive scenario, either physical or social. Although it is not necessarily complete and may be extended, we believe it's general and representative enough to cover a broad range of assistive tasks and scenarios, if not all.

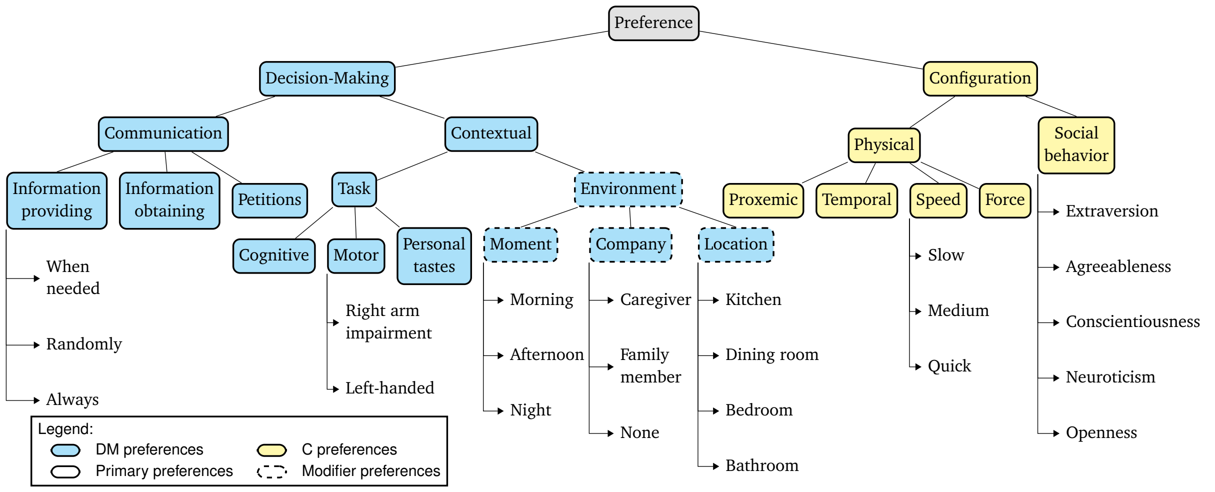


Figure 3.2: Preference taxonomy for assistive physical Human-Robot Interaction.

To begin with, we define two types of preferences: the *primary* preferences and the *modifier* ones. The former are preferences that are directly applicable, while the latter are used to accompany the primary preferences and modify them, effectively conditioning their applicability.

The proposed taxonomy is divided into two main category groups: the decision-making and the configuration preferences (Figure 3.2). The *Decision-making* (DM) preferences are those that help the robot to choose between the different actions that it can execute at a given moment, provided that they all lead to the final goal (see left branch of Figure 3.2). DM preferences are in turn divided into two categories, which are again subdivided into more fine-grained preference types:

- *Communication preferences* regulate the desired amount of different kinds of interaction with the robot. They are subdivided as:
  - Information providing: whether the robot should inform regarding each performed action or should omit unimportant information. It relates to the *verbosity* of the robot.
  - Information obtaining: define if the user prefers the robot to inquire about missing **information** or either it should try to infer it from other sources.

  - Petitions: state if the robot should ask the user to perform some **action** (such as repositioning himself to ease the solution of a task) or if the robot should risk to accomplish the task without bothering the user, provided that no safety issue can arise at any moment.

- *Contextual preferences* define how the robot's behavior may change depending on the execution context, it being defined as the user's environment, place and time. We define four subcategories:

  - Task: state preferences that have implications about the task that is being performed. They may define general user constraints (such as limited right arm mobility) or simple preferences such as how is more comfortable to scoop the spoon when eating. They may be subdivided in:
    - Cognitive preferences related to cognitive disabilities of the user.
    - Motor constraints of the user which may limit the task.
    - Personal tastes are other personal needs and desires.
  - Environment: preferences regarding the execution setting. They are mainly modifier preferences (see Figure 3.2) that accompany primary preferences and limit their application range. We propose a subdivision in three categories:
    - Moment: define the time of the day in which the task is executed, thus preferences may vary depending on, for instance, whether the task is performed in the morning or at night as the state of the person may be different regarding tiredness and mood.
    - Company: preferences concerning the personal elements that are in the environment. User preferences with the robot may be different when a caregiver is also assisting the user in contrast to when a family member is doing so. Besides, the preferences will be others when the user is alone with the robot, given that he may need more support in that case.
    - Location: the preferences related to the location where the user is situated. User preferences may change depending on where the task is being executed. For instance, it may not be the same to fit a shoe while seating on the bed than fitting a slipper while resting on the coach.

On the other hand, *configuration* (C) preferences (see right branch of Figure 3.2) are those preferences that define how an action is to be performed. They are used to tune the parameters of the actions rather than choosing the action sequence that is going to be executed to solve the task. Configuration preferences are also divided into two categories:

- Physical preferences define the physical properties of the (physical) actions. These include:
  - Proxemic: relate to the spatial requirements of the user, the robot and the task.
  - Temporal: define temporal requirements of the user and the task. For instance, the user may prefer to not have the foot lifted for more than one minute.
  - Speed: specify how quick or slow the user wants the robot to move. This relates to the feeling of safety, as the user may get scared if the robot makes sudden fast movements, but may also get impatient when the movements are too slow.
  - Force: some tasks, such as shoe fitting, require pressure against some body parts. The applied force may be limited based on user desires and abilities.
- Social behavior preferences somehow characterize the robot's personality. Following the definition of [51], here we link our taxonomy with the Big Five personality trait taxonomy [53] which describes personality in terms of five traits (extraversion, agreeableness, conscientiousness, neuroticism and openness). With these preferences, the user can define personality-based items. Other elements that could be included as social behavior preferences are the kind of voice, tone, formality level, prose and detail level of the interactive acts.

The proposed taxonomy allows us to define user preferences for Human-Robot Interaction tasks, and more specifically, for Physically Assistive Robots (PAR) to help older adults and handicapped people. However, future assistive robots should be able to perform more than one assistive task. Thus, there is a lot of redundancy when instantiating the preference taxonomy for every specific task. For instance, a user who prefers the robot to move slowly while fitting a shoe will probably prefer it to be slow when dressing a jacket. Or he may have reduced mobility in his right arm, which implies that the user will need special assistance to perform any activity involving this arm. To solve this redundancy, and to ease the description of the preferences, we propose to define them in a two-level manner:

- *Global preferences*, are those that are applicable to most tasks. They define generic user preferences and personal constraints which may be used in any setup.
- *Specific preferences* define activity-related preferences. They only apply to certain cases and during the execution of specific tasks.

Note that we do not restrict the possibility of specific preferences including elements that are already present in the global preferences, and they may even be in conflict by stating opposing elements. We tackle this by setting an importance level in which specific preferences take over

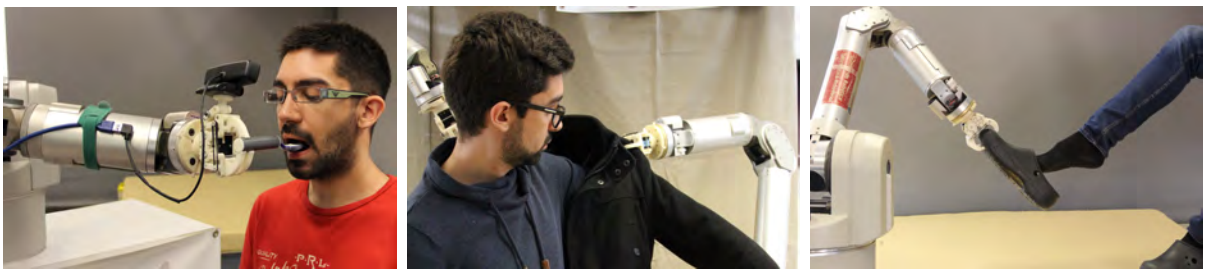

(a) Feeding assistance. (b) Jacket dressing assistance. (c) Shoe fitting assistance.

Figure 3.3: Physically Assistive Robot (PAR) application examples.

the global preferences when a conflict arises. In this way, a specific preference of the same kind of a global preference overrides it, allowing the user to have task-specific tastes without the need of repeating a general desire for every task.

Given that PAR are actually *touching* the human users, safety-related preferences are not taken into consideration. We believe that a Physically Assistive Robot must be safe out-of-the-box, and the user shouldn't be able to modify the safety level. Though strict, this restriction leads to the development of intrinsically safe systems which must not try to perform any action that may potentially hurt a person cohabiting the robot's environment.

3.4 Preference definition examples

This section illustrates how the taxonomy can be instantiated for the different Physically Assistive tasks as the ones shown in Figure 3.3, consisting in feeding, jacket dressing and shoe fitting. To do so, we will define a fictional persona [54] and instantiate her preferences:

— Aunt Mery is an 80 years old granny who lives alone. Although she's healthy, she is suffering from lower back pain and recovering from a fracture in her right arm. Due to these issues, she needs help to carry out some ADLs such as putting shoes on, dressing a jacket to go to the therapist or eating. Thus, a Physically Assistive Robot will help her to maintain some autonomy while she is recovering.

Table 3.1 shows the global preferences, which are applicable to any task. Tables 3.2, 3.3 and 3.4 show the (specific) preferences for the jacket dressing, shoe fitting and feeding tasks, respectively. The "textual definition" column represents what Aunt Mery would say to describe each preference. The specific preferences that override a global preference are marked with an asterisk (*). Note that when two specific preferences collide due to an additional modifier preference, the modified preference precedes the other ones, provided that the modifying condition holds. For instance, Mery prefers to eat slowly in the morning, though a medium velocity (global preference, Table 3.1) is better for any other time of the day (Table 3.4). Also, she cannot wait for more than thirty seconds with the foot lifted when she is alone, but she can hold it up for a minute when there's a family member helping her (Table 3.3). When dressing a coat, a normal force is fine during most of the task (Table 3.2), however she prefers the robot to apply less force when the injured right arm is being dressed. Nevertheless, she prefers to start with the left foot when fitting a shoe (Table 3.3), but the right arm is chosen when dressing a jacket (Table 3.2). The tables demonstrate how, even though some assistive tasks may look similar, the taxonomy allows to freely define different preferences regarding the same task-depending aspects.

Category	Primary	Modifier	Textual definition
Speed	Medium		"I generally don't want the robot to move fast nor slow"
Information Providing	Always		"I prefer that the robot talks to me"
Petitions	Minimum		"I prefer the robot to assist me without bothering"
Social Behavior	Informal and funny		"I like robots that make jokes"
Social Behavior	Formal and polite	not(Company/None)	"I want the robot to be polite when I am not alone"

Table 3.1: Example of Aunt Mery's global (task independent) preferences.

Category	Primary	Modifier	Textual definition
Motor	Right arm first		"The right arm is injured so it's easier to put it first"
	Both arms together	Company/Caregiver	"With the help of the caregiver it's easier to put on both arms together"
	Lateral trajectory		"I like it more when the jacket dressing is started from one side"
	Start position low		"The robot should start from a low position for easier dressing"
Force	Low	Moment/Putting the right sleeve	"The injured arm can't take much pressure"
	Normal		"I prefer the robot not to use too much force when dressing me"

Table 3.2: Examples of Aunt Mery's preferences for the jacket dressing task.

Category	Primary	Modifier	Textual definition
Speed	Slow*	Moment/Night	"I prefer to fit the shoe slower at night as I am more tired"
Motor	Left foot first		"It's more comfortable to put on the left foot first"
	Straight foot		"I don't feel comfortable with the foot turned"
	Left approach		"It's better if the robot approaches for the left"
Petitions	Sometimes*	Company/Caregiver	"The Caregiver helps me understand the robot and how to reposition myself"
Information providing	None*	Company/Caregiver	"The caregiver already gives me enough information"
Temporal	30 sec. lifted		"I can't hold the foot lifted for much time when I'm alone"
	1 min. lifted	Company/Family member	"They help me hold on with the lifted foot"

Table 3.3: Examples of Aunt Mery's preferences for the shoe fitting task.

Category	Primary	Modifier	Textual definition
Speed	Slow*	Moment/Morning	"I like to take my breakfast calmly"
Motor/ Proxemic	Outside feed		"I can't move the spoon but I don't need the robot to insert it in my mouth"
Motor	Straight scooping		"I don't want the robot to move much when I'm biting the spoon"
	Left-side approach	Location/Kitchen	"I'm more comfortable when the robot is in the left side"
	Right-side approach	Location/Dining room	"In the dining room I feel better when the robot is in my right side"
Personal tastes	Low temperature		"I prefer to wait until the food is cooler"
Information providing	High*	Company/None	"I feel more accompanied when the robot talks while eating alone"
Information obtaining	Only when needed		"I don't like to answer questions while eating"
Cognitive	Remind after lunch pills	Moment/Afternoon	"I don't want to forget to take my medicine"

Table 3.4: Examples of Aunt Mery's preferences for the feeding task.

3.5 Summary

In this chapter, we have presented a taxonomy of preferences for assistive scenarios. The taxonomy allows categorizing the preferences the user may have regarding the behavior of the assistant during the task that is to be carried out. The preferences are first divided into the decision-making and configuration categories, depending on whether they are used to choose which action to perform or configure the action that is being executed. Moreover, some of the preferences, called “modifier preferences”, are used to modify the applicability of other preferences. Finally, redundancies in the expression of the preferences are avoided with the definition of global preferences and task-specific preferences. The taxonomy can be useful to define user preferences in a structured way, which can then be used for assistive robotics applications, and more specifically, for those entailing physical interaction. We have exemplified the use of the taxonomy with a user persona whose preferences have been explicitly defined for the tasks of feeding and dressing.

With the proposed taxonomy, have defined what is considered as a preference, and it allows for richer adaptation during the User Tailoring and Execution tuning phases of the FUTE framework defined in Chapter 2. The taxonomy would be instantiated with the user preferences during the User Tailoring, while the Execution tuning would use them to modify the user behavior.

4
Planning techniques for robot behavior adaptation

Having defined in the previous chapter the kind of preferences we will consider, we will now focus on how can such preferences be used to model the user and modify the robot behavior to suit such preferences, adapting to the user over time even when the user preferences were misinterpreted.

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 (FIS), 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 a wrong inference of the initial user model. Finally, some insights in a non-simulated real-world shoe-fitting setup are also provided.

This work has been published in [13].

4.1 Introduction

The adaptation of the robot behavior to a user requires to know some details of the user to which they adapt to. Then those details must be used to guide the robot actions to those that best suit the user. Therefore, many challenges are involved in the adaptation process as the user may not be able to assess which are their preferences or needs and, even if it does, those preferences may be over or underestimated. This means that the robot does not only need to behave according to the preferences but also re-adapt them in case of an imprecise setting of the values. Moreover, how to acquire and use such user preferences still remains an open question.

In this chapter, we propose a method to obtain the actions preferred by the user to then

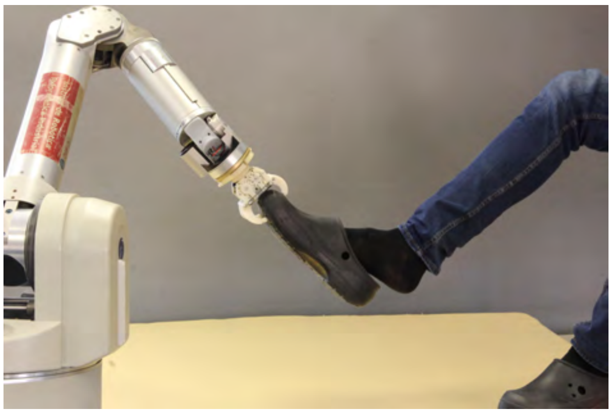


Figure 4.1: Shoe-fitting example.

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 (see Section 2.3) 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 as the one shown in Figure 4.1, 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 Section 3.3. 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.

4.2 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, where robots help people with disabilities or age-related issues [55]. 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.* [30] 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 proposed an online iterative path optimisation method [56]. 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.* [7, 57] use a Baxter robot to put on a sleeve of a jacket to a wooden mannequin. They analyze combinations of user pose and clothing types to detect dressing errors such as cloth snagging and use 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.* [58] 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.* [59] 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.* [31], 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 [30] and [31], 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.* [60], 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 [61, 62] 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.* [25] 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.* [63] 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 [64], 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 by Knox and Stone [65], the human trainer interactively shapes the agent's policy by providing reinforcement signals. A different approach is the one by Griffith *et al.* [66], in which the policy is shaped directly by human feedback rather than using such feedback as a shaping reward.

4.3 User-oriented task planning

Assistive tasks such as shoe-fitting tend to be complex (see Section 1.3). 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. Moreover, as they are intended for users in need of assistance, such users 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.
- $R : S \times A \rightarrow \mathbb{R}$: the reward function.
- $\gamma \in [0, 1)$: discount factor for future rewards degradation.

With this representation, the planner finds a policy $\pi : S \rightarrow A$ 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) [67]. 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 Noisy Indeterministic Deictic rules (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.

In addition, we propose to include the user preferences as a part of the planning domain in the form of expected behavior of the robot (see Section 4.3.1). For example, the ones we have used in the experiments and extracted from the taxonomy of Section 3.3: maximum expected velocity and degree of verbal interaction (Section 4.4). 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 [68], 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 Figure 4.4. Moreover, a similar method is used to obtain a feedback value after each robot interaction, which is used as a scoring method

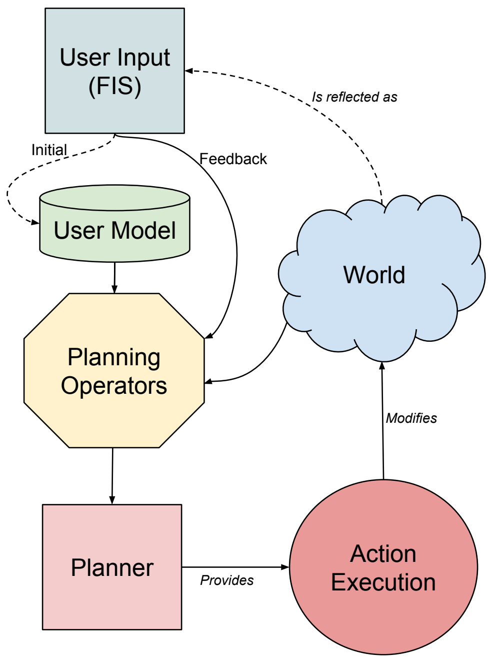


Figure 4.2: System flow representation. Notice the action execution loop from *Planning Operators* - *Planner* - *Action Execution* - *World*, which goes to *User Input* once the task is completed to adapt the *Planning Operators* based on the user feedback.

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.

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 [69, 70], in this chapter 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 the one by Heerink *et al.* [71], would be another useful metric to balance the user feedback.

A system representation is depicted in Figure 4.2. The process is also shown in Algorithm 4.1, where 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.

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

Algorithm 4.1: Preference-based task personalization

```
   – Initialization –
1  userInfo := getUserInfo()
2  userPreferencesPredicates := FIS(userInfo)                      // Section 4.3.2
3  planningDomain := add(userPreferencesPredicates)
4  forall $r \in \{ruleset\}$ do
5      if satisfiesUserModel(r) then
6          updateSatisfyingRule($r$)                    // Use Equations 4.1 and 4.3
7      else
8          updateNonSatisfyingRule($r$)                 // Use Equations 4.2 and 4.4
   – 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 forall $r \in \{usedRules\}$ do
16     updateRuleProbabilities($r$)                             // Use Equation 4.5
   – Update the executed rules based on user feedback –
17 userSatisfaction := getUserSatisfaction()
18 userFeedback := FIS(userSatisfaction)
19 forall $r \in \{usedRules\}$ do
20     if ruleWasSuccessful(r) then
21         updatePenalizations($r$, userFeedback)               // Use Equation 4.6
```

4.3.1 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. This approach does not impede the planner to choose any action but will guide the action selection using the associated costs. Thus, all the reasoning is leveraged to the planner, which is not constrained to select any action. 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 reward), those actions complying with the user model will result in a lower penalization and will be favored by the planner.

We use the shoe-fitting scenario (Section 1.3.2) to explain and test the method. For this shoe-fitting domain, we have used similar actions as the ones in Figure 3.1. Thus, 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 4.3.

- `insertFootInShoe`: 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: insertFootInShoe(F - foot, S - shoe)
Preconditions:
  - reachableFoot(F)
  - shoeInGripper(S)
  - not(footMoving(F))
  - bareFoot(F)
  - correctPose(F)
Effects:
  - shoeInFoot(S, F) & not(bareFoot(F))                (P_os = 0.850)
      -- (Successful outcome)
  - not(footInCorrectPose(F))                          (P_o2 = 0.0625)
  - footMoving(F)                                      (P_o3 = 0.0625)
  - not(inWorkingSpace(F)) & not(reachableFoot(F))     (P_o4 = 0.0250)
```

- `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. The action can be represented with the following NID rule is:

```
Action: inform
Preconditions:
  - not(informedUser)
  - not(askedUser)
Effects:
  -  informedUser  ($P_{o_s} = 1.0$)
       -- (Successful outcome)
```

- `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. The corresponding NID rule is:

```
Action: askUser(F - foot)
Preconditions:
  - not(askedUser)
  - or(not(inWorkingSpace(F)),
       footMoving(F),
       not(footInCorrectPose(F)))
Effects:
  -  askedUser  ($P_{o_s} = 1.0$)
       -- (Successful outcome)
```

Examples of wrong action outcomes are depicted in Figure 4.3. All the actions can be executed either in a *quick*, *intermediate* or *slow* speed, and information may have been given to the user or not before every action execution. The user model predicates are the speed modifier $sm \in \{quick, \ slow, \ intermediate\}$, while the verbosity (information providing) is defined as $vm \in \{verbose, \ not \ verbose\}$. These modifiers relate to the preferences of the user as described in the taxonomy of Chapter 3. So, there are six rules per action, one for each combination of sm and vm. 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

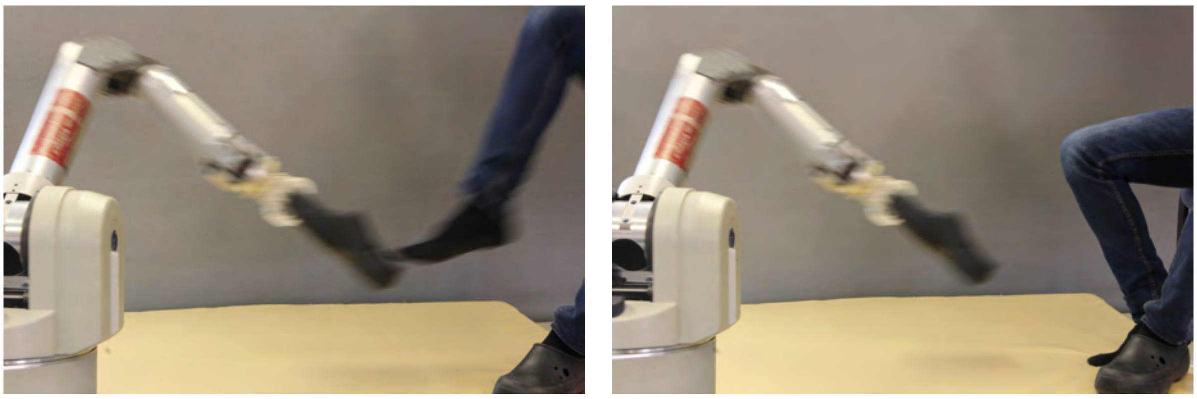

(a) The user moves the foot away from the robot (resulting outcome is `footMoving(right)`).
(b) The user has put the foot aside (resulting outcome: `(not reachableFoot(right))`).

Figure 4.3: Example of shoe-fitting action failures for the `insertFootInShoe` action.

of these predicates are: *reachableFoot(F)*, *footMoving(F)*, *inWorkingSpace(F)*, *shoeInGripper(S)* and *shoeInFoot(S, F)*.

In this chapter, 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 [72]; or similar to the non-primitive tasks of the HTN planning framework [73]. 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 of the FUTE framework) using a learning framework such as the ones presented in [74–77]. 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 Chapter 2. The perception is implemented using an RGB-D sensor such as a Kinect^{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 4.1. Then, if the user was defined as $vm = verbose$ 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 `insertFootInShoe` 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: insertFootInShoe(F)
3: releaseShoe(F)
```

in the non-informative case, and

```
1: informUser
2: approachFoot(F)
3: informUser
4: insertFootInShoe(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: insertFootInShoe(F)
8: informUser
9: releaseShoe(F)
```

4.3.2 Fuzzy user model extraction

As already introduced, we use two simple questions in order to obtain the user traits as an example of how the required initial information can be obtained. This step corresponds to the link between *User Input* and *User Model* in Figure 4.2, and is shown in lines 1-3 from Algorithm 4.1. The answer to the questions is fed to a Mamdani-like Fuzzy Inference System (FIS) [78] built using a simple fuzzy library [79]. The proposed inference system 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 (preferred verbosity).

Another FIS is used to obtain the feedback value, corresponding to the link between *User Input* and *Planning Operators* in Figure 4.2. The feedback computation is also shown in lines 15 and 18 from Algorithm 4.1. 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 4.4 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”.

- Verbosity (*Output*, $[0, 1]$): Includes the terms “yes” and “no”.

- Feedback (*Output*, $[-5, 5]$): Includes the terms “worst”, “bad”, “neutral”, “good” and “best”.

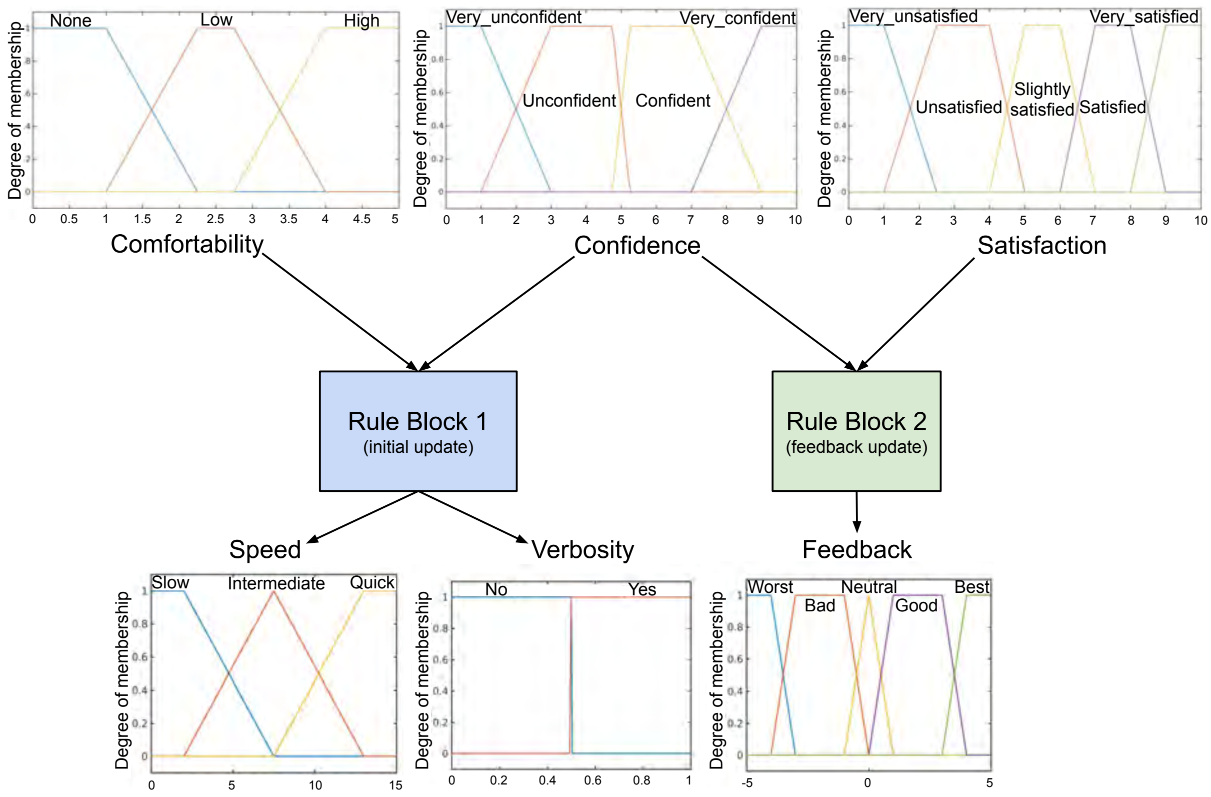


Figure 4.4: Fuzzy Inference System used to obtain the initial user model, as well as its improvement using feedback.

4.3.3 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 4.2. The *Planning Operators* from the figure correspond to the NID rules defined in Section 4.3.1. The initial refinement corresponds to the block comprising lines 4 to 8 in Algorithm 4.1 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

$$P^r_{o_s} = P^r_{o_s} + \frac{1 - P^r_{o_s}}{K}, \tag{4.1}$$

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

$$P^r_{o_s} = P^r_{o_s} - \frac{P^r_{o_s}}{K}. \tag{4.2}$$

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, where K acts as a scaling factor. 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 4.1 to $P^r_{o_s}$, we apply Equation 4.2 to the other rule's outcomes, and vice-versa.

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 $[quick, verbose]$ modifiers will imply a penalization when the user model is defined as $vm = (not\ verbose)$, 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

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

as follows. When the rule is not adequate for the current user model, we update the costs[2] that satisfy the user model as

$$R_c^r = min(R_{max}, max(R_{min}, R_c^r - C)). \tag{4.3}$$

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):

$$R_c^r = min(R_{max}, max(R_{min}, R_c^r + C)). \tag{4.4}$$

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.

4.3.4 Improvement based on user feedback

With the proposed problem definition, the system is able to provide plans that comply with the specified 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 the 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 completed. 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 Algorithm 4.1.

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

Outcome probability update based on executed actions

For the probability update, we use the decreasing m-estimate as defined by [80]. We update the probability of the i_{th} outcome of rule r, P_i^r as

$$P_i^r = \frac{p + \frac{m}{\sqrt{p+n}} P_{i,0}^r}{p + n + \frac{m}{\sqrt{p+n}}}, \tag{4.5}$$

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 of the FUTE framework).

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

$$R_c^r = min(R_{max}, max(R_{min}, R_c^r + C\frac{1}{f})), \tag{4.6}$$

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

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

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 work 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.

4.4 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 $[quick,\ verbose]$, but whose ground-truth real model is $[slow,\ verbose]$, 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 4.2 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 [80] to adapt the probabilities based only on the outcomes of the actions. Similarly, the second one uses the feedback update from Section 4.3.4 (Rule penalization update based on user feedback), 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 4.3.3

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 4.5 shows the results of a user who behaves as $[slow,\ verbose]$, but whose inferred model from the initial questions was defined as $[quick,\ not\ verbose]$. 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

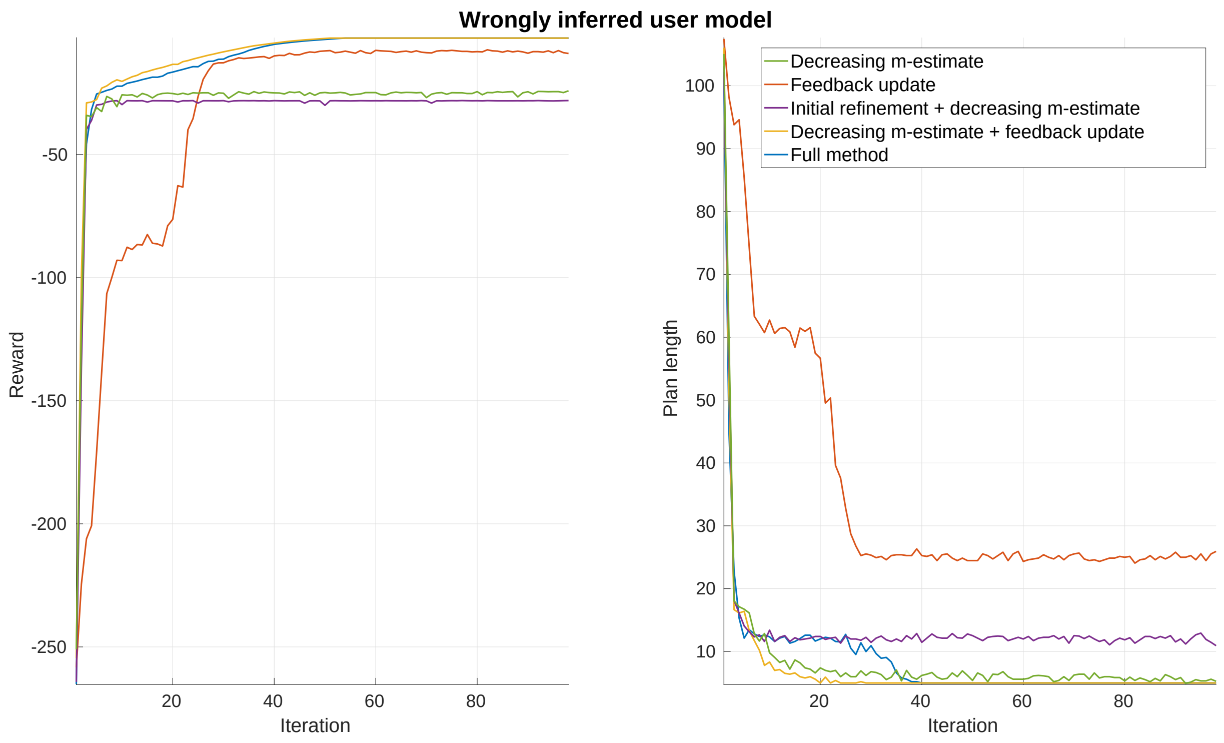


Figure 4.5: Evolution of the rewards and plan lengths using different method combinations. The inferred user model is $[quick,\ not\ verbose]$, but the simulator behaves as a $[slow,\ verbose]$ user.

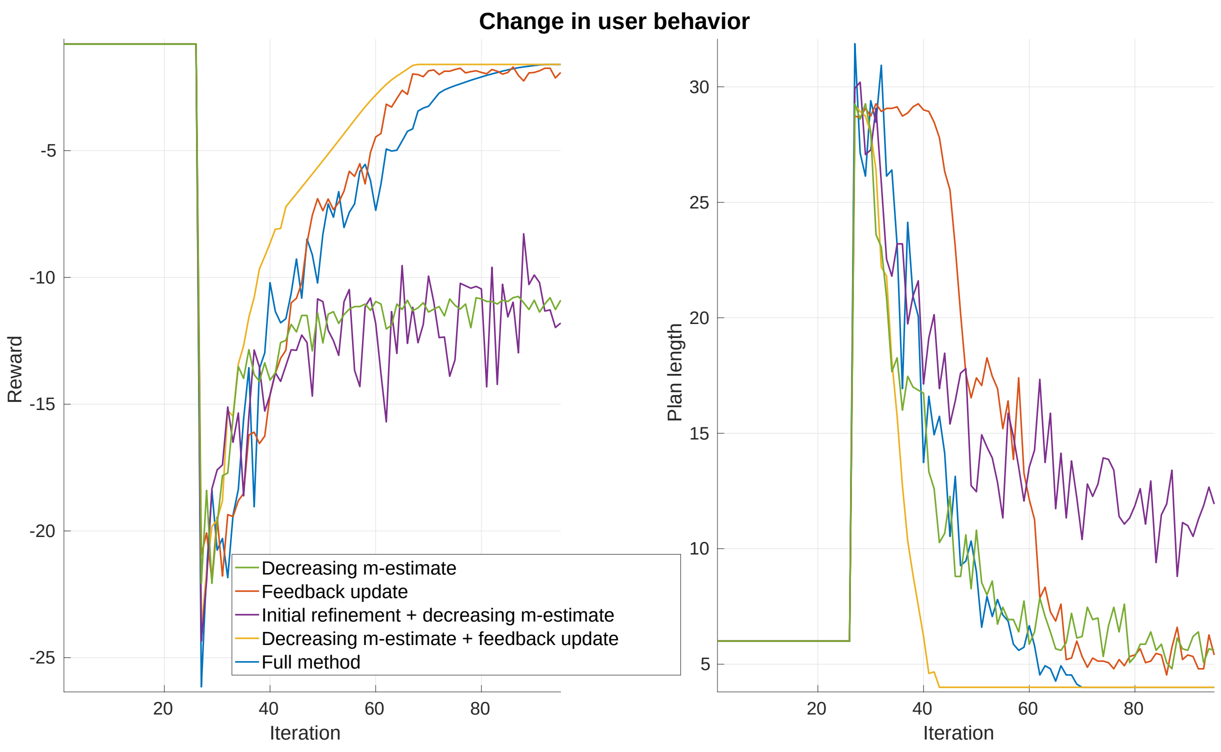


Figure 4.6: Evolution of the rewards and plan lengths using different method combinations. The inferred user model and behavior is $[quick,\ verbose]$, but in iteration 27 the behavior changes to be $[quick,\ not\ verbose]$.

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 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 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

[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.

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 4.6 shows an example of adaptation when the user suddenly changes his/her behavior. In this case, the user was behaving as $[quick,\ verbose]$, 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 $[quick,\ not\ verbose]$ 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 where 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.

4.4.1 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 is shown in Figure 4.7. The experiment presented in this section is meant to demonstrate the usefulness of the proposed approach and evaluate the system. Therefore, the person was instructed to perform some motions to obtain a more realistic scenario for the evaluation. Refer to Chapter 7 for a user-centered evaluation. The robot is a 7 degrees of freedom Barrett® WAM Arm and a Kinect™ camera mounted on the ceiling is used for perception. The software has been developed using the Robot Operating System (ROS) [81]. 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.

The environment state is obtained by processing the point cloud retrieved from the RGB-D camera. Given the presented setup (see Figure 4.7), the point-cloud P is first split 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

[5] ROS package hmi_robin: wiki.ros.org/hmi_robin

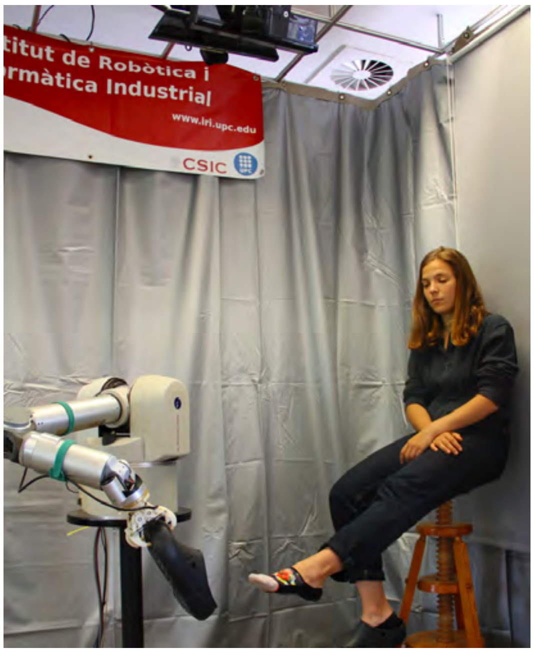
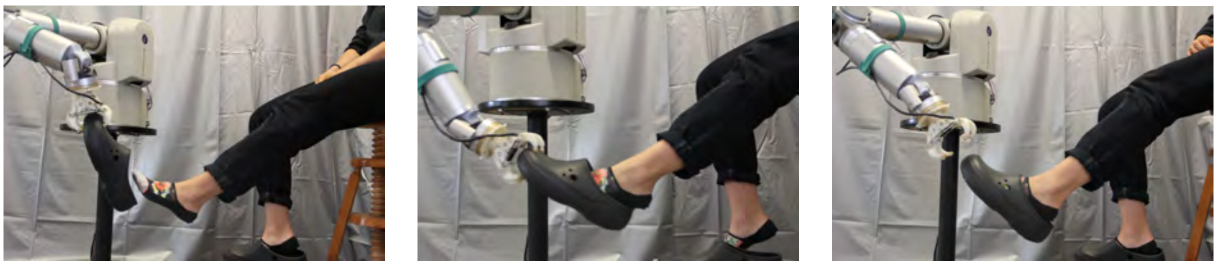


Figure 4.7: Experimental setup with the robot, a user and the ceiling camera.

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 foot tip. 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 4.3) 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 recomputed 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 4.8

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



(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.

Figure 4.8: 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 Figure 4.7).

shows the robot executing the three shoe-fitting task actions[7].

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 non-optimized computations (vision, trajectory generation and planning).

4.5 Summary

In this chapter, 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 a 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

[7]A video demonstration of the shoe fitting task showing the robot adaptation to the user can be found at www.iri.upc.edu/groups/perception/plannedBehaviorAdaptation

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 have shown how the system can 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.

The proposed method uses some preferences from the taxonomy described in Chapter 3 and introduces a way for preference elicitation from the user while using the adaptation framework presented in Chapter 2. We have shown how the Factory setting phase is used to set the default parameters, and how the User Tailoring and Execution tuning phases of the FUTE feed each other back. While the methods presented in this chapter allow for better adaptation, there are still some gaps to be filled on how to further improve the adaptation by taking advantage of low-level adaptations and also how to improve the use of known preferences before adapting to the user changes. These topics will be further analyzed in the following chapters, where "smart" controllers able to adapt to user movements will be joined with our planning approach to ease the definition of assistive skills, and a novel preference suggestion algorithm will be presented.

5 Joining high-level actions with low-level skills

For a safe and successful daily living assistance, far from the highly controlled environment of a factory, robots should be able to adapt to ever-changing situations. Programming such a robot is a tedious process that requires expert knowledge. An alternative is to rely on a high-level planner, but the generic symbolic representations used are not well suited to particular robot executions. Contrarily, motion primitives encode robot motions in a way that can be easily adapted to different situations. In this chapter, we present a combined framework that exploits the advantages of both approaches. The number of required symbolic states is reduced, as motion primitives provide "smart actions" that take the current state and cope online with variations. Symbolic actions can include interactions (e.g., ask and inform) that are difficult to demonstrate. We show that the proposed framework can adapt to the user preferences (in terms of robot speed and robot verbosity), can readjust the trajectories based on the user movements, and can handle unforeseen situations. Experiments are performed in the shoe-dressing scenario. This scenario is particularly interesting because it involves a sufficient number of actions, and the human-robot interaction requires the handling of user preferences and unexpected reactions.

This work has been done in collaboration with IDIAP[1] as part of the I-DRESS project. It has been published in [14].

5.1 Introduction

Physical Human-Robot Interaction (pHRI) is a special case of HRI in which safety becomes a central issue due to the possibility of potentially causing harm to a human user[2]. Therefore, Physically Assistive Robot need to be equipped with two basic skills: *compliant control* to ensure safe motion, and *planning* taking the user into consideration to foresee possible problems and

[1]www.idiap.ch

[2]See Appendix B for more details on safety for Physically Assistive Robots.

deviations in the execution of the task.

Accordingly, adaptive interaction is a key element for the success and acceptance of assistive robots. Some of the challenges that need to be solved are how to transfer such skills to robots in an easy manner, and how to enable robots to cope with user reactions and other issues that may happen during the task, ensuring a robust and safe behavior. In this chapter, we jointly tackle both issues. First of all, we improve the use of learning by kinesthetic demonstration approach to teach the robot “smart” low-level movement primitives, so that it can track the user state and move accordingly. Then, following the adaptation introduced in the previous chapter, a stochastic symbolic planner is used to obtain the sequence of actions to complete the task. If instead, the high-level task planner were used without the low-level primitives, a higher action granularity and more implementation effort would be needed, and tackling the whole problem with the low-level primitives without task planning would require a larger number of demonstrations. Thus, the main advantages of the proposed joint approach are a reduced number of easier demonstrations and less symbolic actions with better error handling and robustness.

As an example, we will use the shoe-fitting scenario again (see Section 1.3.2), where a robotic arm has to put a shoe on a user’s foot. We will use the same fitting actions for the robot, adding shoe grasping. The full set of actions are shoe grasping, approaching the foot, fitting the shoe, and releasing it. Though simple, the task involves physical contact with the user’s foot, which may be harmful. Accordingly, the robot must take this into account, know how the user may behave and suitably adapt its actions to fulfill the task successfully, recovering from inappropriate situations when needed.

5.2 Related work

Service robots in general, and the assistive ones specifically, must perform complex tasks with many particularities. Therefore, joining a high-level symbolic task planner with appropriate low-level motion primitives simplifies the task.

There are many works in literature that address task and motion planning, in a consecutive or integrated way, but most of them focus on the manipulation of still objects, while in our case, we are physically interacting with a person that may move freely.

Gravot *et al.* [82] present a collaborative cooking task with a robot in which a symbolic HTN planner uses cooking recipes to guide the performance of the task and decomposes them into primitive actions, which may be sensing, making specific movements, planning motion or interaction.

Other authors address motion planning as a geometrical problem. De Silva *et al.* [83]

propose an interleaved interface to perform symbolic task planning and geometric planning. The geometric planner is used to compute possible grasps and object locations, taking into account geometric constraints. The symbolic planner is an HTN, and symbolic tasks are related to their Geometric Task Planner's tasks, for which they propose an interleaved backtracking algorithm. The authors apply it in the context of pick-and-place tasks, including human handovers. In a similar manner, Srivastava *et al.* [84] first compute a task plan using a symbolic planner, and then search the instantiations of the pose references used in the plan by means of a motion planner. When such instantiations are not found, partial solutions are identified and extended using the task planner. Another example is the one by Ferrer-Mestres *et al.* [85], where they integrate task and motion planning together, addressing the symbolic and geometrical components of the task simultaneously. Furthermore, Bidot *et al.* [86] present an hybrid task-and-motion planning approach in which task planning is coupled with motion planning and geometric reasoning. Lee *et al.* [87] combine probabilistic activity grammars with low-level motion primitives to learn tasks with reusable structures. Kinesthetic teaching has also been used to learn how to execute structured tasks, such as in the framework presented by Caccavale *et al.* [77].

The adaptive component is essential to solve our kind of tasks in an efficient manner, as well as the use of learning by demonstration to simplify the teaching of the movements. Assistive dressing, the problem we are focusing on in this chapter, has also received some attention from the community. Some of the works devoted to assistive dressing have already been mentioned in Section 4.2, such as the ones by Yamazaki *et al.* [58], Gao *et al.* [30, 56], Klee *et al.* [31], and Chance *et al.* [7, 57]. These works present interesting approaches to model the user space and capabilities, but lack the ability to demonstrate the task in an easy manner that produces smoother movements and shows a nice adaptability to the user movements.

5.3 Planning for the next step

As seen in the previous chapter, physical Human-Robot Interaction tasks such as the ones defined in Section 1.3 are appropriate to be tackled from a task planning point of view. Since the robot is in contact with a user, single reactive behaviors may not be enough to deal with user actions in the long-term horizon of the task. Therefore, it is important to have a plan of the robot's actions that should be performed, solving plan deviations as soon as they occur.

A task planning problem $\Pi = \langle S, A, T, s_0, g \rangle$ is defined by the set of discrete states S, the set A of actions that modify the state, the state transition function $T : S \times A \rightarrow S$, the initial state $s_0 \in S$ and the goal state $g \in S$. A solution to this problem is an action sequence that starts from s_0 that modifies the state using actions in A to end up in g. Each state is described by a set of logic predicates, and each action $a_i \in A; a_i = \{p_{a_i}, e_{a_i}\}$ is composed of the preconditions

$p_{a_i} \in S$, predicates that must be true in order to perform the action, and the effects $e_{a_i} \in S$, which define how the state changes after the execution of the action. Such state is obtained from the internal (known) robot state, but also from a visual system tracking both the user and other objects related to the task.

Since HRI domains are highly non-deterministic as the user is not a controlled agent, the computed plan should consider unexpected effects. For this reason, we rely on a stochastic representation, which allows the definition of a domain in which actions can yield stochastic effects. In this case, the actions' effects e_{a_i} are not just a unique set but many possible sets of outcomes with an associated probability π_j of happening:

$$e_{a_i} = \begin{cases} \pi_1 : e_{a_i}^1 \\ \pi_2 : e_{a_i}^2 \\ \vdots \\ \pi_n : e_{a_i}^n \end{cases} .$$

As mentioned in Section 4.3, this kind of problems are usually represented formally as a Markov Decision Process (MDP) $\langle S, A, P, R, \gamma \rangle$, where $P(s'|s,a)$ defines the probability of going from state s to s' when performing action a, and $R : S \times A \to \mathbb{R}$ is the reward function associating a score to each action. Then, the planner is set up to find an action sequence that maximizes the reward (or, equivalently, minimizes the cost), while taking into account the probability of each action's outcomes. An application example of this type of planning applied to robotized surface cleaning is provided in [80].

Therefore, we can define the actions so that their possible outcomes are based on user reactions, and the planner will compute a plan by considering the probabilities of each effect. As a result, actions that may produce inconvenient outcomes will be less likely to be selected.

Each symbolic action a_i corresponds to one low-level movement primitive (i.e. physical action). In this chapter we will use the method from [75] to learn the motion primitives. These movement primitives are self-adapting motions that can track entities of interest for the task, such as the foot in a shoe-fitting scenario.

Once the plan $P = [a_i, a_j, \dots, a_k]$ has been computed, each action is sequentially carried out. However, there may be cases in which an action produces a non-satisfactory outcome. In such cases, replanning is needed in order to find a new sequence of actions P that brings from the current system's state to the goal state g, and the new plan will be executed.

When interacting with human users, and more specifically when assisting them in a physical manner, it is important to interact with the user, and make clear what the robot is doing. For this reason, we define two interaction actions: inform and ask. The **inform** action is used to provide verbal information to the person before the execution of each movement to avoid misunderstandings or unexpected situations due to the user misinformation about the robot behavior. The **ask** action is used to obtain user's collaboration in cases in which the task cannot be completed. For instance, in the shoe-fitting scenario, the robot would ask the user to put back the foot, when he/she has moved it to an unreachable position, or to avoid moving it when trying to perform a physical contact.

Not less important while interacting with human users, is to adapt to the specific user the robot is assisting. There are no two equal individuals, so the robot should not assist all the users in the same way, but adapt to their preferences and needs. Following our taxonomy of user preferences in assistive scenarios as defined in Section 3.3, in this chapter we will still consider two kinds of preferences: the velocity of movements, and the verbosity level.

Performing the task too slowly may result in user fatigue, and doing it too fast may scare the person. Similarly, a too verbose robot may irritate the user, and a non-informative robot may confuse and scare the user. To cope with these dilemmas, we have defined three speed levels $\alpha \in \{slow,\ medium,\ quick\}$, and two verbosity levels $\beta \in \{verbose,\ \neg verbose\}$. Although a more complex verbosity configuration could be used, we believe two levels are enough for the shoe-fitting task and the explanation of the proposed methods would not benefit from a more complex setup. Building on the concepts presented in Section 4.3, a user model $u = \{\alpha_u, \beta_u\}$ has been added to the planner by means of the predicates α and β. Each action's reward R_i depends on this user model, penalizing such actions that violate it. For completeness, we define a generic reward penalization equations from Sections 4.3.3 and 4.3.4 in a more compact manner:

$$R_i = R_d - P_\alpha \cdot (\alpha_i \neq \alpha_u) - P_\beta \cdot (\beta_i \neq \beta_u), \tag{5.1}$$

where R_d is the default action's reward, P_α is the penalty for not following the user's model speed α_u in the current action a_i executed with speed α_i. Similarly, P_β is the penalty for a wrong verbosity β_u. This way, the planner not only computes a plan to solve the task, but also does it while satisfying the user preferences.

With the explained approach, the robot is able to compute a plan from s_0 to g that complies with the user model. Moreover, when an unexpected behavior arises, the plan is recomputed and its execution is resumed from the new state. This means that the system is able to recover from errors, repeating previous actions when needed in order to return to a previous state or even starting over the task if required. For instance, in a case in which the shoe-fitting is incorrect, the robot would re-grasp the shoe and start the task if far from the foot, or retry the insertion if the foot is close enough.

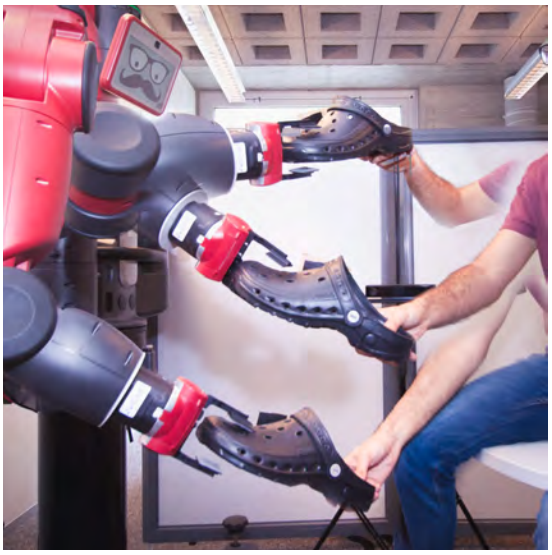


Figure 5.1: By providing demonstrations of a task in several situations, the robot is able to generalize to a wide range of new situations.

5.4 Learned motions

The robot movement is taught by kinesthetically teaching the robot as shown in Figure 5.1. In this chapter, we will use the method presented by Pignat and Calinon [75] where a Hidden Semi-Markov Model (HSMM) is learned by maximizing the likelihood of the demonstration data. The HSMM decomposes the skill into a discrete sequence of spatial distributions. The model encodes the positions of the end-effector, the velocity and orientation of the robot from different coordinate axes. Then, optimal control is used to synthesize the motion of the robot.

The use of this kind of trajectory learning and controller allows for compliant movement of the robot and low-level adaptation to moving targets. The method is able to encode both the position of the end-effector and a moving target such as the foot and learn the motion considering both reference frames, which makes it possible to adapt to changes online while still following the learned trajectory.

5.5 Combining high-level Symbolic Task Planning with low-level Motion Planning

In the previous sections we have defined both system levels and given insights on how they are related. The proposed architecture is depicted in Figure 5.2. The symbolic planner's actions have a direct correspondence with the low-level motion primitives. The high-level planner computes the sequence of actions. Then, the low-level primitives, previously learned by demonstration, are executed.

The strength of this approach is to overcome the limitations of using the two levels separately. Although the same task could be tackled from both points of view, the necessary efforts are highly reduced by the union of both.

On the one hand, performing the full task with the sensorimotor motion primitives would require several demonstrations of all the possible events that may happen throughout the task. This results in the need of designing such demonstrations in a thorough manner, taking into account every case in the scenario.

On the other hand, using a symbolic planner to perform the same task would require to split each of the actions in subactions, obtaining a finer granularity. This not only introduces an overhead in the plan computation (as the domain will have many more actions and outcomes), but also requires the design of the domain such that all the possible outcomes of each action are properly defined. Moreover, this also needs us to implement the control and movements of each action in the robot, be it by demonstration or with another technique.

Therefore, by using both approaches together, we obtain a simpler and more efficient planning domain which is easier to design, implement and debug. Such approach requires only few demonstrations of the full task. Typically, the demonstrations take the form of simple atomic movements instead of complete movements. Moreover, such structure facilitates the addition of verbal interactions and the handling of errors by means of replanning. It provides a way to link high-level actions to low-level control commands, facilitating the modulation of low-level actions and the gradual acquisition of complex skills.

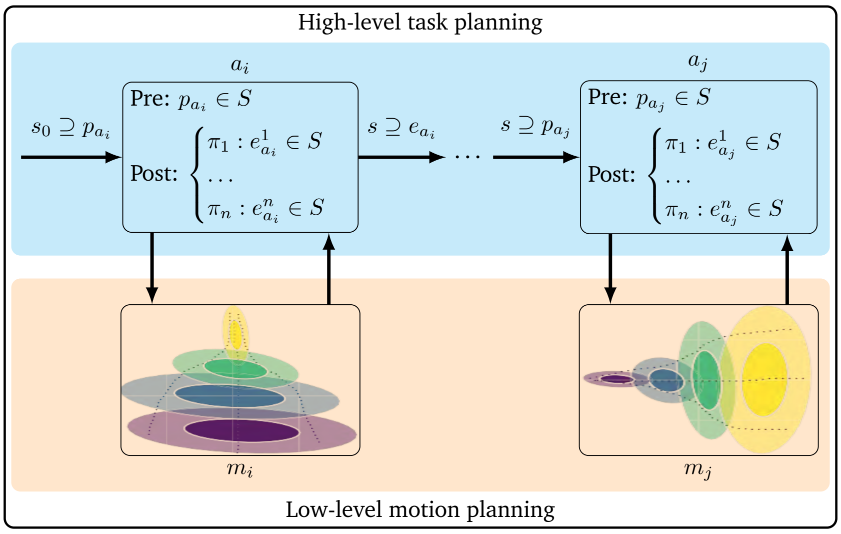

High-level task planning
a_i
Pre: $p_{a_i} \in S$
$s_0 \supseteq p_{a_i}$
Post: $\begin{cases} \pi_1 : e^1_{a_i} \in S \\ \dots \\ \pi_n : e^n_{a_i} \in S \end{cases}$
$s \supseteq e_{a_i}$
$\dots$
$s \supseteq p_{a_j}$
a_j
Pre: $p_{a_j} \in S$
Post: $\begin{cases} \pi_1 : e^1_{a_j} \in S \\ \dots \\ \pi_n : e^n_{a_j} \in S \end{cases}$
m_i
m_j
Low-level motion planning

Figure 5.2: Two-level planning architecture.

Given that the learned motions are adaptable and they do not depend on fixed start and end positions, they are always chainable provided that the high-level actions are also chainable.

5.5.1 High-level state transitions: the shoe fitting example

In order to demonstrate the advantages of the architecture, we show in Figure 5.3 some of the action transitions that are possible in the shoe-fitting task, following up with the same action definitions as in the previous chapters. As it is clearly seen in the graph, the structure of the transitions between actions is quite complex, despite the task requires a low number of actions. Therefore, teaching all the possible transitions by demonstration would result in a tiresome –if not unfeasible– work due to the large number of demonstrations that would be needed. This can be observed in Figure 5.4, were several demonstrations with varied shoe positions and orientations are performed to teach the shoe grasping action. Moreover, the use of the planner permits an easy inclusion of interactive actions such as informing and asking the user, which would have been much harder using a different approach. Thus, the use of the task planner in the high-level loop allows to reuse simple low-level movement primitives, and diminishes the number of demonstrations to just a few demonstrations for each high-level action.

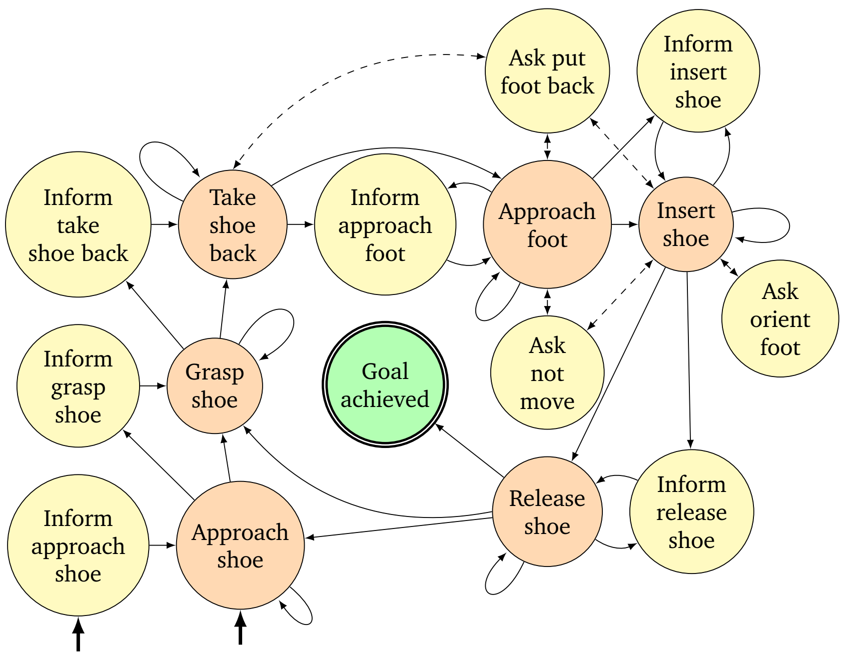


Figure 5.3: Example of action transitions in the shoe-fitting task. Orange nodes represent robot motion actions, while yellow nodes correspond to interaction actions.

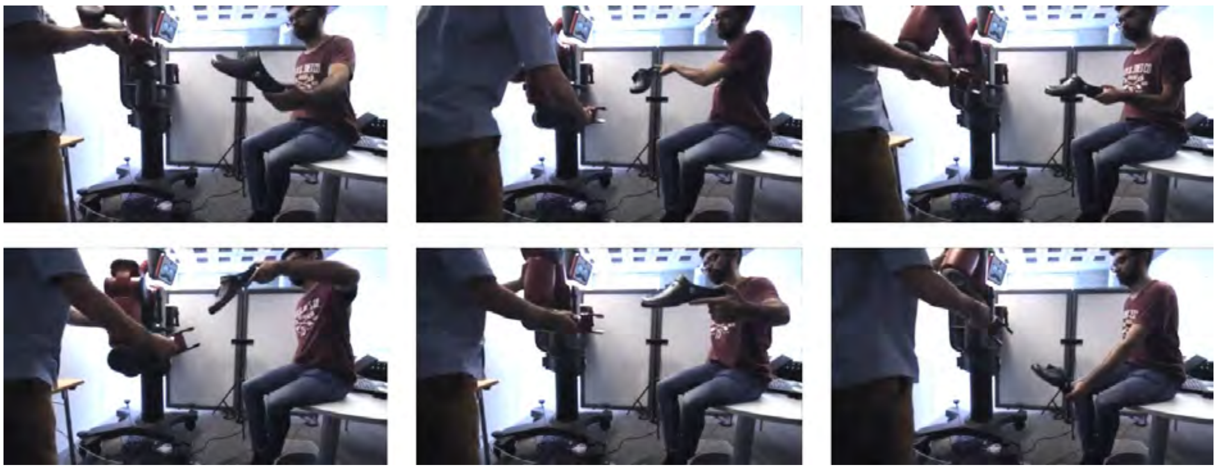


Figure 5.4: Kinesthetic teaching requires repetitions of demonstrations to cope with variability and be able to adapt to changes.

5.6 Experimental evaluation

The proposed two-level architecture has been implemented using a Rethink Robotics' Baxter robot, in a shoe-fitting application. We have used an RGB camera to detect the shoe and the foot of the user. To simplify the perception, augmented reality markers attached to the shoe and the ankle have been used. This allows keeping the symbolic state updated based on real observations.

This section reviews some of the experiments, more details and a visual demonstration of them can be seen in the video[3]. The purpose of these experiments is to demonstrate the feasibility, usefulness, and features of the proposed approach. They have been performed by a single user who was instructed to act in some designed ways for the sake of the demonstration in a more realistic scenario. For instance, the user was asked to make some action fail or move the foot in some moments. For a user evaluation on the tasks used in the thesis refer to Chapter 7.

5.6.1 Experiment 1: Failure recovery after task completion

To demonstrate how the proposed approach is able to cope with non-demonstrated events, we show an experiment in which the user removes the correctly fitted shoe from the foot, as shown in Figure 5.5. The user removes the shoe after the fitting, resulting in an incorrect fit (red node). Once the perception system has updated the state of the environment, the planner is able to detect the situation, and recompute a plan to solve the task by grasping the shoe again. Then, the planner decides to fit the shoe if the foot is still close, or to take it back and approach

[3]See the video at www.iri.upc.edu/groups/perception/twoLevelDressing

the foot again in case the user has moved the foot away. Notice that all the high-level actions and low-level primitives used in this scenario are the same ones that were implemented for the original task. For instance, the shoe grasping action is used either to get the shoe from the user or to pick it up from the foot after the bad positioning of the shoe. Several kinesthetic demonstrations of the situation would have been needed to obtain the same behavior without the high-level task planner.

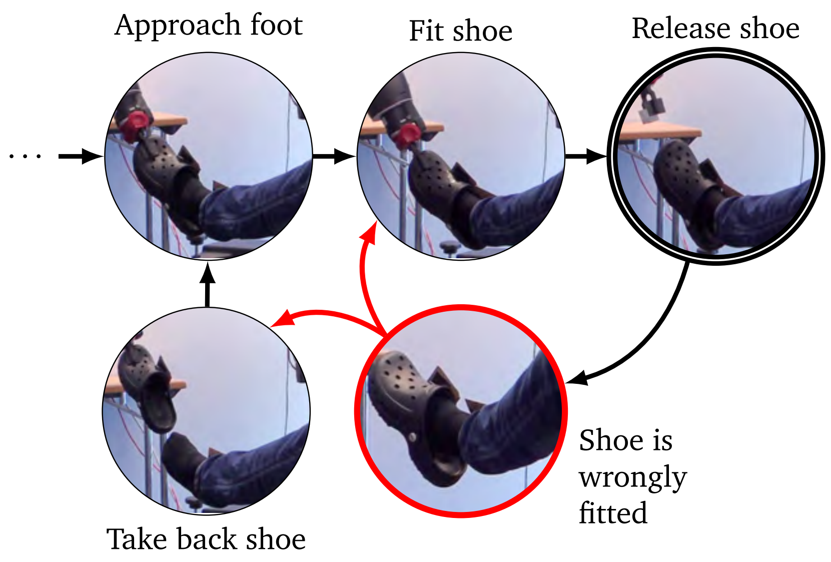


Figure 5.5: Example of error state management with the high-level task planner in a non-demonstrated situation.

5.6.2 Experiment 2: Talking to the user when needed

Here we show how the interaction works. As already introduced, the robot has two interactive actions: ask and inform.

In a normal execution, the robot may be completely silent, or inform the user if it is defined in the user model. The interesting part is when the execution does not go as expected. When fitting the shoe, the user may become tired or scared, resulting in him/her removing the foot from the robot's working space. Then, in order to complete the task, the robot must *ask* the user to put the foot back in the fitting space, or it may *ask* him/her to reduce the motion of the foot for a correct and safe fit, as shown in Figure 5.6. Another case may be one where an action keeps failing because of different user reactions. In such case, the robot *informs* the user about the actions it is performing. With this, the robot may gain user's trust and avoid misunderstandings during completion of the task, even when the model of the user does not take speech into account.

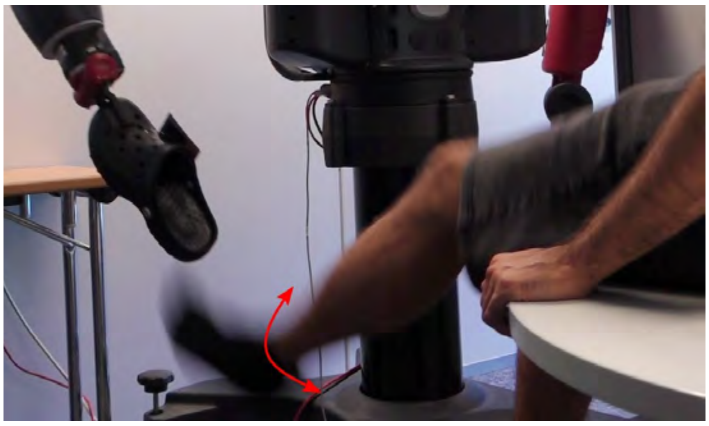


Figure 5.6: Example of user moving the foot. This will trigger an *ask* action to stop the movement, and may also produce a speed change.

5.6.3 Experiment 3: Speed modulation

As already stated, there are many reasons to modify the speed of the actions, among other aspects of the task. In a first case, the robot makes a sudden quick movement that scares the user, as seen in Figure 5.6. In this situation, the user performs a reflex action that moves away the foot, preventing the task to be accomplished. In another case, the robot moves too slowly resulting in fatigue that causes the user to remove the foot from the robot's working space. Such cases are tackled by the high-level planner, which takes into account the speed of the actions and the user model to compute a scaling parameter for the low-level primitives.

5.7 Summary

In this chapter, we have further expanded the methods explained in the previous chapter by adding "smart" low-level primitives able to adapt to the user online while the robot is moving. We have joined them with planning methods in a two-level approach to develop assistive robotics applications. In the high-level, the task planner computes a sequence of actions that will bring the robot to the completion of the task. The low-level reproduces the actions, which are learned by demonstration beforehand, in such a way that they are able to adapt to the current situation by tracking entities of interest for the task.

The proposed approach reduces the number of demonstrations that are needed to implement the task in the robot and makes them simpler and easier to teach. It also lowers the number of symbolic actions that are needed, as well as diminishes the number of replanning attempts. Moreover, it provides better error handling, resulting in a more robust and safe[4] task execution.

[4]For more details about safety refer to Appendix B.

6
Preference suggestions for improved performance

We have shown how smart motion primitives can be used along with the symbolic task planning to improve the adaptation of the robot in assistive tasks, along with adding robustness and easier development. User model adaptation was also explored in Chapter 4 where the taxonomy presented in Chapter 3 was used. However, the preferences described in the taxonomy can be further exploited, as well as the elicitation of new preferences. To this end, this chapter will present the SoPS algorithm to provide suggestions of predicates of a planning problem. We will apply the algorithms to the specific case of planning with user preferences for assistive scenarios, exploiting the preference taxonomy and showing how suggesting preferences can improve task performance in the solution of a planning problem.

This work has been published in [15, 16]

6.1 Introduction

Artificial Intelligence Planning has proved useful to solve many problems in Robotics and Computer Science. Planning systems were traditionally handled by experts in the field, but this trend is now changing as technology evolves and gets closer to lay users. Therefore, as robots and complex decision-making systems enter our homes, a need for communication and explanation of the reasons behind the system's decisions arises.

Suggestions are an example of this kind of communication. A non-expert user may not know all the possible configurations the system may have. Hence, the system itself may suggest potential configurations that can improve its performance. Going even further, it can suggest configurations that can pair with the user-provided one and still improve the system's outcome.

Such suggestions can also be used for explanation purposes. In this case, the system could use suggestions that improve the performance to explain why the performance of the system was not good. Or it can use such elements to show why a specific configuration is better than a different one, which would perform worse.

In this chapter, we analyze the case of providing suggestions for predicates in planning domains. Suggestions are predicate assignments that improve the plan reward, such as preferences over the task execution. As an example, such predicates can be the desired speed in a robotics task. We propose two algorithms capable of providing suggestions. The first one finds out values for unassigned predicates that produce better plans; the second one proposes reasonable changes to already assigned predicates by suggesting values close to the current ones. To do so, our algorithms process a portion of the Space of Plans in search of the best assignment of values to predicates. This subset of the Space of Plans is expressed as a new Plan Space Tree structure that provides a compact representation very convenient for searching and traversing. Then, we demonstrate the ability of the methods proposed to improve the reward obtained by the planner, even when low-performance configurations are initially provided. The methods are evaluated in the three assistive robotics tasks defined in Section 1.3 (shoe-fitting, user feeding and assisted dressing), where the suggestions relate to user preferences that the planner uses to guide the search.

The development of the suggestion methods presented in this chapter required a more standardized approach to task planning with robots. ROSPlan [88] is a framework for task planning in the Robot Operating System (ROS). It simplifies the use of different planners by providing an interface to them and parsing their outputs, as well as having an integrated action dispatcher. ROSPlan has become a standard tool for AI planning in robotics, and we decided to adopt its use for the remainder of this thesis. However, ROSPlan lacked support for any kind of probabilistic planners and, given that probabilistic planning is very useful for handling uncertainty in planning tasks to be carried out by robots, we will first extend ROSPlan to handle such kinds of problems. Moreover, the PPDDL-based approach used beforehand will result insufficient to handle the kind of expressivity we need for our suggestions approach. Therefore, we explored the use of RDDL, which is the standard language for probabilistic planning as it was used in the last editions of the International Probabilistic Planning Competition (IPPC) and most adopted by the community. Therefore, our extension will also integrate the use of RDDL apart from PPDDL.

6.2 Related work

Our proposed work is closely related to and inspired by different topics. We build on top of the concept of planning “excuses” [89], which are defined as the changes needed in the state to find a solution when no plan could be found. This concept was explored by Martínez *et al.* [90] to guide a human teacher when no plan was found. These excuses were also used to find alternative models to explain unexpected states. Similarly, we seek the predicates that can improve the planning performance and provide them as suggestions to the user.

Therefore, our proposed methods are related to the concept of human-aware task planning and human-in-the-loop planning, where the planner takes into account both the human and the robot's actions and abilities to improve task performance. Alami *et al.* [91] proposed a scheme to integrate humans in the robot control architecture. In it, the abilities and constraints from the users, their needs and their preferences are taken into account in the planning process. Cirillo *et al.* [92] proposed a planner able to take into account forecasted human actions that constrain the planner allocation of tasks to the robot, but also create new robot goals. Sekmen and Challa [93] developed a Bayesian Learning algorithm for a robot to model the behaviors and preferences of people. The model is updated based on continuous interactions with the users, which is then used to predict the expected user actions. The authors consider user preferences regarding the scheduling of user tasks and personal customs such as drinking patterns. Young Kwon and Hong Suh [94] predict exogenous events due to human intervention and create plans proactively that improve Human-Robot Interactions. Their method can select when is the best time to perform proactive actions, which results in better interaction and a reduced task execution time. Other examples include the Hierarchical Agent-based Task Planner (HATP) [62], where agents are taken into account as first-class entities and user-defined social rules describe the acceptable behaviors of the agents, allowing the creation of plans that take the user safety and comfort into consideration. Fiore, Clodic and Alami [25] presented a system designed to consider human preferences in Human-Robot Collaboration tasks. In it, the robot can assume different roles and plan the actions for the human, to which it suggests which actions to perform, and also acts as a human assistant. Chakraboti *et al.* [95] show how to project robot intentions during plan executions to assist Human-Robot Interactions using an augmented reality system. The proposed system can reduce the ambiguity over possible plans during task execution and plan generation. In this system, the robot can combine the plan cost with the ability to reveal intentions to improve interaction and task performance. These works mainly assume that the values of the user preferences are known. Contrarily, in this work we use the concept of suggestions to assess how could the task performance be improved when there are unassigned predicates such as preferences by including a user in the planning process to assign values to such predicates.

Our algorithms are based on analyzing the Space of Plans in search of general predicate suggestions, that is, predicates that are missing but that knowing them would help producing better plans. In this paper, we use the example of preference predicates in assistive scenarios. This could also be seen as a preference elicitation process, where we obtain preference suggestions based on already known values. Das *et al.* [96] propose a method for eliciting preferences from a human expert while planning. Their approach uses Hierarchical Task Network (HTN) planning to identify when and where the expert guidance will be useful and seek expert preferences to improve the planner decisions. In our case, we similarly suggest user preferences based on the planner reward function without the need of recurring to the human expert during the planning process. However, in our approach, the expert is the one generating the reward function that uses the preferences. Search in the space of possible plans has been also used by Kim *et al.* [97] to learn to infer final plans in Human Team Planning. Similar plan trees are used by Shmaryahu *et al.* [98] for network penetration testing.

The use of preferences to guide search has been investigated by other authors too. PDDL3 [99] explicitly integrated preferences in the language. They are represented as conditions that do not need to be true to achieve a goal or precondition, but achieving them is desirable. In contrast, we do not use preferences as conditions but we see them as predicates that guide the search by modifying the associated costs and rewards. Sohrabi and McIlraith review Preference-Based Planning (PBP) in [38], where preferences are used to distinguish plans by the quality and argue for the need for reasoning over preferences when generating a plan, obtaining the most-preferred plan. More examples of uses of preferences include another method proposed by Sohrabi, Baier and McIlraith [100], which generates preferred explanations for the observed behavior of the system. A survey on preference-based Reinforcement Learning by Wirth *et al.* [101] reviews works using preference-based reward functions obtained from experts, and a Monte Carlo Tree Search algorithm using preferences to guide search can be found in [102]. Preferences are also used to guide search in BDI agents by Visser *et al.* [103]. Behnke *et al.* [104] present the interaction between the user and the planner as a process to determine user preferences towards the plan. Another Reinforcement Learning algorithm that benefits from the use of preferences is presented by Pinsler *et al.* [105]. Their method learns the reward functions from the robot and human perspectives (user preferences).

Finally, we also found inspiration from the Explainable AI (XAI) and Explainable AI Planning (XAIP) communities. In XAIP [106], the goal is to present the user with explanatory answers to questions regarding action selection, action alternation, efficiency or affordability of the proposed plans. One way of answering such questions is by proposing alternative plans, by replanning from a user-provided state. We find our algorithms closely related to XAIP in the sense that our Space of Plans representation can be used for providing explanations, but the provided suggestions to predicates can also be helpful to present alternative solutions to the user. Measures like plan explicability and predictability can be computed as described by Zhang *et al.* [107]. Such measures can be used to proactively choose plans that are easier to explain. Another example by Eifler *et al.* [108] propose an analysis for explaining the space of possible plans by using plan properties. These properties are boolean functions that capture the aspects of the plan the user cares about.

Even though there has been a lot of research involving preferences, we believe our proposed method is novel in the use of suggestions for improving task performance for planning and decision making, and the use of preferences is a good example of it.

6.3 ROSPlan extension to RDDL

Planning for robotics means planning in dynamic and uncertain domains, in which the outcomes of actions can have a reasonable chance of failure, or non-deterministic effects, for example in complex manipulation tasks and navigation in crowded spaces. Probabilistic planning is an approach to plan under uncertainty, commonly meaning planning with probabilistic action effects. A probabilistic planner tries to maximize the probability of success of a plan. This section presents a standardized integration of probabilistic planners into ROSPlan that allows for reasoning with non-deterministic effects and is agnostic to the probabilistic planner used.

In many domains it is possible to ignore the probabilistic nature of the environment by generating deterministic plans, and replanning when they fail during execution. However, for some problems it is advantageous to consider the probabilities: for example when there is more than one path to the goal and those paths have different associated rewards and probabilities of success, or the state-space includes dead-end states. Given different paths to a goal, the paths with higher associated rewards might counterintuitively be those that are longer, or otherwise the cost structure might be far from obvious. These kinds of problems are termed *probabilistically interesting* [109]. Robotics domains are often probabilistically interesting. For example, an autonomous robot in a dynamic environment can easily move into a state from which it does not have the capability to recover by itself, requiring human intervention. Therefore, robots are often expected to follow the slower, safer paths to the goal to avoid failure. However, by reasoning over the probabilities during planning, more efficient solutions can be found.

The Relational Dynamic Influence Diagram Language (RDDL) is a stochastic domain description language for probabilistic planning. The International Probabilistic Planning Competition (IPPC) uses RDDL [110] for probabilistic planning problems. RDDL is well-suited to describing probabilistically interesting problems, using a dynamic Bayes net formalism [111], as opposed to the effects-based (P)PDDL. Subsequently, both the first and second-place entries in the 2012 IPPC were planners that actively reasoned with probabilities: PROST [112] and Glutton [113].

The ROSPlan framework [88] provides a standard approach for planning in the Robot Operating System (ROS). Until now, one drawback of ROSPlan is that it has been limited to deterministic and contingent planning, using PDDL2.1 [114], and is not suitable for probabilistic planning. The main contributions of this extension are: *(i)* A standardized integration of RDDL and ROSPlan, enabling the straightforward application of the probabilistic planning in robotic domains using ROS. *(ii)* A demonstration of a mobile robot autonomously generating and executing probabilistic plans using this integration in an extensible RDDL domain. We extend ROSPlan to handle RDDL semantics, produce RDDL problem instances, and interface with any RDDL planner that can be used with the RDDLSim server used in the IPPC. In addition, we extend the action interface of ROSPlan, which handles the execution of discrete actions, to reason with non-deterministic effects. To enable distinction between deterministic and non-deterministic effects, we identify two kinds of propositions: *sensed*, whose truth value can be sensed by the agent during execution, and so can be included within a probabilistic effect; and *non-sensed*, which can only produce deterministic effects.[1]

6.3.1 Background on planning under uncertainty

There are numerous approaches addressing uncertainty in the planning and execution process e.g. conformant planning [115], contingent planning [116] or replanning [117]. Other approaches use machine-learning techniques to decrease uncertainty in the planning problem, e.g. [118] learn probabilistic action models and [119] remove uncertainty in state prior to planning by making predictions based on initially known data. Also, there is work on building architectures that involve reactive components to cope with uncertainties in robotics domains [120]. ROSPlan has been used to perform planning for control of multiple-robot systems running with ROS [88, 121]. However, all of these works focus on purely deterministic planning.

Furthermore, probabilistic planning is a standard approach for planning with uncertainty in robotics. An overview of approaches to probabilistic planning is provided in [122]. The most common approach to planning with uncertainties in robotics is modelling the task as a Markov Decision Process (MDP), optionally a Partially Observable Markov Decision Process (POMDP). In contrast to deterministic planning, notably the PDDL2.1 [114] formalism used so far with ROSPlan [88], robotics scenarios must often cope with exogenous and uncontrollable events [123], which can be easily modelled as POMDPs [124]. Solutions to the MDPs for robotics can form policies with finite horizon [125], adopt a satisficing approach [126], or maximize the probability of reaching a goal state [90]. RDDL [110] is well-suited for modelling problems of this kind. It is a dynamic Bayes net formalism, allowing for unrestricted concurrency. This is an essential component in robotics applications, in which the agent must execute the plan in a physical environment. For example, in multi-robot navigation scenarios in which motion is stochastic from the perspective of the planning model.

Atrash and Koenig [127] note that POMDP planning policy graph solutions are similar to the finite-state machines normally used for control. As a result, it has been applied successfully

[1]The source code of the elements described in this section can be found in the main ROSPlan repository, available at github.com/KCL-Planning/ROSPlan

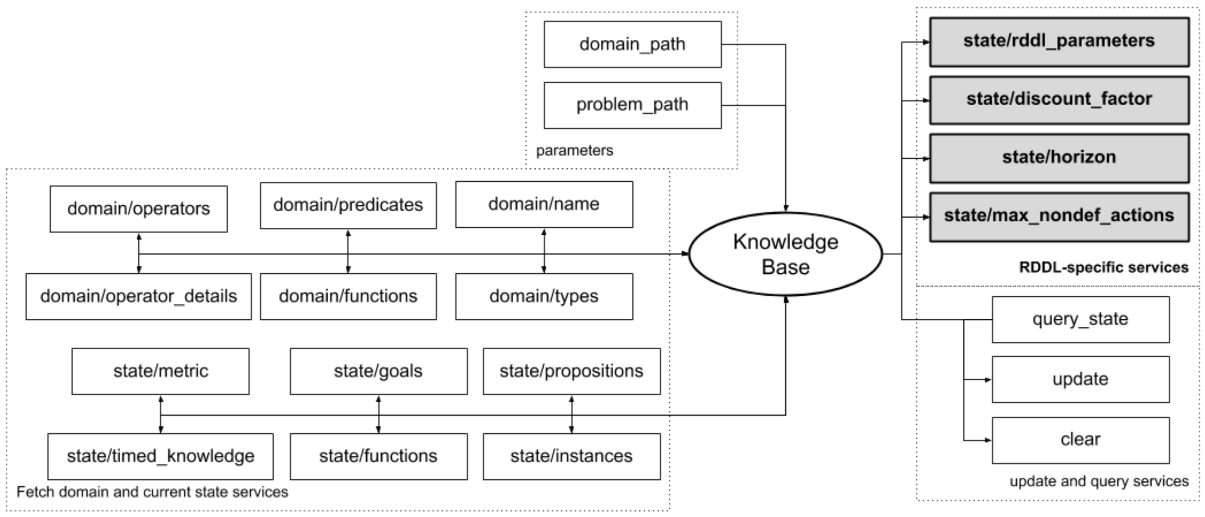


Figure 6.1: ROSPlan's Knowledge Base interface. The RDDL services are highlighted.

in many robotic use cases featuring uncertainty, such as robotic exploration missions [128]; or those with action outcomes that are inherently non-deterministic, such as manipulation problems [129], Human-Robot Interaction [130] and Physically Assistive Robot [13]. The office setting is a common environment for autonomous service robots, and can exhibit these kinds of uncertainty. Examples are collaborative robots servicing human indoor environments [131] and an office-guide robot for social studies [132].

6.3.2 System Description

In order to include the ability of planning with probabilistic domains within the ROSPlan framework, we have designed and implemented a new Knowledge Base and problem generator that are able to handle probabilistic planning problems written in RDDL.

RDDL Knowledge Base

The Knowledge Base (KB) in ROSPlan stores the current model of the environment. It is an interface for updating and fetching a PDDL model in ROS, and primarily consists of a set of ROS services forming this interface. These services are used by many other components of ROSPlan, most of which require state or domain information, such as problem generation and plan execution and validation.

The integration of RDDL with the ROSPlan KB adheres to the existing interface for two reasons: to preserve compatibility with systems already using ROSPlan, and to allow for the interchange of RDDL and PDDL KBs. Therefore, the RDDL KB translates the RDDL domain and problems to PDDL-like structures. Given that RDDL is more expressive than PDDL, the RDDL KB also extends the interface with new ROS services providing RDDL-specific functionality. Figure 6.1 shows the extended KB interface.

To process the RDDL domain into a PDDL-like structure, action-fluents are mapped to PDDL operators, and state-action constraints (also called action-preconditions in newer versions) are encoded as PDDL preconditions in the following way:

1. The constraints are searched to find those of the form $action\text{-}fluent \rightarrow (formula)$.
2. When found, the right hand side is encoded as an action precondition.

We assume the $formula$ only includes conjunctions of state fluents. This is due to a current limitation of ROSPlan, which does not support quantified or disjunctive conditions in PDDL.

Action effects are obtained from conditional probability functions (*cpfs*). This block describes how each fluent changes at each time step, determined by the current state and actions. In order to obtain the effects of an operator, the *cpfs* block is processed for each action fluent. In this processing, the state-fluent of the conditional probability formula is added to the effects of the action when the action fluent appears in the formula, be it alone, inside a quantified expression or along with other expressions (in such case the rest of state fluents are ignored). In the special case of an if-then-else formula, the effect is added when the action fluent appears in the condition of the if clause and the value of the clause is *true*. In case the value is *false*, the negated proposition will be added as an effect.

As a new feature, probabilistic effects are also considered and added to the Knowledge Base. We only consider probabilistic effects to be of the RDDL's Bernoulli distribution and Discrete distribution types. Stochastic effects are processed in a similar way to non-probabilistic ones, but when the result of the *cpf* expression is probabilistic, the effect is added to a new effect list with an associated probability formula.

In addition, assignment effects will be considered similarly to the propositional effects, and they are translated either to constant assignment or to increase and decrease clauses. An assignment effect will be added when there is an if-then-else clause including the action fluent in the condition. The value of the if clause will be added as an assignment effect when it is either a constant value or has the form $fluent \pm expression$ (which will be translated to an increase or decrease of the fluent value). Other cases are not considered.

In order to provide information on exogenous effects, a new operator named *exogenous* is created. This operator has as its effects all the exogenous effects that may happen but are not related to any specific action-fluent. Effects of this kind are otherwise considered in the same way as the effects of other operators. Finally, the reward function is fully instantiated and represented as a PDDL metric function, with the metric set to be maximized. In the case where there is a state-fluent named “goal”, its expression from the *cpfs* block will be included as the PDDL goal.

Although some assumptions are made, such as the conjunctive-only preconditions, it should be noted that these assumptions apply only to the RDDL domain file, which will not be modified when loaded into the KB. Instead, it is passed entirely to the planner. Therefore, although some elements of the domain may be unknown by the KB, the problem is entirely captured, and the planner will still provide correct plans.

Problem Generation

The ROSPlan Problem Interface node is used to generate a problem instance. It fetches the domain details and current state through service calls to a Knowledge Base node and publishes a PDDL problem instance as a string, or writes it to file. To be able to use a planner with a RDDL input, a RDDL Problem interface has been implemented.

The generation of the RDDL problem requires checking operator effects to find which predicates change due to some operators (the state fluents) and which are static for the planning problem (called non-fluents). Additionally, the planning horizon and the discount factor are set by default, or from RDDL-specific services in the KB. A feature of this approach is that as the KB interface is common for both RDDL and PDDL, ROSPlan can generate problems independently of the which KB is used. Thus, a RDDL instance file can be generated from a PDDL instance and vice versa. The requirement is that that both domains share the same structure (i.e., operators and predicates). Therefore, it is now very simple to have both deterministic and probabilistic planners running together, for example, for plan checking and validation or in a planning system composed of both stochastic and deterministic planners.

6.3.3 Online Planning and Execution with RDDL Planners

ROSPlan provides two plan dispatchers: the simple plan dispatcher for non-temporal, sequential plans, and the Esterel plan dispatcher for temporal plans with concurrency. Both dispatchers require as input a complete plan produced offline. For stochastic plan execution with RDDL, a third plan dispatcher was designed and implemented that allows the use of online planners (Figure 6.2: Nodes *Planner Interface* and *RDDL Plan Dispatch*). The online plan dispatcher interleaves plan execution and computation, removing the need of computing an offline plan and replanning when an action fails.

The online dispatcher uses the RDDL Client/Server protocol, also used by the competition server for the IPPC. In each round, the dispatcher obtains the world’s state from the Knowledge Base and sends it to the planner, which returns the actions to be executed in the next time step.

This process is repeated until the planner has reached the horizon defined in the instance file, in which case the planning process can be repeated when the task is not yet finished. With this dispatcher, any RDDL planner that uses the RDDL Client/Server protocol can be used with ROSPlan with no extra effort. Moreover, an offline stochastic planning mode is also supported in which the RDDLSim software [110] is used to simulate a run of the task, and the actions are dispatched as in the case of a deterministic planner, replanning when an action fails to execute. This approach allows to use any IPPC-like RDDL planner available.

Action Execution with Non-deterministic Effects

A robotic system interacting with the real world must keep the symbolic state of the task up to date, based on its sensory inputs. This means updating the Knowledge Base at a fixed rate such that the state is updated before each action is planned and executed. This is crucial in probabilistic planning, as with non-deterministic action outcomes it is not possible to assume that the effects of each action can be applied to the state. Instead, sensing is required to determine which outcome occurred. Therefore, we implemented a new sensing interface (Figure 6.2: *Sensing Interface*) that allows the definition of "sensed predicates and functions", which are those

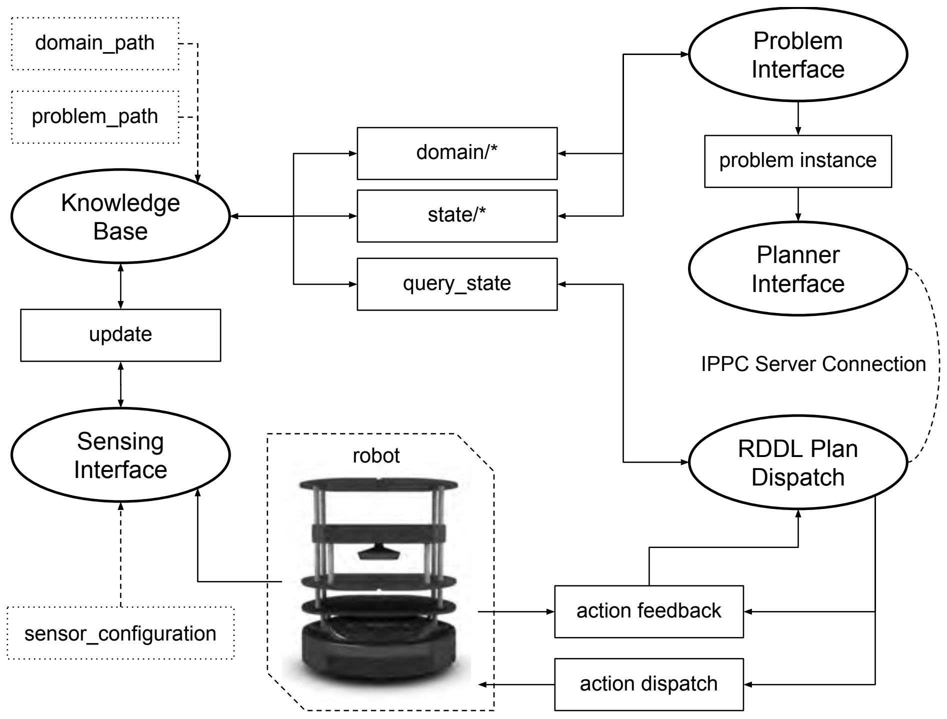


Figure 6.2: The system architecture used in our scenario. ROS nodes are represented by ovals, and implement the ROSPlan interfaces. Message and service topics are represented by solid boxes, parameters by dotted boxes.

whose values are obtained from sensor data.

The sensing interface automatically obtains the sensor data, processes it based on a minimal code definition, and updates the Knowledge Base accordingly at a fixed rate. At the same time, the KB is updated to include the information regarding which propositions are sensed or not, such that effects on the sensed propositions are not automatically applied when an action is executed. The sensed predicates are defined in a configuration file in which is specified: (1) the predicate label; (2) the parameters of the predicate which can be instantiated, and those which are fixed; (3) the sensor containing the required data, expressed as a ROS topic or service; (4) the message type of the data; (5) a single line of code whose result will be the value assigned to the predicate in the KB.

Here we show an example of this configuration for a predicate:

```
1. topics:
2.   docked:
3.     params:
4.       - kenny
5.     topic: /mobile_base/sensors/core
6.     msg_type: kobuki_msgs/SensorState
7.     operation: msg.charger != msg.DISCHARGING
```

This configuration shows (line 3) the name of the predicate, *docked*; (lines 3 and 4) that the single parameter of the predicate is fixed, so that this configuration is sensing the value of the proposition `(docked kenny)`; (lines 5 and 6) the ROS topic to which the sensing interface will subscribe and the message type; and (line 7) a single line of code that returns a Boolean result to be assigned to the proposition. If a more complex processing needs to be done, the interface can be linked with another file containing the implementation for each predicate, in which any kind of program can be defined in order to process the sensor data.

Further discussion on the use of probabilistic planning in robotics domains along with an experimental evaluation can be found in Appendix C.

6.4 Preferences to guide action selection through the reward function

RDDL allows the definition of rich reward functions. Such reward function is computed at each time-step to provide an immediate value to the metric function that is being optimized. In the case of the reward, the goal of the planner is to maximize its value. Therefore, actions that lead to states providing more reward will be favored. This allows the use of suggestible predicates to appear along with actions in the reward function, such that when the suggestible predicate is present with a specific action, it will provide positive or negative reward. This will encourage or penalize the use of the action.

In RDDL, we can do this by defining an if-then-else reward function. The function will has a case for each action and related preference value with the form

```
if (action ∧ (preference_name == @preference_value)) then R,
```

where R is the value that is obtained when the `action` is executed and the `preference_name` predicate is present and has the `@preference_value`.

To ease the description of such predicates in the reward function, we defined a rule-based format that consists on the action, the preference application and the immediate reward value R. This is then formatted and added to the domain automatically. As an example, one could define:

```
getFood, ((?speed ∼= p_speed) ∧ (p_speed ∼= @tl_unknown)) | ((?force ∼= p_force) ∧
    (p_force ∼= @tl_unknown)), -15,
```

which specifies that the action `getFood` receives a penalization with a value of 15 when either the speed or force preferences are not unknown and they have a different value than the one provided by the preference. Another example for the jacket dressing task is:

```
approachBothArms, (p_motor_rightarm == @high) ∧ (p_motor_leftarm == @high), 20,
```

which describes that the `approachBothArms` action is rewarded when the user's arms mobility is defined as high. This will then be converted to:

```
if (exists_{?speed: t_threelevel, ?force: t_threelevel} [approachBothArms(?speed,
    ?force) ∧ ((p_motor_rightarm == @high) ∧ (p_motor_leftarm == @high))]) then 20[2]
```

Consequently, this process simplifies the definition of the reward function, which can become long and complex. Note that in our approach, we expect an expert in the domain to define the rules that will be compiled into the reward function to successfully lead the planner to choose those actions that receive more reward thanks to the preferences.

6.5 Motivation behind providing predicate suggestions

There are not many examples in classical planning where the initial state can be modified at will before starting the task. In classical planning, the planner tries to modify the state using the available actions and operators. However, some elements of the task may be modified when facing the real world. A clear example of it may be that of robotics and, more specifically, collaborative and assistive robotics where humans take part in the planning process, as considered in this thesis.

[2]`t_threelevel` is defined as an enumerable type with three levels: `@high`, `@medium` and `@low`.

In such a context where human help can be used, the system or robot can benefit from human interactions and provide information relevant to the task. Therefore, given a state that is not ideal, it can *suggest* changes or additions to the initial state that may lead to a better performance in the task. These suggestions could be obtained either by questioning the user, asking for a change, or just guessing the state of some unknown predicate, knowing that such information may improve the execution performance.

A clear example, which is the focus of this thesis, is the one of preference and user limitations in an assistive robotics task.

6.5.1 Planning with preferences and limitations

First, we want to ground the definition of preference in the case of our planning domains. Although we will use the same concept of preference as in the previous chapters, here we will define them more formally. Therefore, we will consider the preferences as defined in Chapter 3. Thus, the preferences are used either to guide the action selection process or to modify how a specific action is executed (as a parameter to the action). Preferences in planning can be defined as soft goals and conditions as in PDDL3 [99], or can be related to plan ordering [38].

For completeness, we define again the task planning problem following the notation from Section 5.3.

Definition 6.1. A task planning problem $\Pi = \langle S, A, T, s_0, g \rangle$ is defined by a set of discrete states S, a set of actions A, a state transition function $T : S \times A \rightarrow S$, an initial state $s_0 \in S$ and a goal state $g \in S$. Each state $s \in S$ is an assignment of values to predicates, and each action $a_i \in A$; $a_i = \{p_{a_i}, e_{a_i}\}$ is composed of the preconditions $p_{a_i} \in S$ and the effects $e_{a_i} \in S$.

For more generality, in this chapter we will denote the preference predicates as *suggestible* predicates. We define a suggestible predicate or preference as a predicate that is assigned a certain value, appears along with a certain action in the plan and produces some reward when it is present in the state. Such suggestible predicates do not affect the possibility to reach the goal but affect how the goal is reached and which actions are selected. They are used to guide the search and, instead of being conditions that must hold or identifying a plan as most-preferred, they are predicates that may or may not hold and as a consequence produce different rewards or costs. A suggestible predicate is formally defined as follows.

Definition 6.2. A suggestible predicate $p \in S$ is a predicate such that there is no action $a_i \in A$ in which $p \in e_{a_i}$ or $p \in p_{a_i}$ and $p \notin g$, but it can be that p appears in R, where R is a reward or metric function to be maximized.

As seen in the previous chapters, these preferences may include but are not limited to, the robot's movement speed, proxemics and verbosity.

6.6 Providing suggestions

In this section, we propose the Space of Plans Suggestions (SoPS) algorithm to provide suggestions to a set of predicates $P \subseteq S$.

Definition 6.3. A suggestion $q = \{(p, v) \mid p \in S, v \in Domain(p)\}$ is a set of value assignments to predicates such that the reward increases when planning using them.

Definition 6.4. A Space of Plans is a set of valid action sequences that bring the system from an initial state s_0 to a goal state g.

The algorithm analyzes a subset of the Space of Plans to provide the suggestions. The algorithm's goal is to determine which predicates have more impact on the reward, to suggest those first. Therefore, it needs as an input a subset of the Space of Plans corresponding to the plans obtained by combining the different *suggestible* predicates and obtaining a plan with them, along with their associated plan reward.

This subset of the Space of Plans is compiled as a tree for efficient suggestion search.

6.6.1 The Plan Space Tree

The Space of Plans is compiled into a tree data structure where each branch is a complete plan, similar to the policy trees used in contingent planning [133]. Therefore, all the plans with a common prefix or starting sequence of actions begin at the root node and branch when the plans differ. Accordingly, all the leaves of the tree are actions that produce a goal state.

Each node of the tree keeps a list with the set of suggestible predicates that produced the plan, along with the plan reward. This information is kept at each node for all the plans that reach the node. Moreover, the maximum reachable reward is kept for efficient retrieval of the maximum reachable reward from the node's branch. An example of a Plan Space Tree is shown in Figure 6.3. As it can be observed, each node stores the matrix of predicates for all the plans that go through it, and the index to the maximum reward node. For instance, for the a_1 node, $maxR_{a_1} = \max(maxR_{a_4}, maxR_{a_5}, maxR_{a_6})$.

This representation provides a compact and efficient data structure on which we can perform the search.

In order to populate the tree, the subset of the Space of Plans is generated by executing the planner with changing conditions in the problem file. Therefore, for all the combinations of suggestible predicates, we generate one or more plans (depending on whether stochastic planners are being used). Then, the list of plans is traversed to build the tree, adding new action nodes when there are new branches. When the action node already exists in the tree, the suggestible predicates of the plan along with their associated rewards are added to the node.

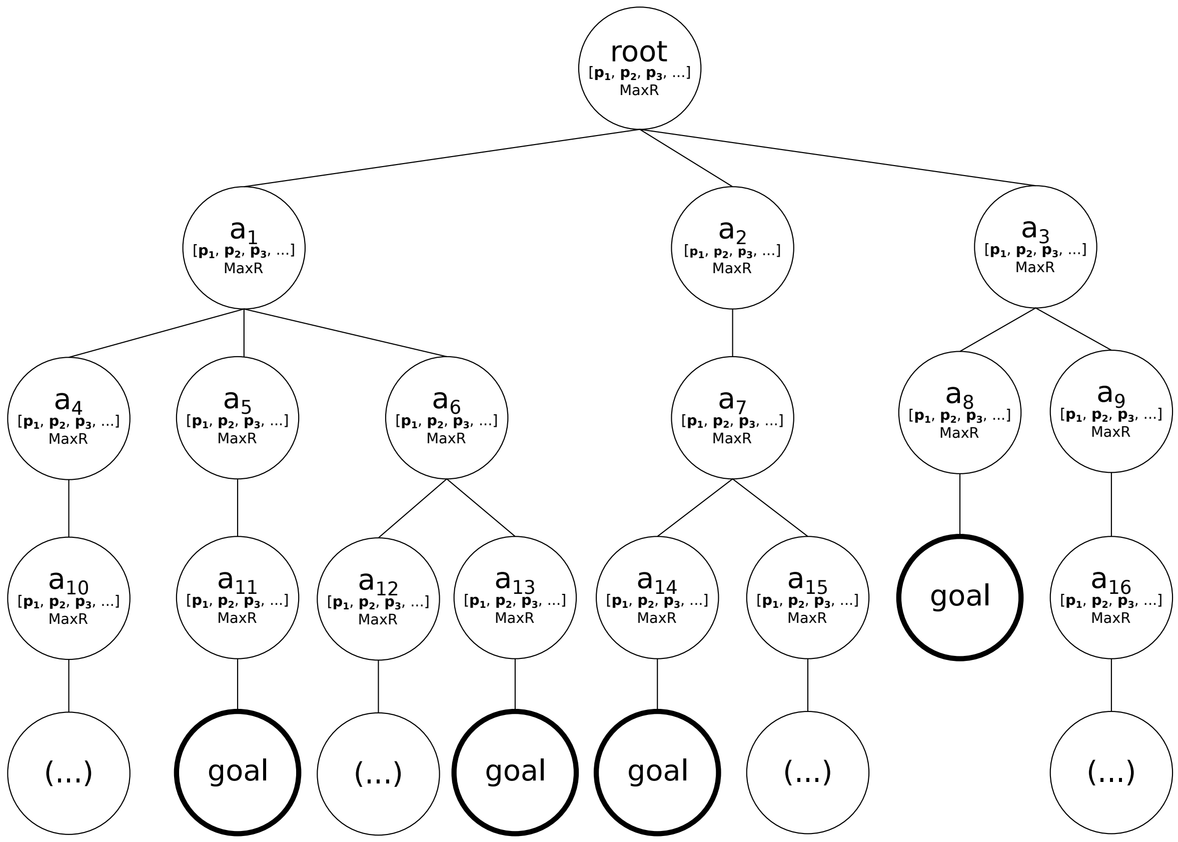


Figure 6.3: Example of a Plan Space Tree. Each node a_i represents an action of the tree. Nodes labeled as goal are leaves whose branch is a complete plan to the goal.

6.6.2 Max-reward traversal

The SoPS algorithm (see Algorithm 6.1) performs a maximum reward traversal of the Plan Space Tree to obtain a set of suggestions to unknown suggestible predicates that improve the plan reward. For this reason, the already known predicates belonging to the suggestible set S are fixed along the tree. To this end, all the branches belonging to plans generated with predicates whose value is different to the fixed one are pruned, and their rewards discarded, keeping only the branches belonging to unknown predicates.

To start, the algorithm searches for the promising nodes in the tree (see the GETPROMISINGNODES procedure). A promising node is a node of the tree such that it is a child with a maximum reachable reward.

Definition 6.5. A promising node m is a node in the Plan Space Tree such that $m \in C_n$ and $\nexists a_i \in C_n, a_i.MaxR \geq m.MaxR$, where C_n is the set of children of the node n and $a_i.MaxR$ is the reward associated to a node a_i.

Algorithm 6.1: SoPS algorithm

```
Input: Plan Space Tree $t$
Output: Set of suggestions $P$
1  procedure GetSuggestions(t)
2      D := GetPromisingNodes(t)
3      m := []
4      forall $d_i \in D$ do
5          m.add(computeMetric($d_i$))
6      return ComputeNodeSuggestion($D_{max(m)}$)
7  procedure GetPromisingNodes(t)
8      children := getMaxRewardChildren(t)            // Children with max. reward
9      $d$ := []
10     forall $c_{max} \in children$ do
11         forall $c \in getChildren(t)$ s.t. $c \neq c_{max}$ do
12             c_diffs := computeDiffs($c_{max}$, $c$)
13             if not empty(c_diffs) then
14                 d.add($c$_diffs)
15         d.join(GetPromisingNodes($c_{max}$))
16     return d
17 procedure ComputeNodeSuggestion(d)
18     s = sum_cols($d$)                              // Sums the columns
19     return $\{i \mid s_i = max(s)\}$
```

For each of those nodes, we compute a Boolean difference matrix D^n (line 12) such that

$$D^n_{i,j} = (p_{i,m} \neq p_{i,j})\ \forall i \in P, j \in C_n \setminus \{m\}, \tag{6.1}$$

where m denotes the child with maximum achievable reward, P is the set of suggestible predicates.With D^n we can compute a set of candidate suggestions for each node n (see GetSuggestions procedure). To do so, we flatten the matrix into a vector d where $d^n_j = \sum_i D^n_{i,j}$. With d, we can obtain the set of suggestions u (see ComputeNodeSuggestion procedure) as

$$u = \arg\max_j d_j. \tag{6.2}$$

Therefore, the candidate suggestions are the predicates belonging to the maximal child whose values are more different in comparison with its siblings. Those are the predicates which have more impact on the difference of reward, and the ones that make this reward maximal.

Along with the candidate suggestion, a significance metric is computed for all the promising nodes (line 5). This metric is an indicator of how different is the maximum reward child of the node in contrast with the other children. We propose the following metric f, which computes the average reward difference between the child with maximum reward m and the rest:

$$f(n) = \frac{\sum_i r_{max} - r_i}{N-1} = r_{max} - \frac{\sum_{\{i \in C | i \neq m\}} r_i}{N-1}, \tag{6.3}$$

where r_{max} is the maximum reward of all the children of the node n, and r_i are the other child rewards.

The rationale behind the metric in Equation 6.3 is that child plans that have a greater average reward difference are better candidates at showing which suggestible predicates can make more difference. Subsequently, the output suggestions are the candidate suggestions of the node with the highest metric. Note that in case of a tie in Equation 6.2, more than one predicate will be suggested. Moreover, along with the predicate that makes the difference, the algorithm provides an assignment to each of the suggested predicates, which are the values assigned to the predicates in the selected node.

The proposed SoPS algorithm (Algorithm 6.1) can be executed iteratively in order to obtain new suggestions until all the suggestible set has been determined. To do so, the values of the known suggestible predicates[3] can be fixed beforehand. More specifically, the algorithm goes over the tree pruning the branches or discarding those that do not satisfy with the fixed predicates. The fixed predicates are then also taken into account in Equation 6.1, where the fixed predicates are ignored in the computation of the differences matrix.

6.6.3 Suggesting changes to known predicates

Once we are able to provide suggestions to unknown predicates, we can go a step further and propose *changes* to some of the fixed (already defined) suggestible predicates. This would provide further improvement of the plan performance, at the cost of slightly modifying the user-defined values.

However, the system shall not completely ignore the defined predicates, as they may be given a specific value for a reason. Therefore, we propose to only modify the predicates when the received suggestion's value is *close* to the defined value. The notion of closeness can be left to the user to be defined. In the case of an ordinal set of values for a predicate, this closeness can just be the arithmetic difference and a defined value of maximum acceptable difference for a change.

Definition 6.6. A change c is a suggestion such that $c = \{(p, v) \mid p \in S, v, v' \in Dom(p), (p, v') \in$

[3]Predicates can be known because they are provided to the algorithm or because they were obtained as suggestions in a previous execution.

$D, sim(v, v') \leq T\}$, where D is the set of predefined predicates, sim is a similarity function, and T a user-defined threshold.

Algorithm 6.2: SoPS-change algorithm

```
Input: Plan Space Tree $t$
Input: Predefined set $D$
Output: Set of suggestions allowing changes $P$
1  P = []
2  s := GetSuggestions($t$)
3  forall $s_i \in s$ do
4      if $predicate(s_i) \in D$ then
5          if $(value(s_i) \neq value(D_{s_i})) \wedge (sim(value(s_i), value(D_{s_i})) \leq T)$ then
6              P.add($s_i$)
7              D.remove($predicate(s_i)$)
8          else
9              t.fixPredicate($D_{s_i}$)        // Prunes the branches not satisfying $D_{s_i}$
10             return SoPS-change($t$, $D$)     // Recursive call
11     else
12         P.add($s_i$)
13 return P
```

The proposed method for changes is the following. First, we obtain the suggestions from the whole Plan Space Tree, ignoring the predefined predicates. If the suggestions are new, we add them to the set. Otherwise, the predefined predicate will take the suggested value if the similarity between the values is close enough, as defined above. Algorithm 6.2 shows the variation of the method including changes.

6.7 Experimental evaluation

To evaluate the proposed algorithms, we have designed three domains in which preferences can play an important role in the decision process and modify the plan to be executed. The domains relate to assistive robotics scenarios, consisting of assistive feeding, shoe-fitting, and assisted jacket dressing.

As described in Section 6.4, the domains have been written in the RDDL language [110]. RDDL allows for richer reward function definitions, suitable for the integration of suggestible predicates as defined above. As a plan solver, we have used the PROST planner [112] has been used to compute the Space of Plans and to execute all the experiments[4]. PROST is a RDDL-based

[4]The source code of the proposed algorithms can be found in github.com/gerardcanal/SoPS

probabilistic planning framework with many different search configurations, originally based on the UCT algorithm [134]. It is a state-of-the-art planner that won the latest IPPC competitions, and we decided to use it due to its good performance. The default IPPC2014 configuration was used for the experiments presented in this section. The experiments have been run using the ROSPlan framework with the RDDL extension described in Section 6.3.

6.7.1 Definition of the domains and preferences

Next we describe the implemented domains and preference options. The domains have been defined such that there are many equivalent actions and many different paths to the same goal. The preferences and suggestible predicates define the final obtained reward and thus guide the planner towards choosing the actions that comply with them. For more details on the definition of preferences refer to Section 3.3. For a demonstration of how do the preferences apply to the three scenarios used in the thesis refer to the supplementary video[5].

Feeding task

The feeding task completes when the user has been fed (at least N spoonfuls completed). It contains the following actions:

- **Get Foot:** uses the cutlery to get the food.
- **Approach straight:** approaches to the user frontally.
- **Approach from below:** approaches the user from below (less intrusively).
- **Approach from the side:** approaches sideways, being always visible but not frontally.
- **Feed straight:** feeds the user by moving in a straight line and exiting in the same way.
- **Feed scooping:** feeds the user and performs a scooping action when exiting to ensure emptying of the food.
- **Wait for user feeding:** waits for the user to get the food.
- **Move away:** retires the robot back to the starting position.

The preferences involved in this task are the head mobility, head proxemics (closeness of the robot to the user), movement speed, applied force, feeding cadence and robot verbosity.

[5]The video demonstration on the use of the preferences for action selection in the assistive robotics scenarios can be found at youtu.be/mVvnigdPJPQ

Shoe-fitting task

The shoe-fitting task is completed when both feet have a fitted shoe and the robot's gripper is empty. It includes the following actions:

- **Request foot reachable:** requests the user to move the foot closer.
- **Request foot visible:** requests the user to put the foot in the robot's sight.
- **Grasp shoe:** it grasps the shoe (from the user, as a handover).
- **Approach from top:** it approaches the user's foot from the top.
- **Approach left:** it approaches the user from the left side.
- **Approach right:** approaches the shoe from the right-hand side.
- **Approach from below:** approaches the user from the below.
- **Insert straight:** inserts the shoe in a straight movement, without forcing the ankle.
- **Insert curved:** shoe insertion forcing a bit the ankle to fit correctly the heel.
- **Insert right/left:** inserts from the side following the foot's shape.
- **Release simple:** releases the shoe and moves away.
- **Release push:** it pushes the shoe a bit before releasing it to ensure fit.

The preferences involved in this task are the foot mobility (for each foot), leg mobility (for each leg), the speed, applied force, verbosity and requests (defines whether requests to the user should be done or not).

Jacket dressing task

The jacket dressing task is completed when both sleeves have been fitted until the shoulder and the robot's grippers are empty. It consists in the following actions:

- **Approach single-arm frontal:** it approaches a sleeve to a single arm from the front (visible to the user).
- **Approach single-arm rear:** it approaches the sleeve to the arm from behind (more comfortable as less movement is involved).
- **Approach arm side:** it approaches a sleeve from the side.

- **Approach both arms:** approaches both sleeves together from behind.
- **Insert sleeve from the front:** inserts the sleeve in a frontal manner (doing so makes it impossible to insert the other sleeve frontally).
- **Insert sleeve straight:** inserts a sleeve from the side, with the stretched arm.
- **Insert both sleeves:** inserts both sleeves together from behind.
- **Drag forearm frontal:** drags the sleeve in the forearm from the front.
- **Drag forearm straight:** drags the forearm sideways.
- **Drag both forearms:** drags both forearms together.
- **Drag upper arm:** drags an upper arm.
- **Drag both upper arms:** drags both upper arms together.
- **Release:** Drags the cloth to the shoulders and releases the garment.

The preferences involved in this task are the arm mobility (for each arm), the speed, the applied force, verbosity and the torso proxemics (how close should the robot go to the user's torso).

6.7.2 Effect of the SoPS algorithm

To demonstrate the effectiveness of the SoPS algorithm, we have compared the obtained suggestions against random preference value assignments, which represent user-provided values. Note that in a real robot scenario the user would provide these preference values. However, in the experiment of this section, these user-provided (fixed) values will be obtained from previous executions of the algorithm or set at random. To do so, we run the algorithm with random sets of fixed preference predicates, starting with sets of size 0 (without any known predicates) and adding one predicate each time until all the suggestible predicates are known. Thus, the algorithm returns suggestions to the yet unassigned suggestible predicates. Each obtained suggestion is then fixed (and assumed known), and new suggestions are further requested until all the suggestible predicates have been assigned. In this case, the predicates known beforehand (the random sample) were fixed in the Plan Space Tree and the affected branches were pruned-out. Therefore, those predicates are not taken into account by the algorithm.

For each step (new suggestion obtained), we have created 50 sets of random samples of predicates. Afterward, the SoPS algorithm was used to obtain suggestions for the unknown predicates, getting one suggestion at each step until the total number of suggestible predicates was reached.

After computing each suggestion, the planner was used to obtain a new plan, and its final reward was stored. Given that the PROST planner uses stochastic methods and its solution is not deterministic, each plan was computed 20 times. This value was experimentally defined as the number of executions that provided a sufficient number of plans to capture the variability of the obtainable plans for each configuration of the presented scenarios. The results shown in this section are the average of all the 1000 executions (including both the 20 repeated planning attempts and the 50 random samples).

Figure 6.4 shows the results obtained using the explained procedure for the feeding domain. In this domain, we consider 6 possible preferences or predicates (detailed in Section 6.7.1). The SoPS line (in blue) shows the mean reward that can be obtained when no preferences are known (0 known predicates), and the reward that can be obtained when the most promising preferences are determined (up to 6).

As it can be observed, the use of the SoPS algorithm highly improves the obtainable reward in all the cases. This is because, even when random fixed predicates don't give much reward, the algorithm finds suggestions for the other predicates that can improve the total reward. The fact that the initial reward (first point in every line) increases as more predicates are known can

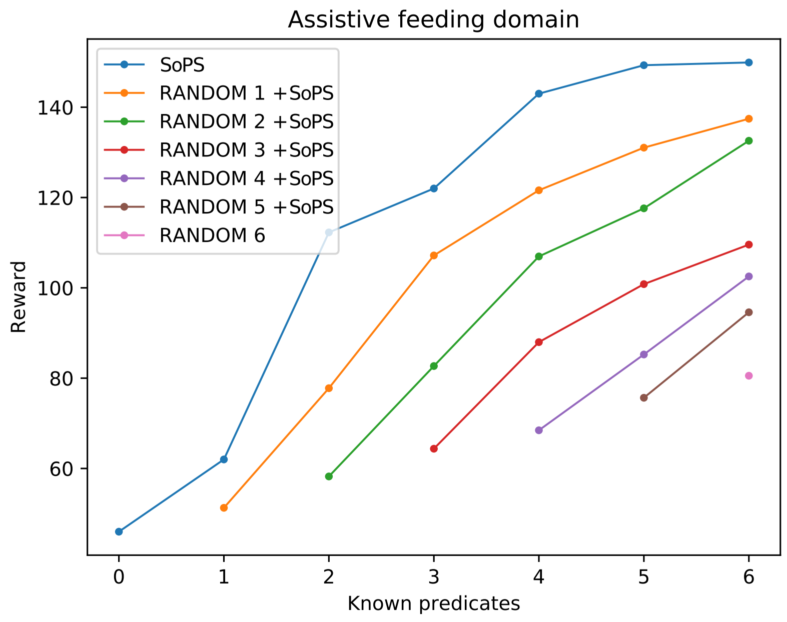


Figure 6.4: Results with different set-ups for the feeding domain. Observe how the suggestions provide better rewards in all the cases, even when the system starts with random fixed values for the suggestible predicates.

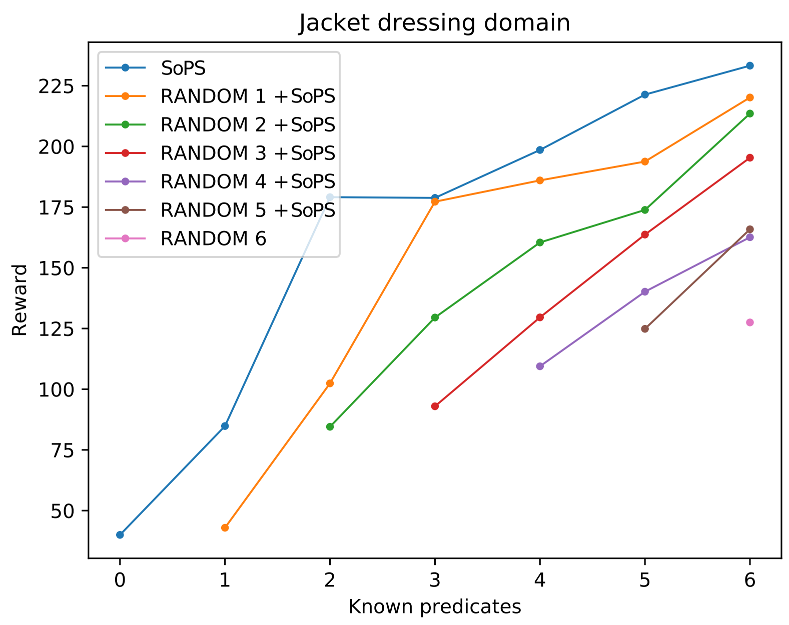


Figure 6.5: Results with different set-ups for the jacket dressing domain. In this case, there is some correlation between some predicates, but the algorithm still produces an improved reward.

be explained because having an extra predicate increments the initial reward (as the suggestible predicates provide extra reward). Note that, as 6 predicates are the maximum, SoPS cannot be applied in the “random 6” case and thus only the mean reward is depicted.

A similar result is can be seen for the jacket dressing domain in Figure 6.5. Interestingly, a plateau is found between the second and third predicates for the SoPS (blue line). This is a result of two predicates that were suggested together due to being tightly coupled predicates that provide the reward when they are together. In the evaluation, we keep only one of the suggested predicates at each step for easier comparison. Therefore, when obtaining the third predicate the algorithm provides two suggestions. After fixing the first one (predicate 3), the algorithm returns the second suggestion for predicate 4 (which was already suggested in the previous iteration). Therefore, the reward of both predicates is obtained when the second one is suggested.

Finally, Figure 6.6 shows the same behavior regarding the random pre-assignment of predicates and the algorithm, but plateaus can be observed at the end of the assignments. This is due to the superfluous predicates present in the domain. These superfluous predicates do not increase the reward. Therefore, those are obtained as a suggestion once the useful predicates have been already suggested. A more detailed analysis of the superfluous predicates can be found in Section 6.7.4.

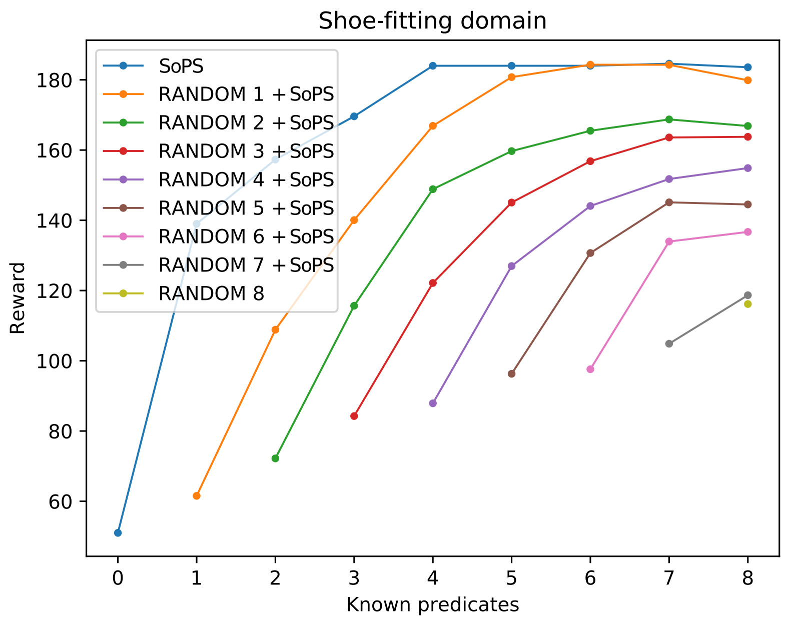


Figure 6.6: Results with different set-ups for the shoe-fitting domain. This domain has some predicates that do not improve the reward, and those are suggested at the end when the maximum reward has been obtained.

6.7.3 Improvements by allowing changes with SoPS-change

A new experiment has been performed to analyze the effects of allowing changes in fixed suggestible predicates with the SoPS-change algorithm. These changes are suggestions to already assigned predicates keeping into account the current value. The SoPS-change experiment will be performed, as in the previous section, generating random assignments for the value of the preferences. However, this time changes will be allowed if the suggested predicate was already assigned. In general, any change can be allowed by specifying enough distance as a parameter. Given that our domains include physical scenarios, we believe big changes should not be allowed. Therefore, for this experiment suggested changes are only accepted when the suggestion obtained is at distance one from the assigned value, so bigger changes are not allowed. In case there is a suggested change that can not be accepted, we fix that preference to the already assigned value and continue with the following suggestions.

Figure 6.7 shows the maximum obtainable reward for the feeding case when preference values are fixed, starting with N preassigned predicates (horizontal axis) and using the suggested predicates from the algorithms to assign the rest. The figure compares the changes approach to the standard SoPS version. In both cases, the reward values are obtained from the tree, and they represent the maximum obtainable reward as stated in the Plan Space Tree. Therefore, no new plans are computed this time. Observe that, as in the previous section, the higher the number of random fixed predicates the lower the reward is. But, when changes are allowed, the reward decay is much slower. Notice this happens even in a conservative approach in which only small changes are allowed (so when the suggestion is highly different from the fixed predicate, the suggestion is ignored). The same behavior can be observed for the other domains, in Figure 6.9 and Figure 6.8.

6.7.4 Finding superfluous suggestions

The results obtained from the shoe-fitting domain in Figure 6.6 show that predicates that do not provide much reward get suggested at the end (as the most promising ones are suggested earlier). To confirm this and analyze its implications, we performed another experiment where more suggestible predicates that do not help to increase the reward were added. We call refer to these suggestible predicates as superfluous predicates.

To this end, we have run the same experimental set-up of Section 6.7.2 with slightly modified domains. In them, we added two predicates which are not taken into account in the reward function, but allow them to be suggested, being added to the Plan Space Tree. Later we have executed the SoPS algorithm in them, the results of which are shown in Figure 6.10.

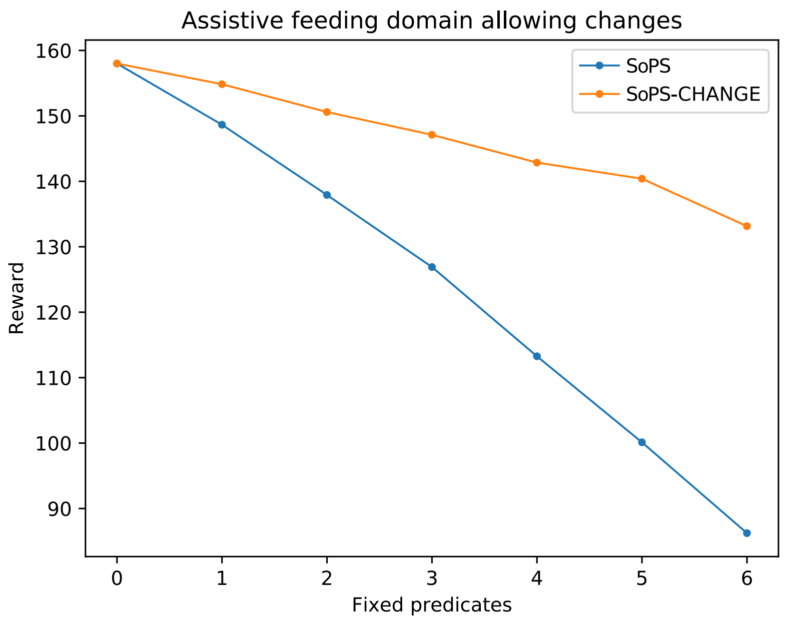


Figure 6.7: Results for the feeding domain allowing changes. As the number of fixed predicates increases, the reward decreases (as they may not be optimal). Suggesting changes close to the fixed predicates allows to improve the obtained reward.

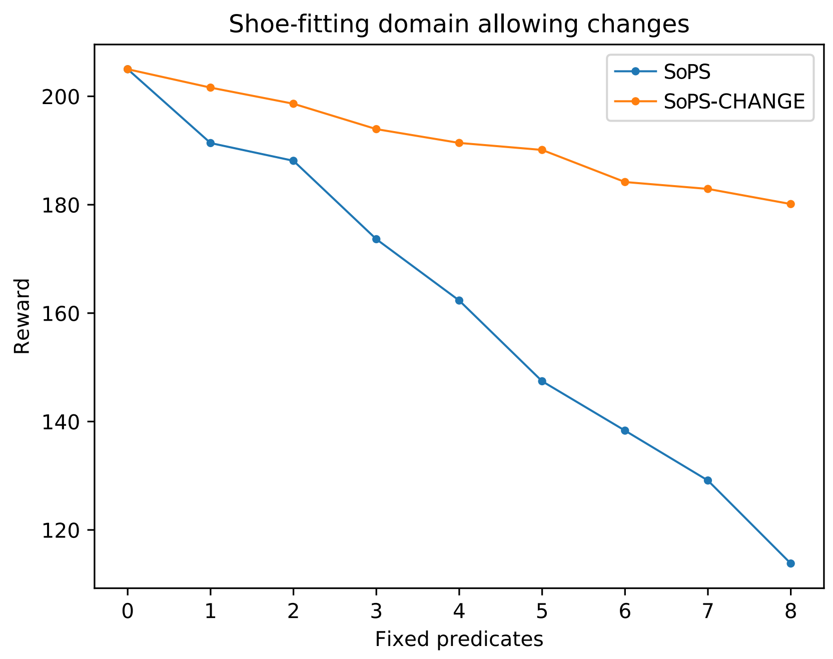


Figure 6.8: Results for the shoe-fitting domain allowing changes. This domain shows a similar trend, successfully improving the obtained reward with the suggested changes.

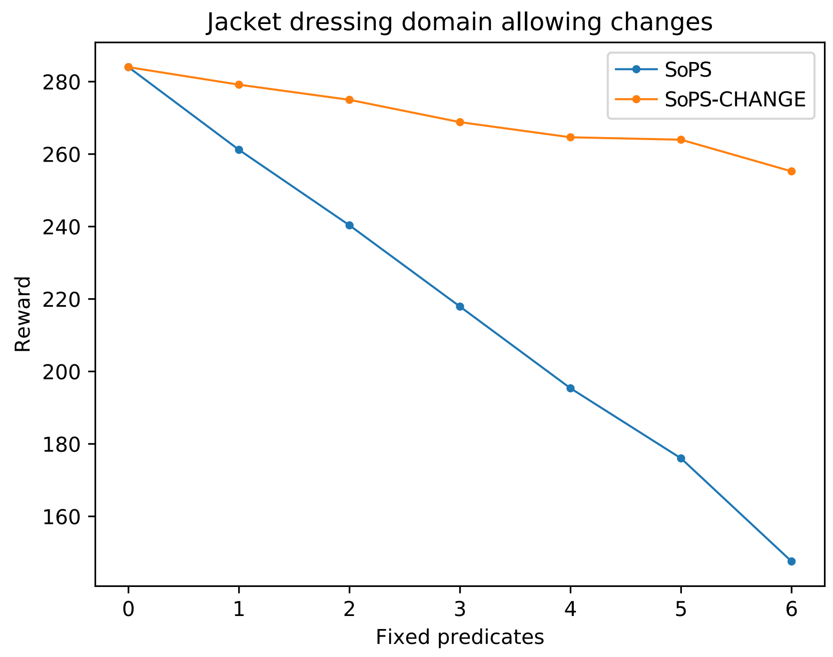


Figure 6.9: Results for the jacket dressing domain allowing changes. Change suggestions to the fixed values with a distance of one are enough to improve the final reward.

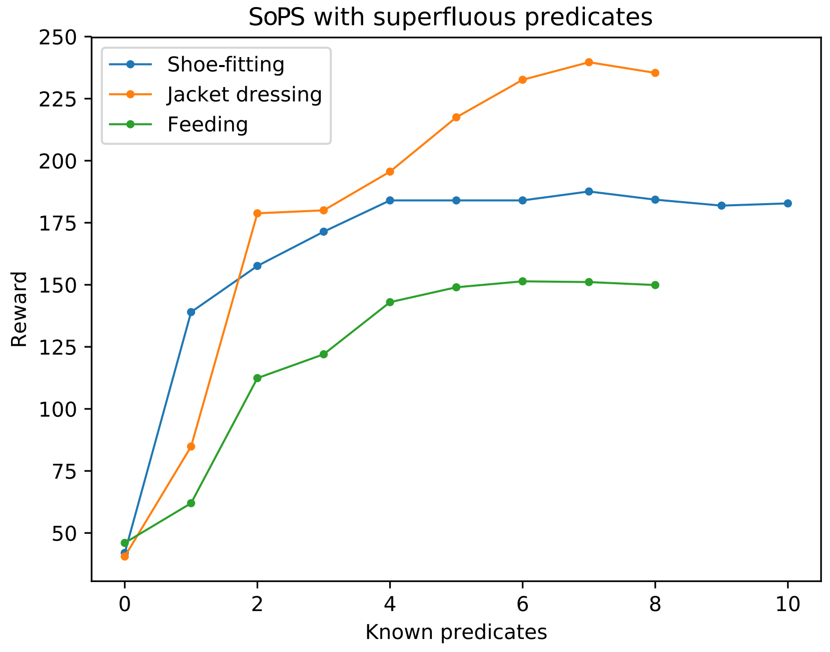


Figure 6.10: Results with the different domains including superfluous predicates. Our method is able to maximize the reward ignoring the predicates not providing more reward, which are suggested at the end.

As it can be seen, the reward function tends to saturate around the last predicates, while keeping the same shape as in the previous experiment. In the case of the shoe-fitting, it is clear that there are many superfluous predicates. The feeding case also shows a third potential superfluous or less-useful predicate, while the jacket dressing shows that most of the predicates are useful. Slight variations of the tails of the reward plots are due to the stochasticity of the results (which are again an average of all the plan executions).

Consequently, it can be seen that the SoPS algorithm can also be used to determine whether there are superfluous predicates in a domain, which can be used to decrease the size of the search space. However, it can be seen that superfluous predicates can't usually be detected while obtaining the suggestions, but only when all the suggestible predicates have been obtained. Even so, the computation of all the suggestions is efficient and quick enough to be possible to pre-compute the superfluous predicates beforehand.

6.8 Summary

In this work, we have presented an algorithm to provide suggestions for assigning values to predicates in planning domains. We have defined the concept of suggestible predicates, which are those predicates that help the planner by guiding the search and providing more reward under some circumstances. To do so, we extended ROSPlan to be able to handle stochastic planners and languages such as RDDL. Then, we have proposed the SoPS algorithm that uses a Plan Space Tree built from a pre-computed subset of the Space of Plans. The algorithm traverses the tree to obtain suggestions for predicates such that the final plan reward is maximized. A variation of this algorithm that suggests changes to already assigned predicates has also been proposed. These changes are considered taking into account the current assigned value of the predicate.

The algorithms were evaluated with three assistive robotics domains in which the suggestible predicates are preferences of the user to define the robot's behavior. Results show that using the values selected by the algorithms improves more the reward in comparison with a random selection of the values when computing new plans.

The methods proposed in this chapter can be used in many other domains apart from the already shown here. The algorithms can also be used for plan explainability and other Explainable AI set-ups. We believe the suggestions provided by our algorithms could also be used to explain a non-expert user why the planner took an action or another, as well as to help the user in selecting the best configuration based on their needs, explaining that assigning a specific value to a predicate can lead to better plans. Although some more work can be done in this direction, we believe these algorithms can be a good tool for providing plan explanations, as well as powerful algorithms to analyze the Space of Plans.

Moreover, the algorithms presented here can be used to elicit unknown user preferences. An initial elicitation method was presented in Chapter 4, where a Fuzzy Inference System was used to get the values of the preference predicates. The user model presented may work well with a few predicates, but when the number of predicates grows the number of user questions does too. The methods presented in this chapter can then be used to refine the model obtained by the FIS, ultimately leading to a better adaptation and task performance.

7
Evaluating the use of preferences through HRI

Until now, we have defined a personalization framework and also defined the potential preferences that can be applied to Physically Assistive Robotics tasks (Chapters 2 and 3), and studied how to integrate such preferences in an autonomous manner by means of AI Planners (Chapters 4 and 5). Having implemented autonomous robots able to assist users in different ways based on preferences, there is a question to be solved which is whether the users are able to understand when their preferences are employed and to assess and distinguish between different robot behaviors produced by different preferences. We will answer the question in this chapter, where we perform a Human-Robot Interaction study to evaluate the use of preferences. The experiments have been performed with real users performing the three tasks defined in Section 1.3 with a real robot.

This work has been published in [18].

7.1 Introduction

The success of any Assistive Robotics task depends on the interaction with the user. User collaboration is key to have satisfactory assistance, and preferences should play an important role to achieve this collaboration. If the user feels comfortable and safe, the trust in the robotic system will increase, which will lead to overall increased satisfaction. The ability to adapt to users should increase their satisfaction, and the best manner of assessing the impact of the adaptation is by trying it with real users.

This is even clearer in the case of physical Human-Robot Interaction, where the *interaction* is one of its main elements. When touching users with a robot, its behavior must be clear, legible and understandable by the user at any moment. Failure to do so may result in rejection of the complexity of use or fatal outcomes such as user harm. This thesis focuses on the behavior adaptation of the robot and the user. We defined a framework for robot adaptation (Chapter 2), a taxonomy of possible preferences (Chapter 3), and analyzed different methods of adaptation by the use of task planners (Chapters 4 and 5). Finally, we showed how could these preferences be suggested to improve the performance of the tasks (Chapter 6). However, the part of the user interaction has been left out to focus on how to create robust behaviors and adaptations. Therefore, we are now ready to proceed with a user evaluation to assess whether the use of the preferences as we have envisaged in this thesis is correct and leads to better interaction.

In that direction, this chapter presents a user study that evaluates if users can identify in which executions of the assistive tasks the robot is using their chosen preferences, without previous examples. We also consider whether the tasks fulfilling the user's preferences are found more pleasant and whether the able users see potential benefits for dependent users. Finally, the Almere model [135] has been used to evaluate the robot acceptance. For this purpose, the three assistive tasks described in Section 1.3 have been considered. Two robot arms have been used to assist the users in performing the tasks in a fully autonomous manner. The users have experienced each task two times, answering a questionnaire of their experience after each task. The following sections will detail the used experimental methodology, evaluation measures, and experiment results.

7.2 Related work

Physically Assistive Robots require a proper interaction with the assisted user to successfully complete the task. Given the importance of such kind of applications, many works intend to solve some or part of the assistive tasks. Silva *et al.* [136] propose a modular robotic arm for assisted feeding. Vila *et al.* [17] (see Appendix B) assume that impacts may occur accidentally and analyze impact forces and safety measures for robot-assisted feeding. A taxonomy of manipulation strategies for feeding is presented by Bhattacharjee *et al.* [137]. To do so, they created a dataset of food manipulation. Following up, bite acquisition is further studied by Gallenberger *et al.* [138], where adaptive strategies select the manipulation primitive to use with each food item. A user study with 25 participants was performed in which users had to determine how easy was to bite off the fork when the robot presented the food.

In a similar scenario to the jacket dressing that we consider in this thesis, a hospital gown is dressed by Erickson *et al.* [139], where a deep recurrent model is used to predict the garment forces applied to the user. This is further applied for improving the robot's control, using only haptic and kinematic data from the robot's end-effector. Zhang *et al.* [140] propose a tracking method based on Bayesian Networks in latent spaces to fuse robot pose and force for camera-less estimation of user postures while dressing. They applied it to the dressing of a sleeveless jacket. The system is evaluated with able-bodied users who had some of the movements restricted by braces to simulate user limitations.

Most of these works focus on the independent living of the users by providing more autonomy. However, such autonomy can also be seen as the control the users have over the robots as described by Lee and Riek [141]. Moreover, Chien *et al.* [142] show that providing customized designs and settings, as well as positive experiences, can improve the acceptance and use of assistive robots by older adults. To this end, the use of preferences may have a great impact on the autonomy that Physically Assistive Robot provide. Therefore, other authors have also focused on the need for personalization and adaptation for better user assistance. Kapusta *et al.* [143] present a task optimization of robot-assisted dressing, TOORAD, where a plan is generated for both the person and the robot, by using a simulation model including geometry and kinematics of the human, the robot, and the environment. This helps to provide personalized plans for users. A study with six participants with physical disabilities, with the system successfully assisting four of them. The authors state the need for variation on the forms of assistance, which we try to assess in this chapter.

Moreover, personalization and user preferences in HRI has also been widely acknowledged in a broader range of topics. Tapus *et al.* [144] developed a Socially Assistive Robot (SAR) to assist, encourage and interact with post-stroke patients in rehabilitation. The personality of the robot is modified to match the user's one, modifying elements such as the levels of extroversion and introversion. They also modify robot proxemics and speed and vocal content, similarly to some of our proposed preferences. The authors demonstrate that users prefer the robot that has an adapted behavior to match their personality, leading to improved task performance. Moro *et al.* [145] propose a behavior personalization method for SAR consisting of the combination of Learning from Demonstration and Reinforcement Learning (RL). In it, the caregivers demonstrate different behaviors for the robot, which are used to learn general behaviors. Then, RL is used to obtain a policy that selects the most appropriate behavior given the user's cognitive level. Preferences for hand-overs are analyzed by Cakmak *et al.* [146], where robot and object configurations are adapted to the user preferences that relate to the position and orientation of the object. Lee *et al.* [147] carried out a long-term field experiment with a snack delivery robot. Participants interacted with the robot over two months and the robot adapted their social interactions. The robot was able to remember the user's favorite snacks, usage patterns and robot behaviors. Results showed an increase in rapport with the robot and the users and increased participants' cooperation. Learning of user preferences based on ratings is performed in [24], where preferences are learned from user ratings over time. Three models of users are defined, which integrate preference profiles. They show its applicability in preferences over light animations in a mobile robot. Chevalier *et al.* [148] designed a personalized HRI environment for individuals with autism spectrum disorder. The user's personality (introversion and extroversion) is used to create a model that predicts user preferences by Cruz-Maya and Tapus [149]. The user evaluation performed shows that the behaviors generated by the model were more preferred by the users. Preferences for social interaction parameters such as distance and speeds are evaluated by Rossi *et al.* [150], where comfortable stopping distance is evaluated against human pose and user personality. The authors state the importance of learning the preferences in order to adapt the robot behavior, which would foster acceptance by the users and safety feelings. User preferences for robot motion planning are learned by Wilde *et al.* [151]. User path ranking is used to learn user constraints regarding the task, obtianing the preferred path for the user. Similarly, Hayes *et al.* [152] learn user preferences over continued interaction to improve navigation tasks. Preferences for service robots are studied by Torres *et al.* [153], where a preference reasoning is proposed. Preferences over service robotics scenarios such as object placement and user tastes are used to interact with the user.

In addition to all this, assistive robotics has the problem of acceptance. Society is still reluctant to accept social robots for domestic purposes, as analyzed by de Graaf *et al.* [154]. The authors state that it is vital to include opinion of future users in order to adapt the robots to their preferences. A long-term acceptance study with older adults was performed by Piasek and Wieczorowska-Tobis [155], where high acceptance was demonstrated by users in need of social assistance, with this user segment being open to facilitating technologies. Another study in the same direction was performed by Smarr *et al.* [156]. In it, twenty-one older adults completed questionnaires regarding their preferences and openness to assistive robotics. The results show a preference for instrumental tasks such as housekeeping or medication reminders but were less open to receiving assistance in tasks such as shaving or hairdressing. A similar analysis is performed by Deutsch *et al.* [157], where a qualitative study with thirty cognitively-able individuals. Results show many opportunities for home robots, and some user needs that feel threatened by the inclusion of such devices. However, the current generation shift may lead to inconclusive results when experimenting with current older adults. To this end, Gessl, Schlögl and Mevenkamp [158] study the perceptions and acceptance of future older adults. The study spans 188 users from 20 to 60 years of age, where they found correlations between age, gender and personality with technology acceptance. The authors state that personality plays a significant role in the acceptance of assistive robotics technologies, which we believe strengthens the need for personalized and preference-based robotic behaviors. Other similar studies like the one by Biswas *et al.* [159] try to assess whether older adults are different from young users when interacting with robots, in order to understand how communication preferences change by age. Results show similar preferences for the older people and under 21 samples than the middle group aged from 22 to 64 in elements such as preferring speech over a tablet use.

Although many important works have focused on the use of preferences, we are not aware of any of them analyzing how well can the users assess the correct use of their own preferences, which is what we study in this chapter. We believe this is a key factor to increase user acceptance and improve the overall interaction and assistance.

7.3 Methodology

While it may seem apparent that preferences and adaptation can improve a Physically Assistive Robotics task, there are not many experiments proving so. However, the design of a system able to modify its behavior based on user preferences does not imply such a system to be adaptive and consistent with the preferences, as this is a subjective measure of the user. And, although the user may be able to distinguish the behaviors while comparing them, this would not usually be the desired case for assistive robotics at homes, where the system should be more plug-and-play and managed by non-technical users, without the need of knowing how are the different preferences translated to actual robot behaviors.

Therefore, the main purpose of this study is to determine whether a user can identify when the robot's behavior is guided by his/her own selection of preferences, given our system and setup. We also want to know if the preferences give a feeling of better assistance. Finally, we want to find out if the behaviors produced by different preferences are observed as different by the users, to see not only whether the user can see when their preferences are used, but also when they're not and the different behavior is observed.

Given this, our hypotheses are: (H1) the preferences modify the behavior of the robot in an expectable way, (H2) the interaction with the robot is more pleasant when preferences are used, and (H3) different preferences will produce clearly different robot behaviors identifiable by the user (regardless of their beliefs about the use of their preferences).

Accordingly, the input parameters will be the chosen preferences by the user. These will be fed into a task planner that will choose the different actions to be executed based on the preferences provided.

The following subsections will detail the setup, scenarios and experimental methodology.

7.3.1 Scenarios

We have used the three Physically Assistive Robotics tasks defined in Section 1.3 to be carried out by the participants. In what follows, we detail how the tasks have been used to evaluate the use of the preferences along with a description of the preference values that the user could choose per each task. The preferences are the ones used over all the thesis, and extracted from the taxonomy defined in Section 3.3. The actions in the domain are the ones defined in

Section 6.7.1. Examples of the setups for the different tasks are shown in Figure 7.1.

- **Assistive feeding:** The robot feeds the user one spoonful. The spoon is empty for the experiments. The task-related preference is the proximity, which states whether the user wants the robot to be close, far or at intermediate distance. This is used by the planner to decide whether to use an action that inserts the spoon in the user's mouth, or another one that lets the user get the food while the robot waits. More details on the task can be found in Section 1.3.1.

- **Shoe fitting:** The robot approaches a shoe to the user, who prepares the foot in front of it, and the shoe is fitted. Then, the robot releases the shoe and the task is finished. The preference to be chosen relates to the movement of the right foot, which could be either nothing, a bit or a lot/normal movement. Users were allowed to freely simulate an ankle injury that prevented them to move the foot. The robot behavior changed in the way the shoe was fitted, and in the amount of force that was applied against the foot. More details on the task can be found in Section 1.3.2.

- **Jacket (open sleeved garment) dressing:** The robot has the garment grasped with two arms, and it dresses the user. Possible behaviors are starting with one arm or the other, or inserting both together. This could be done in each part of the task, be it the sleeve insertion, lower arm dragging or upper-arm dressing. In this task, the user could choose how much the right arm could be moved, again simulating injury if desired. The possible values were no movement, a bit or normal movement. This preference guided the behavior of the robot by starting for the right arm or using actions that would not affect the arm movement, either by forcing the user to move or by stretching the arm with the garment. More details on the task can be found in Section 1.3.3.

As already mentioned, these tasks are widely addressed by the PAR community, given that

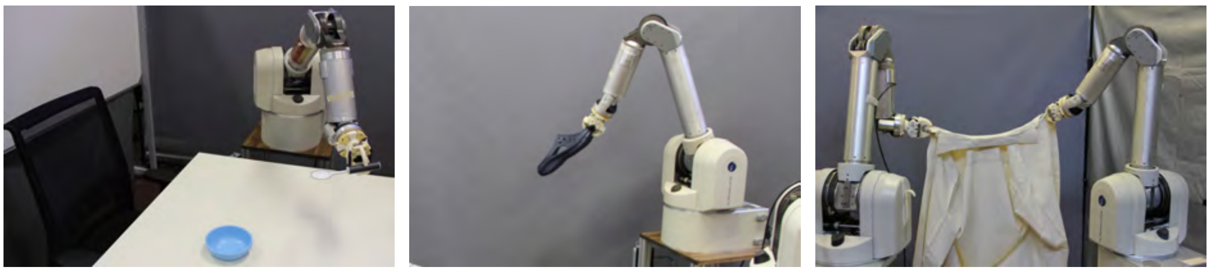

(a) Assistive feeding setup (b) Shoe fitting setup (c) Sleeved garment dressing setup

Figure 7.1: Setup used for the experiments in the three tasks.

they have a potentially high impact on the dependent users' lives as they represent some of the most basic needs without which a human cannot live with dignity, while still being safe enough to prevent user harm.

Given that we are only focused on the behavior changes observed by the users, we have safely implemented the tasks by compromising some elements in favor of safety. Therefore, we have implemented trajectories learned kinesthetically in joint space to prevent any potential misbehavior or unexpected movement of the robot due to sensor faults, misdetections or potential singularities in the inverse kinematics that could harm the user.

We believe that, although this may hinder a bit the experience of the users, it does not impede them the assessment of the behavior of the robot neither the ability to discern the cases in which the preferences are being used, while still being able to participate in the task and experience the assistance first-hand in a safe environment.

Apart from the task-related preferences, the user could choose two additional preferences for each task (but with the possibility of providing different values per each task). We decided to use the same two preferences to simplify the user's task. These preferences are:

- **Robot speed:** defines the speed at which the robot moves, using fuzzy terms. The options were slow, intermediate or fast. No demonstration of the speeds was performed beforehand, so it was based only on user intuition.

- **Robot verbosity:** specifies whether the robot should speak or not. In our scenarios, speaking relates to informing the user before each task was being performed. An example of speech is: "I am approaching your right hand". Possible values for the preference are sometimes, always or never. As this is a preference and not a constraint, the robot may decide to speak even in cases when the preference was not to speak. For instance, it may speak to inform the users that it will not insert the food into their mouth but instead wait for them to get the food while the robot stays still.

The tasks have been implemented in the robot and behaviors have been designed for each of them. The decision making is performed using task planning, and the preferences are used in the same way as described in Chapter 4 and using the ROSPlan extension described in Section 6.3. During the tasks, the robot was acting in a fully autonomous way without external interventions.

7.3.2 Material

The experiments were carried out using two Barrett WAM® robotic arms. The feeding and shoe fitting tasks were using only one arm, while the jacket dressing task employed both of them.

Additionally, each user was given a single-use spoon for the feeding task. This task also needed a table, a plate, and a chair. No food was supplied to the users, both for health and safety reasons. The shoe-fitting task required a shoe, which was chosen to be a Crocs^{TM}-like shoe, and a stool to sit the person in. Finally, the jacket dressing task was using a big size sleeved shirt that would fit all the users and was wide enough to prevent harm by the garment getting stuck while fitting.

7.3.3 Participants

Thirty participants volunteered to take part in the experiments. All of them were healthy, cognitively-aware, and able-bodied adults. Even though those users are no the target population for our applications, we believe the state of the Physically Assistive Robotics systems are not mature enough to be used with potential end-users. We needed some part of flexibility from the user as well as full cognitive capacities to understand the task, the test and what was their role in the experiment, which was to focus on the robot's behavior and see whether there were changes and their preferences were used.

Such experimentation in a care home may have been a source of stress for the patients and may have led to inconclusive results if the goal of the task was not understood, apart from possibly generating false expectations to the patients of such care homes. The fact that nowadays we are still far to see real Physically Assistive Robots helping real users made us shift the focus to younger users who may need one of these robots in the future [158]. Thus, we believe this chose of participants does not harm the experiment but helps to understand how potential future users see this kind of systems, and whether the preferences are well understood. Therefore, we specifically targeted the study to healthy people from 20 to 40 years. Those are users who are more familiarized with technology and personalization of devices such as smartphones and, thus, able to understand the goals of the experiment.

7.3.4 Procedure

The experiments were conducted at IRI's Perception and Manipulation lab[1]. The study was designed as a within-subject experiment in which the participants were exposed to different conditions of the same task. The work presented in this chapter has been approved by the Ethical Committee of the Spanish Council of Scientific Research (CSIC) in report number 025/2019.

Each user started the experiment by learning about the purpose of the study and the tasks to be performed. After learning about the experiment procedure they signed the participant consent form. Then, they were asked about their preferences for the robot behavior, and this was done before each task was executed. All the tasks were performed two times by each participant. In each of them, one execution would use the user's preferences to guide the robot's

[1]www.iri.upc.edu/research/perception

behavior, and the other execution would use explicitly different preferences to the ones selected. There was also a control task for each user in which the chosen preferences were not used, and both executions of the task were using the same set of preferences. This control task should show no perceived differences in the robot's behavior. The user did not know in which run would the preferences be used and was only told that it may be that there were used in the first execution, the second, all of them or none. After each task, the user would answer some questions regarding the task, to state where the preferences were used (where options were in the first run, in the second, or dubious).

In order to balance the executions, the following strategy was used. Three possible cases were defined for each task:

- **CASE A:** The task will be executed two times without preferences.
- **CASE B:** First execution will use the preferences, the second won't.
- **CASE C:** First execution will not use the preferences, the second execution will.

Then, we allocated the cases to the tasks such that each user was experimenting the three cases (one per each task) and balancing between tasks. Therefore, the first user would start with the tasks following cases A, B, and C, the next one would experiment cases B, C and A, and so on. This way, all the tasks were allocated the different cases for different users. Therefore, it was not possible that one task was always executed without preferences, neither that it included preferences in the same order in all the executions.

At the end of the three tasks, the users completed the survey by answering a randomized set of questions from the Almere model [135]. The full questionnaire used to survey the users during the experiments is reproduced in Appendix D.

7.4 Results

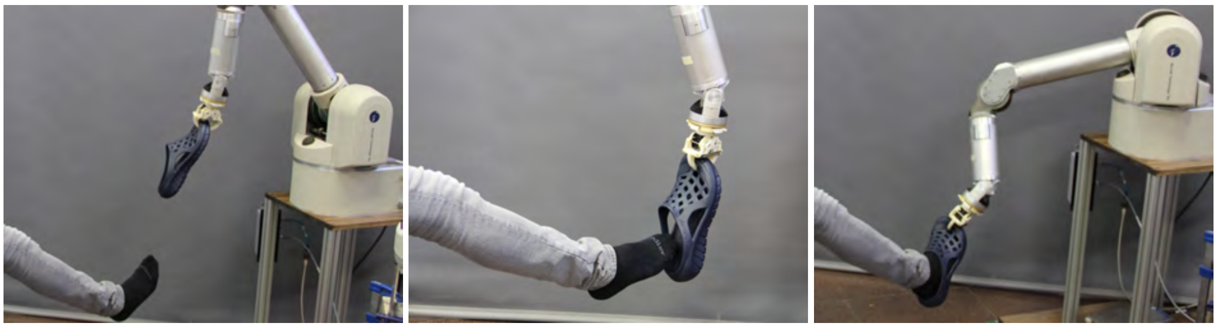
Thirty users (86% self-identified as male) aged between 20 and 39 (mean of 26) years old participated in the experiment. Images of some participating users performing each task can be seen in Figure 7.2. For a more detailed example refer to the video demonstration[2].

7.4.1 Preference guessing

The first question the users faced after each task had been executed two times was to tell in which of the trials they believed their preferences had been used. Their options were that the

[2]The video demonstration showing different users performing the three assistive tasks can be found at www.iri.upc.edu/groups/perception/assistivePreferencesEvaluation

(a) User feeding example.


(b) Fitting a shoe to a user.

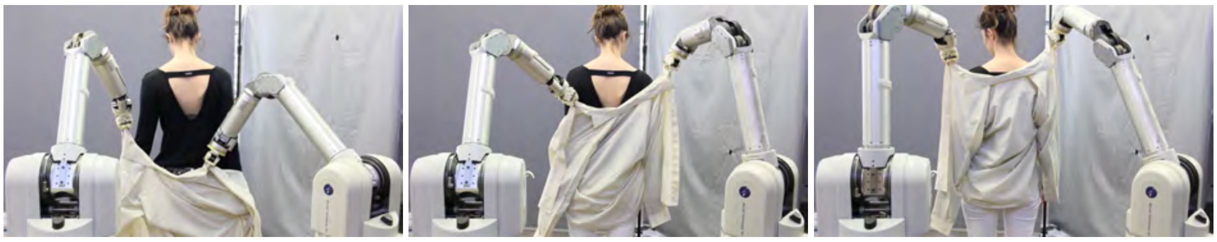
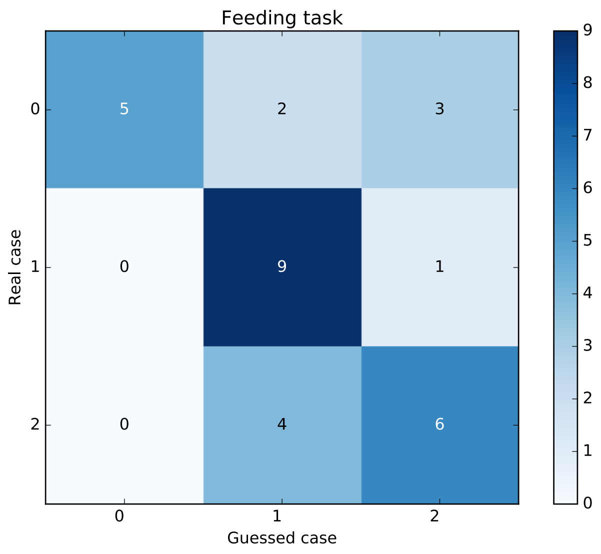

(c) Dressing of a sleeved garment.

Figure 7.2: Users participating in the experiment.

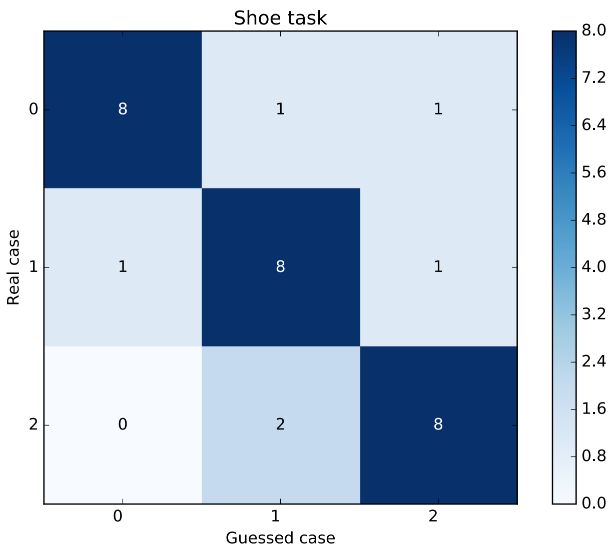
preferences were used in the first trial, in the second or they were not sure (i.e no difference found).

The results are shown in Figure 7.3. As it can be seen, there were 66.67% of correct guesses for the feeding task, 80% for the shoe task and 70% for the jacket dressing. If joined together, the total amounts to 72.22% of successful guesses in the trial where the preferences were used.

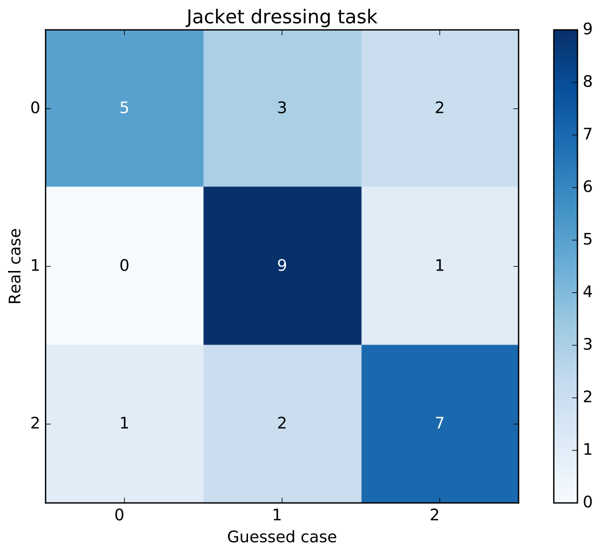
As it can be seen in the plots, most of the cases were successfully identified. The most misguessed case was the one in which no preferences were involved. The observed behavior of the users was that sometimes, even when they were displaying a clear doubt, they were selecting trial 1 or trial 2. Asked after the test, they state that they thought that one of the two trials had to involve preferences as they were asked in the beginning and that they may have not noticed something during the task execution. Even with it, all the cases were correctly identified, being the feeding task the most difficult one. We attribute this to the fact that being the most dangerous task, the differences in speed and movements were not that much distinguishable. To assess significance, we performed a Chi-squared test on the correctly guessed results with a confidence level of 95%, obtaining a p-value of $0.006 < 0.05$.



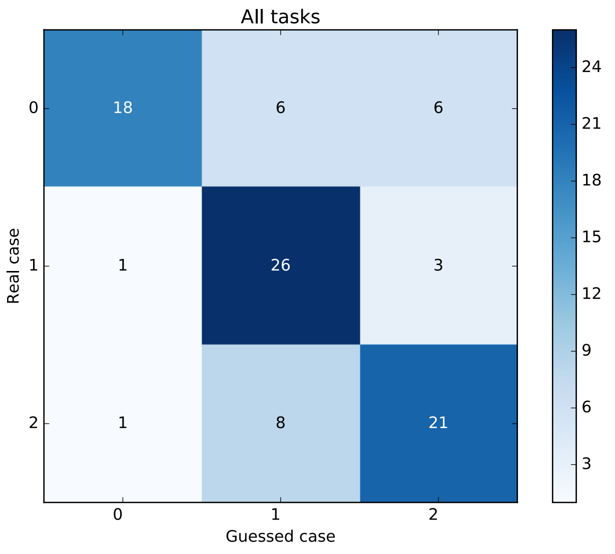
(a) Guesses for the feeding task



(b) Guesses for the shoe fitting task.



(c) Guesses for the jacket dressing task.



(d) Combined guesses for all the tasks.

Figure 7.3: Results of guessed preference trial per task. The 0 represents doubt/no preferences, 1 represents in the first trial and 2 in the second one.

	FEEDING	SHOE	JACKET	ALL
5	20.00%	23.33%	**36.67%**	26.67%
4	**43.33%**	**33.33%**	**36.67%**	**37.78%**
3	20.00%	30.00%	26.67%	25.56%
2	6.67%	10.00%	0.00%	5.56%
1	10.00%	3.33%	0.00%	4.44%

Table 7.1: Frequencies of the 5-point Likert scale for the pleasantness of interaction. Considering all results.

	FEEDING	SHOE	JACKET	ALL
5	**40.00%**	**43.75%**	**56.25%**	**46.81%**
4	20.00%	25.00%	37.50%	27.66%
3	26.67%	18.75%	6.25%	17.02%
2	6.67%	12.50%	0.00%	6.38%
1	6.67%	0.00%	0.00%	2.13%

Table 7.2: Frequencies of the 5-point Likert scale for the pleasantness of interaction. Only correctly guessed state when preferences were present are considered.

	FEEDING			SHOE			JACKET			ALL		
	OP	NP	SGP	OP	NP	SGP	OP	NP	SGP	OP	NP	SGP
MEAN	**3.85**	3	3.8	3.8	3.3	**4**	4.4	3.5	**4.5**	4.02	3.27	**4.11**
MEDIAN	**4**	3.5	**4**	**4**	3	**4**	4.5	3	**5**	**4**	3	**4**
MODE	4	4	**5**	**5**	3	**5**	**5**	3	**5**	**5**	3	**5**

Table 7.3: Descriptive statistics for the pleasantness. OP are the results only considering the cases where preferences were present. NP are the cases where preferences were not present. SGP are the results when there were preferences and the users correctly guessed in which trial.

7.4.2 Pleasantness of the interaction

After each task, the users were also asked about the pleasantness of the interaction when preferences were present. When no preferences were available, the users tended to answer either a low score or middle score to express neutrality.

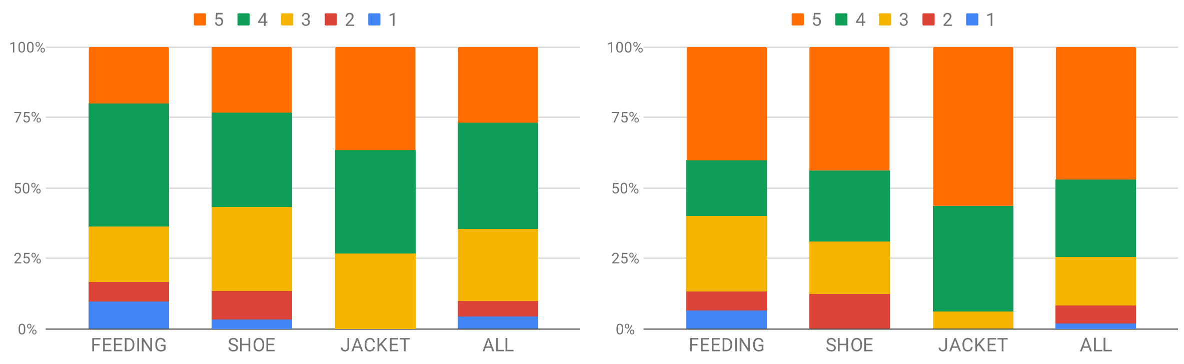
Figure 7.4a and Table 7.1 show the results obtained for the pleasantness of the interaction considering all the answers, while Figure 7.4b and Table 7.2 show only the answers in the cases where the preferences were successfully identified (and there were preferences). It can be observed that the pleasantness levels increase when the preferences are correctly identified, as when taking into consideration all the answers there are more low scores, while considering only the guessed results the good scores (values 4 and 5) are more present. The effect is less clear in the feeding task, but there is still a clear increase in the pleasantness. These results would support our hypothesis H2, where the use of user preferences leads to a more pleasant task. Table 7.3 provides further descriptive statistics on the obtained Likert-scale responses, showing that the results were higher in the cases where preferences were present. also supporting that when preferences are not present the overall pleasantness is lower. A Chi-squared significance test gives us confidence in the results, obtaining a p-value < 0.05 both for the aggregated results of all the tasks and for the shoe fitting and jacket tasks independently.

7.4.3 Differential behavior

Our last hypothesis was that the use of preferences would lead to clearly different robot behaviors, independently of the chosen preferences. To assess this, we asked the users about whether the behavior was assessed as different using a 5-point Likert scale question. We would expect high scores for cases with preferences, while the cases in which preferences were not present should show disagreement. Figure 7.5 and Tables 7.4–7.5 show a comparison of the obtained results when preferences are not present and when they are applied. As it can be seen, there's much more confusion (a score of 3) and disagreement when preferences were not used, while the use of preferences shows more agreement in the difference of behaviors between the first and second trial. Similarly, Table 7.6 shows the same trend, where cases using preferences were identified as having a clearly different behavior. In this question, there is no difference regarding the correct guessing of the use of the preferences. A Chi-squared test for this question also showed significance for the shoe-fitting, jacket dressing and aggregated results, for which we can consider our hypothesis accepted.

7.4.4 Potential usefulness of Assistive Robotics

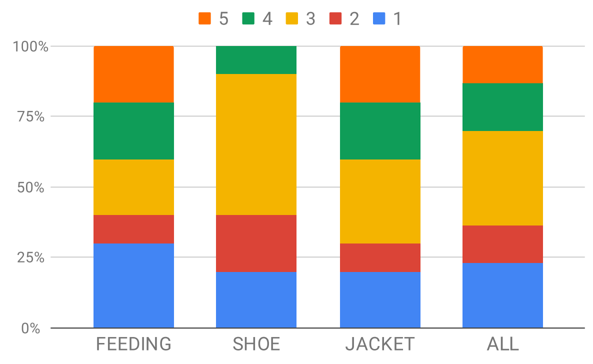
Our targeted users did not require this kind of assistance, but everyone can relate or think of cases in which PAR may be of use. Therefore, we asked them about their beliefs on the potential uses of these assistive robotics tasks.



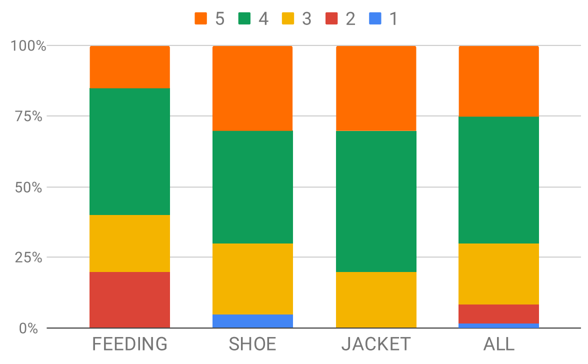
(a) Pleasantness answers

(b) Pleasantness answers considering only successfully guessed state (with preferences)

Figure 7.4: Results of the pleasantness when using preferences. Observe that 4 and 5 are the dominant answers overall. The results improve with an increase of agreement answers when only the successfully guessed state is taken into account, meaning that the use of preferences increases the task pleasantness.



(a) Behavior difference when preferences are not present



(b) Behavior difference when preferences are present (independent of correct guess)

Figure 7.5: Results of behavior difference per task with and without preferences. Note that when preferences are present, the users agree on an observed behavior difference, while when no preferences are present the users get more confused.

	FEEDING	SHOE	JACKET	ALL
5	20.00%	0.00%	20.00%	13.33%
4	20.00%	10.00%	20.00%	16.67%
3	20.00%	**50.00%**	**30.00%**	**33.33%**
2	10.00%	20.00%	10.00%	13.33%
1	**30.00%**	20.00%	20.00%	23.33%

Table 7.4: Frequencies of the 5-point Likert scale for the difference of behavior. Only when no preferences were applied

	FEEDING	SHOE	JACKET	ALL
5	15.00%	30.00%	30.00%	25.00%
4	**45.00%**	**40.00%**	**50.00%**	**45.00%**
3	20.00%	25.00%	20.00%	21.67%
2	20.00%	0.00%	0.00%	6.67%
1	0.00%	5.00%	0.00%	1.67%

Table 7.5: Frequencies of the 5-point Likert scale for the difference of behavior. Only when preferences were present regardless of user's guess.

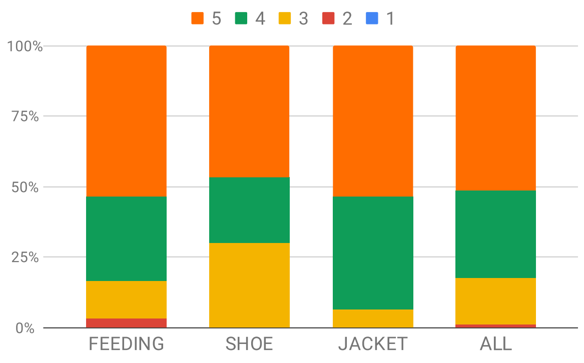
The collected answers are summarized in Figure 7.6. Both charts show a clear opinion in favor of the use of Physically Assistive Robots to help dependent people. Even though our experiment was using a prototype and some user adaptation was removed, most of the users agreed that the use of preferences such as the ones proposed in this thesis can improve the assistance for those in need. In a more general question, they also confirmed that those applications can be helpful for dependent users. As it can be seen in Figure 7.6a, the majority of the users agreed that the preferences improve the assistance. Similar results are observed in the helpfulness, where users also confirmed that these tasks can provide effective assistance to users in need.

Finally, we have used the Almere model [135] to get further insights into the acceptance of this kind of robots. Even though the model was designed for older adult users, we wanted to see whether we may get some insights into the potential future acceptance of assistive robotics arms among older adults and other dependent users. Therefore, our users also answered the questions from the Almere model. The questions were provided randomized and adapted to the experiment at hand, appealing to their imagination in some questions, where they were told to put themselves in the role of a dependent user. The Social Presence (SP) construct was not included in the questionnaire given that there was no conversation between the robot and the user.

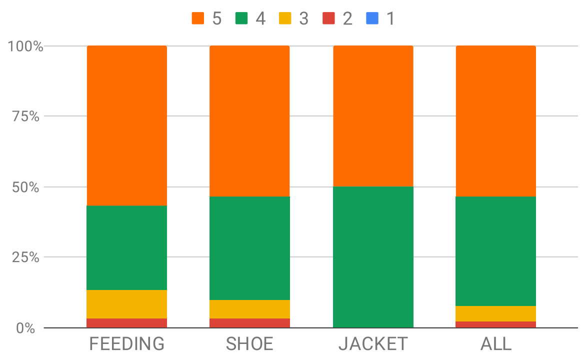
The obtained results are summarized in Table 7.7. Even though not all the constructs provided reliable answers, we believe they give us good insights into what the users think about the system. Above all, we want to notice the constructs as Anxiety (ANX - evoking anxious reactions) which shows the lowest scores. The Attitude Towards Technology (ATT) shows positive feelings about the applicability of the technology, being the one with the highest score. Trust and Intention To Use (ITU) show a wider range of opinions, while the Perceived Usefulness (PU) and Perceived Adaptivity (PAD) give insights of good acceptance of the system and perception of a useful and adaptive system, which is even enjoyable (PENJ).

	FEEDING			SHOE			JACKET			ALL		
	OP	NP	SGP	OP	NP	SGP	OP	NP	SGP	OP	NP	SGP
MEAN	3.55	2.9	**3.6**	3.9	2.5	**4.19**	4.1	3.1	**4.19**	3.85	2.83	**4**
MEDIAN	**4**	3	**4**	**4**	3	**4**	**4**	3	**4**	**4**	3	**4**
MODE	**4**	1	**4**	**4**	3	**4**	**4**	3	**4**	**4**	3	**4**

Table 7.6: Descriptive statistics for the behavior difference. OP are the results only considering the cases where preferences were present. NP are the cases where preferences were not present. SGP are the results when there were preferences and the users correctly guessed in which trial.



(a) Perceived improvement of the assistance of dependent people by the use of preferences



(b) Perceived helpfulness of the applications for dependent people

Figure 7.6: Answers for the improvement of the preferences and the perceived helpfulness of the assistive tasks.

Construct	Mean ± SD	Alpha	Construct	Mean ± SD	Alpha
ANX	2.71 ± 0.85	0.72	PEOU	3.81 ± 0.56	0.46
ATT	4.33 ± 0.54	0.68	PS	3.02 ± 0.74	0.64
FC	3.51 ± 0.74	0.31	PU	3.99 ± 0.59	0.55
ITU	3.02 ± 1.08	0.91	SI	3.87 ± 0.64	0
PAD	4.08 ± 0.44	0.20	Trust	3.42 ± 1.05	0.78
PENJ	3.92 ± 0.70	0.77			

Table 7.7: Results of the Almere model [135] analysis.

7.5 Summary

Physically Assistive Robots may have a huge impact on future society and home care for dependent people. In this chapter, we have carried out a user evaluation with thirty healthy subjects in order to assess the effect of the use of preferences to modify the behavior of the robot. Our experiment intended to determine whether the users were able to assess when a robot was using their chosen preferences in an assistive task and when it was not. Moreover, we evaluated the impact of such preferences in the perceived pleasantness of the task, and also the opinions on future usability of the proposed assistive tasks.

The obtained results allow us to confirm that most users are able to determine when their preferences are being used, meaning preferences are self-explanatory even when the users did not have any previous experience to compare with. We have also observed that in the absence of preferences, some users may try to assess that there were preferences even when there were not. And even in that case, the results show a clear understanding of the use of the preferences by the robot. When it comes to pleasantness, we can conclude that the use of preferences led to a more pleasant sensation for the users. Furthermore, the users show clear agreement in that applications such as feeding, shoe fitting, and dressing assistance will be helpful for dependent users, and preferences and personalization will improve this assistance. Finally, regarding the acceptance, we can't have conclusive results given that our system was a prototype and our users were not end-users, but our preliminary results on the topic show that the users can trust the system without having much anxiety over it. The system was intuitive and easy to use, for which most of the users would use a system like this in case of need.

Given the obtained results, we can confirm our hypothesis and conclude that our use of preferences for physically assistive tasks modify the behavior of the robot in an expectable and legible manner and that doing so results in a better experience for the user. We have also observed that different preferences produce clearly different behaviors, as expressed by the users. We have gotten important insights on how the preferences can play a role in physical assistance. However, we believe this study should not end up here, and a follow-up would be helpful to get further intuition by extending it to users with some kind of disability or dependency degree, but still with full cognitive capacities in order to understand the goals of the study. For this, a more complete system would be needed, with full vision capacities to interact and adapt to the user and safe control strategies to prevent any harm.

This chapter has wrapped up the methods explained over all the chapters in a final user evaluation. In it, we have seen how the Execution tuning step of the FUTE framework of Chapter 2 successfully adapts a pre-trained robot to the current user while performing a Physically Assistive Robotics task. We have seen how the preferences of the taxonomy defined in Chapter 3 are employed in this personalization scenario, and applied the planning techniques for robot behavior adaptation explained in Chapters 4 and 5. Furthermore, we have done so by using the ROSPlan extensions detailed in Chapter 6 and Appendix C.

8 Conclusions

This thesis has been devoted to robot behavior adaptation and personalization to a user. And more specifically, to adapt the actions of Physically Assistive Robots while helping users perform their Activities of Daily Living. A video demonstration of the robot performing the three assistive scenarios is provided as supplementary material[1].

Assistive Robotics has the potential of greatly improving the lives of many patients around the world. In the current global context, with an aging workforce, a lack of nurses [1,2] and the expectancy of an increase in stroke cases in Europe and the rest of the world [8], such assistive devices will be more needed than ever. Still, there are extra advantages to the rise of such devices apart from helping this potentially depleted healthcare workforce. We strongly believe this has many benefits for the users, which is what drove this thesis since its beginning. As described by Williams *et al.* [10], dependent patients that suffered a loss of independence may have a great psychological impact due to their condition: loss of self-worth and self-identity, feelings of burden and guilt. Similarly, Boström *et al.* [11] found an association between depressive symptoms and the lack of independence in the tasks of transfer and dressing. Therefore, the possibility of providing such users in need with a device able to help them regain their autonomy and independence would greatly impact their well-being.

The overall contributions of the thesis can be summarized as follows:

1. The definition in Chapter 2 of a methodology for robot behavior adaptation and personalization, which we called FUTE. The framework defines how and when we should personalize robots, considering that non-technical users such as healthcare professionals will be the ones readapting the robot. The framework was then used to define the kind of preferences that would be involved in assistive tasks. This was done in Chapter 3 by the description of the preference taxonomy. This categorization permits defining any kind

[1]The demonstration video can be found at youtu.be/uiKl2Q1PzOk, and examples of behavior adaptation can be seen at youtu.be/mVvnigdPJPQ

of preference. We argue that user limitations may also be included there in the form of preference, thus resulting in a comprehensive and complete classification of preferences and needs. This is important as it grounds the concepts and definitions and lets us establish a common language to speak about preferences. The applicability of the preferences has been analyzed, both in Chapter 3 but also in the rest of the thesis where the defined preferences have been successfully used to alter the robot behavior. Chapter 7 confirms that the selection of preferences is descriptive enough for the users to recognize when their selected preferences are being used, provides a better experience, and improves the performance of the system.

2. The development of planning techniques for behavior adaptation. We have shown how the use of task planners is convenient for assistive robotics tasks. Given the inherently dangerous nature of these tasks, planning how the task will be performed is rather essential and more robust than a reactive behavior (which is also needed). Moreover, the use of such techniques allows us to define different behaviors and combine them. We have exploited the use of planners and joined them with the defined preferences. Therefore, we can define different equivalent actions with different performances and let the planner choose which is the most suitable one. This has been done in Chapter 4 by linking the action outcomes and costs with the preferences acquired from a user model. The model is filled by a Fuzzy Inference System, which is the result of asking two simple unrelated questions to the user. We also show how these costs and outcome probabilities can be used to re-adapt the behavior to changes in the user. This is helpful in two ways: first, it allows the system to cope with wrongly inferred user models and secondly, it is flexible to changes in the user's behavior, adapting to it again. Therefore, the method works effectively for long-term adaptation to the user. We further develop this in Chapter 5, where the algorithms are combined with low-level reactive controllers that adapt online to user movements. We show how this approach results in easier teaching of the robot with fewer demonstrations needed, and that the symbolic description is also easier, with a lower number of actions and symbolic states. These methods provide the robustness that physically assistive domains need as well as seamless user adaptations.

3. An extension of the ROSPlan framework for probabilistic planners and the introduction of suggestible planning predicates. ROSPlan's extension allows the use of more languages and paradigms with robotic environments in an easy and familiar manner, which we believe is a contribution to both the robotics and AI planning communities. With it, we were able to use RDDL for richer task definitions. RDDL allowed us to define reward functions that depend on the previously described preferences. Such preferences are then

linked to actions by value, resulting in the planner selecting the actions that comply with each preference's value. With these methods, Chapter 6 defined the SoPS algorithms to provide suggestions of planning predicates (represented as preferences). The algorithms perform a systematic analysis of a subset of the space of plans and can manage partial assignments of the preferences. Thus, this allows the system to suggest new preferences based on the values of the already known preferences, and those suggestions would result in improved task performance. Furthermore, we believe the proposed algorithms can be useful for the XAI and XAIP communities, as the suggestions and plan space tree definition can be used for plan explanation. This also complements the FIS initialization proposed in Chapter 3, as the FIS method would not scale well for many more preferences.

4. A novel HRI study on assistive robotics and preferences. To the best of our knowledge, this is the first one involving assistive robotics and user preferences to modify the robot's behavior. In the study, described in Chapter 7, we revealed that the users can successfully identify the changes in the robot behavior. Moreover, they can do it without previous interactions with the assistive execution of the robot and using only their own intuition. This confirms that the methods developed during the thesis can be used for behavior adaptation in assistive robotics scenarios and that and that we are developing the definitive tools for the caregivers of the future.

Overall, this thesis has analyzed the whole process of behavior adaptation with assistive robots, from the description of the personalization process and definition of preferences to the use of Artificial Intelligence techniques for the acquisition of the preferences and the personalized action selection. Our approach has been to define a reward function (or total plan cost) such that it depends on the preferences, those being linked to the available actions. Then, we leave all the reasoning to the solver, usually a planner, which then decides how to account for the preferences. This novel approach doesn't directly define preferences as soft goals but integrates them into the domain and metric function definition. This makes the planner try to optimize such function, as it would usually do, and doing so it also optimizes the use of the preferences. Therefore, our algorithms do not require a planner that is compatible with PDDL3. Instead, this thesis has included standard PDDL, PPDDL, and RDDL throughout.

Finally, the main theme of the thesis has been evaluated in a real robot environment with real users in three different assistive scenarios: assistive feeding, shoe-fitting and, jacket dressing, while exploring other aspects such as safety (Appendix B) and applicability of AI planning paradigms (Appendix C).

8.1 Future work

This thesis has studied the personalization of assistive robots from its definition to its implementation. Nevertheless, there is still much room for improvement and research directions to follow to be able to integrate this kind of device into society. Some of the potential future extensions of this thesis are:

- **Preference elicitation and inference:** There is still much room for improvement in the acquisition of the preferences. The FIS proposed in Chapter 4 is one manner of doing so. Other possible techniques include preference mining from user data and using recommender systems techniques such as collaborative filtering as done in [26]. The main problem of such approaches is data acquisition, as well as generalization. Other approaches include the use of games for probing the users, assessing their behavior in front of the robot and checking their limitations without the users noticing. We believe approaches like this would work better than plainly asking for the preferences as, probably, not even the users will know their preferences regarding the behavior of the robot.

- **System evaluation by potential users:** We evaluated the use of preferences in Chapter 7 with promising results. However, as discussed in the chapter, we believe these autonomous assistive systems are not ready nor safe enough for testing with real dependent users. Therefore, we leave it for future work and hope to see advances in this direction soon, as that will be the final test to assess their usability. This does not only apply to the users, but also to caregivers and non-technical professionals who should be able to provide the adaptation. Chapter 2 presented our personalization framework and argued that non-technical users should be able to kinesthetically re-teach the behaviors, but a thorough user analysis with healthcare professionals should be performed. This was out of the scope of this thesis, which was focused on the autonomous adaptation of the robot.

- **Benchmarking and evaluation:** Regarding the evaluation, we believe that the community should also focus on the development of benchmarks and objective evaluation tools. Although user studies provide insightful conclusions, we advocate for the creation of methodologies that allow the *objective* analysis and comparison of approaches for robot behavior adaptation to the user as well as the effectiveness of the assistance provided to the user. Through standardization of competitions, this would promote fair research and advances in such topics.

- **Safety:** Another key point in assistive robotics is safety. Chapter 5 added some robustness to the adaptation which is linked with safety, and an initial analysis was developed in

Appendix B. Adapting to the user in a safe manner is a must, even when user's own preferences are in conflict with that. Moreover, behavior adaptation could also be used to provide a safer performance rather than only using the preferences. And apart from that, a general analysis and definition of procedures for safety in Physically Assistive Robots will bring these robots closer helping users at their homes.

- **Adaptive compliance:** Linked to safety, an extra level of adaptation would be the one of compliance. Modifying the stiffness factor of the robot during the development of the task would add an extra layer of adaptation and safety. For now, we modified the stiffness factor based on user preferences for entire tasks (see Chapter 7), but doing so during a trajectory could be a good improvement.

- **Better communication between low and high-levels:** Chapter 5 showed the benefits of having synergies between low-level adaptations and high-level ones (during planning). However, we believe such integration could be further analyzed. For instance, sharing the probability values computed for the low-level trajectory in the high-level planner, or even using the likelihood of a generated movement primitive to assess the chances of the action succeeding. This would then allow the learning of such events at both levels, simplifying, even more, the development of the robotic task.

- **Computer vision:** We have not tackled the computer vision methods in this thesis but used already available software or built simple segmentation-based algorithms to get the needed information. However, visual sensing is an essential part of the assistive process as the person needs to be correctly identified. Furthermore, user reactions should be considered in order to react to them and better adapt the system. As an example, the feedback input from the user in Chapter 4 could be combined with user sensing. Thus, emotion recognition and affective computing techniques would be really helpful in assistive domains. Moreover, specific algorithms for each assistive task should be developed to have a final and robust system.

- **Natural interaction:** Similarly, the use of Natural Language Processing (NLP) techniques would greatly improve the interaction. We are used to interacting by means of speech and gestures, and having such an interface would help the use of the system. And not only speech but detecting sounds of approval, denial or groans as non-verbal audible cues would complement emotion recognition techniques to detect the user response to the system. Besides, other non-verbal cues such as gestures provide a natural interaction with the user [160] and would effectively complete an assistive system.

We hope that the research performed around this doctoral thesis will help foster the advances in assistive robotics, and we expect to see such devices helping people and improving our society in the near future. Nonetheless, there is still a long way to go and research to perform in this direction, and we are longing to see many improvements soon.

A
List of publications

This section details the complete list of accepted and submitted publications since the beginning of the PhD:

Journals

1. **G. Canal**, G. Alenyà and C. Torras, "Adapting robot task planning to user preferences: an assistive shoe dressing example", *Autonomous Robots*, vol. 43, pp. 1343–1356, August 2019. *Published online in 2018*.

2. **G. Canal**, C. Torras and G. Alenyà, "SoPS: Generating predicate Suggestions based on the Space of Plans. A planning with preferences example", *Submitted*, 2020.

3. **G. Canal**, C. Torras and G. Alenyà, "Are preferences useful for better assistance? – A Physically Assistive Robotics user study", *Submitted*, 2019.

Conferences

4. **G. Canal**, G. Alenyà and C. Torras, "Personalization Framework for Adaptive Robotic Feeding Assistance", in *International Conference on Social Robotics (ICSR)*, pp. 22–31, Springer International Publishing, November 2016.

5. **G. Canal**, G. Alenyà and C. Torras, "A taxonomy of preferences for physically assistive robots", in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 292–297, August 2017.

6. **G. Canal**, E. Pignat, G. Alenyà, S. Calinon and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3273–3278, May 2018.

7. **G. Canal**, M. Cashmore, S. Krivić, G. Alenyà, D. Magazzeni and C. Torras, "Probabilistic Planning for Robotics with ROSPlan", in *Towards Autonomous Robotic Systems*, pp. 236–250, Springer International Publishing, July 2019.

Workshops

8. M. Vila, **G. Canal** and G. Alenyà, “Towards safety in Physically Assistive Robots: eating assistance”, in *Robots for Assisted Living Workshop at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

B
Safety in adaptive Physically Assistive Robots

The inclusion of home robotics poses many research and ethical challenges, being this more clear in the case of Physically Assistive Robots where there may be close contact in highly sensitive areas of the body. Thus, when having such physical contacts, potential user harm is much more probable, so safety must be taken into account.

Therefore, safety is one of the base elements to build trust in robots. Accordingly, safety should be the main focus of research in the PAR community. In this appendix, we will focus on the safety aspects of a physically assistive task such as helping a user to eat autonomously. We propose safety measures in two ways. The first one is preventive, monitoring the user and ensuring to perform the actions in safe moments. The second one is focused on recovering from unavoidable issues such as impacts, stopping the robot before it can harm anybody and recovering from that in order to finish the task.

Part of this appendix was presented in [17].

B.1 Safety strategies

Given that safety is essential in any interaction task involving direct contact with a user, we have defined two main safety strategies to prevent the undesired collisions which may occur. Taking inspiration of other automation fields, we consider the passive and active safety options for PAR.

- **Passive safety** are traditionally measures used to minimize harm in case of an abnormal event. Here, we can consider two kinds of safety. The *completely passive safety* which is achieved by using a compliant robot controller such as [75, 161]. In it, the control signal based on the position error is minimal and in case of impact low forces will be applied. However, after the impact the position error is still present so the robot will continuously try to apply some force to try to reach the desired destination. Note that using this controller exists a trade-off between compliance and movement precision. Another kind is the *partially passive safety*, where the maximum contact force is limited. To do so, a force sensor (mounted between the robot end-effector and the gripper) can be used to obtain force and torque values at the end-effector, which can be used to detect unwanted contacts and react when the maximum force is exceeded.

- **Active safety** tries to prevent abnormal situations or accidents. This includes attention mechanisms where the robot will only proceed when it has the user's attention, or when the user is in the correct position to achieve the task. It also involves elements such as

	Minimum value [N]	Maximum value [N]
Enter mouth	-1.5	4
Exit mouth	-7.1	4.5

Table B.1: Force limits for trajectories of insertion and extraction of the cutlery inside mouth.

user tracking for better adaptation, and planning of the actions to perform, which can prevent dangerous situations. The inclusion of user preferences and robot personalization could also be considered as active safety, as they can prevent abnormal situations caused by unexpected robot behavior from the user.

The following sections will describe how the two safety protocols can be used to prevent undesired collisions, focusing in the feeding task.

B.2 Safety analysis for autonomous user feeding

Having the a human user in the loop, as in the case of physical Human-Robot Interaction, necessarily involves potentially unexpected movements or reactions. These can ultimately result in unintended collisions with the user, resulting in potential harm or discomfort.

From the tasks proposed in Section 1.3, the feeding one is the most sensible to unintended collisions as it is inherently invasive due to the insertion of the cutlery in the user's mouth [162]. Furthermore, the proximity of the end-effector of the robot to the head of the person makes it necessary to add extra safety measures. Therefore, we will use this task as an study example of other safety measures.

The feeding interaction for the study will develop as follows. Once the spoon is loaded with food, the robot waits for the user's attention to approach the mouth. Then, the robot moves to a *pre-feeding position* (around 20 cm in front of the mouth). The robot then waits for the user to open his mouth while he/she is looking directly to the spoon. When this occurs, the robot feeds the user, goes back and starts the process again. The loading of the spoon is done using a pre-programmed motion as it has no influence on the safety.

The attention detection has been done by detecting the a forward head orientation using the OpenFace [163] library. The mouth opening detection has been done by processing the facial landmarks obtained with the OpenPose library [164].

B.2.1 Force limitation for unexpected contacts

In order to stop the robot in an appropriate fashion, force thresholds in the direction of the contact have been set for the potentially harmful situations. In the feeding scenario those are: the cutlery insertion into the mouth where low forces are expected, and the mouth exit where force is inherently part of the task. Table B.1 shows the defined thresholds for these situations.

When a force limit is exceeded the robot remains one second in the *waiting position* that consists on gravity compensation. When finished, if the spoon is full, the robot will move to the *pre-feeding position* and wait for the mouth opening. Alternatively, the robot will go back and re-start the feeding process again.

B.2.2 Passive safety evaluation

In order to evaluate the safety measures, two experiments have been performed. The first one has been used to analyze the safety of the system and thus they have been carried out without real users. The second one involves real users in a controlled scenario.

In the following experiments, the passive measure was achieved reproducing a pre-learned trajectory and reproduced in a compliant mode, without including the tracking and low-level adaptation explained above. This was done for better evaluation of the active safety measures.

Completely passive vs. partially passive safety

To perform this experiment a picture of a person opening the mouth has been fixed on a wood panel. This wood panel is strong enough to support the robot's force without moving or bending.

This experiment consists on the robot moving towards the picture with the same movement that it performs when entering the user's mouth. However, in this experiment the robot will impact with the wood panel.

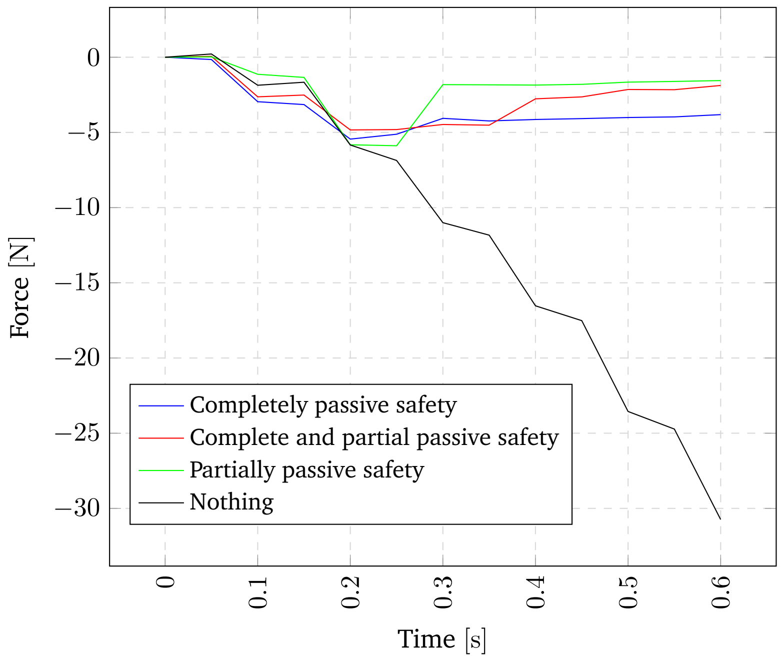


Figure B.1: Comparison of the force for the four setups in the perpendicular axis.

This experiment was performed by combining different setups including completely passive safety (compliant control) and partially passive safety by force limitation. The impact forces registered in the axis perpendicular to the user's mouth are shown in Figure B.1. As it can be observed, the setups without a compliant controller decrease similarly and have the same impact force which is $-5.8N$. The partially passive safety by force limitation setup has an increase of force after its peak and remains in $-1.6N$ as it enters the *waiting position*. On the other hand, the force in the setup without any passive security continues decreasing. The compliance setup

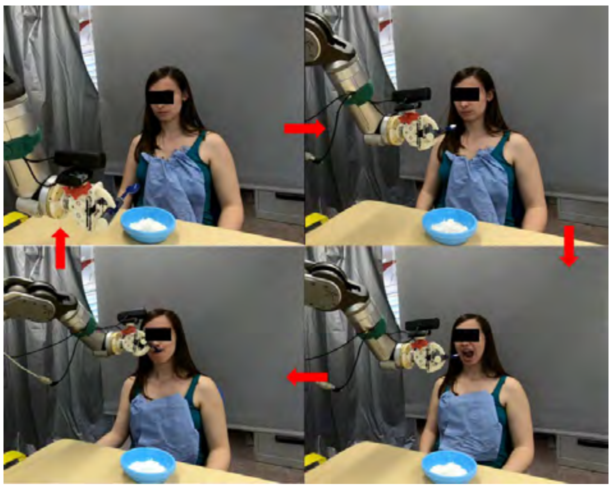


Figure B.2: Successful execution of the feeding task

decreases slower reaching a peak of $-5.1N$. After this peak it remains in $-4N$ as it is trying to reach the desired position. Finally, the compliance and force limited setup (complete and partial passive safety) has the slowest decrease reaching a peak of $-4.8N$. After 0.2s of applying a force between $-4.8N$ and $-4.5N$, the force increases up to $-1.6N$ and remains there as the robot has entered the *waiting position*.

With this results it is clear that both safety strategies offer a safe task performance as the peak forces never reach harmful levels. Therefore, setups with at least some degree of passive safety can be used.

However, during the majority of the experiments conducted with passive security there was food spilling and thus, the task could not be finished properly.

Passive safety offers a safer operation as the robot reaches lower forces. However, the difference of peak forces between the compliant and force limited is only of $1N$, so it is not a determining factor. On the other hand, passive security does not increase the applied force over time which can discomfort the user. Moreover, there exist a great difference between compliant and non-compliant setups as the ones with compliance have less precision which causes food spilling.

Pilot user study on feeding safety

A prototype of the application was tested in 104 executions with 10 able-bodied participants. Each user was asked to perform some specific tests with anomalies and some free-form tests. An example of a successful execution can be observed in Figure B.2.

With the pilot study, we evaluate the different safety strategies available. First of all, we evaluate some preventive (active) safety. To do so, users were asked to look at a side and turn the head to compute the average reaction time of the robot to a change in the visual state of the user. In the head orientation experiment, we had an average reaction time of 0.46 seconds, with all the users' movement detected correctly.

Then, the mouth openness detection was assessed in a similar way. In this case, the average response time was 0.44 seconds, although some users were not correctly detected by the face

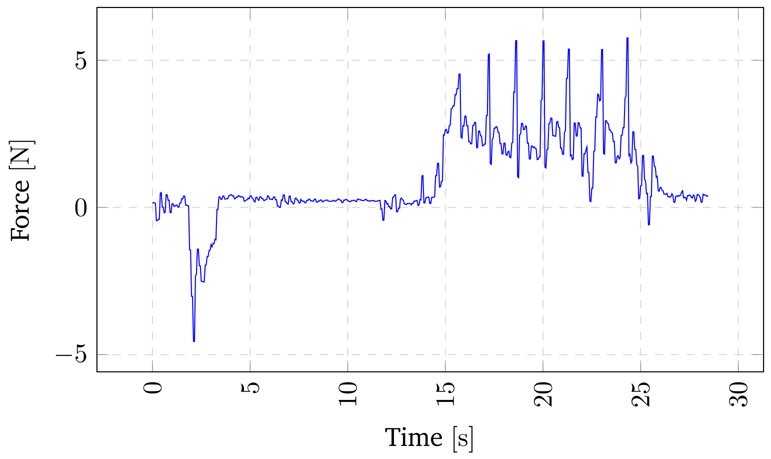


Figure B.3: Force in the y-axis (perpendicular to the user) of the impact between the user face and the spoon ($t = 2s$) and the user retaining the spoon ($t \in 15..25s$)

landmarking library, which highlights the importance of having the low-level safety exposed above.

Finally, users agreed to perform tests to assess the forces involved in contacts occurred while feeding. We performed an impact experiment and also a spoon retention one. In the first one, the robot was impacting with the user's face when entering the mouth. In the second, the user retained the spoon with their bite, not letting the robot perform the exiting motion. Note that the users were free to move away from the robot in case they felt threatened or potential harm was involved.

An example trace of the forces involved in this experiment is in Figure B.3. The first element is the impact, shown around the second 2 of the execution. In this case, the force reaches the $-4.5N$. Then, the robot remains in *pre-feeding position* and performs the motion again, this time entering the mouth (second 15). Around second 17, the robot tries to leave the mouth but feels the user retention so it enters the *waiting position* to avoid any harm, and retries until a successful exit motion can be performed (second 25). The higher peak in this case is of $5.7N$ when trying to leave the mouth.

After the experiments, the users were surveyed and all of them agreed in stating that the impacting and retaining forces were not harmful, and that they felt comfortable during the experiment. Therefore, this safety measures guarantee that in the case of an unavoidable impact, although not pleasant, it will not be harmful for the user. Moreover, it is also safe for the user to retain the spoon or even move it while it is inside the mouth.

C
ROSPlan's Probabilistic Planning evaluation

This appendix extends Section 6.3 with an evaluation on the use of probabilistic planning in robotics domains. Hence, we demonstrate the usability of the proposed extension for probabilistic domains. We provide a comparison of the performance of the probabilistic and deterministic options on the same problem under different conditions followed by a discussion on the observed results. Note that this appendix does not intend to conclude whether any planning approach is better but rather to provide insights on when one may be more suitable than others.

The ROSPlan extension has been tested in a mobile robotics scenario where we have defined a challenging *print-fetching* domain where the robot is used as a service robot for fetching printed documents in an office (Figure C.1). HRI supplements the lack of manipulation abilities of the used robot, thus allowing it to perform this task. A real-world evaluation is carried out in an environment with high uncertainty.

This evaluation is part of the work presented in [15].

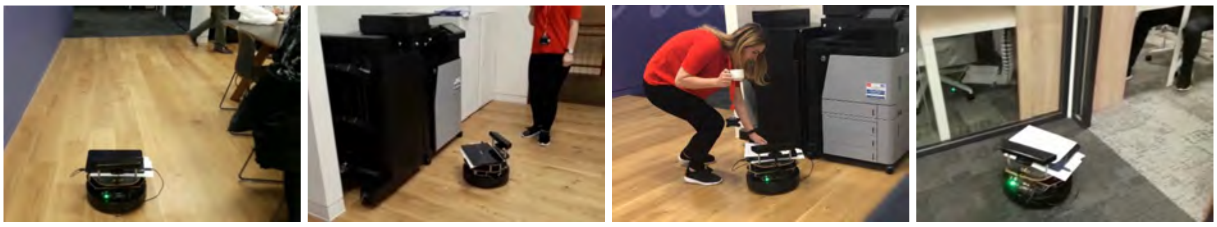
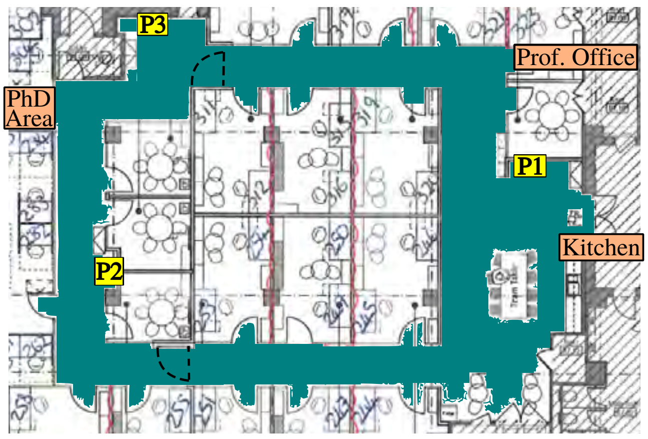


Figure C.1: The scenario in which we test the proposed system is an office environment. A mobile robot, the TurtleBot 2[1] is used for the print-fetching service. When the robot gets a request for fetching prints, it decides from which printer to collect them. Since it is not equipped with an arm, it asks a random nearby person to put prints on it, and delivers them to the user.

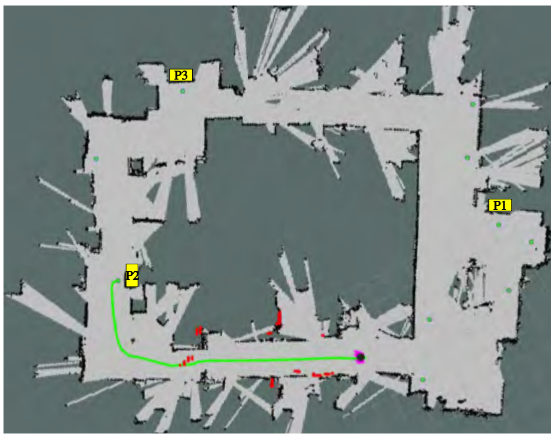
C.1 Example System and Scenario

We have used the RDDL nodes in our example scenario, using the system architecture shown in Figure 6.2. In this system, the RDDL Knowledge Base loads the RDDL domain and initial state. The Problem Interface requests the domain and state information to generate a RDDL problem instance. The Planner Interface and RDDL Plan Dispatch communicate through the IPPC server

[1]www.turtlebot.com



(a) The layout of office environment where the robot is operating. The corridor is marked with the green color and printers are marked with yellow boxes. The orange boxes denote potential goal destinations.



(b) A screenshot of the visualization tool RViz taken while performing experiments. It shows the map of the corridor and a green line indicating the robot's current path.

Figure C.2: Map layouts of the proposed scenario description.

interface, as described above, suggesting and dispatching actions. The sensing interface is also being used to instantiate the predicates based on sensor data and update the state accordingly.

To demonstrate the effectiveness of the developed framework, we have tested it in a scenario in which a mobile robot fetches printed documents in a large office building. This scenario involves a high degree of uncertainty, since the environment is dynamic and humans can obstruct the corridors and printers. The scenario also involves human-robot interaction, which is intrinsically uncertain.

Scenario description

The robot operates in a single-floor office environment with 16 offices shown in Figure C.2. There are three printers distributed along the corridor. The robot can trigger printing on any of these printers when a request is made. Since the mobile robot is not equipped with an arm, the robot can request human assistance to place the papers onto its tray. There are many employees working in this area, and the corridor is usually dynamic. The robot relies on the fact that someone will pass by and assist the robot upon request. However, it can happen that there is no one at the printer and the robot has to wait or go to another printer. Once the documents are on the carrier, the robot brings them to the person who made request. It is important to note that printers can be occupied, in which case the robot will have to wait. Moreover, the robot will know whether there is somebody there to assist or if the printer is busy until it has arrived to the printer. Figure C.1 shows an example of the scenario.

This scenario could be well-suited to be modeled as a Partially Observable Markov Decision Process (POMDP), as there are fluents that cannot be known until observed, such as the presence or absence of people near the printer. Also, it could be modeled as an Stochastic Shortest Path (SSP) problem, given that the scenario is goal-oriented in that the robot has to deliver the printed papers to a specific location. However, given the lack of available out-of-the-box solvers for both POMDPs and SSPs, we have modeled the problem as an MDP where a positive reward is given only once the goal is reached.

C.1.1 Print-fetching domain

In order to run the scenario on both PDDL and RDDL planners, a domain model has to be written in each language[2]. A fragment of the RDDL domain is shown in Figure C.3 and a fragment of the PDDL domain for the task is shown in Figure C.4. These figures show the *goto_waypoint* action and demonstrate the differences between the ways in which the domains are used to model the same action. In the RDDL domain the *cpfs* used to describe the effects of the action fluents are distributed throughout the domain description. Care must be taken to ensure that the state transition in both domains remains identical, with the exception of probabilistic effects. While the RDDLSim software used to run the IPPC includes an automatic translation from RDDL to a subset of PPDDL, to properly determinize the domain we performed this translation by hand. In future work we intend to investigate the prospect of using the Knowledge Base to perform this determinization automatically.

```
// State fluents
robot_at(robot, waypoint): { state-fluent, bool, default = false };
docked(robot): { state-fluent, bool, default = false };
visited(waypoint): { state-fluent, bool, default = false };

// Action fluents
goto_waypoint(robot, waypoint, waypoint): { action-fluent, bool, default = false };

cpfs {
  robot_at'(?r, ?w) =
    if (exists_{?w1: waypoint} (goto_waypoint(?r, ?w1, ?w))) then true
    else if (exists_{?w1: waypoint} (goto_waypoint(?r, ?w, ?w1))) then false
    else robot_at(?r, ?w);
  visited'(?w) =
    visited(?w) | (exists_{?r:robot, ?w1: waypoint} [goto_waypoint(?r, ?w1, ?w)]);

  asked_load'(?r) =
    if (exists_{?wf: waypoint, ?wt: waypoint} [goto_waypoint(?r, ?wf, ?wt)]) then false
    else if (ask_load(?r)) then true
    else asked_load(?r);

  asked_unload'(?r) =
    if (exists_{?wf: waypoint, ?wt: waypoint} [goto_waypoint(?r, ?wf, ?wt)]) then false
    else if (ask_unload(?r)) then true
    else asked_unload(?r);
}
```

Figure C.3: Fragment of the RDDL domain for the print-fetching scenario, showing the *robot_at* state fluent, *goto_waypoint* action fluent and cpfs that describes the transition of the state fluent.

[2]Both PDDL and RDDL domains can be found here: github.com/m312z/KCL-Turtlebot/tree/master/domains

```
(:action goto_waypoint
  :parameters (?v - robot ?from ?to - waypoint)
  :precondition (and
    (robot_at ?v ?from)
    (localised ?v)
    (undocked ?v))
  :effect (and
    (not (robot_at ?v ?from)) (robot_at ?v ?to)
    (increase (total-cost) (distance ?from ?to))))
```

Figure C.4: Fragment of the PDDL domain for the print-fetching scenario, showing the *goto_waypoint* action.

RDDL domain description

The print-fetching domain in RDDL is made of seven action fluents: one for moving (*goto_waypoint*), two actions for interacting with the user and asking him/her to load or take the printed papers, two for waiting for the user to do it, and the ones for docking and undocking the robot to the charging station. A fluent named *goal* is used to specify the goal condition, such that the final reward is given only once the goal is reached, thus simulating a goal-oriented MDP. In the print-fetching domain the goal is to deliver the printed papers to a specific location. The domain has two stochastic fluents, both sampled from a Bernoulli distribution. One represents whether there is somebody to help the robot in one location, and the second specifies whether a printer is being used or not, being the parameter of the Bernoulli distribution dependant on the location. Finally, the reward function provides a positive reward when the goal is reached and the robot is docked, and then some penalizations, considered as costs, for moving (weighted by the distance of the moving action), waiting in a printer where there is nobody to help, and waiting in a printer which is busy.

C.2 Experiments

In our experiments we used a mobile robot (TurtleBot 2). The robot is equipped with a Kinect sensor which is used for both mapping and navigation [165]. Experiments were run in a real-world office environment where people were performing their regular daily activities. Therefore, corridors were crowded and the robot had to avoid obstacles while performing the task. All actions used in the scenario were implemented, apart for the detection of paper placement and human presence perception, which were simulated. An implementation of these actions is not in the scope of this evaluation.

We tested the system architecture shown in Figure 6.2 using the probabilistic planner PROST [112] and compared with the default ROSPlan system using the PDDL2.1 planner Metric-FF [166]. The goal for both planners is to deliver the printed papers in the shortest time. There were two sources of uncertainty in the scenario whose prior probabilities were modeled in the RDDL domain: (1) the presence of people near the printer and (2) the occupancy of the printer. The values are given in Table C.1. When using the deterministic planner (Metric-FF), the system replanned in the case of an action failure.

C.2.1 Results

We performed three different real-world robotic experiments which represented three situations obtained by sampling the events of person near the printer and occupancy of the printer. These experiments are described in Table C.2. For each experiment we applied both planning approaches in five executions. A fourth experiment has been simulated.

As a measure of effectiveness we compare total time of execution, time of planning and robot travel distance. To measure execution time, we measure from the start of planning until the robot completes the task. To measure planning time: (1) in the case of Metric-FF replanning can be performed several times, so the total planning time is the sum of these planning episodes; (2) in the case of PROST, planning is performed before each action is taken so total planning time is the sum of the time to produce each action. The travel distance is the length of the path that the robot traveled.

Printer	Events	Probability of the event
P1	Occupancy	0.5
P1	Nearby person	0.9
P2	Occupancy	0.2
P2	Nearby person	0.4
P3	Occupancy	0.8
P3	Nearby person	0.5

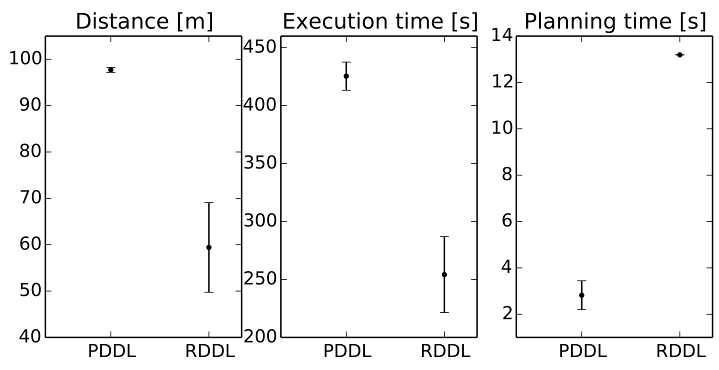
Table C.1: Prior probabilities of events in the experimental setup. The same values are used in the problem definition of RDDL.

Experiments	Start position	Delivery goal	Printer	Printers occupancy	Nearby person
			P1	free	yes
1	Prof. Office	PhD Area	P2	free	no
			P3	free	no
			P1	free	yes
2	PhD Area	Kitchen	P2	free	yes
			P3	busy	yes
			P1	busy	yes
3	Docking station	Prof. Office	P2	free	no
			P3	busy	yes

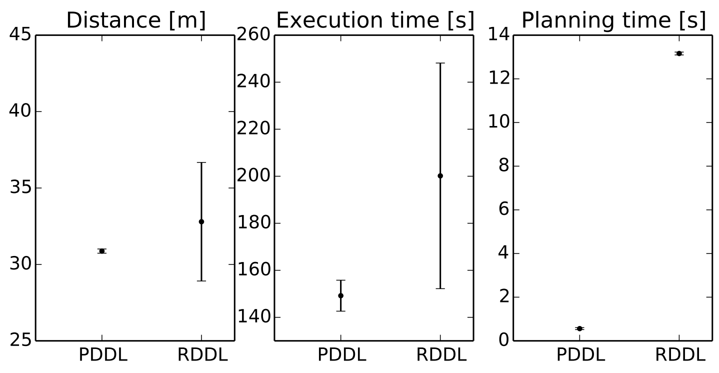
Table C.2: Experimental setups. For each setup and planning approach we run 5 tests.

The results of all three experiments are shown in Figure C.5. Experiment 1 demonstrates the advantage of probabilistic planning. In this set up, the robot can only succeed in printer $P1$, though when only the traveled distance is considered, $P3$ would be the best option. In order to minimize the expected duration of the plan, the Metric-FF planner chose to visit printers $P2$ and $P3$ without taking probabilities of events into account. As these printers were empty, the plan execution failed and replanning was performed both times, to finally succeed when visiting $P1$. On average, the Metric-FF planner had to replan 4 times in each test run of this experiment. In contrast, the probabilistic approach attempted to use printer $P1$ first[3].

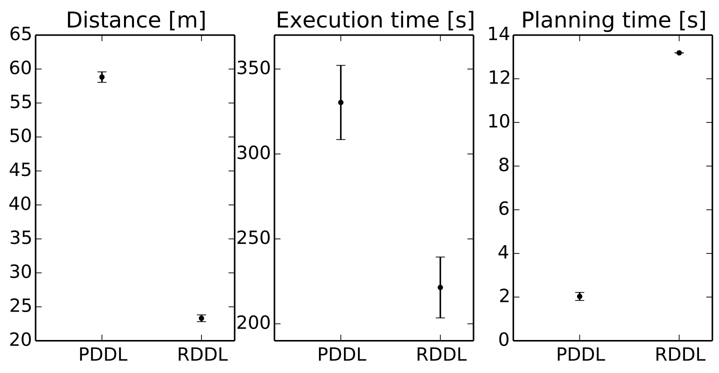
[3]A video demonstration of this setup can be found in youtu.be/aozTz4Ex7PI



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure C.5: Experimental results, showing mean values with standard deviations of the robot travel distance, test execution time and planning time for the first 3 experiments. In each experiment, 5 tests were performed for each approach.

Experiment 2 shows a simple case where conditions are optimal for a deterministic planner (no unexpected effects). In this case, $P1$ and $P2$ are the best option to select. As expected, the Metric-FF planner did not have to replan at all, as the best solution was the one selected in the first attempt. Therefore, it exhibits a shorter planning and execution time than the probabilistic planning approach. The distance is still approximately the same, because only in one case did PROST not find the optimal solution.

Experiment 3 shows a case where the available printers are busy, therefore forcing the robot to either wait for the printer to become available or to try another printer. In this case, we simulate the printer to be busy for one action execution. Therefore the printer will become available if the robot waits until a timeout and checks again, or if the robot goes to another location and comes back to a visited printer which was busy. The observed behavior for the deterministic planner in this case was to visit the closest printer $P1$, which was busy, then visit printer $P2$, which was empty, the printer $P3$, which is also busy, to finally succeed at $P1$. In contrast, the stochastic planner went to printer $P1$, waited for it until timeout, and then waited again, obtaining the papers in this second step. This behavior was obtained due to the planner having the certainty of eventually having someone to help at printer $P1$, though there was uncertainty of succeeding if other printers were visited.

The standard deviation (σ) in distance and execution time is small for PDDL, and large for RDDL. This is because the deterministic planner always chooses the plan that is optimal in time and distance, and in fact the σ comes only from real-world execution. The variance seen in PDDL is due to the navigation system and person interaction. In contrast, PROST produces different plans depending upon the probabilities of events, which can vary greatly in execution time and distance traveled. The σ in planning time is greater for the PDDL planner. This is due to the impact of the replanning attempts.

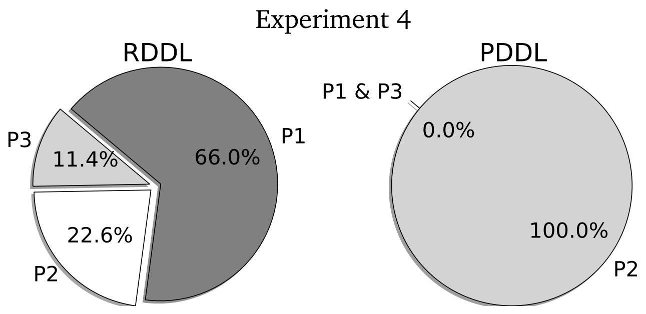


Figure C.6: Experimental results, showing the distribution of the first printer selected across all plans by each planner in Experiment 4. In each experiment, 500 tests were made in simulation.

A final simulated experiment has been performed to further show the effects of the probabilities in the planning scenario we proposed. The setup for this experiment was the robot starting at the *PhD Area*, and the delivery goal was the *Professor's office*. For this experiment, 500 executions with both the deterministic planner and the stochastic one are carried out, and we take into account only the action of the plan that shows the first chosen printer. As it can be seen in the results from Figure C.5, the deterministic planner always chose to go to $P2$, which is the one providing shortest travel distance. In contrast, the stochastic planner has different choices, leading to a distribution that resembles the one shown in Table C.1, selecting to visit most of the times $P1$, then $P2$ and finally $P3$. Therefore, given that $P2$ is less likely to have people around to help the robot, the deterministic planner is more prone to fail in such setup.

C.3 Discussion

The focus of this appendix was to describe a use-case for the integration of probabilistic planning into ROSPlan (Section 6.3), and to demonstrate the execution of probabilistic plans in real-time robotics scenarios. This has involved the implementation of RDDL models into the ROSPlan KB, and an online dispatcher that uses the RDDL Client/Server protocol.

This appendix is not intended to make a comparison of deterministic vs. probabilistic approaches. Our experiments show that both approaches have advantages, and a more thorough discussion can be found in [109]. Many factors determine which planning approach is better suited to the domain and problem. For example, whether the domain is probabilistically interesting and whether probabilities are known. Also, whether or not it is necessary to have an optimal plan, or that from a given initial state the same plan is always generated for execution.

Although the use of a probabilistic planner may result in shorter paths and faster plan execution, from the perspective of domain modeling we found it was more intuitive to use an action-oriented language. Another element to take into account is that, while the handling of uncertainties by means of probabilistic planning can be useful in robotics and real-world scenarios, those probabilities must be coherent with the real-world. Such probabilities are often hard to obtain or estimate, and will usually need some kind of learning or adaptation to the real world.

D
Questionnaire used for the HRI evaluation

Following we reproduce the questionnaire that was presented to the users for the evaluation performed in Chapter 7.

Demographics

1. Gender (To which gender identity do you mostly identify?)

   ☐ Male
   ☐ Female
   ☐ Other (Specify)

2. Age *[Numerical value]*

Assistive feeding

3. The robot was using my preferences in the...

   ☐ First trial
   ☐ Second trial
   ☐ I am not sure

4. The interaction with the robot was more pleasant when it was using the preferences

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

5. The behavior of the robot was significantly different between both trials

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

6. This application would be helpful for dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

7. The use of the preferences in this application would significantly improve the assistance of dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Shoe fitting

8. The robot was using my preferences in the...

   ☐ First trial
   ☐ Second trial
   ☐ I am not sure

9. The interaction with the robot was more pleasant when it was using the preferences

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

10. The behavior of the robot was significantly different between both trials

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

11. This application would be helpful for dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

12. The use of the preferences in this application would significantly improve the assistance of dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Jacket dressing

13. The robot was using my preferences in the...

   ☐ First trial
   ☐ Second trial
   ☐ I am not sure

14. The interaction with the robot was more pleasant when it was using the preferences

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

15. The behavior of the robot was significantly different between both trials

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

16. This application would be helpful for dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

17. The use of the preferences in this application would significantly improve the assistance of dependant people

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

General - Almere Model[1]

In the following questions, when "the robot" is mentioned, imagine a final robotic product able to assist the user in the same way you have experienced. Notice the used robot is a prototype.

Anxiety (ANX)

18. If I should use the robot, I would be afraid to make mistakes with it

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

19. If I should use the robot, I would be afraid to break something

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

20. I find the robot scary

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

21. I find the robot intimidating

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

[1]Note that questions related to the Almere model [135] were presented to the users section-less and randomized.

Attitude Towards Technology (ATT)

22. I think it's a good idea to use the robot (in general)

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

23. I think it's a good idea to use the robot (for people who may need it)

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

24. The robot would make life more interesting (of people in need/dependant people)

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

25. It's good to make use of the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Facilitating Conditions (FC)

26. I have everything I need to use the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

27. I know enough of the robot to make good use of it

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Intention to Use (ITU)

Assume the case in which you would need assistance to perform the task activities -feeding, dressing, shoe fitting-.

28. I think I would use the robot during the next few days

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

29. I'm certain to use the robot during the next few days

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

30. I plan to use the robot during the next few days

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Perceived Adaptivity (PAD)

31. I think the robot can be adaptive to what I (may) need

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

32. I think the robot will only do what I need at that particular moment

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

33. I think the robot will/could help me when I consider it to be necessary

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Perceived Enjoyment (PENJ)

34. I enjoy the robot talking to me (in case it talked)

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

35. I enjoy doing things with the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

36. I find the robot enjoyable

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

37. I find the robot fascinating

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

38. I find the robot boring

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Perceived Ease of Use (PEOU)

39. I think I will know quickly how to use the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

40. I find the robot easy to use

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

41. I think I can use the robot without any help

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

42. I think I can use the robot when there is someone around to help me

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

43. I think I can use the robot when I have a good manual

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Perceived Sociability (PS)

44. I consider the robot a pleasant conversational partner

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

45. I find the robot pleasant to interact with

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

46. I feel the robot understands me

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

47. I think the robot is nice

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Perceived Usefulness (PU)

Assume the case in which you would need assistance to perform the task activities -feeding, dressing, shoe fitting-.

48. I think the robot is useful to me (if I needed assistance to perform the tasks)

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

49. It would be convenient for me to have the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

50. I think the robot can help me with many things

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Social Influence (SI)

Assume the case in which you would need assistance to perform the task activities -feeding, dressing, shoe fitting-.

51. I think the caregivers would like me using the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

52. I think it would give a good impression if I should use the robot

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Trust

53. I would trust the robot if it gave me advice

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

54. I would follow the advice the robot gives me

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Acronyms

ADLs Activities of Daily Living. 1, 2, 5, 27, 31, 35, 121, 154, *Glossary:* Acitivities of Daily Living

AI Artificial Intelligence. 4, 29, 78, 102, 123, 152–154

CNA Certified Nursing Assistant. 2, 3

FIS Fuzzy Inference System. 4, 12, 39, 44, 45, 50, 51, 60, 102, 122–124

FSA Finite State Automaton. 28

FUTE Factory setting, User Tailoring, Execution tuning. 12, 13, 15–17, 26, 27, 38, 40, 49, 54, 61, 119, 121, *see Section 2.3*

HRI Human-Robot Interaction. 5, 16, 28, 30, 31, 34, 63, 66, 77, 81, 103, 105, 123, 135, 143, 153, *Glossary:* Human-Robot Interaction

HSMM Hidden Semi-Markov Model. 68

HTN Hierarchical Task Network. 29, 30, 42, 49, 64, 65, 77

IMU Inertial Measurement Unit. 41

IPPC International Probabilistic Planning Competition. 76, 79, 80, 83, 84, 93, 135, 137

KB Knowledge Base. *Usually used to refer ROSPlan's component*. 81–85, 135, 137, 142

MDP Markov Decision Process. 43, 66, 80, 136–138, 151, 154, *see Section 4.3*

NID Noisy Indeterministic Deictic rules. 43, 44, 46–48, 52, 60

NPRS Numerical Pain Rating Scale. 44

PAR Physically Assistive Robot. 2, 5, 12, 13, 28, 34, 35, 63, 81, 103–105, 107, 108, 110, 115, 116, 118, 119, 121, 125, 129, 153, *Glossary:* Physically Assistive Robot

PBP Preference-Based Planning. 29, 78, *Glossary:* Preference-Based Planning

PDDL Planning Domain Description Language. 29, 78–83, 87, 123, 137, 138, 141, 154

pHRI Physical Human-Robot Interaction. 13, 32, 63, 65, 103, 130, *Glossary:* Physical Human-Robot Interaction

POMDP Partially Observable Markov Decision Process. 80, 136, 137, 154

PPDDL Probabilistic Planning Domain Description Language. 4, 76, 123, 137, 154

ProMP Probabilistic Movement Primitive. 20, 24–26

RDDL Relational Dynamic Influence Diagram Language. 4, 13, 76, 79–86, 92, 93, 102, 122, 123, 135, 137–139, 141, 142, 154

RL Reinforcement Learning. 78, 105

ROS Robot Operating System. 58, 76, 79–82, 84, 85, 154

SAR Socially Assistive Robot. 2, 16, 105, *Glossary:* Socially Assistive Robot

SSP Stochastic Shortest Path. 136, 137

XAI Explainable AI. 78, 102, 123, 152

XAIP Explainable AI Planning. 78, 123

Glossary

Acitivities of Daily Living Any of a number of routine tasks and functions a person must be able to perform in order to maintain independence. *(Oxford Dictionary)* Those include daily self-care activities such as bathing, dressing, personal hygiene, grooming, toileting and continence, and feeding, among others. 1, 2, 5, 31, 121, 151, 154, *Acronym:* ADLs

Compliant robot control Robot controller that allows external perturbations while still following the intended trajectory. These kind of controllers are suitable for tasks involving humans as the human can alter the robot trajectory, and harmful forces against the user are not applied in case of collisions. Example controllers are [20,75]. 5, 9, 15, 23, 24, 63, 68, 125, 129, 131

Conformant planning Type of planning under uncertainty without the possibility of observing the state (no sensing actions are available). A solution to the conformant planning problem is a sequence of actions from the initial state to the goal one. A solution is predicted to lead to the goal state regardless of the outcomes of the nondeterministic actions, or from which initial state the execution began. 80

Contingent planning Type of planning under uncertainty where the agent does not have complete information of the world but it has sensors to observe the environment (sensing actions). The plan is usually represented as a decision tree where each node is a set of states, and at each step different actions are performed under different circumstances. 79, 80

Deterministic planning Type of planning where the actions have single effect that is expected to happen always. 79, 80, 83, 84, 86, 135, 138, 140–142, *Task planning usually refers to deterministic planning. Also denoted as classical planning*

Human-Robot Interaction Branch of research in robotics that studies the interactions between humans and robots. It is a multidisciplinary field that involves Human-Computer Interaction (HCI), robotics, Artificial Intelligence and social sciences such as psychology and sociology. 5, 16, 28, 30, 31, 34, 77, 81, 103, 151, *Acronym:* HRI

Motion Planning Branch of robotics that focuses in the problem of robot movement and positioning. A solution to the motion planning problem consists of a sequence of valid robot configurations that moves the robot from one position to a different target one, typically avoiding potential obstacles. 41, 64, 65, 68, 106, *related terms are: path planning, geometric planning, robot navigation*

Physical Human-Robot Interaction Branch of research in HRI that focus on interactions where the robot has physical contact with a human, be it in assistive or any other robot-related task. Physically Assistive Robot (PAR) can be considered as a subset of it. 151, *Acronym:* pHRI

Physically Assistive Robot Robotic system designed to provide physical assistance to the person in need. The typical assistance would be to help the user to perform Activities of Daily Living (ADLs) in an an independent and autonomous manner, with the help of the robot. Physical tasks involve physical contact between the robot and the user. Examples of tasks include feeding, dressing, grooming, combing or shaving. 2, 5, 12, 28, 34, 35, 63, 81, 104, 105, 110, 116, 118, 121, 125, 129, 151, 153, *Acronym:* PAR

Preference-Based Planning Type of planning that focuses on the generation of plans that satisfy user preferences over the plans. Plans are ranked by quality based on the user preferences, and the plan satisfying more preferences is usually considered the best one. 29, 30, 78, 151, *Acronym:* PBP

Probabilistic planning Type of planning under uncertainty where the actions may have different effects based on some probabilities, and exogenous effects may also be present. They are usually represented as an MDP or a POMDP. 4, 13, 40, 43, 60, 64, 76, 79–81, 83–85, 88, 92, 93, 102, 122, 135, 139–142, *also known as: stochastic planning*

Replanning Strategy for planning under uncertainty where a classical deterministic planner is used to compute a plan, and the plan is recomputed every time the plan fails or it is no longer valid. 73, 78–80, 83, 84, 138, 139, 141

ROSPlan A modular framework that provides a collection of tools for Artificial Intelligence Planning in a ROS system. ROSPlan has a variety of nodes which encapsulate planning, problem generation, and plan execution. It possesses a simple interface, and links to common ROS libraries. 4, 13, 76, 79–84, 93, 102, 109, 119, 122, 135, 138, 142, 151, *see ROSPlan's website*

Socially Assistive Robot Robotic system designed to provide assistance to human users, constraining that assistance to be through non-physical social interaction. SAR focus on achieving specific convalescence, rehabilitation, training, or education goals by addressing social rather than physical interaction [5, 167]. 2, 16, 105, 152, *Acronym:* SAR

Spasmodic movement Movement caused by, subject to, or in the nature of a sudden involuntary muscular contraction or convulsive movement *(Oxford Dictionary)*. 7, 8

Task planning Branch of Artificial Intelligence involved in problem solving. The goal of a planning algorithm is to find a *plan*, which is a sequence of actions from an *initial* state to a *goal* state. Planning problems are defined formally in languages such as PDDL, PPDDL or RDDL. Planning problems are defined with a domain file where the applicable actions are defined. Such action definitions include preconditions for its application and effects of use. A problem file describes the initial symbolic state of the environment and the desired goal or target state. A planning algorithm will try to find a plan to change the state of the environment from the initial state to the goal one by using the defined actions. 2, 4, 5, 12, 13, 29, 30, 39–41, 43–47, 49, 50, 52–55, 57–61, 63–65, 68, 70, 72, 73, 75–80, 84, 86–88, 96, 103, 104, 109, 119, 122, 123, 135, 139–142, 152–155, *generally also denoted as: Symbolic (Task) planning, AI planning, Automated planning and scheduling. A formal definition can be found in Section 5.3*

Temporal planning Type of planning where the duration of the actions is taken into consideration in order to find a sequence of actions. Actions may be temporally overlapping and concurrent. 83

Bibliography

[1] K. M. Daniel and C. Y. Smith, “Present and future needs for nurses”, *Journal of Applied Biobehavioral Research*, vol. 23, no. 1, 2018.

[2] M. Marć, A. Bartosiewicz, J. Burzyńska, Z. Chmiel, and P. Januszewicz, “A nursing shortage – a prospect of global and local policies”, *International Nursing Review*, vol. 66, no. 1, pp. 9–16, 2019.

[3] C. Torras, “Assistive robotics: Research challenges and ethics education initiatives”, *Dilemata*, no. 30, pp. 63–77, 2019.

[4] T. L. Chen, M. Ciocarlie, S. Cousins, P. M. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C. H. King, D. A. Lazewatsky, A. E. Leeper, H. Nguyen, A. Paepcke, C. Pantofaru, W. D. Smart, and L. Takayama, “Robots for humanity: using assistive robotics to empower people with disabilities”, *IEEE Robotics Automation Magazine*, vol. 20, no. 1, pp. 30–39, 2013.

[5] D. Feil-Seifer and M. J. Matarić, “Defining socially assistive robotics”, in *9th International Conference on Rehabilitation Robotics*, pp. 465–468, June 2005.

[6] G. Canal, G. Alenyà, and C. Torras, “Personalization Framework for Adaptive Robotic Feeding Assistance”, in *International Conference on Social Robotics (ICSR)*, pp. 22–31, Springer International Publishing, November 2016.

[7] G. Chance, A. Camilleri, B. Winstone, P. Caleb-Solly, and S. Dogramadzi, “An assistive robot to support dressing – strategies for planning and error handling”, in *6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 774–780, June 2016.

[8] E. Stevens, E. Emmett, Y. Wang, C. McKevitt, and C. Wolfe, *The Burden of Stroke in Europe*. Stroke Alliance for Europe, 5 2017.

[9] M. A. Gignac and C. Cott, “A conceptual model of independence and dependence for adults with chronic physical illness and disability”, *Social Science & Medicine*, vol. 47, pp. 739–753, 9 1998.

[10] A. M. Williams, G. Christopher, and E. Jenkinson, “The psychological impact of dependency in adults with chronic fatigue syndrome/myalgic encephalomyelitis: A qualitative exploration”, *Health Psychology*, 2016.

[11] G. Boström, M. Conradsson, E. Rosendahl, P. Nordström, Y. Gustafson, and H. Littbrand, “Functional capacity and dependency in transfer and dressing are associated with depressive symptoms in older people”, *Clinical interventions in aging*, vol. 9, p. 249, 2014.

[12] G. Canal, G. Alenyà, and C. Torras, “A taxonomy of preferences for physically assistive robots”, in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 292–297, August 2017.

[13] G. Canal, G. Alenyà, and C. Torras, “Adapting robot task planning to user preferences: an assistive shoe dressing example”, *Autonomous Robots*, vol. 43, pp. 1343–1356, Aug 2019.

[14] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive HRI: application to dressing assistance", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3273–3278, May 2018.

[15] G. Canal, M. Cashmore, S. Krivić, G. Alenyà, D. Magazzeni, and C. Torras, "Probabilistic Planning for Robotics with ROSPlan", in *Towards Autonomous Robotic Systems*, pp. 236–250, Springer International Publishing, 2019.

[16] G. Canal, C. Torras, and G. Alenyà, "SoPS: Generating predicate Suggestions based on the Space of Plans. A planning with preferences example", *Submitted*, 2020.

[17] M. Vila, G. Canal, and G. Alenyà, "Towards safety in Physically Assistive Robots: eating assistance", in *Robots for Assisted Living Workshop at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[18] G. Canal, C. Torras, and G. Alenyà, "Are preferences useful for better assistance? – A Physically Assistive Robotics user study", *Submitted*, 2019.

[19] A. H. Maslow, "A theory of human motivation", *Psychological Review*, vol. 50, no. 4, pp. 370–396, 1943.

[20] A. Colomé, D. Pardo, G. Alenyà, and C. Torras, "External force estimation during compliant robot manipulation", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3535–3540, 2013.

[21] C. Clabaugh, G. Ragusa, F. Sha, and M. Matarić, "Designing a socially assistive robot for personalized number concepts learning in preschool children", in *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 314–319, Aug 2015.

[22] J. Greczek, E. Short, C. Clabaugh, K. Swift-Spong, and M. J. Matarić, "Socially assistive robotics for personalized education for children", in *AAAI Fall Symposium on Artificial Intelligence and Human-Robot Interaction*, 2014.

[23] D. Leyzberg, S. Spaulding, and B. Scassellati, "Personalizing robot tutors to individuals' learning differences", in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 423–430, ACM, 2014.

[24] K. Baraka and M. Veloso, "Adaptive interaction of persistent robots to user temporal preferences", in *Social Robotics*, pp. 61–71, Springer International Publishing, 2015.

[25] M. Fiore, A. Clodic, and R. Alami, "On planning and task achievement modalities for human-robot collaboration", in *Experimental Robotics: The 14th International Symposium on Experimental Robotics*, pp. 293–306, Springer International Publishing, 2016.

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

[27] S. Chernova and M. Veloso, “Interactive policy learning through confidence-based autonomy”, *Journal of Artificial Intelligence Research*, vol. 34, pp. 1–25, 2009.

[28] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, “Learning and generalization of complex tasks from unstructured demonstrations”, in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 5239–5246, IEEE, 2012.

[29] M. Lawitzky, J. R. Medina, D. Lee, and S. Hirche, “Feedback motion planning and learning from demonstration in physical robotic assistance: differences and synergies”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3646–3652, IEEE, 2012.

[30] Y. Gao, H. J. Chang, and Y. Demiris, “User modelling for personalised dressing assistance by humanoid robots”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1840–1845, 2015.

[31] S. D. Klee, B. Q. Ferreira, R. Silva, J. P. Costeira, F. S. Melo, and M. Veloso, “Personalized assistance for dressing users”, in *Social Robotics*, pp. 359–369, Springer International Publishing, 2015.

[32] W. K. Song, W. J. Song, Y. Kim, and J. Kim, “Usability test of KNRC self-feeding robot”, in *International Conference on Rehabilitation Robotics*, pp. 1–5, 2013.

[33] X. Zhang, X. Wang, B. Wang, T. Sugi, and M. Nakamura, “Real-time control strategy for emg-drive meal assistance robot — my spoon”, in *International Conference on Control, Automation and Systems*, pp. 800–803, Oct 2008.

[34] M. Topping, “An Overview of the Development of Handy 1, a Rehabilitation Robot to Assist the Severely Disabled”, *Intelligent and Robotic Systems*, vol. 34, no. 3, pp. 253–263, 2002.

[35] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Probabilistic movement primitives”, in *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[36] A. Colomé, G. Neumann, J. Peters, and C. Torras, “Dimensionality reduction for probabilistic movement primitives”, in *IEEE-RAS International Conference on Humanoid Robots*, pp. 794–800, 2014.

[37] G. Pigozzi, A. Tsoukiàs, and P. Viappiani, “Preferences in artificial intelligence”, *Annals of Mathematics and Artificial Intelligence*, vol. 77, pp. 361–401, 2016.

[38] J. A. Baier and S. McIlraith, “Planning with preferences”, *AI Magazine*, vol. 29, no. 4, pp. 25–36, 2008.

[39] N. Li, W. Cushing, S. Kambhampati, and S. Yoon, “Learning Probabilistic Hierarchical Task Networks as Probabilistic Context-Free Grammars to Capture User Preferences”, *ACM Transactions on Intelligent Systems and Technology*, vol. 5, pp. 1–32, 4 2014.

[40] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos, "Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners", *Artificial Intelligence*, vol. 173, no. 5–6, pp. 619 – 668, 2009.

[41] S. Sohrabi, J. A. Baier, and S. A. McIlraith, "HTN planning with preferences", in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1790–1797, 2009.

[42] T. C. Son and E. Pontelli, "Planning with preferences using logic programming", *Theory and Practice of Logic Programming*, vol. 6, pp. 559–607, 9 2006.

[43] B. D. Argall and A. G. Billard, "A survey of Tactile Human–Robot Interactions", *Robotics and Autonomous Systems*, vol. 58, no. 10, pp. 1159–1176, 2010.

[44] H. Yanco and J. Drury, "Classifying human-robot interaction: an updated taxonomy", in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2841–2846, IEEE, 2004.

[45] C. Krauss and S. Arbanowski, "Social Preference Ontologies for Enriching User and Item Data in Recommendation Systems", in *IEEE International Conference on Data Mining Workshop (ICDMW 2014)*, pp. 365–372, 2014.

[46] C. M. Bastemeijer, L. Voogt, J. P. van Ewijk, and J. A. Hazelzet, "What do patient values and preferences mean? A taxonomy based on a systematic review of qualitative papers", *Patient Education and Counseling*, 12 2016.

[47] J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction", *Journal of Human-Robot Interaction*, vol. 3, no. 2, p. 74, 2014.

[48] H. Peng, C. Zhou, H. Hu, F. Chao, and J. Li, "Robotic dance in social robotics - A taxonomy", *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 3, pp. 281–293, 2015.

[49] J. Shim and R. C. Arkin, "A Taxonomy of Robot Deception and Its Benefits in HRI", in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2328–2335, IEEE, 10 2013.

[50] T. J. Wiltshire, E. J. C. Lobato, J. Velez, F. G. Jentsch, and S. M. Fiore, "An interdisciplinary taxonomy of social cues and signals in the service of engineering robotic social intelligence", in *Unmanned Systems Technology XVI*, vol. 9084, 6 2014.

[51] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots", *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[52] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots", *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[53] O. P. John and S. Srivastava, “The big five trait taxonomy: History, measurement, and theoretical perspectives”, *Handbook of personality: Theory and research*, vol. 2, no. 1999, pp. 102–138, 1999.

[54] C. LeRouge, J. Ma, S. Sneha, and K. Tolle, “User profiles and personas in the design and development of consumer health technologies”, *International Journal of Medical Informatics*, vol. 82, no. 11, pp. e251 – e268, 2013.

[55] H. Robinson, B. MacDonald, and E. Broadbent, “The role of healthcare robots for older people at home: A review”, *International Journal of Social Robotics*, vol. 6, no. 4, pp. 575–591, 2014.

[56] Y. Gao, H. J. Chang, and Y. Demiris, “Iterative path optimisation for personalised dressing assistance using vision and force information”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4398–4403, Oct 2016.

[57] G. Chance, A. Jevtić, P. Caleb-Solly, and S. Dogramadzi, “A quantitative analysis of dressing dynamics for robotic dressing assistance”, *Frontiers in Robotics and AI*, vol. 4, 2017.

[58] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, “Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions”, in *IEEE/SICE International Symposium on System Integration*, pp. 564–570, Dec 2014.

[59] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, “Reinforcement learning of clothing assistance with a dual-arm robot”, in *11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 733–738, IEEE, 2011.

[60] S. Alili, M. Warnier, M. Ali, and R. Alami, “Planning and plan-execution for human-robot cooperative task achievement”, in *19th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 19–23, 2009.

[61] R. Lallement, L. De Silva, and R. Alami, “Hatp: An htn planner for robotics”, in *2nd ICAPS Workshop on Planning and Robotics*, 2014.

[62] L. de Silva, R. Lallement, and R. Alami, “The hatp hierarchical planner: Formalisation and an initial study of its usability and practicality”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6465–6472, September 2015.

[63] G. Castellano, I. Leite, and A. Paiva, “Detecting perceived quality of interaction with a robot using contextual features”, *Autonomous Robots*, pp. 1–17, 2016.

[64] A. L. Thomaz and C. Breazeal, “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’06, pp. 1000–1005, AAAI Press, 2006.

[65] W. B. Knox and P. Stone, “Interactively shaping agents via human reinforcement: The tamer framework”, in *The Fifth International Conference on Knowledge Capture*, September 2009.

[66] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz, “Policy shaping: Integrating human feedback with reinforcement learning”, in *Advances in Neural Information Processing Systems 26*, pp. 2625–2633, 2013.

[67] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, “Learning symbolic models of stochastic domains”, *Journal of Artificial Intelligence Research*, vol. 29, pp. 309–352, 2007.

[68] E. McLafferty and A. Farley, “Assessing pain in patients”, *Nursing Standard*, vol. 22, no. 25, pp. 42–46, 2008.

[69] B. Kühnlenz, S. Sosnowski, M. Buß, D. Wollherr, K. Kühnlenz, and M. Buss, “Increasing Helpfulness towards a Robot by Emotional Adaption to the User”, *International Journal of Social Robotics*, vol. 5, no. 4, pp. 457–476, 2013.

[70] B. M. Muir, “Trust between humans and machines, and the design of decision aids”, *International Journal of Man-Machine Studies*, vol. 27, no. 5, pp. 527 – 539, 1987.

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

[72] N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke, and T. Asfour, “The robot software framework armarx”, *Information Technology*, vol. 57, no. 2, pp. 99–111, 2015.

[73] K. Erol, J. Hendler, and D. S. Nau, “HTN planning: Complexity and expressivity”, in *AAAI*, vol. 94, pp. 1123–1128, 1994.

[74] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, “Learning physical collaborative robot behaviors from human demonstrations”, *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 513–527, 2016.

[75] E. Pignat and S. Calinon, “Learning adaptive dressing assistance from human demonstration”, *Robotics and Autonomous Systems*, vol. 93, pp. 61–75, 2017.

[76] A. Pervez and D. Lee, “Learning task-parameterized dynamic movement primitives using mixture of gmms”, *Intelligent Service Robotics*, vol. 11, pp. 61–78, Jan 2018.

[77] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, “Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction”, *Autonomous Robots*, vol. 43, pp. 1291–1307, Aug 2019.

[78] E. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1 – 13, 1975.

[79] J. Rada-Vilela, “fuzzylite: a fuzzy logic control library”, 2014. Accessed 14 Aug 2019.

[80] D. Martínez, G. Alenyà, and C. Torras, “Planning robot manipulation to clean planar surfaces”, *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 23 – 32, 2015.

[81] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system", in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[82] F. Gravot, A. Haneda, K. Okada, and M. Inaba, "Cooking for humanoid robot, a task that needs symbolic and geometric reasonings", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 462–467, 2006.

[83] L. de Silva, A. K. Pandey, and R. Alami, "An interface for interleaved symbolic-geometric planning and backtracking", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 232–239, 2013.

[84] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 639–646, 2014.

[85] J. Ferrer-Mestres, G. Frances, and H. Geffner, "Planning with state constraints and its application to combined task and motion planning", in *Workshop on Planning and Robotics (PLANROB)*, pp. 13–22, 2015.

[86] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, "Geometric backtracking for combined task and motion planning in robotic systems", *Artificial Intelligence*, vol. 247, pp. 229 – 265, 2017.

[87] K. Lee, Y. Su, T.-K. Kim, and Y. Demiris, "A syntactic approach to robot imitation learning using probabilistic activity grammars", *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1323–1334, 2013.

[88] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system", in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2015.

[89] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found", in *Proceedings of the International Conference on International Conference on Automated Planning and Scheduling (ICAPS)*, ICAPS'10, pp. 81–88, AAAI Press, 2010.

[90] D. Martínez, G. Alenyà, and C. Torras, "Relational reinforcement learning with guided demonstrations", *Artificial Intelligence*, vol. 247, pp. 295–312, 2017.

[91] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila, "Toward human-aware robot task planning", in *AAAI spring symposium*, pp. 39–46, 2006.

[92] M. Cirillo, L. Karlsson, and A. Saffiotti, "Human-aware task planning: An application to mobile robots", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 1, no. 2, p. 15, 2010.

[93] A. Sekmen and P. Challa, "Assessment of adaptive human–robot interactions", *Knowledge-Based Systems*, vol. 42, pp. 49 – 59, 2013.

[94] W. Y. Kwon and I. H. Suh, "Planning of proactive behaviors for human–robot cooperative tasks under uncertainty", *Knowledge-Based Systems*, vol. 72, pp. 81 – 95, 2014.

[95] T. Chakraborti, S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Projection-aware task planning and execution for human-in-the-loop operation of robots in a mixed-reality workspace", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4476–4482, IEEE, 2018.

[96] M. Das, P. Odom, M. R. Islam, J. R. J. Doppa, D. Roth, and S. Natarajan, "Planning with actively eliciting preferences", *Knowledge-Based Systems*, vol. 165, pp. 219 – 227, 2019.

[97] J. Kim, M. E. Woicik, M. C. Gombolay, S.-H. Son, and J. A. Shah, "Learning to infer final plans in human team planning", in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 4771–4779, 7 2018.

[98] D. Shmaryahu, G. Shani, J. Hoffmann, and M. Steinmetz, "Constructing plan trees for simulated penetration testing", in *The 26th international conference on automated planning and scheduling*, vol. 121, 2016.

[99] A. Gerevini and D. Long, "Plan constraints and preferences in pddl3", tech. rep., 2005.

[100] S. Sohrabi, J. A. Baier, and S. A. McIlraith, "Preferred explanations: Theory and generation via planning", in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[101] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4945–4990, 2017.

[102] T. Joppen, C. Wirth, and J. Fürnkranz, "Preference-based monte carlo tree search", in *KI 2018: Advances in Artificial Intelligence*, pp. 327–340, Springer International Publishing, 2018.

[103] S. Visser, J. Thangarajah, J. Harland, and F. Dignum, "Preference-based reasoning in bdi agent systems", *Autonomous Agents and Multi-Agent Systems*, vol. 30, pp. 291–330, Mar 2016.

[104] G. Behnke, B. Leichtmann, P. Bercher, D. Höller, V. Nitsch, M. Baumann, and S. Biundo, "Help me make a dinner! challenges when assisting humans in action planning", in *Proceedings of the International Conference on Companion Technology, Ulm*, vol. 11, p. 2017, 2017.

[105] R. Pinsler, R. Akrour, T. Osa, J. Peters, and G. Neumann, "Sample and feedback efficient hierarchical reinforcement learning from human preferences", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 596–601, IEEE, 2018.

[106] M. Fox, D. Long, and D. Magazzeni, "Explainable planning", *CoRR*, vol. abs/1709.10256, 2017.

[107] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, and S. Kambhampati, “Plan explicability and predictability for robot task planning”, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1313–1320, IEEE, 2017.

[108] R. Eifler, M. Cashmore, J. Hoffmann, D. Magazzeni, and M. Steinmetz, “Explaining the space of plans through plan-property dependencies”, in *ICAPS-19 Workshop on Explainable Planning*, 2019.

[109] I. Little and S. Thiebaux, “Probabilistic planning vs replanning”, in *ICAPS Workshop on Planning Competitions: Past, Present, and Future*, 2007.

[110] S. Sanner, “Relational Dynamic Influence Diagram Language (RDDL): Language Description.” Online: users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf, 2010.

[111] T. Dean and K. Kanazawa, “A model for reasoning about persistence and causation”, *Computational intelligence*, vol. 5, no. 2, pp. 142–150, 1989.

[112] T. Keller and P. Eyerich, “PROST: Probabilistic planning based on UCT.”, in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[113] A. Kolobov, P. Dai, M. Mausam, and D. S. Weld, “Reverse iterative deepening for finite-horizon MDPs with large branching factors”, in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

[114] M. Fox and D. Long, “PDDL2.1: An extension to PDDL for expressing temporal planning domains”, *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[115] B. D. Smith, K. Rajan, and N. Muscettola, “Knowledge acquisition for the onboard planner of an autonomous spacecraft”, in *EKAW*, 1997.

[116] R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, “Experiences with an architecture for intelligent, reactive agents”, *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 237–256, 1997.

[117] S. W. Yoon, A. Fern, and R. Givan, “FF-Replan: A baseline for probabilistic planning.”, in *International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 352–359, 2007.

[118] S. J. Celorrio, F. Fernández, and D. Borrajo, “The PELA architecture: Integrating planning and learning to improve execution”, in *AAAI*, 2008.

[119] S. Krivic, M. Cashmore, D. Magazzeni, B. Ridder, S. Szedmak, and J. Piater, “Decreasing Uncertainty in Planning with State Prediction”, in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2032–2038, 8 2017.

[120] L. Iocchi, L. Jeanpierre, M. T. Lázaro, and A.-I. Mouaddib, “A practical framework for robust decision-theoretic planning and execution for service robots”, in *International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 486–494, 2016.

[121] R. D. Buksz, M. Cashmore, B. Krarup, D. Magazzeni, and B. C. Ridder, “Strategic-tactical planning for autonomous underwater vehicles over long horizons”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[122] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Elsevier, 2004.

[123] D. Martínez, G. Alenyà, T. Ribeiro, K. Inoue, and C. Torras, “Relational reinforcement learning for planning with exogenous effects”, *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2689–2732, 2017.

[124] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning”, in *International Conference on Machine Learning (ICML)*, pp. 157–163, 1994.

[125] C. Boutilier, T. Dean, and S. Hanks, “Decision-theoretic planning: Structural assumptions and computational leverage”, *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[126] N. Kushmerick, S. Hanks, and D. S. Weld, “An algorithm for probabilistic planning”, *Artificial Intelligence*, vol. 76, no. 1-2, pp. 239–286, 1995.

[127] A. Atrash and S. Koenig, “Probabilistic planning for behavior-based robots”, in *FLAIRS*, pp. 531–535, 2001.

[128] T. Smith and R. Simmons, *Probabilistic planning for robotic exploration*. PhD thesis, Carnegie Mellon University, The Robotics Institute, 2007.

[129] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, “Grasping POMDPs”, in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4685–4692, 2007.

[130] J. Hoey, A. Von Bertoldi, P. Poupart, and A. Mihailidis, “Assisting persons with dementia during handwashing using a partially observable markov decision process”, *Vision Systems*, vol. 65, p. 66, 2007.

[131] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, T. Kollar, C. Mericli, M. Samadi, S. Brandão, and R. Ventura, “Cobots: Collaborative robots servicing multi-floor buildings”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5446–5447, 2012.

[132] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, “Design of an office-guide robot for social interaction studies”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4965–4970, 2006.

[133] J. Hoffmann and R. Brafman, “Contingent planning via heuristic forward search with implicit belief states”, in *Proceedings of the 2005 International Conference on Planning and Scheduling (ICAPS)*, vol. 2005, 2005.

[134] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning”, in *European Conference on Machine Learning (ECML)*, pp. 282–293, Springer, 2006.

[135] M. Heerink, B. Kröse, V. Evers, and B. Wielinga, "Assessing Acceptance of Assistive Social Agent Technology by Older Adults: the Almere Model", *International Journal of Social Robotics*, vol. 2, pp. 361–375, Dec 2010.

[136] C. Silva, J. Vongkulbhisal, M. Marques, J. P. Costeira, and M. Veloso, "Feedbot - a robotic arm for autonomous assisted feeding", in *Progress in Artificial Intelligence*, pp. 486–497, Springer International Publishing, 2017.

[137] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa, "Towards robotic feeding: Role of haptics in fork-based food manipulation", *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1485–1492, 2019.

[138] D. Gallenberger, T. Bhattacharjee, Y. Kim, and S. S. Srinivasa, "Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding", in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 267–276, IEEE, 2019.

[139] Z. Erickson, H. M. Clever, G. Turk, C. K. Liu, and C. C. Kemp, "Deep haptic model predictive control for robot-assisted dressing", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, May 2018.

[140] F. Zhang, A. Cully, and Y. Demiris, "Probabilistic real-time user posture tracking for personalized robot-assisted dressing", *IEEE Transactions on Robotics*, pp. 1–16, 2019.

[141] H. R. Lee and L. D. Riek, "Reframing assistive robots to promote successful aging", *ACM Transactions on Human-Robot Interaction*, vol. 7, pp. 11:1–11:23, May 2018.

[142] S.-E. Chien, L. Chu, H.-H. Lee, C.-C. Yang, F.-H. Lin, P.-L. Yang, T.-M. Wang, and S.-L. Yeh, "Age difference in perceived ease of use, curiosity, and implicit negative attitude toward robots", *ACM Trans. Hum.-Robot Interact.*, vol. 8, pp. 9:1–9:19, June 2019.

[143] A. Kapusta, Z. Erickson, H. M. Clever, W. Yu, C. K. Liu, G. Turk, and C. C. Kemp, "Personalized collaborative plans for robot-assisted dressing via optimization and simulation", *Autonomous Robots*, Jun 2019.

[144] A. Tapus, C. Ţăpuş, and M. J. Matarić, "User—robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy", *Intelligent Service Robotics*, vol. 1, p. 169, Feb 2008.

[145] C. Moro, G. Nejat, and A. Mihailidis, "Learning and personalizing socially assistive robot behaviors to aid with activities of daily living", *ACM Transactions on Human-Robot Interaction*, vol. 7, pp. 15:1–15:25, Oct. 2018.

[146] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S. Kiesler, "Human preferences for robot-human hand-over configurations", in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1986–1993, September 2011.

[147] M. K. Lee, J. Forlizzi, S. Kiesler, P. Rybski, J. Antanitis, and S. Savetsila, "Personalization in hri: A longitudinal field experiment", in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 319–326, 2012.

[148] P. Chevalier, J.-C. Martin, B. Isableu, C. Bazile, and A. Tapus, “Impact of sensory preferences of individuals with autism on the recognition of emotions expressed by two robots, an avatar, and a human”, *Autonomous Robots*, vol. 41, no. 3, pp. 613–635, 2017.

[149] A. Cruz-Maya and A. Tapus, “Learning users’ and personality-gender preferences in close human-robot interaction”, in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 791–798, August 2017.

[150] S. Rossi, M. Staffa, L. Bove, R. Capasso, and G. Ercolano, “User’s personality and activity influence on hri comfortable distances”, in *Social Robotics*, pp. 167–177, Springer International Publishing, 2017.

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

[152] C. J. Hayes, M. Marge, E. Stump, C. Bonial, C. Voss, and S. G. Hill, “Towards learning user preferences for remote robot navigation”, in *Proceedings of the RSS 2018 Workshop on Models and Representations for Human-Robot Communication*, 2018.

[153] I. Torres, N. Hernández, A. Rodríguez, G. Fuentes, and L. A. Pineda, “Reasoning with preferences in service robots”, *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 5105–5114, 2019.

[154] M. M. de Graaf, S. Ben Allouch, and J. A. van Dijk, “Why would i use this in my home? a model of domestic social robot acceptance”, *Human–Computer Interaction*, vol. 34, no. 2, pp. 115–173, 2019.

[155] J. Piasek and K. Wieczorowska-Tobis, “Acceptance and long-term use of a social robot by elderly users in a domestic environment”, in *2018 11th International Conference on Human System Interaction (HSI)*, pp. 478–482, IEEE, 2018.

[156] C.-A. Smarr, A. Prakash, J. M. Beer, T. L. Mitzner, C. C. Kemp, and W. A. Rogers, “Older adults’ preferences for and acceptance of robot assistance for everyday living tasks”, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 56, no. 1, pp. 153–157, 2012.

[157] I. Deutsch, H. Erel, M. Paz, G. Hoffman, and O. Zuckerman, “Home robotic devices for older adults: Opportunities and concerns”, *Computers in Human Behavior*, vol. 98, pp. 122 – 133, 2019.

[158] A. Gessl, S. Schlögl, and N. Mevenkamp, “On the perceptions and acceptance of artificially intelligent robotics and the psychology of the future elderly”, *Behaviour & Information Technology*, pp. 1–20, 2019.

[159] M. Biswas, M. Romeo, A. Cangelosi, and R. B. Jones, “Are older people any different from younger people in the way they want to interact with robots? scenario based survey”, *Journal on Multimodal User Interfaces*, Jul 2019.

[160] G. Canal, S. Escalera, and C. Angulo, “A real-time Human-Robot Interaction system based on gestures for assistive scenarios”, *Computer Vision and Image Understanding*, vol. 149, pp. 65–77, 2016. Special issue on Assistive Computer Vision and Robotics - “Assistive Solutions for Mobility, Communication and HMI”.

[161] A. Colomé, A. Planells, and C. Torras, “A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments”, in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5649–5654, 2015.

[162] I. Naotunna, C. J. Perera, C. Sandaruwan, R. A. R. C. Gopura, and T. D. Lalitharatne, “Meal assistance robots: A review on current status, challenges and future directions”, in *IEEE/SICE International Symposium on System Integration (SII)*, pp. 211–216, Dec 2015.

[163] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L. Morency, “Openface 2.0: Facial behavior analysis toolkit”, in *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pp. 59–66, May 2018.

[164] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299, 2017.

[165] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters”, *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[166] J. Hoffmann, “The Metric-FF planning system: Translating “ignoring delete lists” to numeric state variables”, *Journal of Artificial Intelligence Research*, vol. 20, pp. 291–341, 2003.

[167] M. J. Matarić, J. Eriksson, D. J. Feil-Seifer, and C. J. Winstein, “Socially assistive robotics for post-stroke rehabilitation”, *Journal of NeuroEngineering and Rehabilitation*, vol. 4, no. 1, p. 5, 2007.