Frail human assistance by a humanoid robot using multi-contact planning and physical interaction using : multi-contact planning and physical interaction

Anastasia Bolotnikova

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Systèmes Automatiques et Microélectroniques

École doctorale : Information, Structures, Systèmes

Unité de recherche UMR5506

Frail human assistance by a humanoid robot using multi-contact planning and physical interaction

Présentée par Anastasia Bolotnikova

Le 25 mars 2021

Sous la direction de Abderrahmane Kheddar et Sébastien Courtois

Devant le jury composé de

Sandra Hirche, Professor, Technical University Munich	Président de jury
Dana Kulić, Professor, Monash University	Rapporteur
Vincent Padois, Directeur de Recherche, Inria Bordeaux	Rapporteur
Alessandro De Luca, Professor, Sapienza University of Rome	Examinateur
Sébastien Courtois, Docteur, SoftBank Robotics Europe	Co-encadrant de thèse
Abderrahmane Kheddar, Directeur de Recherche, CNRS	Directeur de thèse

UNIVERSITÉ DE MONTPELLIER

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my Ph.D supervisor Professor Abderrahmane Kheddar for giving me a great chance to conduct research on such an interesting, exciting and significant topic. His valuable guidance and support during these years were a great contribution to my professional development.

Secondly, I would like to thank my Ph.D co-supervisor Sébastien Courtois, from SoftBank Robotics Europe, for helping, supporting and advising me during the course of my work at SoftBank. I have learned many important aspects of the research work in the industrial settings thanks to his supervision.

Additionally, I would like to thank the ANRT (National Agency for Research and Technology) for managing the CIFRE Ph.D scheme, which gave me an incredibly valuable opportunity to conduct my research work while being associated with the industrial partner, SoftBank Robotics Europe. This allowed me to gain the very useful perspective of the importance and role of research in the industrial context.

I would like to thank my colleagues from SoftBank Robotics Europe for a welcoming working environment and for always being ready to help. A very special gratitude for their help and support goes out to Michel Besombes, Vincent Bonnet, Maxime Busy, Maxime Caniot, Alban Laflaquière, Sébastien Dalibard.

I would also like to thank the members of the Interactive Digital Humans research group for contributing to the fruitful and supporting research work environment. A special gratitude goes out to the research engineers Arnaud Tanguy and Pierre Gergondet for their great help with the software development. I would also like to thank Adrien Escande for his assistance and helpful advice.

I am extremely grateful to Professor Dana Kulić, Professor Vincent Padois, Professor Sandra Hirche and Professor Alessandro De Luca for being part of my thesis jury and taking time to review my work and share their feedback on the research conducted in the frame of this thesis.

I would also like to thank the L'Oréal-UNESCO For Women in Science program for creating an amazingly supportive community for fostering career development of young female researchers.

Last but not least, I would like to thank my family, friends and my mother in particular for an incredible support. Without them none of this would be possible!

Anastasia Bolotnikova, 11 January 2021, Montpellier, France

Abstract

As the percentage of the elderly population is rising worldwide, the demand imposed on modern society for providing a sufficient amount of skilled workers in the caregiving sector is becoming increasingly harder to fulfill. Utilization of robotic technologies in physical assistance in a home support context can contribute to sustaining a frail person's autonomy and quality of life. We envision the use of the humanoid robot technology for providing daily assistance with physical motion tasks. More specifically, in this work, we focus on the usage of Pepper humanoid robot platform, mass-produced by SoftBank Robotics. The choice of the platform is motivated by its affordability, user-friendly design and multi-modal communication capabilities.

First, we propose and develop a proprioceptive sensor based contact detection method. In order to maintain low cost of the platform, our method for contact detection aims at using only the available Pepper's sensors to detect collision with the environment, namely a contact event during physical human-robot interaction. We detail the integration of the proposed method as a feedback signal in the whole-body controller to react to human touch in real-time.

Secondly, we investigate the whole-body humanoid robot posture planning problem in the assistive physical human-robot interaction context. We augment the non-linear optimization based posture generation framework with necessary components that allow us to plan a robot attitude in contact with a human point cloud. The proposed human point cloud processing pipeline provides the necessary data structures to formulate the posture generation problem for a robot to initiate a physical assistance task.

Then, we present a fully autonomous interaction scenario for initiating a physical assistance process. A Finite-State Machine and task-space Quadratic Programming based controller is developed for a robot to navigate towards a human, perform a multimodal communication and establish several physical contacts in a fully autonomous fashion. The controller performance is demonstrated in real experiments with a human subject. All the software tools developed to perform whole-body task space Quadratic Programming based control on SoftBank humanoid robots are made publicly available and are documented in detail.

Finally, we study the problem of partial physical assistance in motion. We present a control methodology that enables a humanoid robot to supply the assistive forces necessary to help a frail human to achieve a desired performance of a motion task. We present and discuss the simulation results of the proposed method.

We conclude this work with discussion of the achieved results and the future perspectives of research in the area of humanoid-human interaction for physical assistance.

Keywords

Contact observer, humanoid robots, whole-body control, multimodal communication, physical human-robot interaction.

Résumé de la thèse

Alors que le pourcentage de seniors parmi la population mondiale augmente, la quantité de personnel soignant qualifié pour l'aide à cette catégorie de personnes est elle en perpétuelle diminution. Cette thèse défend l'idée que l'utilisation des technologies robotiques pour une assistance physique pourrait contribuer à maintenir l'autonomie et la qualité de vie des personnes fragiles, et par conséquent un maintien à domicile plus long. Les robots humanoïdes peuvent prendre part à une telle vision, notamment pour effectuer les tâches à valeurs non ajoutées pour le personnel soignant ou la famille.

Au cours de ce travail, une attention plus particulière est portée à l'utilisation de la plateforme humanoïde, Pepper, premier robot humanoïde produit en grande série. Le choix de cette plateforme est motivé par son accessibilité lié à son coût mais également par sa conception voulue sociale, qui le rend convivial ; ses capacités de communication multimodales facilitent grandement certaines tâches.

Dans ce cadre, nous avons développé une méthode de détection de contact basée sur les capteurs proprioceptifs. Afin de maintenir le coût du robot, la détection du contact utilise uniquement les capteurs déjà présents sur le robot. Il s'agit pour le robot de détecter les collisions avec l'environnement, et plus spécifiquement la détection de l'interaction physique homme/robot. L'intégration de la méthode proposée passe par l'analyse du signal de retour des capteurs pour ajuster la réponse en temps réel du robot à l'événement de contact détecté.

Ensuite, nous avons abordé le problème de la planification de la posture des robots humanoïdes dans le contexte de l'interaction physique homme-robot. Nous avons revu le framework de génération de posture basé sur l'optimisation non linéaire avec les composants nécessaires qui permettent de planifier une posture en contact avec un nuage de points issu de la perception de la personne à assister. Le pipeline de traitement du nuage de points proposé fournit les structures de données nécessaires pour formuler le problème de génération de posture pour qu'un robot puisse initier une tâche d'assistance physique.

Suite aux discussions avec un centre EHPAD, un premier scénario d'interaction entièrement autonome est proposé pour initier le processus d'assistance. Un contrôleur dans l'espace des tâches formulé comme un programme quadratique est développé pour que Pepper puisse atteindre une personne, effectuer une communication multimodale et établir plusieurs contacts physiques de manière totalement autonome. La performance du contrôleur est démontrée par une expérience réelle. Tous les outils logiciels développés pour effectuer le contrôle du corps entier des robots humanoïdes de SoftBank sont mis à la disposition du public et sont documentés en détail.

Enfin, nous avons entamé le problème de l'assistance physique à des mouvements prédéfinis. Nous présentons une méthodologie de contrôle adaptative qui permet au robot Pepper de fournir les forces d'assistance nécessaires pour accompagner un mouvement effectué (ici le bras) par une personne avec une suppléance des couples articulaires. Nous présentons les résultats préliminaires pour l'approche proposée.

Nous concluons notre thèse par une discussion sur les résultats obtenus et les perspectives futures de la recherche concernant l'interaction physique homme-robot pour l'assistance physique au mouvement.

Mots clés: observateur de contact, robots humanoïdes, contrôle du corps entier, communication multimodale, interaction physique homme-robot.

Contents

NOMENCLATURE

Acronyms and abbreviations

2D / 3D / 6D	2/3/6 dimensional
CM	Confidence Map
CNN	Convolutional Neural Network
CoM	Center of Mass
Covid19	Coronavirus disease 2019
DC	Direct Current
DCM	Device Communication Manager
DoF	Degree of Freedom
FD	Forward Dynamics
FN / FP	False Negative / False Positive
FoV	Field of view
FSM	Finite State Machine
FSR	Force Sensitive Resistor
GJK	Gilbert-Johnson-Keerthi
IBVS	Image Based Visual Servoing
ID	Inverse Dynamics

IMU	Inertial Measurement Unit
LED	Light-emitting diode
LM	Linear Model
LERoll / RERoll	Left/Right Elbow Roll
LSPitch / RSPitch	Left/Right Shoulder Pitch
LSRoll / RSRoll	Left/Right Shoulder Roll
MAX	Maximum Absolute Error
MC_RTC	Multi-contact Real Time Control (framework)
MQP	Multi-robot Quadratic Programming (controller)
NN	Neural Network
NRMSE	Normalized Root-Mean-Square Error
NURBS	Non-Uniform Rational B-Spline
OS	Operating System
PAF	Part Affinity Field
PBVS	Position Based Visual Servoing
PC	Personal Computer
PD / PI	Proportional Derivative / Proportional Integral
PG	Posture Generator
PID	Proportional Integral Derivative
QP	Quadratic Programming
RGB / RGBD	Red Green Blue / Depth
RMSE	Root Mean Square Error
ROS	Robot Operating System (robotics middleware)

SBRE	SoftBank Robotics Europe
TP	True Positive
URDF	Universal Robot Description Format
V-SLAM / SLAM	Visual Simultaneous localization and mapping
pHRI / HRI	physical Human-Robot Interaction

List of symbols

$\epsilon, \dot{\epsilon}$	joint position tracking error and its derivative
$q_d, \dot{q}_d, \ddot{q}_d$	desired joint position, velocity and acceleration
q	joint position
K_p, K_d	proportinal and derivative gains
u	voltage
e	back-electromotive force
K_e	back-electromotive force constant
L	inductance
R	resistance
i	electrical current
t	time
ω	motor speed
τ_m	motor torque
K_t	torque constant
J_m	motor inertia
μ	motor friction constant
τ_l	load torque

$M(q)$	inertia matrix
$c(q, \dot{q})$	Coriolis, centrifugal and gravity forces
τ_f	friction torque
τ_B	spring-damper regularization torque term
τ_{ext}	external torque
θ	motor angle
ϕ	motor-link angle difference
α	motor-link backlash gap
$K_{\{x\}v}, K_{\{x\}s}$	viscous and static friction coeficient
$K_{\phi p}, K_{\phi d}$	stiffness and damping coefficients
τ_{ld}	desired load torque
τ_d	desired joint torque
ϵ_{exp}	expected joint position tracking error
r	contact observer signal
δ	contact observer threshold
K_c, K_v	compliance and velocity gains
$\mathcal{T}$	assistance task instruction
ϕ	posture generator cost function
u, v	NURBS curve parameters
f	robot contact wrenches (stacked vector)
$P^e = \{p^e, R^e\}$	contact point frame position and orientation
$\mathcal{S}$	NURBS surface
UV	NURBS surface parametric space
$\Omega^{\mathcal{C}}$	subspace of NURBS surface parametric space

D	point cloud
$N_{x,y}(z)$	B-spline basis function
$\mathcal{C}$	constraining closed curve
g_k	point cloud projected onto NURBS surface perametric space
$\vec{n}_{t_k}$	curve normal at a point
s	curve order
q^{pref}	preferred robot posture
h^{avg}	sub-cloud average point
$\mathcal{P}$	posture task (QP objective)
$\mathcal{B}$	mobile base position task (QP objective)
$\mathcal{C}$	center of mass task (QP objective)
q_d	posture set point
$\epsilon_{\mathcal{P}}$	posture task error
$\delta_{\mathcal{P}}$	posture task completion threshold
$\mathcal{V}$	PBVS task (QP objective)
${}^aX_b$	frame transformation between bodies a and b
$\mathcal{O}$	IBVS task (QP objective)

INTRODUCTION

By the middle of the 21st century, the projected percentage of people over 60 years of age will reach over 20% of the world population and almost 35% of the population in Europe; this growth is most critical in Japan; even third world countries will be affected. The problem is therefore global, and in Europe it is quite alarming. People over 65 year old are the world's most rapidly growing demographic. By the year of 2050, 16% of the global population is expected to represent this age group, which would constitute a 7% increase from the 9% of the global population falling into this age category in 2019 un2 (2019). This imposes a global problem, when there are more people in the world requiring help than those who can provide it. As a result, it will become increasingly difficult to provide quality assistance to those who need it. That is why we need to think today, and in an urgent manner, how we can arrange life in society so that elderly and frail people who require daily assistance with various tasks have access to healthcare services and care that they need. Moreover, these challenges must be addressed with a careful consideration of all the relevant social, cultural, ethical and economics constraints.

The various robotics technologies are expected to play a pivotal role in addressing the challenges of the lack of workforce in the caregiving sector Niemelä and Melkas (2019). Many researchers all over the world are dedicating their efforts to achieve safe, reliable and comfortable human-robot interaction. The research in this rapidly expanding area is useful in industrial settings, where people collaborate with the robots, also often called *cobots* in this context, to perform manufacturing tasks (Fig. 1). At the same time, advancing human-robot interaction technologies is also advantageous for the healthcare settings, where robots can help human caregivers to take care of the frail people, or even in the home settings to provide required daily assistance and allow a frail person to remain in the comfort of their own home (Fig. 2).

FIGURE 1 – *Physical human-robot interaction with* cobots *in industrial settings.*

The research presented in this thesis focuses on tackling the problem imposed by the growing need of robotization of the healthcare and caregiving sectors with the use

FIGURE 2 – *Physical human-robot interaction for providing daily assistance to frail and elderly.*

of the social domotic humanoid robots, namely the Pepper humanoid robots. Most people have already encountered the Pepper humanoid robot, mass-produced by SoftBank Robotics Pandey and Gelin (2018), either in real life or on television or Internet. This is a world first affordable humanoid robot specifically designed for a user-friendly social interaction and for providing the sense of companionship, which makes it a good candidate for providing a high quality of care for frail people by making them feel both safe and comfortable. Pepper robots can already move around the room with an omnidirectional mobile base, detect people in its environment, have simple dialogues with people and recognize principal human emotions. However, in order for this platform to become a major player in addressing the challenges of the aging population, the robot needs to have the skills for interacting with humans physically in a safe, reliable and comfortable manner. In the future, we want humanoid robots, such as Pepper, to efficiently assist frail people with daily physical tasks. So what needs to be done to move towards this goal?

The primary skill that a robot needs to possess to actively engage in and appropriately react to the human-robot physical interaction is the ability to detect the contact events. The current design of a Pepper humanoid robot includes few tactile sensors, at the back of the hands and on the head, that allow for a simple tactile interaction with the robot, e.g. touch on the robot's head can trigger a certain pre-programmed robot reaction. In order to appropriately engage in a more advanced physical interaction, however, the robot ideally must be able to detect a contact not only in some specific parts of its body, but on any of its links. Yet, the installation of the tactile or force sensors all over the platform's body would result in an insurmountable increase of the platform cost. Thus, an alternative solution must be considered. In this work, we propose a methodology for a contact event detection, that only uses the sensors that are already available of the current version of the Pepper robot. We develop and study a joint position tracking error discrepancy based contact observer. Using the output of an acceleration-resolved whole-body Quadratic Programming based controller, a decision tree model is trained on a collected dataset of a certain robot motion task to predict a normal, or expected, joint position tracking error. The prediction of the error is compared with the readings of the robot encoders to detect a discrepancy. If the discrepancy is higher than a predefined threshold value, a contact event is detected for a certain robot link. This information is passed as a feedback signal to the controller that determines what will be the robot reaction to the detected contact event, e.g. stop moving or comply to the touch. The proposed methodology allows to create contact-aware whole-body controllers for humanoid robots without requiring any increase in the platform cost.

Besides being contact-aware, the humanoid robot needs to be able to autonomously plan its posture for physical interaction with a human. All the robot-, human-, environment and physical interaction task related constraints must be taken into account when formulating the posture planning problem to ensure that a feasible and safe posture for a humanoid to physically interact with a human can be computed. Similar whole-body posture planning methods have already been investigated and developed for the industrial context, for instance. In this work, we investigate what are the additional components required to re-formulate a posture planning optimization problem to include also human and assistance task constraints and objectives. We develop a human point cloud processing pipeline that supplies the necessary data structures to the nonlinear optimization based posture planning framework. A constrained parametric surface, fitted on an assistance task related human body part point cloud is used for a robot to autonomously plan the exact contact location on the surface. The strictly convex collision hulls fitted on the point cloud areas that represent other human body parts are used to formulate collision avoidance constraints. Additional elements, such as maximum contact force constraints or humanoid head orientation objective, can be added to the problem formulation if the task requires. The solution of such an optimization problem is a feasible, safe and appropriate posture suitable for the humanoid to initiate the physical assistance task.

Before a humanoid robot starts to engage in physical interaction with a human, two important aspects need to be taken care of. First, the robot must autonomously navigate to the position in the environment that is sufficiently close with respect to the human for the robot to reach a person to establish the physical contacts. In this work, we use a wide-angle depth camera on the Pepper robot prototype, which allows us to perform a sufficiently accurate human detection even in a very close proximity to a person. The feedback from the camera and the human detection module is used as feedback in the Position Based Visual Servoing task of the whole-body controller to regulate the autonomous robot navigation process.

Once the robot reaches a position relatively close to the human, the second important aspect that requires careful implementation is the comprehensive communication of robot intent. Before engaging in the physical interaction, the robot must ensure that the human is comfortable and prepared for the interaction process. In this work, we augment the whole-body controller with multimodal communication features. The Pepper robot uses verbal, visual and body language modalities to communicate its intentions of establishing physical contacts with the human. With this controller feature, we ensure that the human is well informed about the following actions that the robot will perform, which increases the chances of a smooth interaction process.

Finally, once the assistance process is correctly initiated, the robot needs to serve as a source of the assistive force. In the physical assistance task, two sources of force are contributing to the motion task. First is the force generated by the frail person themselves and the other is the assistive force supplied by the robot. As the human force cannot be measured or known a priori, the robot must adapt to the presence of this unknown variable and regulate its contribution to the motion task accordingly. In this work, we propose a control method for adapting humanoid robot contribution to the known motion task while accounting for the presence of unknown human contribution. Identifying a model of a reference torque needed for the task and estimating human contribution with a combination of observer and predictive experience based model, the required robot torque contribution to the task is adapted. The interaction

force required to supply the required assistive torque is computed by a whole-body humanoid robot motion controller.

These are just some of the main robot skills required for bringing a humanoid robot technology closer to being used in the physical assistance context. Besides the mentioned skills, which we focus on in this work, better human language and human motion understanding as well as extensive user-studies etc. are also contributing to the list of research areas which require attention for enabling humanoids to assist people efficiently with daily life tasks.

The work conducted in the frame of this thesis is presented in this manuscript according to the following structure:

- In Chapter 1 we present the proposed contact observer approach. First, we state the problem and discuss the state-of-the-art. Secondly, the proposed method is described in detail and first experimental results are demonstrated Bolotnikova et al. (2018b).
- Then, in Chapter 2, we present a detailed description of the integration of the proposed contact observer methodology as a feedback signal in a real-time humanoid robot whole-body controller. We conclude this chapter with a presentation of the results from the real physical interaction experiments with a human subject Bolotnikova et al. (2018a).
- Chapter 3 is dedicated to the multi-contact posture planning method developed for the physical human-robot interaction. The posture generation framework, previously used in the industrial context, is augmented with the components required for planning a contact on a constrained parametric surface fitted to the human point cloud. Several human-robot physical interaction scenarios are used to demonstrate the resulting multi-contact humanoid robot postures computed using the augmented posture generation framework Bolotnikova et al. (2020b).
- Chapter 4 describes the whole-body controller developed for a fully autonomous initiation of human physical assistance by a humanoid robot. The controller implementation details and experimental results with a human subject are presented. Moreover, the controller code used both for simulation and real experiments is made publicly available Bolotnikova et al. (2020a).
- In Chapter 5 an adaptive force control method for humanoid-to-human assistance with a priori known motion task is proposed. The simulation results of the proposed methodology are presented and discussed Bolotnikova et al. (2021a).

We conclude our work with discussion of the achieved results and the future research perspectives in the area of human-robot interaction in the daily life assistance context.

Appendix A presents a software toolkit developed in the frame of this thesis Bolotnikova et al. (2021b). These are the software components that allow to perform a whole-body task-space Quadratic Programming based control on SoftBank humanoid robots. The software components are developed with human-robot interaction application in mind. An additional scenario of daily assistance, namely medicine delivery, is used to demonstrate the modularity, ease of use and wide area of application of the developed software.

CHAPTER 1

CONTACT OBSERVER

For a humanoid robot to actively participate in a physical interaction with its environment or a human user, a real-time solution for a contact observer is required. In this chapter, we present the methodology proposed for proprioceptive sensor based contact sensing suitable for affordable personal robots with no force/torque or electric current sensing. According to the proposed method, we combine robot model knowledge and the output of an acceleration resolved quadratic programming whole-body controller to make a prediction of the expected position. The predicted expected position error is then compared to the error measured by the robot encoders for computing the contact observer signal. At the end of this chapter, we demonstrate the performance of our proposed approach in the experiments of contact detection and estimation of collision direction and intensity on a real humanoid robot Pepper platform controlled by a task-space multi-objective quadratic programming controller.

1.1 Introduction

The goal of the work presented in this chapter is to propose, develop and test the contact observer methodology suitable for implementation on a Pepper humanoid robot. The main motivation for such method development is to use the contact observer to regulate robot behaviour in the context of physical Human-Robot Interaction (pHRI). The use of force/torque sensors is not available on the current Pepper platforms. The addition of such sensors to the platform would result in an unacceptable increase of the platform cost, weight and complexity. As a consequence, the main challenge addressed in this chapter is that the contact sensing must be addressed by using only the sensors which are available on the Pepper robot platform.

The Pepper humanoid robot platform (shown on Fig. 1.1), produced by the SoftBank Robotics Europe (SBRE) company, is often presented in various public places and events. People, especially children, express great interest in interacting with the robot. That also includes the interest to interact with the robot physically, by touching its face, head, hands etc. So far, meaningful physical interaction, where the robot can detect and react to the touch, was limited to only a few tactile sensors, installed on the robot's head and hands. Often, however, people excitedly touch various other robot links, while the robot can express no reaction as it remains "unaware" of these contacts. Enabling ro-

FIGURE 1.1 – *The platform used in our study: Pepper humanoid robot that is widely used in costumer service and research spheres.*

bust whole-body contact observer for the Pepper robot has a potential of bringing the pHRI experience to a whole new level with meaningful robot reaction to various external contacts and ability to interact with the environment through taking contacts and applying forces, not to mention that this functionality can serve as a safety reflex when human touch can damage robot structure or *vice versa* Pandey et al. (2014).

The practical implementation aspects of a contact sensing solution is particularly challenging for low-cost personal robots, such as Pepper, where the embedded sensors are limited and the design mechanics and kinematics do not obey high precision requirements. In this chapter, we describe an approach which aims to overcome those limitations and enable whole-body contact sensing for Pepper.

First, we review the progress made in the proprioceptive sensor based contact sensing in recent years and outline why existing approaches could not be adapted for the Pepper platform (Sec. 1.2). Then, we present our proposed methodology (Sec. 1.3, Sec. 1.4). Finally, we demonstrate the performance of the proposed approach (Sec. 1.5) and draw conclusions with discussion on current method limitations and further developments (Sec. 1.6).

The contributions of the work presented in this chapter are the following:

1. We derive a formula for expected tracking error computation for a Direct Current (DC) motor controlled with PD scheme;

2. We describe the process of non-linear system identification for expected tracking error prediction based on the knowledge of desired trajectory and the robot model;

3. With the ability to predict which part of the position tracking error is related to the normal collision-free motion, we propose a novel contact observer signal, which incorporates direction and intensity information of the collision event;

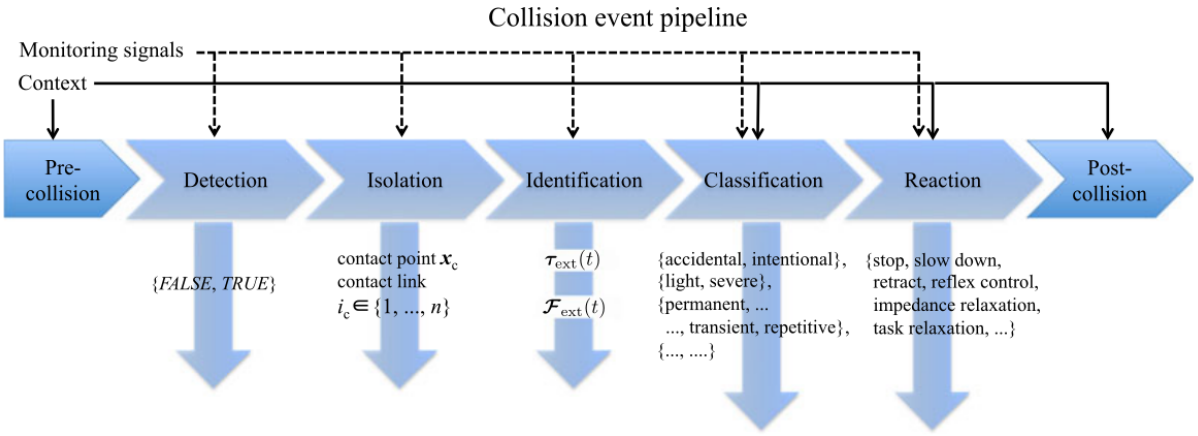


FIGURE 1.2 – *The collision event pipeline [Haddadin et al. (2017)].*

4. We perform experiments with a Pepper robot platform and demonstrate high sensitivity of our proposed contact observer and good performance of contact detection and identification of collision direction and intensity for various robot motions.

1.2 Background

Various methods have been proposed over the years for proprioceptive sensor based contact sensing for robotic manipulators. The overview of several such techniques, namely the direct estimation of the external torque, energy, velocity and momentum observers, is well documented in the survey paper on robot collisions Haddadin et al. (2017), where the collision event pipeline is also introduced (Fig. 1.2). In this chapter, we focus on three main phases of the collision event pipeline: (i) detection– *did collision occur*?, (ii) isolation– *where on the robot collision occurred*?, (iii) identification– *what is the direction and intensity of the collision*?

Among all the methods, presented in the survey, the best performing one proved to be the momentum observer De Luca and Mattone (2005), as it can effectively address all three main phases of the collision event pipeline while avoiding the estimation of the joint acceleration and mass matrix inversion.

This method has been extended for the application on floating base (humanoid) systems in Flacco et al. (2016). The momentum observer has also been augmented to include common non-linear effects, namely large backlash and friction, commonly encountered on low-cost platforms Flacco and Kheddar (2017); updated momentum observer was implemented and tested on the Romeo robot arm, which is also developed and manufactured by SBRE. Another interesting work in this area has addressed reconstruction of the interaction forces in static conditions; it was implemented and tested on a small humanoid robot from SBRE named NAO Mattioli and Vendittelli (2017).

Initially, the momentum observer was introduced for a single contact isolation, however, it was also used as a base for the multi-contact isolation method in Manuelli and Tedrake (2016). The momentum observer can be used more efficiently in combination with force sensor measurements, when a force/torque sensor is installed either at the robot base for fixed platforms Buondonno and De Luca (2016) or when force

sensors are present on some of the many robot links, as was done for humanoid Atlas in Vorndamme et al. (2017). Such extensions of the momentum observer, however, are not applicable to low-cost robots due to the lack of force sensing devices on the platform, mainly because of their cost and the logistics they require.

There are several reasons, why the classical momentum observer (that we investigated and tried) cannot be efficiently applied to platforms like Pepper robot, namely:

1. Motor-side friction is significant and will appear as an external torque in the residual vector unless friction compensation is appropriately implemented;
2. Significant motor-link backlash and flexibility in some joints violates the assumption that motor and joint angles coincide and consequently torque from the motor is not always well transmitted to the link-side;
3. Motor torque, τ_m, which is usually estimated from electric current and current to torque constants, cannot (for now) be exploited from Pepper due to current measurements being absolute and down-sampled; if this issue could be resolved in the future, it can be an addition to our presented method.

We could use the method developed in Flacco and Kheddar (2017) to overcome friction and backlash; yet it requires having two encoders per joint. As for now, the motor-side encoder measurements are inaccessible from the robot's central memory to measure and account for the motor-link backlash.

The momentum observer method other shortcomings are the estimation delay due to the necessity to set a high observer gain and the requirement to apply filtering to the estimated noisy signal, which in turn results in an even lower collision sensitivity and consequently adds a further delay to the collision detection.

An alternative method, which is also included in the survey paper, is the direct estimation of the external joint torque τ_{ext}. This method is the most straightforward and simplest to implement (Eq. 1.1).

$$\hat{\tau}_{\text{ext}} = \hat{M}(q)\ddot{q} + \hat{c}(q, \dot{q}) - \tau_m \tag{1.1}$$

where $q, \dot{q}, \ddot{q}$ are the robot joint position, velocity and acceleration vectors, $\hat{M}$ is the estimate of the inertia matrix, $\hat{c}$ is the vector of the estimated Coriolis, centrifugal and gravity forces and τ_m is the vector of the measured motor torques. This method, however, requires to estimate joint acceleration $\ddot{q}$, typically obtained via double derivation of the joint position q measurements that results in a very noisy signal. A method has been proposed recently to overcome this issue by generating a high-accuracy and high-bandwidth joint acceleration and velocity estimates via fusion of the proprioceptive sensor measurements and the measurements from the links IMU sensors Birjandi et al. (2020a). This method has been further improved to reduce robot dynamics model errors' influence on the sensitivity of the collision detection Birjandi and Haddadin (2020). These developments led to the improved collision detection accuracy and sensitivity compared to the state of the art methods, such as the momentum observer. Nevertheless, the utilization of this method requires to use joint torque sensor measurements and robot links IMU measurements, not available on the current Pepper robot.

In the view of the aforementioned constraints, instead of using the momentum observer method or the observer-extended direct method, we address the whole-body

contact observer for Pepper by means of monitoring the difference between measured position tracking error and predicted expected position tracking error (i.e. without external torques during free or static motions) given known robot model and motion "intentions". In our work, Pepper is controlled by an acceleration resolved quadratic programming controller (QP) Bouyarmane and Kheddar (2011); Bouyarmane et al. (2019), which we use to compute desired acceleration (and subsequently velocity and position by numerical integration) and desired model-based joint torque for a given motion task. We use those quantities to train a machine learning model that makes a prediction of the expected tracking error value. Note, that the use of machine learning techniques has recently gained popularity in the realm of novel collision detection methods based on proprioceptive sensor measurements, Zwiener et al. (2018); Heo et al. (2019) to name a few. Indeed, such approaches allow to exploit the data, which can easily be collected, to avoid dealing directly with issues like model errors or the effects that are hard to model accurately, such as backlash or friction.

In the next section of this chapter, we present the detailed developments of our approach for expected tracking error prediction and contact observer signal computation.

1.3 Proposed contact observer method

For the development of the contact observer method, we are challenged to use only position tracking error to extract the collision event information: intensity, direction and link. However, we assume the condition of having compliant (low PD gains with or without feedforward terms) semi-reversible or totally reversible actuators. This assumption holds in case of the Pepper platform, which was designed to be safe (low gains) and semi-reversible, hence inherently compliant.

In static settings, when the join position tracking error value does not vary significantly and remains very small (near zero), the contact event monitoring based on joint position tracking error is trivial –collision or contact event causes the static tracking error to change and reveals the collision intensity (i.e. stronger collision causes larger deviation from the static near zero error), direction (positive or negative deviation) and link (last joint in the chain with tracking error exceeding a given threshold) information, assuming the external contact direction is such that it causes joint displacement (which doesn't happen if the direction of the contact force is normal to the joint axis).

When the robot moves, the problem becomes more complex –the tracking error is not constant as in the static case; its dynamics (i.e. the increase or decrease of the tracking error) is not always caused by collision. Indeed, it is due to the fact that because of the dynamics (inertia, Coriolis...) and the posture configuration (w.r.t gravity) each joint might not yet reach desired steady-state position. In this case we need to be able to distinguish when the tracking error increase is caused by a collision event and when it simply caused by the free joint motion dynamics.

To define a joint position tracking error based contact observer signal, we eliminate from the tracking error the part that refers to normal/expected joint motion and leave only the part of the tracking error which is caused by a collision. In order to achieve this, we identify the relationship between the robot motion intention in terms of desired trajectory and the expected joint position tracking error that would occur if such motion is not blocked by any external collision. In the following, one degree of free-

dom toy example, we show the rational ground that drives our reasoning. Consider a DC motor regulated by a proportional-derivative (PD) controller, with gains K_p and K_d respectively. Subtracting joint position sensor measurements q from desired joint position target q_d gives the tracking error value $\epsilon = q_d - q$. The value of ϵ is used as a feedback in the PD controller to compute desired voltage input u (Eq. 1.2).

$$u = K_p\epsilon + K_d\dot{\epsilon} \tag{1.2}$$

The electric equation of a DC motor with resistance R and inductance L is given as Eq. 1.3.

$$u - e = L\frac{di}{dt} + Ri \tag{1.3}$$

where e is the back-electromotive force that is proportional to the motor speed ω with proportionality constant K_e (Eq. 1.4).

$$e = K_e\omega \tag{1.4}$$

The motor torque is proportional to the electrical current i with proportionality constant K_t (Eq. 1.5).

$$\tau_m = K_t i \tag{1.5}$$

The dynamic equation of the motor is given as Eq. 1.6

$$\tau_m = J_m\dot{\omega} + \mu\omega + \tau_l \tag{1.6}$$

where J_m is the motor inertia, μ is the motor friction constant and τ_l is the load torque that includes motor-link friction and backlash effects (Eq. 1.7)

$$\tau_l = M(q)\ddot{q} + c(q,\dot{q}) + \tau_f + \tau_B - \tau_{\text{ext}} \tag{1.7}$$

where M is the load inertia and c combines Coriolis, centrifugal and gravity forces. Following the friction modeling principles, used for the Romeo arm in Flacco and Kheddar (2017), τ_f depends on the motor-link backlash and is expressed as Eq. 1.8.

$$\tau_f = \begin{cases} K_{\phi v}\dot{\phi} + K_{\phi s}\text{sign}(\dot{\phi}) + \\ \quad K_{qv}\dot{q} + K_{qs}\text{sign}(\dot{q}) & |\phi| < \alpha \\ K_{\theta v}\dot{\theta} + K_{\theta s}\text{sign}(\dot{\theta}) & \text{otherwise} \end{cases} \tag{1.8}$$

where θ is the motor angle, $\phi = q - \theta$ is the difference between the joint and the motor angles, α is the size of the motor-link backlash gap and $K_{\{x\}v}, K_{\{x\}s}$ are viscous and static friction coefficients respectively.

When $|\phi| < \alpha$, the motor is moving inside the backlash gap, hence, no torque is transfered from the motor to the load (i.e. $\tau_l = 0$). Otherwise, the motor is in contact with one of the borders of the backlash gap and the load is moving together with the motor. The τ_B term in Eq. 1.7 is a spring-damper regularization term to model the effect due to the motor-link backlash interaction (Eq. 1.9).

$$\tau_B = \begin{cases} -M(q)\ddot{q} - c(q,\dot{q}) - \tau_f + \tau_{\text{ext}} & |\phi| < \alpha \\ K_{\phi p}(\phi + \alpha) + K_{\phi d}\dot{\phi} & \phi \geq \alpha \\ K_{\phi p}(\phi - \alpha) + K_{\phi d}\dot{\phi} & \phi \leq -\alpha \end{cases} \tag{1.9}$$

Here, the "inside backlash equation" (i.e. $|\phi| < \alpha$ case) simply nulls the load torque as seen from the actuator; $K_{\phi p}$ and $K_{\phi d}$ are the stiffness and damping coefficients. Note that both τ_f and τ_B can be thought of as $\xi(q, \dot{q}, \theta, \dot{\theta}, \alpha)$ for simplicity.

The variable τ_{ext} in Eq. 1.7 is any other external torque, e.g. that caused by an external collision on the link.

Substituting (1.4) and (1.5) into (1.3) gives Eq. 1.10.

$$u = K_e\omega + R\frac{\tau_m}{K_t} \tag{1.10}$$

We neglect $L\frac{di}{dt}$ term due to its relative insignificance compared to e and Ri.

Substituting (1.2) and (1.6) into (1.10) gives the analytical relation between tracking error ϵ and the external torque applied on the load τ_{ext} (Eq 1.11).

$$K_p\epsilon + K_d\dot{\epsilon} = \frac{R}{K_t}J_m\dot{\omega} + \left(\frac{R}{K_t}\mu + K_e\right)\omega + \frac{R}{K_t}\left(M(q)\ddot{q} + c(q, \dot{q}) + \tau_f + \tau_B - \tau_{\text{ext}}\right) \tag{1.11}$$

Same relation can be derived for other types of control schemes (PID, PI, etc.) in analogous form.

For the contact observer method development, this relation can be exploited in the following ways. First, it shows that by measuring $q, \dot{q}, \ddot{q}, \omega, \dot{\omega}, \epsilon$ and $\dot{\epsilon}$ and knowing robot model (M, c), motor properties (R, K_t, K_e, μ, J_m) and controller gains (K_p, K_d), as well as all other constants present in Eq. 1.11, the value of τ_{ext} can be computed directly without necessity to measure motor torque or electric current. Secondly, assuming the motion of the load free of external collisions, i.e. $\tau_{\text{ext}} = 0$ we can use Eq. 1.11 in order to compute expected (under free motion assumption) tracking error ϵ_{exp} from the value of desired position, speed and acceleration of the load $q_d, \dot{q}_d, \ddot{q}_d$ (Eq. 1.12).

$$\epsilon_{\text{exp}} = \frac{RJ_mN}{K_tK_p}\ddot{q}_d + \left(\frac{R}{K_t}\mu + K_e\right)\frac{N}{K_p}\dot{q}_d + \frac{R}{K_tK_p}\Bigg(\underbrace{M(q_d)\ddot{q}_d + c(q_d, \dot{q}_d)}_{\text{desired load torque } \tau_{ld}} + \tau_f + \tau_B\Bigg) - \frac{K_d}{K_p}\dot{\epsilon}_{\text{exp}} \tag{1.12}$$

where $\dfrac{1}{N}$ is the gear reduction ratio ($\omega = N\dot{q}$). Yet, in the presence of significant backlash, it is more accurate to model ω as a function of $\dot{q}$ that also includes the motor-link backlash effect ($\omega = \xi(\dot{q}, \phi, \alpha)$). Finally, Eq. 1.11 shows that tracking error has direct relation to the external torque and thus can potentially be used to reconstruct external collision forces.

In our work, we cannot directly evaluate Eq. 1.12, because we do not know precisely R, K_t, K_e, J_m, μ, and the value of $\dot{\epsilon}_{\text{exp}}$ cannot be computed before computing ϵ_{exp}. Additionally, we do not have the access to the motor side encoder to measure θ, ω and $\dot{\omega}$, which would allow us to handle the backlash appropriately and compute τ_f and τ_B terms. Thus, instead of evaluating Eq. 1.12 directly, we choose to identify a non-linear model from a sample robot motion recording dataset to approximate Eq. 1.12 using a set of available desired motion related variables, namely $\dot{q}_d, \ddot{q}_d, \tau_{ld}$ and ϵ.

For the identification, we selected a binary-tree prediction model Breiman et al. (1984). The non-smooth activation function of a binary-tree non-linear model estimator is suitable in our particular case, because it is capable of modeling sudden abrupt changes in the tracking error signal, unlike other non-linear model estimators with smooth activation function, such ass sigmoid or wavelet networks Zhang and Benveniste (1992), which we also experimented with.

The final form of the $\tilde{\epsilon}_{\text{exp}}$ expression is Eq. 1.13.

$$\tilde{\epsilon}_{\text{exp}}(t) = \texttt{binary\_tree}(\dot{q}_d, \ddot{q}_d, \tau_{ld}) \tag{1.13}$$

With the identified model capable to predict the expected joint position tracking error for a given desired motion feature vector, the part of the joint position tracking error that is related only to the collision event can be computed by subtracting expected error value from the measured error to compute our contact observer signal r (Eq. 1.14).

$$r = \epsilon - \epsilon_{\text{exp}} \tag{1.14}$$

In the following section, we describe how the model identification process is performed and discuss the resulting performance of expected tracking error prediction.

1.4 Expected position tracking error model identification

Hereafter, we describe the process of model identification for the expected joint position tracking error prediction (Eq. 1.13).

In this example, we focus on the Pepper left shoulder roll joint (denoted for short as *LSRoll*). In order to identify a model for expected tracking error prediction for *LSRoll*, we record the robot motion data free of external collisions while controlling Pepper via the QP controller with a single posture task in the objective function. The controller is set up to generate a sequence of various motions including moving between the joint limits with randomly selected small or big offsets from the joint limits and moving the joint to various random setpoints. In the middle of each data acquisition process the configuration of a previous joint (*LSPitch*) and a the next joint (*LERoll*) in the chain change to new randomly selected setpoints and the main joint, *LSRoll*, repeats the motion sequence again. With such setup our intention is to identify the model which is "aware" of the change in the configuration of other nearby joints. Note that this is possible due to the desired load torque, τ_{ld}, being part of the feature vector to the non-linear model estimator, as it incorporates the robot model knowledge. The joint stiffness value is set to 100% for all joints in our experiments.

For the sample joint motion dataset, 7 different sequences of *LSRoll* joint motion are recorded with various QP posture task stiffness values of the QP posture task. Posture task stiffness varies from 2 to 5 in estimation dataset recordings. This data is used to identify the parameters of a binary tree and to evaluate accuracy of tracking error reconstruction on estimation dataset. Average resulting accuracy in terms of Normalized Root-Mean-Square Error (NRMSE) of the joint position tracking error prediction over 7 data sequence recordings used in the estimation process is 77.29%. We consider it to be satisfactory performance on the estimation dataset and proceed to evaluate performance of this model on "unseen" test data.

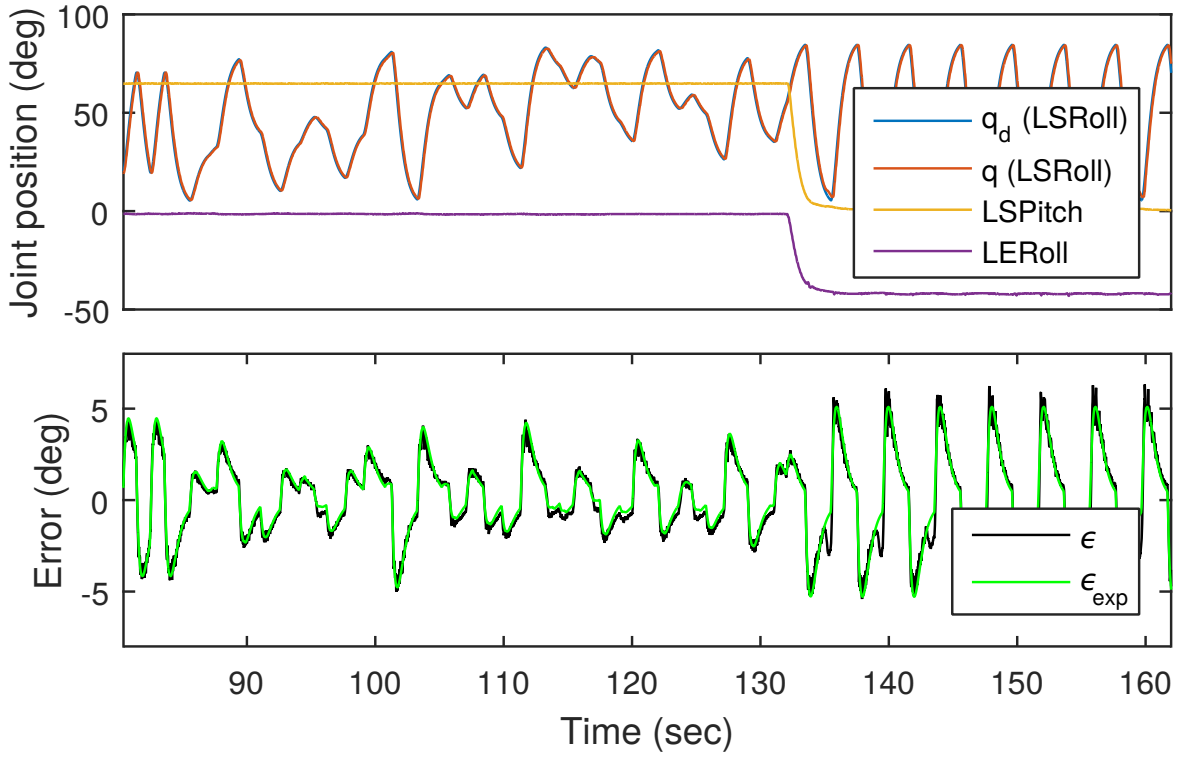


FIGURE 1.3 – *Evaluation of the identified binary tree model for the expected joint position tracking error prediction on test data: joint trajectories (top); expected tracking error prediction for* LSRoll *joint (bottom). Overall tracking error prediction accuracy:* 81.11%.

To collect the test data, we use the same QP controller but with new randomly selected offset and setpoint parameters and increase significantly the stiffness of the QP posture task, setting it to 9, in order to trigger motion with higher speed (and thus larger tracking error). The accuracy of tracking error prediction on the test data set is 81.11%. The plot of a segment of the joints trajectories from this experiment (test dataset) and error reconstruction plot are shown in Fig. 1.3.

The results presented in Fig. 1.3 demonstrate that the identified prediction model generalizes well to unseen data and accurately predicts the value of the position tracking error. We also see that the model is robust to changes of the QP task stiffness and changes of the configuration of other joints in the chain.

For achieving the best possible prediction performance, model identification has to be performed for every motor separately and possibly repeatedly, as the motors and the gear system wear out with time. However, we have observed that the left body side joints' models can perform equally well for the right side, as the same motor/load types are used and they wear out approximately equally. Even more so, we observed that the same motor types can "share" a model. Pepper has 17 joints and uses 5 different types of motors in total. Thus, in general, it is sufficient to perform the model identification for only 5 different Pepper motor types.

In the next section, we demonstrate how the identified model performs when applied to the data sequence with external collisions.

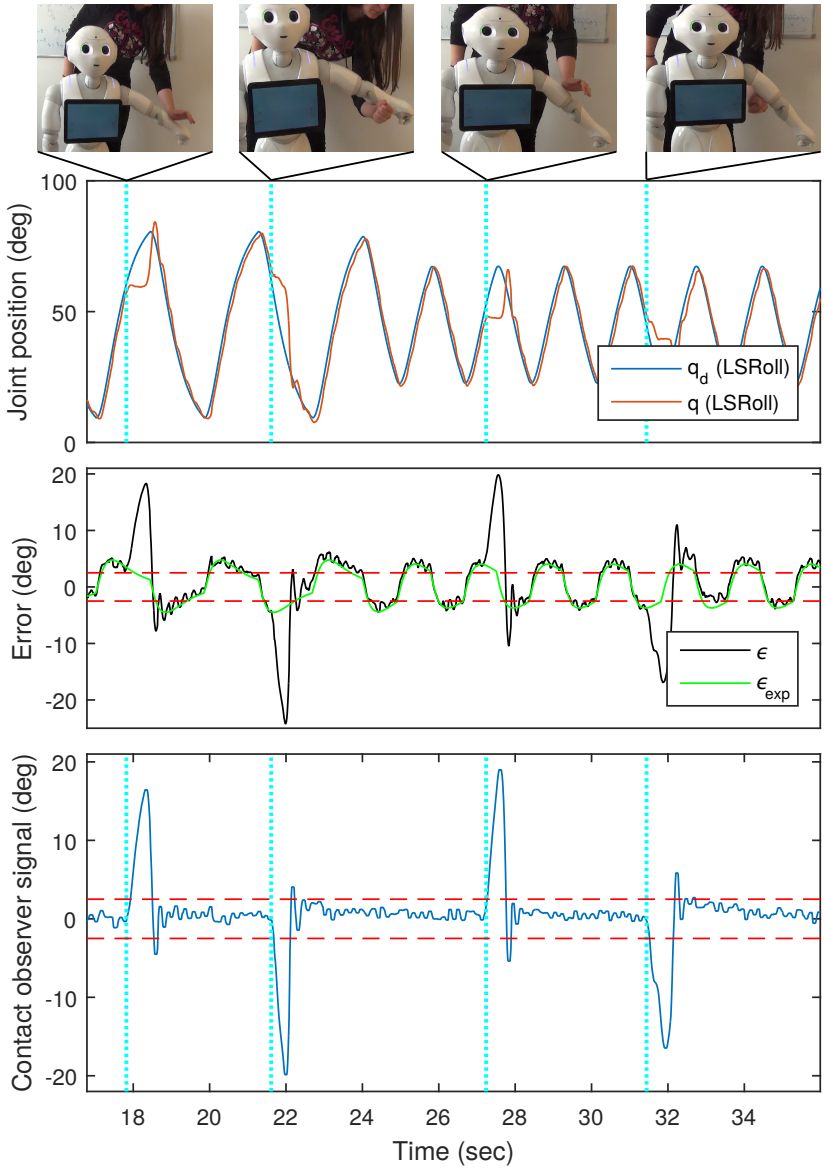


FIGURE 1.4 – LSRoll *expected tracking error model evaluation: joint trajectories (top); expected tracking error prediction (middle); contact observer signal r (bottom). Dashed blue lines show the start of the contact events. Dashed red lines indicate the threshold δ for contact detection.*

1.5 Experimental Results

Now, we showcase how our proposed contact observer signal r (Eq. 1.14) is used for the contact detection and identification of contact direction and intensity.

In the experiments reported in this section, a median filter over 11 latest samples of r is applied to reduce noise and eliminate occasional spikes in the signal. The time-step between two consecutive samples is 12ms. We set a fixed threshold $\delta = 2.5°$ for the contact detection. Whenever $|r| > \delta$ we consider that a collision/contact occurred. The threshold δ can also be interpreted as an external torque sensitivity threshold, meaning that any external force, which results in such a τ_{ext} at the joint that causes the displacement beyond δ, can be detected by the proposed method. The sign and magnitude of r reveal the direction and collision intensity information respectively.

We use the same QP controller, described in the previous section, to generate a

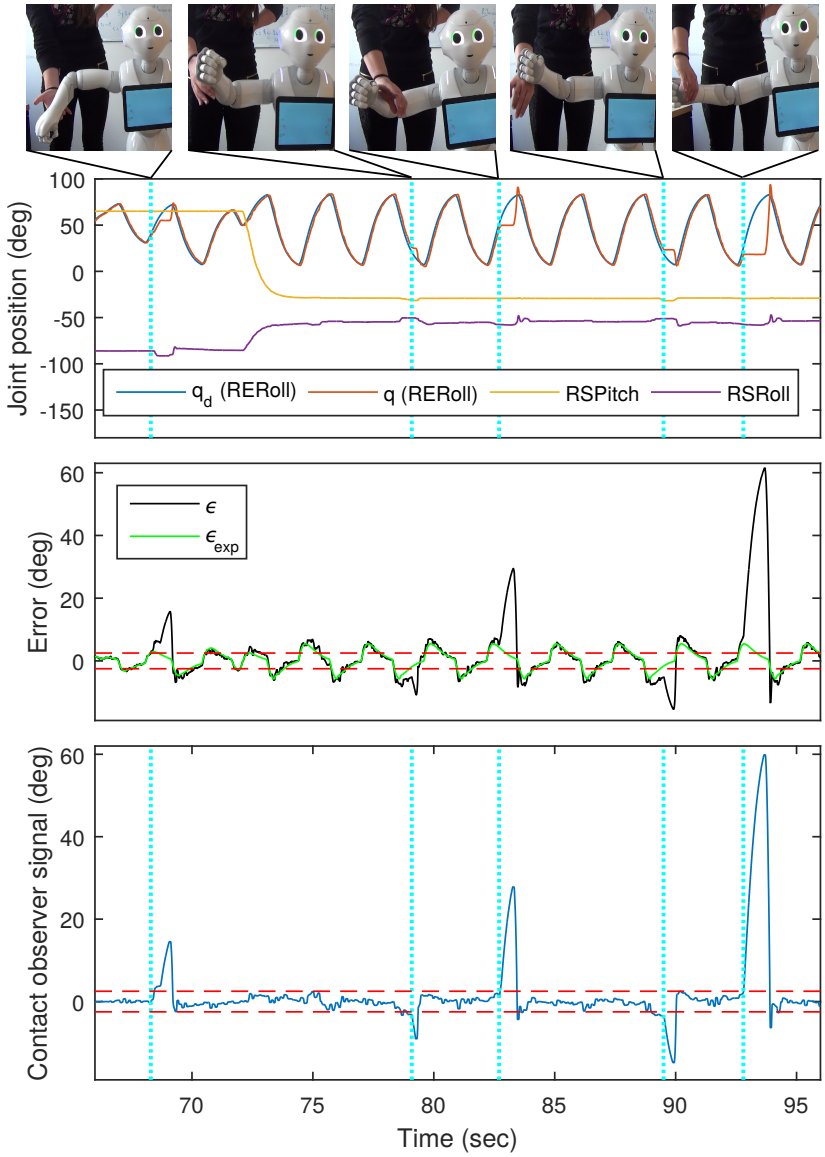


FIGURE 1.5 – *Right elbow roll joint model evaluation: joint trajectories (top); expected tracking error prediction (middle); contact observer signal r (bottom). Dashed blue lines show the start of the contacts. Dashed red lines indicate the threshold for contact detection.*

sequence of the left arm joints motion with new randomly selected parameters for offsets and setpoints. The QP posture task stiffness is set to 8. During the execution of the motion several external collisions are triggered by touching the robot's left arm. The plot in Fig. 1.4 shows a ∼20 second segment of the results from this experiment. We repeat a similar experiment for the right arm elbow roll joint *REROll* with posture task stiffness set to 12. A segment of *REROll* experiment results is presented in Fig. 1.5. The extended presentation of these results can be seen in the video accompanying this work [1].

The results indicate that our proposed method is capable of making precise prediction of expected tracking error and, thus, produce a contact observer signal r which

1. Video titled "Contact Observer for Humanoid Robot Pepper based on Tracking Joint Position Discrepancies" is hosted online at the IDH LIRMM YouTube channel: https://youtu.be/nY9zMG0EsnM

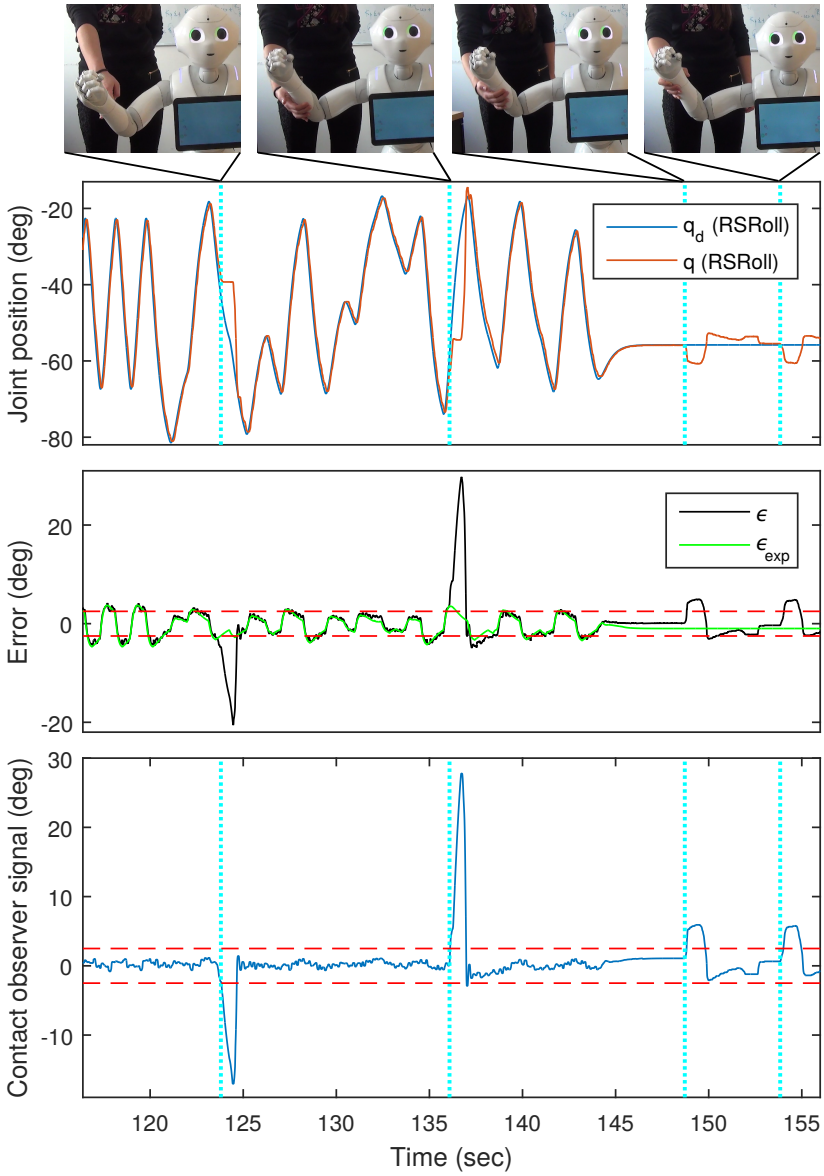


FIGURE 1.6 – *The model identified using* LSRoll *joint sample data applied to the test data of* RSRoll *joint. The models of the left body side generalize well for the right body side joints.*

remains below threshold δ when there is no collision, i.e. when the joint moves freely. When collision occurs (indicated in the plots by dashed vertical blue lines), r exceeds the fixed threshold. Moreover, the direction and the intensity information about the collision event is correctly represented via the sign and magnitude of r.

The middle plot of the Fig. 1.6 demonstrates the segment of experiment where the binary tree model trained using recordings of *LSRoll* joint's data is applied to predict expected tracking error for the right shoulder roll joint (*RSRoll*). This result confirms that models identified on left side body joints generalize well for the right side body joints, eliminating the necessity to train separate models for every robot joint.

The Tab. 1.1 reports the total amount of false positive #FP ($r > \delta$ without contact), false negative #FN ($r \leq \delta$ with contact) and true positive #TP ($r > \delta$ with contact) contact detections across three experiments.

Note that usually after a strong external collision, when the joint quickly returns

Experiment name	**#FP**	**#FN**	**#TP**
LSRoll experiment	2	2	18
REROll experiment	0	3	19
RSRoll (with LSRoll model)	0	3	18
Total:	**2**	**8**	**55**

TABLE 1.1 – *Number of false positive, false negative and true positive contact detections across three experiments*

to its desired position, r exceeds the threshold for a brief amount of time due to the impact. Such cases of exceeding the threshold are not considered as false positives in Tab. 1.1. We also note that most of the false negative cases reported in Tab. 1.1 occurred due to the large flexibility of the hip-roll joint, which enables the upper body of Pepper to move when an arm is pushed/pulled. This reduces the amount of the position tracking error in the arm joints and can lead to a false negative contact detection. Of course, this issue would be resolved when the whole body (including the floating wheeled part) is considered in the contact detection method.

1.6 Conclusion

In this chapter, we have derived a simplified expression for computing the expected value of the position tracking error of a DC motor controlled with PD scheme given the knowledge of desired joint trajectory and desired load torque. This expression revealed that, under some conditions of compliance (low PD gains with feedforward terms and/or reversibility) the expected tracking error prediction does not require knowledge of neither the motor current nor the motor torque.

We described the process of non-linear model identification and presented the results of expected tracking error prediction, which show good accuracy and generalization properties of the identified models. We demonstrated how prediction of expected tracking error can be used for computing the contact observer signal, which incorporates intensity and direction information of the collision event.

In its initial form, the proposed approach still exhibited some false positive contacts detections and was not integrated as a feedback signal in real-time robot control. In the following development stages, we focus on reducing the amount of false positive detections and making the proposed system more robust. The goal of the work, presented in the next chapter, is to enable the proposed contact observer method to be integrated as a real-time feedback signal for a whole-body control of the Pepper robot in pHRI applications and regulation of the robot active compliance to touch.

CHAPTER 2

COMPLIANT ROBOT MOTION CONTROL

In this chapter, we present developments of a real-time compliant motion control for a personal humanoid robot Pepper. Our approach allows to interpret and react to human guidance through touch using only joint encoder measurements to monitor contact direction and intensity for both static and moving links. This novel method is developed with consideration of the minimal sensor requirement of the hardware platform to meet high affordability criteria. We demonstrate performances in the pHRI experiments with the humanoid robot Pepper.

2.1 Introduction

In various fields, such as manufacturing or health care, an efficient automation may require work tasks to be performed by a robot in close-contact with a human. To achieve safe and efficient collaboration, it is important that agents understand each other's intentions and react to them appropriately. In this work, we focus on the interpretation and correct online reaction of a humanoid robot Pepper to human touch.

The human-robot collaboration must be safe, efficient and optimal in terms of cost. Thus, a minimal sensor set-up is preferable in order to reduce robot manufacturing and maintenance costs. As a result, we focus on more cost-efficient, but in practice a more challenging way of contact sensing by using a minimum setup of proprioceptive sensors only. That is to say, we use only robot's joint encoder measurements to control robot reaction to external contact events.

Recent developments in the area of proprioceptive sensor based contact monitoring have shown promising results for serial manipulators Haddadin et al. (2017), humanoids Flacco et al. (2016) and also a special case of affordable personal robots, such as Pepper, used in this work, with high motor-joint backlash and friction Flacco and Kheddar (2017). In our work, described in the previous chapter, we proposed a contact observer method that uses data-driven machine learning techniques to detect a colliding link and monitor contact direction and intensity. This method was developed to fit well into minimal sensor set-up, as only joint encoder measurements are exploited for contact detection for both moving and static links.

In this chapter, we describe the modifications made to our initial proposed contact observer method in order to increase its accuracy and generalization properties in the

case of multi-joint motions. Then, we demonstrate its use as a real-time feedback signal for regulating compliant motions of the arm of a humanoid robot Pepper.

2.2 Background

In order to control active compliant motion of a robot, it is necessary to sense or estimate the external forces applied on the robot's body. A sophisticated yet expensive solution would require to cover the entire robot structure with a *haptic skin*. Such an approach, however, is not suitable for our application due to the cost of adaptation of additional sensors (even with cheap components) and their maintenance; the introduction of additional weight to the robot structure; additional power consumption etc. Kheddar and Billard (2011); Mittendorfer et al. (2015); Yogeswaran et al. (2015); Cheng et al. (2019).

The aforementioned cost and maintenance related constraints are not specific to our use case, but are important in many other areas of robot-environment or robot-human physical interaction as well. As a result, researches have directed their efforts towards achieving contact sensing by using must-be proprioceptive sensors only (i.e. those usually available on a robot platform), such as joint encoders, motors' electric current or torque sensors, etc. A thorough overview of such contact sensing methods is presented in Haddadin et al. (2017). Methods, which were initially developed for serial manipulators, have been further developed to be applied on humanoid robots, e.g. Flacco et al. (2016); Mattioli and Vendittelli (2017); Vorndamme et al. (2017).

In Mattioli and Vendittelli (2017) a proprioceptive sensor based external force reconstruction was used as a feedback in the control loop to regulate robot interaction forces with the environment –under assumption of static conditions for external force reconstruction. The goal of our work, presented in this chapter, is to develop a method that works equally well for controlling a compliant motion of both static and moving robot links. Compliant reaction to human touch using proprioceptive sensors has been demonstrated in Flacco and Kheddar (2017) in the experiment with a Romeo humanoid robot arm. However, this method could not be adapted for the current version of the Pepper platform. This is due to the inability to measure motor-link backlash and to measure or accurately estimate motor torque, which in turn prohibits to estimate external torque by means of the momentum observer method De Luca and Mattone (2005). The latter method was used in robot control to achieve safe reaction to external collisions in pHRI context, see examples in De Luca and Mattone (2005); Haddadin et al. (2008); De Luca and Ferrajoli (2008); Parusel et al. (2011).

To address a challenging task of contact sensing without using cover haptics, joint torque or even motors' electric current sensors; we proposed a position tracking error based contact observer, described in the previous chapter. This method is based on monitoring the discrepancy between the measured position tracking error and the *expected* one for a given desired free motion trajectory. First, a machine learning model is trained on a sample dataset with appropriate set of desired trajectory related features to predict the *expected* position tracking error value for collision-free joint motion. Then, for every iteration of the control loop, the trained model is used to predict the expected position tracking error, assuming no contact has occurred. The contact is detected whenever there is significant discrepancy between measured and predicted (and therefore

expected) error values.

In our initial study, this proposed method has shown promising results for contact detection and for monitoring both the contact intensity and direction on a sample collected robot motion dataset, which we analyzed offline. Now, we exploit our proposed contact observer method in real-time control for regulating compliant motion of the Pepper robot for physical interaction with a human. In the work presented in this chapter, we demonstrate that an enhanced version of our contact observer can be exploited for the online interpretation of the human touch and regulation of compliant robot motion.

The use of disturbance observers for the regulation of robot compliant motion has been explored in several works. In Hyon et al. (2007) whole-body compliant motion in human-humanoid interaction settings was proposed for maintaining balance of the torque-controlled bipedal humanoid platform in the presence of unknown external forces. A momentum based disturbance observer was applied to the floating-base model of a humanoid robot in Ott et al. (2013) to detect external forces and integrate them into a whole-body control for kinesthetic teaching and simultaneous compliant balancing. These methods, however, require a measurement of the joint torques, which may not be available and may also be challenging to estimate in case of highly geared and flexible joints encountered on many humanoid platforms, including also the Pepper robot which we use in this work.

In Kim et al. (2018) disturbance observer based compliant humanoid motion control scheme, that can compensate for high joint elasticity, was proposed including the modeling of flexible joints that was performed using the measurement of motor-joint backlash angle. On the current position-controlled Pepper platform, access to the motor side encoder measurements is not available through the robot's centralized memory, prohibiting to handle motor-joint backlash and joint flexibility. There are also no motor torque sensors on the platform. Furthermore, estimation of the motor torque from electric current measurements is not feasible, due to the measurements being passed to the centralized memory in absolute and down-sampled form. Nevertheless, in our work we aim to overcome those constraints by exploiting the machine learning data-driven approach for the disturbance observer.

Following in this chapter, we describe modifications of the contact observer w.r.t our initial study presented in Chapter 1. First, we present the details about the robustified expected position tracking error prediction model training process (Sec. 2.3). Then, we describe the compliance control scheme and controller implementation details (Sec. 2.4). Finally, we demonstrate results of the compliant motion regulation using the proposed method achieved with Pepper in pHRI experiments (Sec. 2.5).

2.3 Contact Observer for Multi-Joint Motions

The use of r signal (Eq. 1.14) for contact monitoring has been demonstrated in the previous chapter in several experiments with Pepper left and right arm joints, with one joint moving at a time, i.e. 1 degree of freedom (DoF) case. Occasionally false positive (FP) and false negative (FN) contact detections occurred during the experiments. Furthermore, when the system was tested on more complex motions, e.g. with all arm joints moving simultaneously with arbitrary speed to random set-points, the trained

expected tracking error prediction models turned out to be not general enough to adapt to such motions. The reason for this is mainly the fact that sudden motions of one joint can have significant influence on the tracking error value of some other nearby joints, especially in the presence of a significant motor-link backlash.

Such false detection cases have to be eliminated or minimized in order to use r as a feedback signal in the active compliance control. We identified the main reasons for the false detection cases. First, due to the insufficient prediction accuracy of the binary tree prediction model for some new data (not used for the model training), r value can exceed threshold $\pm\delta$ even though external collision is not present, meaning that prediction model fails to generalize to "unseen" data and predict value of ϵ_{exp} correctly. Secondly, in some circumstances (e.g. singularity or near joint limits configurations), external contact force does not cause r to exceed $\pm\delta$, thus the contact remains unnoticed. Similarly, in case of light contacts, r does not exceed $\pm\delta$, which can be addressed by lowering the value of δ (which was set to 2.5° in previous work). However, to do that the overall prediction accuracy has to be improved, for both seen and unseen data, which may not be achievable due to the accuracy-generalization trade-off. And, finally, as mentioned previously, the prediction of expected tracking error of a joint depends on the motion of other joints. Therefore, *relevant* variables related to other joints' desired trajectories must be identified and included in the prediction feature vector.

The aim is to ensure that the trained models generalize well to a wide range of complex motions, while encountering fewest possible false positive contact detections. At the same time, it is equally important that prediction is computed within the control time constraint for all joints. Therefore, the use of complex models that take very long time to compute the prediction is prohibited, which leads to potentially sacrificing the maximum achievable prediction accuracy. Thus, the trade-off between model complexity and feature vector size and model accuracy must be properly handled.

In several application areas of machine learning, so called, ensemble based techniques have often shown to perform better in practice compared to single classification or regression models Dietterich (2000). In particular, *boosting* is a machine learning technique for ensemble model training that is suitable for our goal to battle false positive contact detection cases Drucker and Cortes (1996).

According to the boosting methodology, a cascade of several models is trained successively; every new model $k = 2, \ldots, N$ (N being the total number of models in the ensemble) is trained taking into consideration the performance of the previously trained models $1, \ldots, k-1$. At every new iteration k, previously known inaccurate predictions of ϵ_{exp} are given more weight before training of k^{th} model takes place. As a result, every new model in the ensemble is trained to make a better prediction, especially in those cases where previous models performed poorly. This results in minimization of the maximum prediction error and consequently fewer amount of false positive contact detection cases. We chose a *cascade of boosted decision trees* as the prediction model to be used for training a more robust model for expected position error prediction. We use `XGBoost` library implementation of this technique Chen and Guestrin (2016).

To achieve the best possible prediction model accuracy and generalization, while minimizing the required computation time, we make several important choices about model hyperparameters. The process of hyperparameter tuning is described in Sections 2.3.1–2.3.3 for *LShoulderRoll* joint, also referred to as joint nr. 6 or *LSRoll* for short (with corresponding variables denoted as $\epsilon^6, q^6, \ddot{q}^6$ etc.). Analysis for other joints is per-

formed in the same manner.

2.3.1 Feature Vector Components

On a sample recording of 45 minutes of collision-free data (with time step of 12 ms, i.e 225000 samples in total) we analyze the importance of different variables for ϵ^6_{exp} prediction to select the most relevant variables for the feature vector. We train a boosted ensemble of decision trees with a feature vector that includes the desired motion related variables for all the main left arm joints. The results of relative variable importance computed from the trained model are summarized in Fig. 2.1.

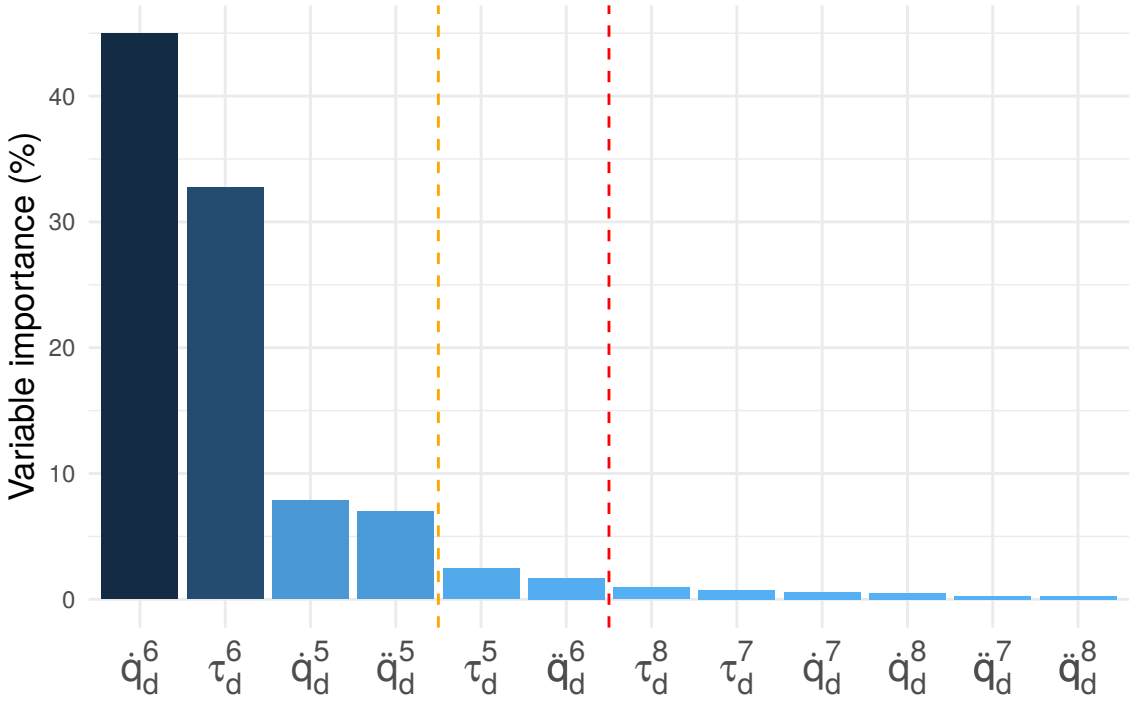


FIGURE 2.1 – *Variable importance for ϵ^6_{exp} prediction.*

As can be seen from the plot, the most important variables for ϵ^6_{exp} prediction are desired speed and torque of the target joint, which confirms our previous findings described in the previous chapter. However, desired speed, acceleration and torque of the nearby joint, *LShoulderPitch* (joint nr. 5), also have significant weight for prediction of ϵ^6_{exp}. These features even outweigh the importance of the $\ddot{q}^6_d$ feature. The variables after the orange dashed line on the plot are considered of medium-importance and variables after the red dashed line are considered as non-important.

To make a more informed decision on the final size of the feature vector, we perform k-fold cross-validation with $k = 4$ (75% samples for training and 25% for validation) and analyze how the root-mean-square error (RMSE) and maximum absolute prediction error (MAX) on validation sets are affected as we incrementally include more of most important variables into prediction feature vector. The mean RMSE evaluated over 4 cross-validation folds only improves significantly if 3 to 4 most important variables are used (see Fig. 2.2). The mean MAX improves significantly until 6 most important variables are used and stops improving and even deteriorates after this point.

Based on these observations, we chose to include 6 most important variables in the final feature vector.

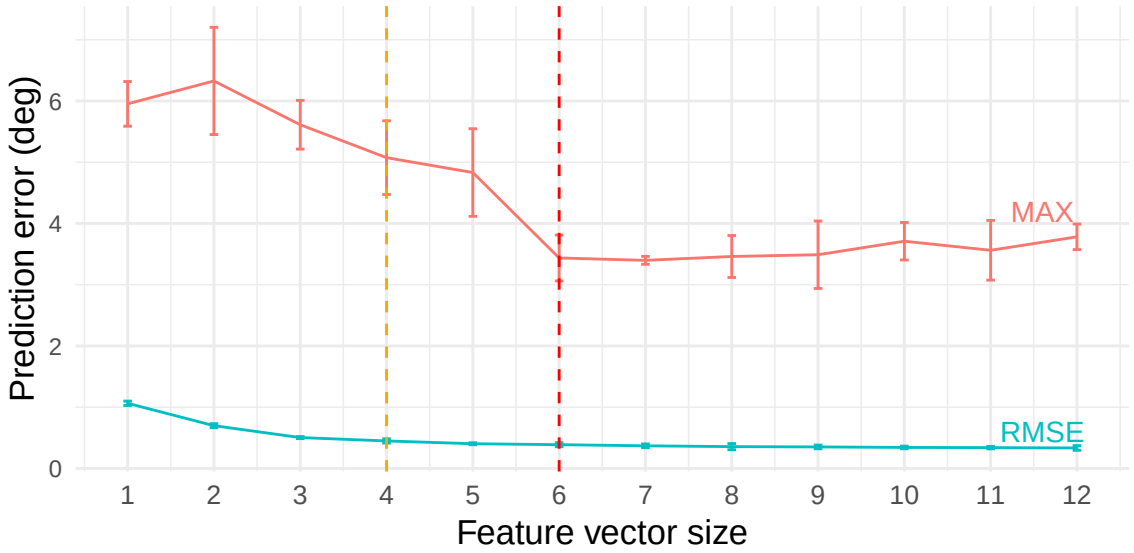


FIGURE 2.2 – *Prediction error with various feature vector sizes.*

2.3.2 Ensemble Size and Learning rate

After selecting the components of the feature vector, we identify an optimal value for the ensemble size N, and a learning rate η of the model. Since we intend to use the trained model in the real-time application, we must select the model with the smallest possible ensemble size. Selecting a higher learning rate for the training allows to achieve good performance with fewer number of sub-models in the ensemble. However, higher learning rate may result in lower model accuracy, as in this case the training algorithm is only allowed to make "big steps" towards an optimal solution. We analyze the mean cross-validation RMSE for various sizes of ensemble and for different values of the learning rate. Fig. 2.3 illustrates the performance of models of various sizes trained with three different learning rates.

We see from this plot that lower η values do not result in significant improvement of performance in terms of the mean RMSE over k cross-validation folds as the size of the ensemble increases. The difference in performance between the best and worst performing models is $< 0.02°$. Therefore, we choose to use $\eta = 0.3$ with model size $N = 50$ as our optimal choice, since it provides a good performance while keeping ensemble size N relatively low.

2.3.3 Individual Decision Tree Maximum Depth

Now that other model hyperparameters are selected, we identify an optimal size of the individual decision trees. We train several models with $N = 50$ and $\eta = 0.3$ and analyze mean RMSE and MAX computed for training and validation folds. We also compute the rate of false positive detections as a total amount of absolute validation prediction errors exceeding the threshold $\pm\delta$ divided by the total number of the validation samples. The results are shown in Fig. 2.4.

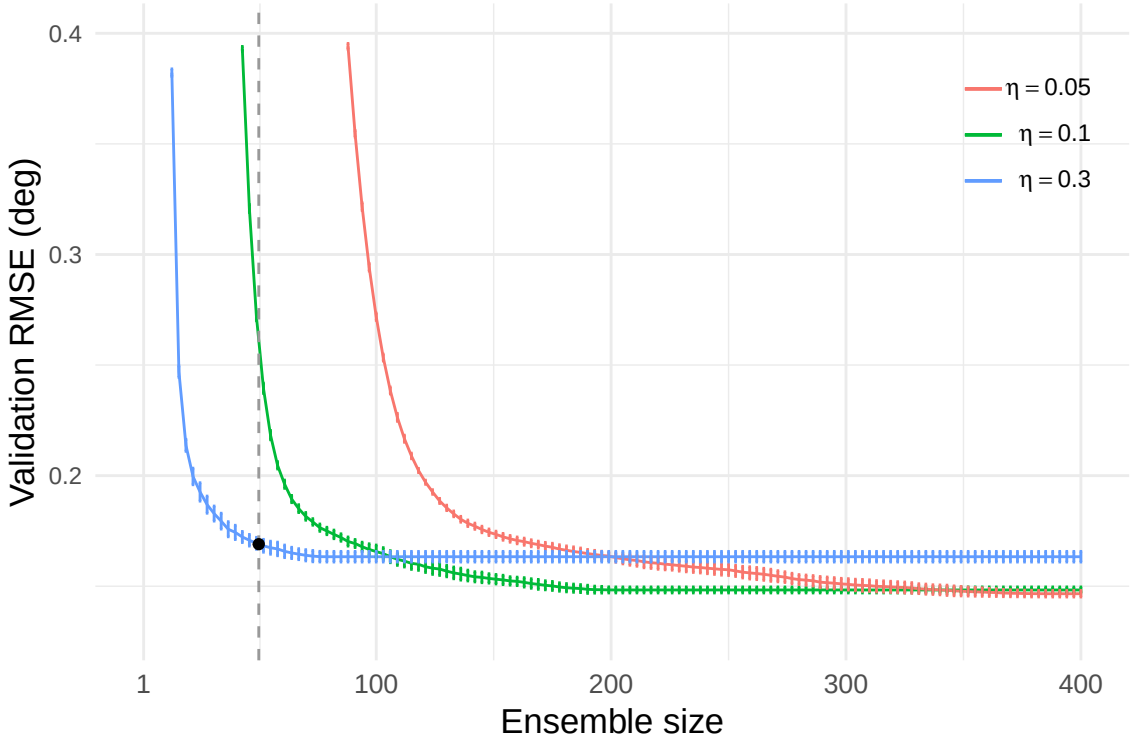


FIGURE 2.3 – *Mean validation RMSE for various ensemble sizes and learning rates.*

The results show that with maximum tree depth of 7 the false positive rate decreases to $< 0.1\%$. The values of RMSE for both the validation and the training data are acceptably low, always below soft and hard thresholds indicated as orange and red dashed lines on the plot respectively. The maximum absolute prediction error decreases for both seen and unseen data until maximum tree depth reaches 8. After this point, the trained model becomes too complex and overfits the training data (training error decreases, while validation error remains the same or increases). Note that adapting regularization strategies did not help to improve the performance of the model on validation data. Based on these observations, we chose the final value for the maximum tree depth to be set to 8.

With this final decision on hyperparameter values, we are able to achieve the performance of the contact observer that is acceptable for the use of r as a feedback in a real-time control for active compliance. The next section presents the compliant motion control implementation details.

2.4 Compliant Motion Control

In this section, we describe the QP controller design that allows to control compliant robot motion using the contact observer signal as a feedback signal on collisions. The overview of the entire process is presented by pseudocode in Algorithm 1. For simplicity, consider a QP controller with one posture task in the objective function, and typical kinematics constraints, such as self-collision avoidance and joint limits. The posture

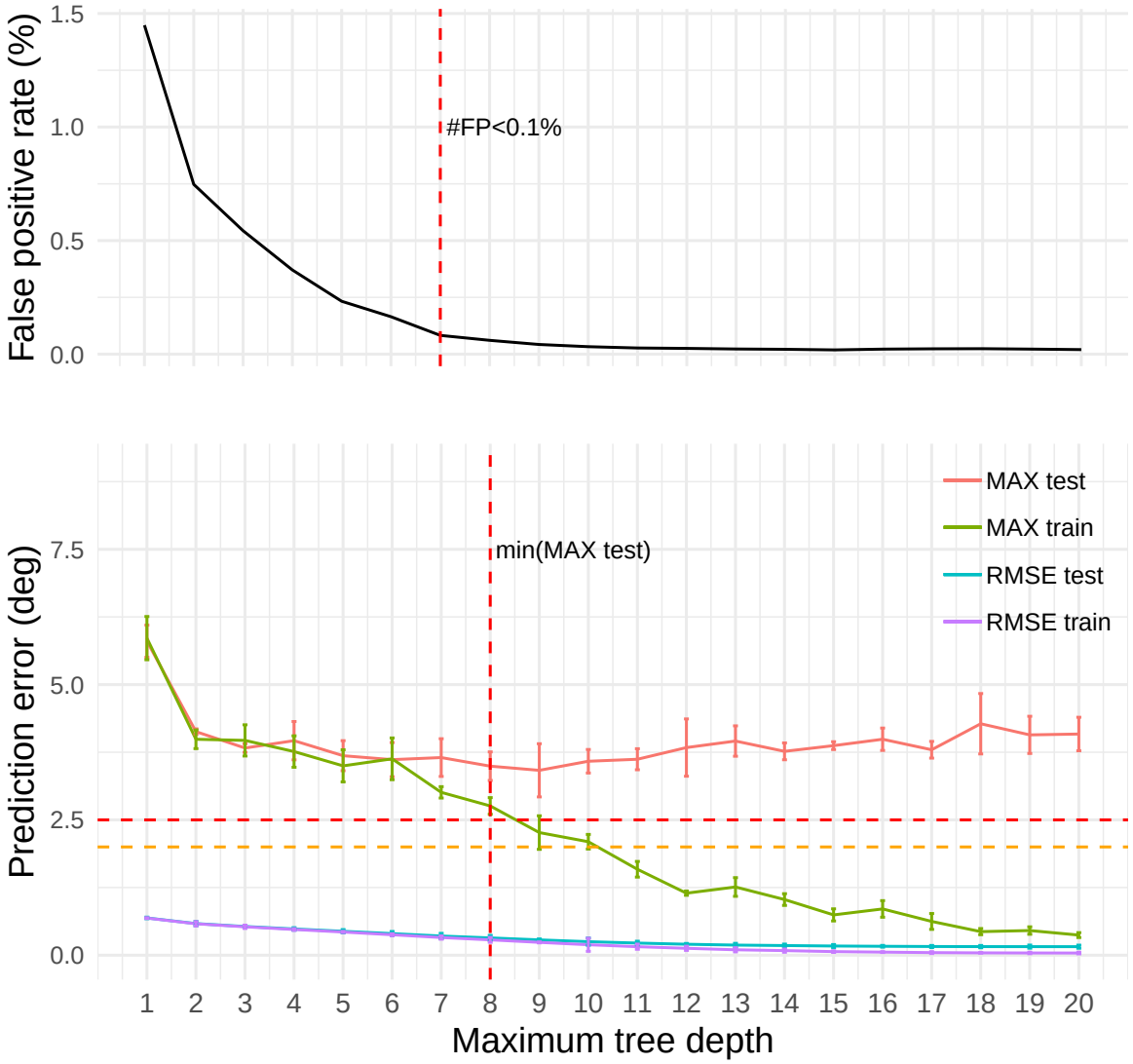


FIGURE 2.4 – *Optimal max individual tree depth selection from false positive rate (top) analysis and training/test error (bottom).*

task specifies a target posture, q_{target} [1]. The output of the QP controller, $\ddot{q}_d$, generates a whole-body motion that brings the robot joints closer to q_{target}, while satisfying the kinematics and self-collision avoidance constraints. The rate of convergence of the joints to q_{target} is regulated via the QP posture task stiffness gain.

Before the QP controller starts to regulate robot motions, the initial robot sensor values, q_{init}, are read. For the first iteration of the controller execution, timestamp $t = 0$, we assume that no collision with the environment occurred, i.e. $r(0) = \mathbf{0}$. At this point, the posture task target position is equal to the values of the initial sensor reading ($q_{\text{target}} = q_{\text{init}}$). The QP controller computes $\ddot{q}_d$, that is integrated twice to obtain desired joint setpoint, q_d, that is sent to the low-level PD robot servo to progressively bring the robot joints closer to q_{target}. In the first iteration, since $q_{\text{target}} = q_{\text{init}}$ the robot does not move, i.e. $\ddot{q}_d \approx \mathbf{0}$.

Using the known robot model and QP output, the desired torque τ_d is computed.

1. Note that q_{target} is the *final* target robot posture, whereas q_d is intermediate desired joints position command which iteratively and incrementally brings a robot from its initial state to q_{target}.

Now the ϵ_{exp} can be computed using the trained prediction models described in the previous section. At the same time, the measured position tracking error is computed as $\epsilon = q_d - q_{\text{init}}$. Finally, the contact observer signal, $r = \epsilon - \epsilon_{\text{exp}}$, is computed and passed to the next QP iteration along with the new sensor readings.

Starting from the second iteration, if $|r| < \delta$ the posture task remains unchanged. Otherwise, if a discrepancy between ϵ and ϵ_{exp} has occurred (i.e. $|r| \geq \delta$), before the QP controller is inferred, the robot QP posture task has to be readjusted to comply in the direction of estimated disturbance. The compliant motion of every joint is regulated via Eq. 2.1

$$q_{\text{target}}(t) = q_d(t - \Delta t) - K_c r + K_v \dot{q}_d(t - \Delta t) \tag{2.1}$$

where K_c and K_v are compliance and velocity gains respectively, which are tuned for every joint. The posture task is updated in the QP objective function with new q_{target}, and the entire process repeats until the controller is stopped.

The compliance gain, K_c, regulates how much influence contact observer signal r has on updated target posture. Lower values of K_c allow the target posture to be updated only slightly, which means that QP will not compute high $\ddot{q}_d$ and the robot will comply stiffly (little amount with lower speed). On the contrary, a high K_c causes the q_{target} of the posture task to change significantly resulting in high compliance and faster motion, as the $\ddot{q}_d$ is proportional to the QP task error. The velocity gain, K_v, regulates the proportional contribution of joint desired speed to facilitate compliance of moving joints (those that started to comply in the previous iterations).

Note that this compliant motion control strategy can be augmented with any other QP tasks (e.g. set point, visual servoing, force tasks, etc.), by adding them to the QP objective function via higher level planning algorithm either along with compliant posture task or within periods when $|r| < \delta$. In such a case, a proper tuning of QP tasks' weight and stiffness values can influence greatly the overall performance of the complex QP controller.

2.5 Experimental Results

We use the proposed and robustified contact observer methodology, integrated as a real-time feedback signal for compliant motion control via the QP controller, in two pHRI experimental scenarios. The threshold for contact detection, δ, is set to 2.5° for all joints in all experiments.

In the first scenario, a robot is making a motion that initiates the process of assisting a human to stand up from a sitting position. The robot moves its arm towards the human back and stops moving further if it detects a contact (i.e. K_c and K_v gains of Eq. 2.1 are set to zero once the contact is detected). If no contact is detected the robot continues to move its arm until joint limits. The motion is repeated several times with varying values of the target *LShoulderPitch* (*LSPitch* for short) joint position, allowing to take contact lower or higher on the back of a human. The values of the desired joint positions and the contact observer of the *LSRoll* are shown in Fig. 2.5. When the contact observer signal exceeds the threshold, the commanded position for the *LSRoll* joint is altered such that the robot stops instead of continuing to move forward. The snapshots of this experiment are shown in Fig. 2.6.

In the second experimental scenario, the robot moves 4 joints of the left arm ran-

Algorithm 1: The pseudocode of algorithm for the QP controller regulating joint compliance via r signal

$q \leftarrow$ ROBOT.GET SENSORS()
$r \leftarrow \mathbf{0}$
CNTRL $\leftarrow$ INITIALIZE CONTROLLER()
POSTURE TASK $\leftarrow$ INITIALIZE POSTURE(q)
CNTRL $\leftarrow$ INSERT TASK(POSTURE TASK)
CNTRL.RUN()
while CNTRL.RUNNING() **do**
  if $|r| \geq \delta$ **then**
    $q_{\text{target}}(t) \leftarrow$ COMPLY JOINTS($r, q_d(t-1), \dot{q}_d(t-1)$)
    CNTRL.POSTURE TASK $\leftarrow$ UPDATE TASK($q_{\text{target}}(t)$)
  end if
  $\ddot{q}_d \leftarrow$ SOLVE QP(POSTURE TASK)
  $\dot{q}_d \leftarrow$ INTEGRATE($\ddot{q}_d$) $q_d \leftarrow$ INTEGRATE($\dot{q}_d$)
  ROBOT.SETJOINTANGLES(q_d)
  $\tau_d \leftarrow$ INV. DYNAMICS($\ddot{q}_d, \dot{q}_d, q_d$, ROBOT MODEL)
  $\epsilon_{\text{exp}} \leftarrow$ PREDICT($\ddot{q}_d, \dot{q}_d, q_d, \tau_d$)
  $\epsilon = q_d - q$
  $r = \epsilon - \epsilon_{\text{exp}}$
  $q \leftarrow$ ROBOT.GET SENSORS()
end while

domly. When the contact is detected, Eq. 2.1 with positive K_c and K_v gains, is used to guide the robot motions through human touch and comply the joints of the arm. The values for K_c and K_v gains are set manually for every joint; exact values used in the experiments are presented in Tab. 2.1.

	LShoulderPitch	*LShoulderRoll*	*LElbowYaw*	*LElbowRoll*
K_c	17	5	5	17
K_v	0.08	0.02	0.02	0.02

TABLE 2.1 – *Gain values for main left arm joints.*

The subsequent snapshots of a sample arm motion compliant to the human touch are shown on Fig. 2.7.

The plot segment of desired position of the shoulder roll joint and corresponding contact observer signals are shown in Fig. 2.8. Due to the random set-points being commanded to the robot during the free motion –to show that the method can be applied for movement of arbitrary position and speed, the compliance of the joints may not be obvious from the plot. However, looking closely one sees that at the moments of contacts (indicated with dashed blue lines) the desired joint position is being altered taking into account the direction in which the contact is applied.

Extended presentation of the results of both experiments is included in the video accompanying this work [2].

2. Video titled "Compliant Robot Motion Regulated via Proprioceptive Sensor Based Contact Observer" is hosted online at the IDH LIRMM YouTube channel: https://youtu.be/NnVgbZqZebU

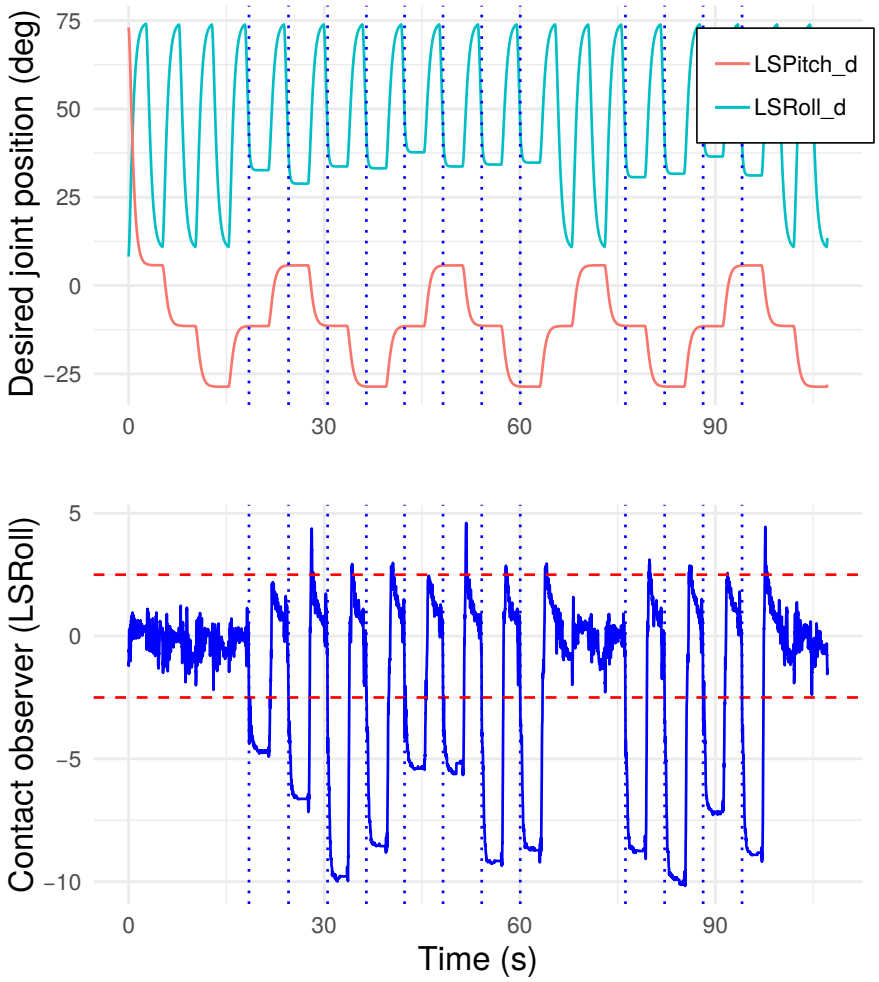


FIGURE 2.5 – *Robot stops moving the arm forward when it comes in contact with a human subject's back. The dotted blue lines show the start of the contact events, red dashed lines show the threshold values* $\pm\delta$.

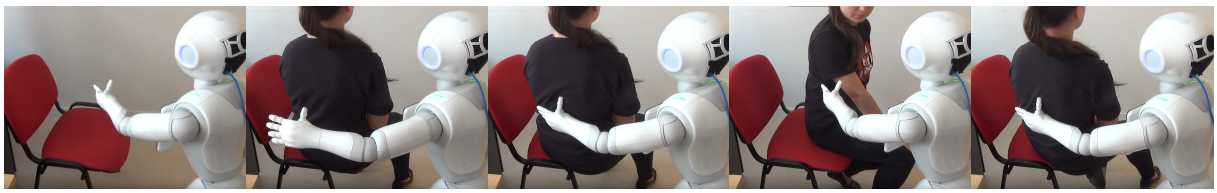


FIGURE 2.6 – *The snapshots of the experiment where the robot's arm motion is altered depending on whether it comes in contact with a human's back or not. The snapshots explained from left to right: (i) no contact,* LSRoll *joint is at its forward limit; (ii) no contact,* LSRoll *is at its backward limit; (iii) first contact, robot's arm stops moving before reaching the limit; (iv) human moves away, no contact,* LSRoll *is at its forward limit; (v) human moves back, robot's arm stops moving before reaching the limit.*

2.6 Conclusion

In this chapter, we have detailed the process of integration of a joint position tracking error discrepancy based contact observer as a real-time feedback signal in the QP controller for regulating compliant robot motion. The theoretical and implementation

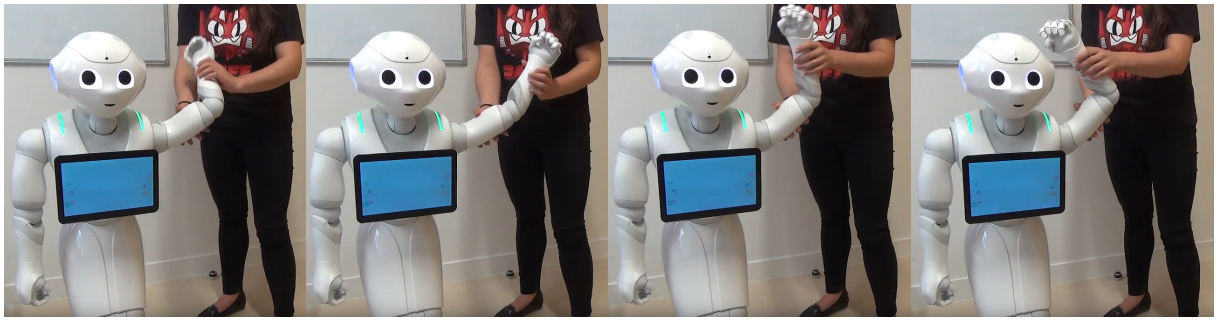


FIGURE 2.7 – *The snapshots of the experiment where the robot's arm motion is compliant to the human's touch.*

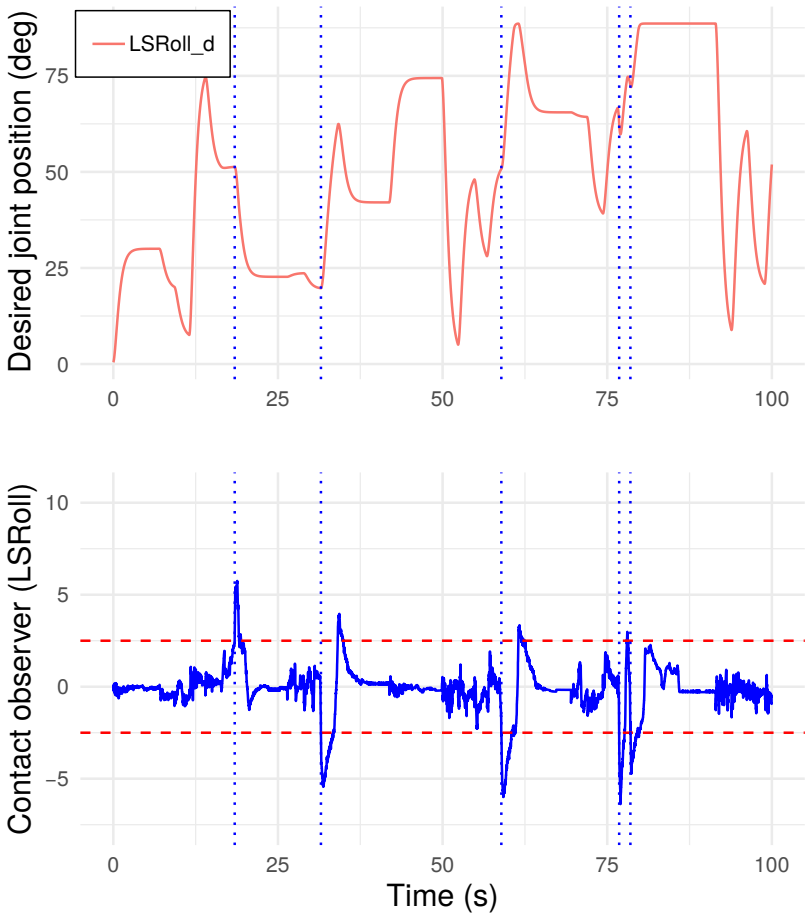


FIGURE 2.8 – *Robot moves joints to random set-points. When contact is detected, joint position is set according to Eq. 2.1. The dotted blue lines show the start of the contact events, red dashed lines show* $\pm\delta$.

details of contact observer robustification and compliance controller design have been described. Finally, we demonstrated the performance of the proposed method in pHRI experiments with humanoid robot Pepper.

The following Chapter 3 is dedicated to the whole-body feasible multi-contact posture planning for pHRI applications. Chapter 4 presents the whole-body controller for a pHRI scenario that utilizes the compliant robot motion control strategy, presented in this chapter, to establish several physical contacts with a human subject.

CHAPTER 3

MULTI-CONTACT PLANNING ON HUMANS

For robots to interact with humans in close proximity safely and efficiently, a specialized method to compute whole-body robot posture and plan contact locations is required. In our work, a humanoid robot is envisioned to be used as a caregiver that is assisting a human with a physical task. In this chapter, we present the proposed method for formulating and initializing a non-linear optimization posture generation problem using an intuitive description of the assistance task and the result of a human point cloud processing. The proposed method allows to plan whole-body posture and contact locations on a task-specific surface of a human body, under robot equilibrium, friction cone, torque/joint limits, collision avoidance, and assistance task inherent constraints. The proposed framework can uniformly handle any arbitrary surface generated from point clouds, for autonomously planning the contact locations and interaction forces on potentially moving, movable, and deformable surfaces, which occur in direct physical human-robot interaction. At the end of this chapter, we present and discuss the results of posture generation using the proposed method for pHRI scenarios.

3.1 Introduction

A humanoid robot can potentially be used as a reconfigurable and mobile multi-functional assistive support structure. One such platform can be used to assist in various tasks, unlike simpler robots that are designed for a specific task. Moreover, a familiar anthropomorphic appearance of the robot can result in overall better usability due to higher likeability Staffa and Rossi (2016), more intuitive communication Mitzner et al. (2014); Torta et al. (2014) and easier gain of user trust and acceptance of the technology Li et al. (2010).

This chapter is dedicated to the development of the multi-contact planning methodology that works directly on a human point cloud. The central question of multi-contact planning is the computation of a feasible multi-contact configuration of a humanoid robot. In what follows, we answer the question: how to compute a feasible robot posture in contact with a human, and plan contact locations on a surface of a

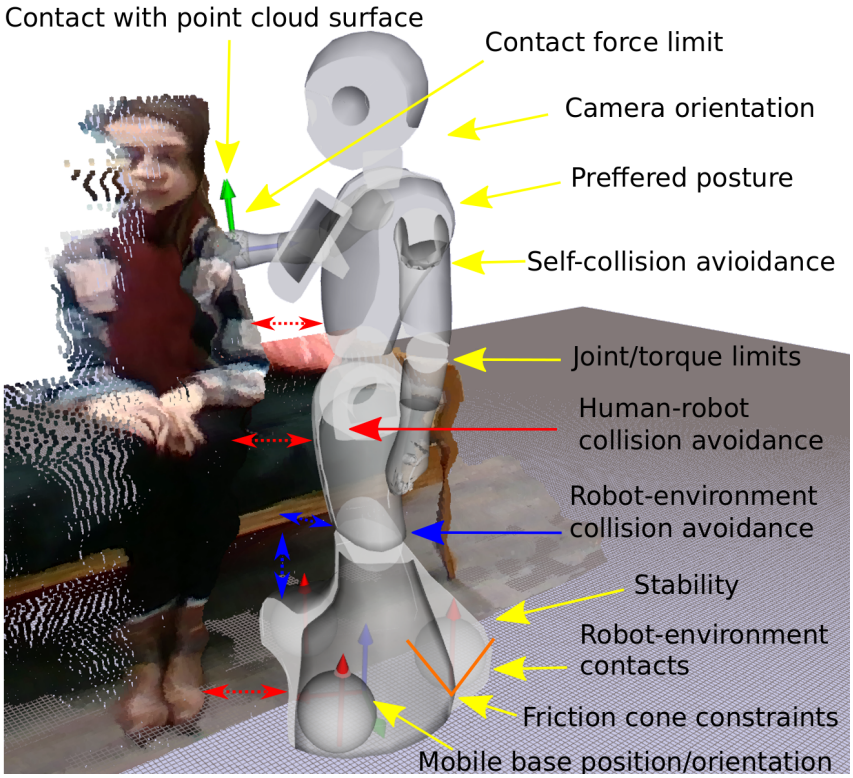


FIGURE 3.1 – *Safe and feasible humanoid robot posture for pHRI.*

human body part, while accounting for human safety and comfort, as well as robot structure and assistance task inherent constraints? (Fig. 3.1)

In our work, we introduce additional constraints to the posture generator (PG) formulated as non-linear optimization on non-euclidean manifolds Brossette et al. (2018). Multi-contact planning follows care-givers guidance to build and initialize PG, which includes the robot, assistance task and human inherent constraints and objectives. We present newly integrated PG constraints that plan contact locations on a surface of a human body part specified in the assistance task description. We fit a Non-Uniform Rational B-Spline (NURBS) surface, trimmed by a NURBS curve, on a segmented point cloud, that represents a human body part, as acquired from the embedded robot's camera, and use it to formulate geometric contact constraints. Additionally, we use collision avoidance with human point cloud constraints to ensure pHRI safety. More explicitly, the contributions of the work presented in this chapter are following:

1. A construction and initialization of PG for pHRI from an intuitive description of an assistance task (Sec. 3.3);

2. A formulation of constraints for contact location planning on a human point cloud surface (Sec. 3.3.1);

3. A human point cloud processing for contact planning and collision avoidance PG constraints (Sec. 3.3.2);

4. An evaluation of the proposed method in pHRI scenarios (Sec. 3.4).

3.2 Related works

Multi-contact planning has been addressed in various works and is currently a hot topic in humanoid research. Yet, it has never been extended to human physical assistance. To our best knowledge, our work is the first to consider such a perspective. At the heart of multi-contact planning, there is a so-called PG that generates on-demand (i.e. requests from the search strategy) the possible contact candidates.

Multi-contact planning methods treated mostly interaction with static and rigid environments, for which exact models are available. Existing methods address multi-contact planning by building a search tree Escande et al. (2013) or using a cascade coupling Tonneau et al. (2018), both based on a frequent inference of the PG (viewed as generalized inverse kinematics). First, the contact surfaces are elected and the corresponding contact posture is found, if exists, that realizes these contacts. Other methods embed contact planning directly with the motion problem formulation. For instance, in Arreguit et al. (2018) the contact sequence is predefined, the rigid and static environment is modelled by flat circular surfaces, and the contact locations of each humanoid end-effector are part of the optimization variables, so they are free to move inside the predefined contact surfaces. Similarly, the contact locations on flat surfaces, and also the contact sequence itself, have been incorporated into an optimization problem by the use of mixed-integer optimization Ibanez et al. (2014); Deits and Tedrake (2014); Ponton et al. (2016), non-linear trajectory optimization Dai et al. (2014) or augmentation of the contact creation related decision variables Mordatch et al. (2012). Those are some of the most outstanding works in multi-contact motion planning, but so far none have ever addressed contact planning on a human body for assistance.

One could be tempted to extend previous multi-contact planning to pHRI. In practice there are interesting simplifications such as the fact that (i) a human represents a closed-form almost know surface for contact planning, and (ii) assistance must follow recommendations from geriatric, care-givers and doctor professionals, which means that the type of contact to achieve a given assistance are known and must be followed. However, there are also some difficulties such as the fact that (i) human is articulated, and (ii) its surface is varying with clothes and deformations which require using direct perception to plan contacts, e.g. point clouds. The PG on point cloud has been explored in preliminary experiments using plane segmentation for stair climbing Oßwald et al. (2011), and in multi-contact navigation planning on flat and rigid surfaces Brossette et al. (2013). In pHRI, however, a basic plane fitted on a point cloud cannot well-represent a human body part surface. The inclusion of a trimmed NURBS surface into the PG, proposed in our work, allows to achieve high flexibility for the modelling of a surface for contact location planning. As a result, our proposed framework can handle a wide range of various pHRI scenarios.

In our work, we use a single RGBD camera and human links location 2D probability heatmaps, from OpenPose library Cao et al. (2017), to perform human link point cloud segmentation (Sec. 3.3.2). The output of the segmentation is then used to construct a parametric surface for contact location planning and convex hulls of human body parts for collision avoidance constraints (Sec. 3.3.1). Several works dedicated to the problem of the semantic meaning of human body parts in a point cloud exist in the literature. For instance, the method to fit an entire 3D human body model to the point cloud was proposed in Kwok et al. (2014). This method, however, requires two RGBD cameras, a

set of precomputed human body templates and takes around 30s to complete. Additionally, the output of this method is a volumetric mesh of a human body, which cannot be efficiently incorporated into the contact point location search optimization problem that requires continuous parametric surface as an input. A faster method for human pose estimation via skeleton fitting on a point cloud acquired by multiple RGBD cameras was proposed by Barros et al. (2015). This method, first, roughly initializes the human skeleton on a point cloud. Then, points in the cloud are assigned the limb class to which they are closest to. This method is reported to perform under 1 s, and could potentially be used in our framework for point cloud segmentation. However, it requires two RGBD sensors calibrated setup, whereas we opt to use one RGBD sensor for lower set-up complexity and potential use of the method with the on-board robot camera. The machine learning-based labelling of point cloud elements into human body part classes was proposed in Buys et al. (2014). The output of this method could also potentially substitute OpenPose probability heatmaps in our framework (Fig. 3.4). However, the performance of this method on our sample data has shown to be far less superior compared to that of the OpenPose, which made it unusable in our work.

In the following Sec. 3.3, we describe in detail our proposed methodology for whole-body robot posture planning for pHRI. Our proposed method requires a point cloud from a single RGBD camera and an intuitive description of the pHRI task to generate a safe and feasible whole-body robot posture suitable for the pHRI task. The proposed method also allows to find appropriate optimal contact locations on a finely defined human body part surface, fitted directly on the human point cloud. To the best of our knowledge, no PG framework has such functionality, which is critical for pHRI applications.

3.3 Proposed method

In the world, where robots will be working with human caregivers for assisting patients in secondary care tasks, it is paramount that high-level intuitive human commands can be translated into low-level robot motion planning and control objectives. It is especially critical to make sure that there is no increased workload of caregivers due to the introduction of robots into their workspace, a problem that often goes unnoticed in the process of introduction of new technology Niemelä and Melkas (2019). Indeed, in the caregiving sector, assistance know-how and practices should be instructed to the robot from health professionals' knowledge and practices Khatib et al. (1999). The objective of our work, presented in this chapter, is to enable the robot to generate safe feasible postures for engaging in pHRI as instructed from the high-level instructions given to the robot by non-(robotics) experts.

For a given assistance task instruction, here we denote it as $\mathcal{T}$, the optimal floating base location (i.e. position and orientation) and joints configuration of a humanoid robot, that we call robot *posture* denoted as $\boldsymbol{q}$, have to be planned simultaneously. Additionally, the PG algorithm needs to autonomously plan the suitable, appropriate and feasible contact locations on the patient body, where assistive forces should be applied. The computed posture must enable the robot to supply or resist required contact forces to assist a human as required while ensuring its own equilibrium, human comfort and safety. To compute such feasible posture, we propose to formulate a non-linear optimi-

zation problem described by Eq. 3.1. [1]

$$\min_{\boldsymbol{q},\boldsymbol{u},\boldsymbol{v},\boldsymbol{f}} \quad \phi(\boldsymbol{q},\boldsymbol{u},\boldsymbol{v},\boldsymbol{f}) \tag{3.1a}$$

$$\text{s. t.} \begin{cases} \text{joint and torque limits} & (3.1b) \\ \text{self–collision avoidance} & (3.1c) \\ \text{environment–robot collision avoidance} & (3.1d) \\ \text{environment–robot contacts} & (3.1e) \\ \text{equilibrium and friction cone constraints} & (3.1f) \\ \text{human–robot collision avoidance} & (3.1g) \\ \boldsymbol{f}_i^{\min} \leq \boldsymbol{f}_i^c \leq \boldsymbol{f}_i^{\max} \quad \forall i \in [1,m] & (3.1h) \\ \boldsymbol{P}_i^e = \mathcal{S}_i(u_i,v_i) \quad \forall i \in [1,m] & (3.1i) \\ u_i, v_i \in \Omega_i^{\mathcal{C}} \quad \forall i \in [1,m] & (3.1j) \end{cases}$$

Here, ϕ is a cost function –i.e. a function of the decision variables, to minimize (e.g. distance to a desired posture, torques, contact forces, etc.). The explicit expression of ϕ is given in Sec. 3.4, Eq. 3.5a. The constraints Eq. 3.1b–3.1h have been previously discussed in Brossette et al. (2018) and are described in detail in Brossette (2016). Some are bounds on the decision variables and others are more general bounds. The novel constraints Eq. 3.1i–3.1j, which enable the PG to plan contact location on a surface of the human body part, are introduced in this work (Sec. 3.3.1).

An assistance task instruction $\mathcal{T}$, for instance, could be the following: $\mathcal{T} = \{$Lightly contact the patient on the left arm with the robot's right hand and turn the robot's head to look at her/his face$\}$. Such description of the task determines the structure of the PG optimization problem, it's constraints and objectives. More precisely, it dictates the total number of robot-human contacts m, robot end-effector to be used for making contact $\boldsymbol{P}_i^e$, the patient body parts to contact $\mathcal{S}_i$ (Eq. 3.1i) (trimmed by $\Omega_i^{\mathcal{C}}$ if necessary (Eq. 3.1j)), class of force bounds that is appropriate for the task $\boldsymbol{f}_i^{\min}, \boldsymbol{f}_i^{\max}$ (Eq. 3.1h), and robot head orientation objective (see Eq. 3.5a).

To plan the contact locations on a surface of a human body part, as dictated by $\mathcal{T}$, the constraint expressed via Eq. 3.1i is added to the optimization problem. This constraint restricts the robot-human contact to lie on the NURBS surface $\mathcal{S}_i$ fitted to the point cloud of the segmented human body part. To allow our framework to equally handle any arbitrary point cloud or special-cases, we use additional curve enclosure constraint, Eq. 3.1j, that trims away areas in the parametric space of $\mathcal{S}_i$ that are not suitable for establishing a contact (e.g. the areas not covered by the point cloud, sensitive or delicate areas of the human body surface). The NURBS surface parameters u_i and v_i are stacked into decision variables $\boldsymbol{u}$ and $\boldsymbol{v}$ of our PG on manifold, respectively. The robot contact forces $\boldsymbol{f}_i^c$ are stacked into a PG decision variables vector $\boldsymbol{f}$.

Depending on the problem and its size, non-linear optimization is not deterministic in general, see discussions in Brossette et al. (2018). Nevertheless, we suggest that the information that can be extracted from $\mathcal{T}$ and the information of the human perception in the environment (Sec. 3.3.2), allows us to have very good initial guesses. This allows the solution, when it exists, to be found relatively fast. We bring practical examples of such task-aware PG initialization in the Sec. 3.4. As in any gradient descent approach,

1. Matrices and vectors are in bold; scalars in non-bold lower-case; descriptive functions in calligraphic font.

there is no guarantee that the solution is a global optimum nor that it could be systematically found when it exists. In the latter case, we perturb the initial guess.

We detail the implementation of constraints Eq. 3.1i and Eq. 3.1j in the following section 3.3.1, present the details of a human point cloud processing pipeline in Sec. 3.3.2 and, finally, exemplify the use of our approach on sample pHRI tasks in Sec. 3.4.

3.3.1 Contact constrained to a surface fitted on a point cloud

In this section, we detail the constraints for planning a contact location on an point cloud surface (Eq. 3.1i–3.1j).

Given a point cloud $\boldsymbol{D} = \{\boldsymbol{p}_k | k = 0, \cdots, K\}$ and an initial guess of the control points locations $\boldsymbol{P}_{ij}^{\text{init}}$ of the NURBS surface $\mathcal{S}$, the control points position update $\boldsymbol{c}_{ij}$, that fits $\mathcal{S}$ onto $\boldsymbol{D}$, can be computed via quadratic optimization that minimizes the Euclidean distance between points $\boldsymbol{p}_k$ and corresponding closest points projected onto surface as $\boldsymbol{\mathcal{S}}(\boldsymbol{u}_k, \boldsymbol{v}_k)$, see Eqs. 3.2a–3.2b.

$$\min_{\boldsymbol{c}_{ij}} \quad \sum_k \left(p_k - \frac{\sum_i \sum_j N_{i,b}(u_k) N_{j,r}(v_k)(\boldsymbol{P}_{ij}^{\text{init}} + \boldsymbol{c}_{ij})}{\sum_i \sum_j N_{i,b}(u_k) N_{j,r}(v_k)} \right)^2 \tag{3.2a}$$

$$0 \leq u_k, v_k \leq 1 \quad u_k, v_k \in \mathbb{R} \tag{3.2b}$$

The parameters b and r denote surface order in directions U and V of the surface parametric space, respectively. The nonrational B-spline basis functions are denoted as $N_{x,y}(z)$. The number of control points can be either predefined or adjusted in the fitting process Piegl and Tiller (2012); Dimitrov et al. (2016); Flöry (2009).

Example of a point cloud $\boldsymbol{D}$ and a fitted NURBS surface $\mathcal{S}$ are shown in Fig. 3.2 (top). This figure also illustrates the next issue we need to address, which is the four-sided nature of the NURBS surface. Since NURBS parametric space is four-sided, surface fitted to an arbitrary point cloud likely needs to be trimmed by fitting a constraining closed curve $\mathcal{C}$ that encloses the point cloud, thus defining a subspace of the surface parametric space $\Omega^{\mathcal{C}} \subset UV$ that is suitable for making a contact. The control points of $\mathcal{C}$ are found by, first, projecting 3D points $\boldsymbol{p}_k$ into $UV \subset \mathbb{R}^2$ space, to get corresponding 2D points $\boldsymbol{g}_k$. Given the initial guess of curve control points location $\boldsymbol{P}_i^{\text{init}}$, so-called, footpoint parameter t_k is computed for every $\boldsymbol{g}_k$, so that point $\boldsymbol{\mathcal{C}}(t_k)$ on the curve is the closest point to $\boldsymbol{g}_k$ and $\vec{\boldsymbol{n}}_{t_k}$ is curve normal at this point. The constraining curve fitting process consists in finding curve control points position update values $\boldsymbol{c}_i$ by solving the optimization problem Eqs. 3.3a–3.3b Flöry and Hofer (2008).

$$\min_{\boldsymbol{c}_i} \quad \sum_k w_k \left(g_k - \frac{\sum_i N_{i,s}(t_k)(\boldsymbol{P}_i^{\text{init}} + \boldsymbol{c}_i)}{\sum_u N_{i,s}(t_k)} \right)^2 \tag{3.3a}$$

$$(\boldsymbol{g}_k - \boldsymbol{\mathcal{C}}(t_k))^T \cdot \vec{\boldsymbol{n}}_{t_k} \leq 0 \tag{3.3b}$$

where s is the curve order and w_k are the point's weights, which are lower for the interior points and higher for the points which are closest to the curve. The points $\boldsymbol{g}_k$, projected onto the UV space of $\mathcal{S}$, along with fitted constraining curve $\mathcal{C}$, are shown in Fig. 3.2 (top). Note that if $\boldsymbol{D}$ represents a rather four-sided real surface, the fitting of $\mathcal{C}$ may not be required, as the surface underlying such a point cloud will already be well-defined by $\mathcal{S}$ (i.e. nothing to "trim out" with $\mathcal{C}$).

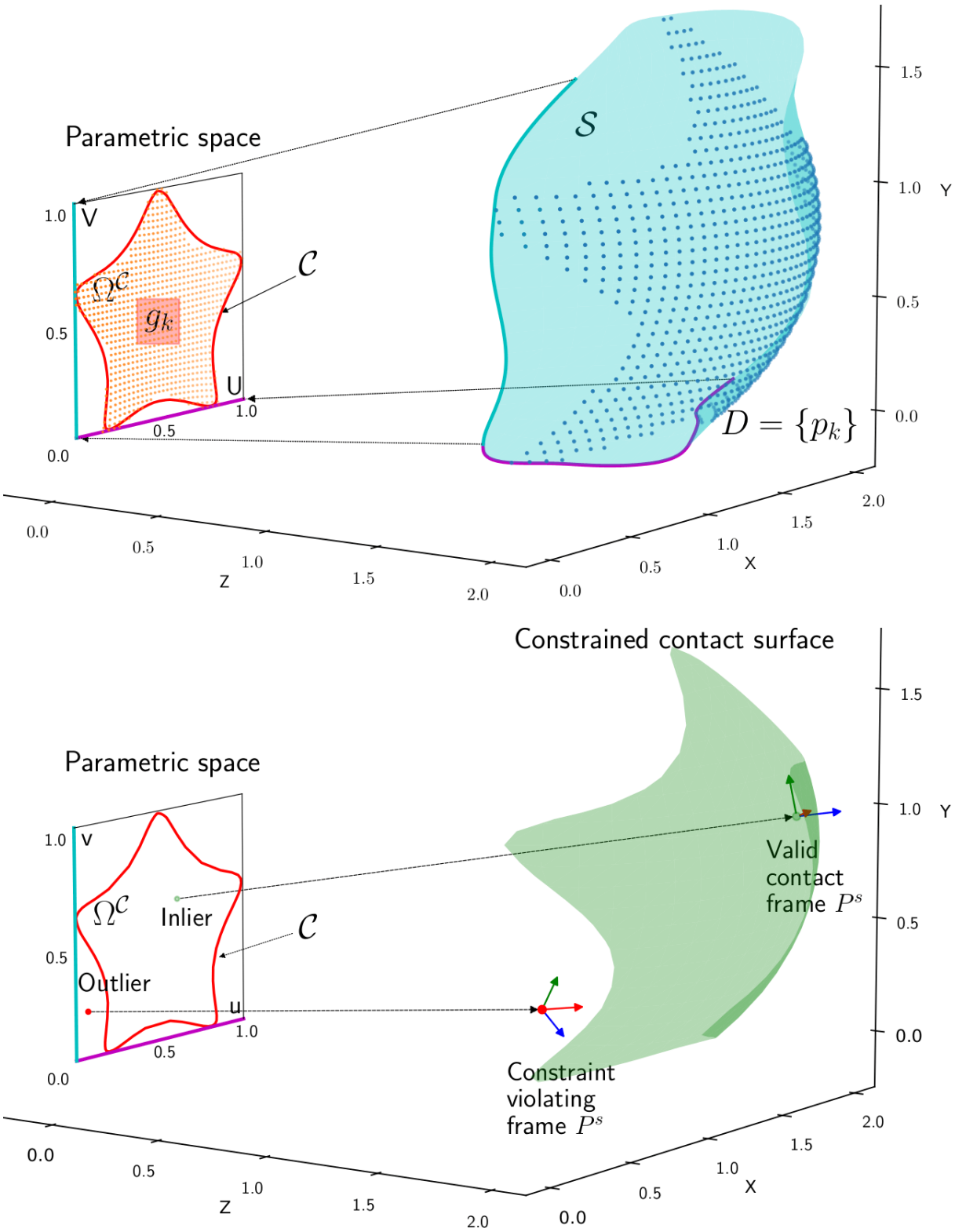


FIGURE 3.2 – *NURBS surface and curve fitting to point cloud (top), trimmed contact surface constraint representation (bottom).*

After $\mathcal{S}$ and $\mathcal{C}$ control point values are adjusted to fit $\boldsymbol{D}$, the goal is to find such values of u^* and v^* so that the contact point location $\boldsymbol{P}^e \in \mathcal{S}(u^*, v^*)$ is on the area of NURBS surface covered by the point cloud (i.e. $u^*, v^* \in \Omega^{\mathcal{C}}$), while satisfying all the other PG constraints Eqs. 4.1b– 3.1h.

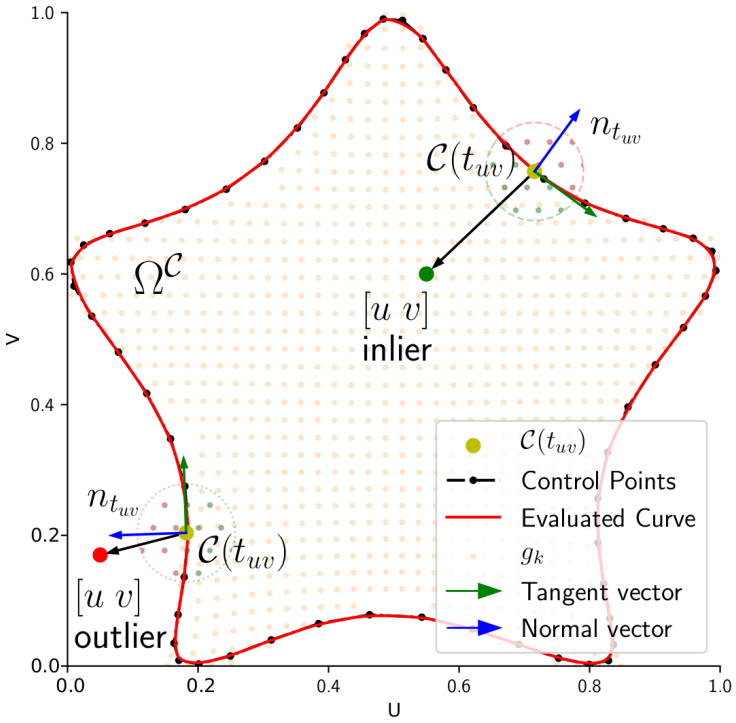


FIGURE 3.3 – *Curve enclosure constraint on UV space.*

The fitted surface and the constraining curve can now be incorporated into the PG formulation. We consider here only one contact for the purpose of presentation clarity. The end-effector frame for contact is $\boldsymbol{P}^e = \{\boldsymbol{p}^e, \boldsymbol{R}^e = (\vec{\boldsymbol{x}}^e, \vec{\boldsymbol{y}}^e, \vec{\boldsymbol{z}}^e)\}$, with frame position $\boldsymbol{p}^e$ and orientation $\boldsymbol{R}^e$ w.r.t. the world frame. We add surface parameters as additional decision variables u and v, and add the constraints Eqs. 3.4c–3.4j to the PG problem.

$$\min_{\boldsymbol{q},u,v,\boldsymbol{f}} \quad \phi(\boldsymbol{q}, u, v, \boldsymbol{f}) \tag{3.4a}$$

$$\text{s. t.} \quad \text{eq. 4.1b–eq. 3.1h} \tag{3.4b}$$

$$\boldsymbol{p^s} = \frac{\sum_i \sum_j N_{i,b}(u) N_{j,r}(v) \boldsymbol{P}_{ij}}{\sum_i \sum_j N_{i,b}(u) N_{j,r}(v)} \tag{3.4c}$$

$$0 \leq u, v \leq 1 \quad u, v \in \mathbb{R} \tag{3.4d}$$

$$([u\ v] - \boldsymbol{\mathcal{C}}(t_{uv}))^T \cdot \vec{\boldsymbol{n}}_{t_{uv}} \leq 0 \tag{3.4e}$$

$$\boldsymbol{du} = \frac{\partial \mathcal{S}(u,v)}{\partial u} \qquad \boldsymbol{dv} = \frac{\partial \mathcal{S}(u,v)}{\partial v} \tag{3.4f}$$

$$\boldsymbol{z}^s = \boldsymbol{du} \times \boldsymbol{dv} \qquad \boldsymbol{y}^s = \boldsymbol{z}^s \times \boldsymbol{du} \tag{3.4g}$$

$$\boldsymbol{R}^s = \left(\frac{\boldsymbol{du}}{|\boldsymbol{du}|}, \frac{\boldsymbol{y}^s}{|\boldsymbol{y}^s|}, \frac{\boldsymbol{z}^s}{|\boldsymbol{z}^s|} \right) \tag{3.4h}$$

$$\overrightarrow{\boldsymbol{p}^s \boldsymbol{p}^e} \cdot \boldsymbol{R}^s = (0,0,0) \tag{3.4i}$$

$$\vec{\boldsymbol{z}}^e \cdot \boldsymbol{R}^s_{xy} = (0,0) \qquad \vec{\boldsymbol{z}}^e \cdot \boldsymbol{R}^s_z \geq 0 \tag{3.4j}$$

The constraints Eqs. 3.4c–3.4e ensure that the contact point $\boldsymbol{p}^s$ lies on the authorized surface area. The constraints Eqs. 3.4f–3.4h compute the surface contact frame orientation $\boldsymbol{R}^s$ at point $\boldsymbol{p}^s$ with Z-axis aligned with the surface normal. The constraints Eqs. 3.4i– 3.4j align the robot's end-effector frame with the surface contact frame in 3 translational directions and 2 orientational axes. The robot is free to choose its contac-

ting end-effector orientation only around the surface normal. These constraints are illustrated in Fig. 3.2 (bottom); specifically Eq. 3.4e is illustrated in Fig. 3.3. Here, the variable $t_{uv} \in [0\ 1]$, from the curve parametric space, is a footpoint parameter of $[u\ v]$ point, computed on previous optimization iteration, such that $\boldsymbol{\mathcal{C}}(t_{uv})$ is the closest point on the curve to $[u\ v]$ and $\vec{\boldsymbol{n}}_{t_{uv}}$ is the curve's normal at this point.

The solution to Eq. 3.4 is an optimal whole-body robot posture and the contact location on the point cloud surface, approximated by the trimmed NURBS surface, that satisfies joint and torque limits, maintains robot statically stable, keeps interaction forces inside the friction cones, avoids collisions and satisfies contact force bounds.

In the following Sec. 3.3.2, we detail the human point cloud segmentation that supplies the input $\boldsymbol{D}$ for the construction of the contact constraints of the proposed PG framework. We also describe how the point cloud segmentation is used to construct human-robot collision avoidance constraints of PG (Eq. 3.1g).

3.3.2 Processing of human point cloud

Our proposed robot-human contact planning, described in Sec. 3.3.1, can be used for assistance pHRI tasks, as long as, the point cloud to contact $\boldsymbol{D}$ (e.g. human shoulder, human back) is properly segmented out from an entire scene observed by an RGBD camera. Here, we present the point cloud segmentation pipeline that supplies input to our PG. The overview of the entire pipeline is shown in Fig. 3.4.

First, an RGB image is processed by a two-branch multi-stage convolutional neural network (CNN) from the OpenPose library Cao et al. (2017). This CNN predicts confidence maps (CM) for 25 main human body keypoints, by assigning the likelihood of the presence of a particular human body part to every image pixel. Simultaneously, CNN predicts, so-called, Part Affinity Fields (PAF), which encode the location and orientation of human body parts in the 2D image.

In our work, we use CM and PAF to compute human body parts 2D masks for point cloud segmentation. We threshold PAF and CM of all the body parts to consider only high likelihood pixels ($\geq \sigma = 40\%$), which are assigned a 100% likelihood after thresholding. The pixels with likelihood below σ are assigned 0% likelihood. As a result, we obtain black and white images that represent 2D masks of human body parts.

We combine all resulting 2D masks to obtain 16 masks for body part segmentation from point cloud (different total number of masks can be used depending on the use-case). We dilate the resulting masks to remove small holes and expand the borders. The head mask is augmented by adding an ellipse of estimated head width and height around the face centre. The torso mask is augmented with a polygon that connects 3-4 visible torso keypoints (assuming that at least 3 torso keypoints are visible). Further, subtraction of body parts masks that are likely to be occluding torso (e.g. arms, forearms, hands) is performed on the torso mask.

Once the masks for all individual body parts are computed, a depth image and camera intrinsic parameters are used to compute a 3D point cloud of the entire scene. We apply human body parts 2D masks on a point cloud to segment 16 sub-clouds which contain only those 3D points that are likely to belong to each particular human body part.

The result of the segmentation is used to select a sub-cloud $\boldsymbol{D}$ that, according to

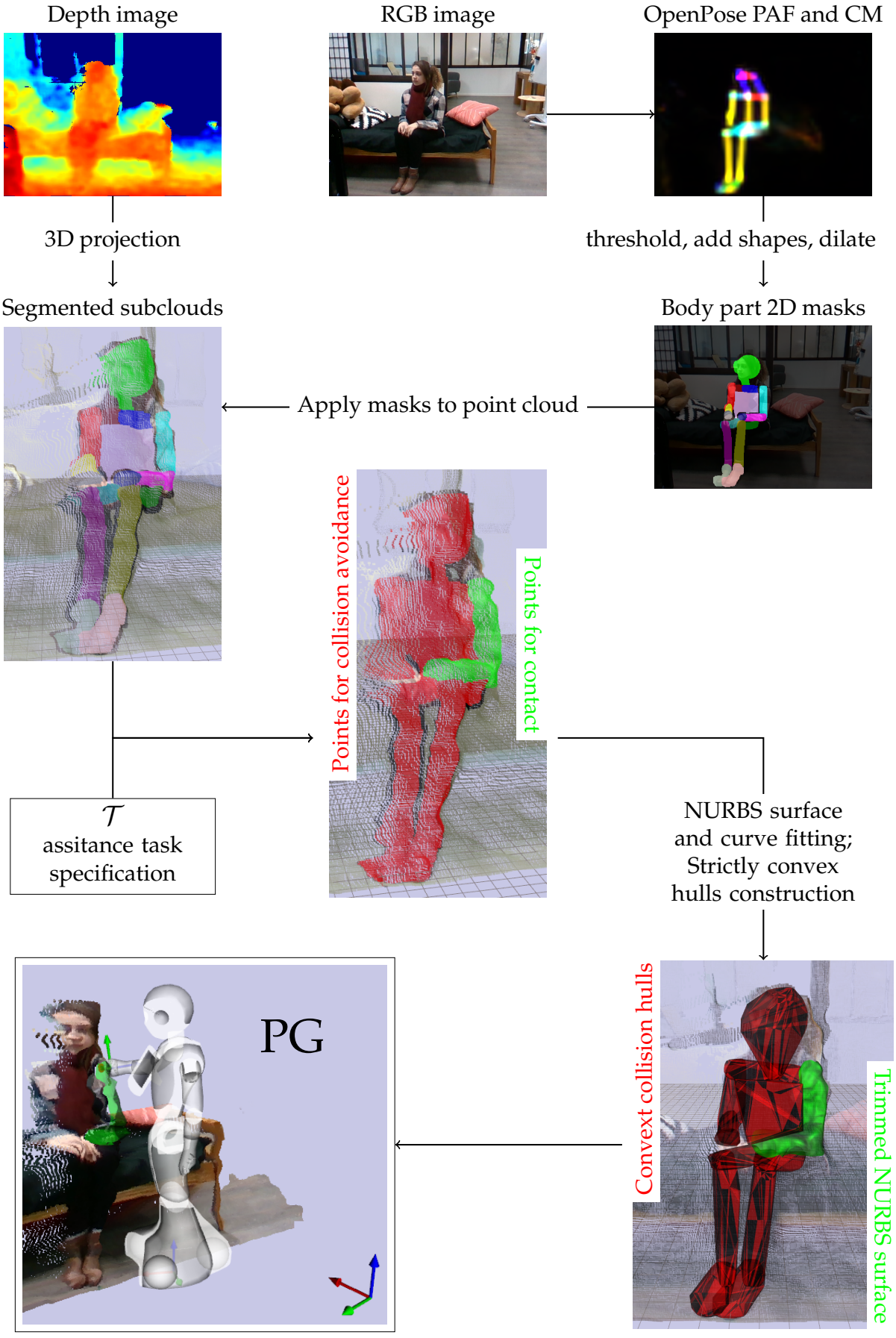


FIGURE 3.4 – *Point cloud processing pipeline to supply input for the posture optimization problem formulation for pHRI assistance task.*

predefined task $\mathcal{T}$, is to be used for establishing a contact. This sub-cloud $\boldsymbol{D}$ is filtered, downsampled made fit a NURBS surface $\mathcal{S}$ and a trimming curve $\mathcal{C}$, using NURBS algorithms from Point Cloud Library Piegl and Tiller (2012); Rusu and Cousins (2011), and building contact constraints Eqs. 3.4c–3.4j.

All segmented sub-clouds are used to create (strictly) convex hulls of human body parts for human-robot non-desired collision avoidance (Eq. 3.1g). The PG's collision avoidance is implemented using an efficient GJK distance algorithm for proximity queries Escande et al. (2014b). The cloud $\boldsymbol{D}$, representing a body link for the contact creation, is also used to define collision avoidance constraints with all robot links but the end-effector used for contact as specified in $\mathcal{T}$.

Our proposed method does not require costly estimation of a human model parameters, its floating base position and orientation or joint angles. Nevertheless, it provides all the necessary information, extracted directly from a point cloud, for computing a safe and feasible multi-contact posture suitable for initiating pHRI for a well specified and intuitively formulated assistance task. There are ongoing efforts in the 3D computer graphics and vision communities to provide directly reliable 3D pose and joint configuration of humans in any posture, see e.g. Xu et al. (2018). Shall this be one day readily available in reasonable computation time and reliability, we can simply replace our pipeline with it, eventually pre-fit a personalized NURB on it, and use it as an input for our PG. It won't make our PG faster but we will gain in the perception side (i.e. the construction of the PG problem).

We exemplify how our proposed method performs on sample point clouds in the following Sec. 3.4.

3.4 Experimental Results

In this section, we present the evaluation of the performance of our proposed method for whole-body posture generation and contact location planning on a human point cloud. First, we outline the method implementation details (Sec. 3.4.1). Then, we discuss the results of NURBS surface and curve fitting to the segmented human body part point cloud (Sec. 3.4.2). And finally, we present the results of the robot posture and contact locations generated using the proposed method for three sample pHRI scenarios in the context of human care and assistance (Sec. 3.4.3–3.4.5). We conclude this section with discussion on method limitations and future work perspectives (Sec. 3.4.6).

3.4.1 Implementation details

The PG framework Brossette et al. (2018), that we use and extend for the pHRI use-case in our work, is highly versatile and is not robot specific. Given just a URDF file description of Pepper robot, that we use as a platform in our work, all the basic PG constraints (Eqs. 3.1b–3.1f) are automatically constructed using this PG framework.

The novel PG constraints (Eqs. 3.4c–3.4j, and 3.1g) are constructed based on the results of the point cloud processing in each particular scenario. The only robot specific parts of the PG formulation in our scenarios is a frame constraint for Pepper mobile base, which is free to move only in XY plane and around Z-axis of the world reference frame, and 3 contact constraints with the ground one per each robot wheel.

The RGBD data for each scenario was collected and then processed offline. We present the resulting safe and feasible multi-contact postures for pHRI, computed by our proposed method, visualized in RViz together with the corresponding point clouds and trimmed contact NURBS surfaces on the right side of Figs. 3.5–3.7. The plots on the left side of Figs. 3.5–3.7 show cost function (Eq. 3.5a) convergence, for each scenario, that indicate the optimality of the resulting PG solutions. The convergence criterion of PG is thoroughly described in Brossette (2016) (see p. 78).

3.4.2 Results of surface and constraining curve fitting

We assume that after segmentation and filtering a point cloud accurately represents the underlying real surface. We consider the trimmed NURBS surface to well represent the actual surface of the human body when the average squared fitting errors, Eq. 3.2a and Eq. 3.3a, are below 1^2mm and 0.005^2, for surface and curve, respectively. Fitting the constraining curve with a lower tolerance threshold is significantly slower and more importantly useless. The human body is compliant already, and in the online experiments, the person might move a bit too. Thus, the robot must be controlled to reach the person and establish the contacts in closed-loop. One potential continuation of this work could exploit the robot's configuration, that is computed by our PG, to serve as a target for the closed-loop QP Bouyarmane et al. (2019) controller, that will achieve the desired contacts and postures at best using online perception and measured contacts Bolotnikova et al. (2018b,a). In such a controller, contacts will be made using guarded motion to absorb surface uncertainties. For example, when a motion supporting contact is required on the patient's back, it won't be required at mm precision.

3.4.3 Scenario 1: Attracting human's attention

For our first experiment, we consider a use-case where the robot attracts a human's attention by performing a light touch. For this use-case, the task for the robot is $\mathcal{T}_1 =$ {Lightly contact the patient on the left upper limb with your right hand and look at her/his face}. The points of the human left upper limb are extracted from the point cloud for fitting a trimmed NURBS surface for contact constraint with the robot's right end-effector. Other segmented sub-clouds are used to create convex hulls for collision avoidance.

Since we know the task to perform a priori, we can do a *task- and human-aware initialization* of PG. We initialize the robot posture with the mobile base in front of and facing the surface to contact (as detected from RGBD data), the right end-effector slightly raised and turned to be prepared for a contact, and the left end-effector in a downward position. We denote such robot configuration as *preferred posture* $\boldsymbol{q}^{\text{pref}}$, which can be dictated for specific classes of assistance tasks from human knowledge and expertise.

We use $\boldsymbol{q}^{\text{pref}}$ in the PG cost function to keep the final result to be close to the *preferred posture*. We also define the robot camera orientation objective, to ensure that it is oriented towards the human head sub-cloud average point $\boldsymbol{h}^{\text{avg}}$. Finally, the force bounds, which can be defined by medical professionals, for the 'light' human-robot contact interaction forces are set to $\boldsymbol{f}^{\min} = \{-0.05, -0.05, 0.5\}$, $\boldsymbol{f}^{\max} = \{0.05, 0.05, 3.0\}$

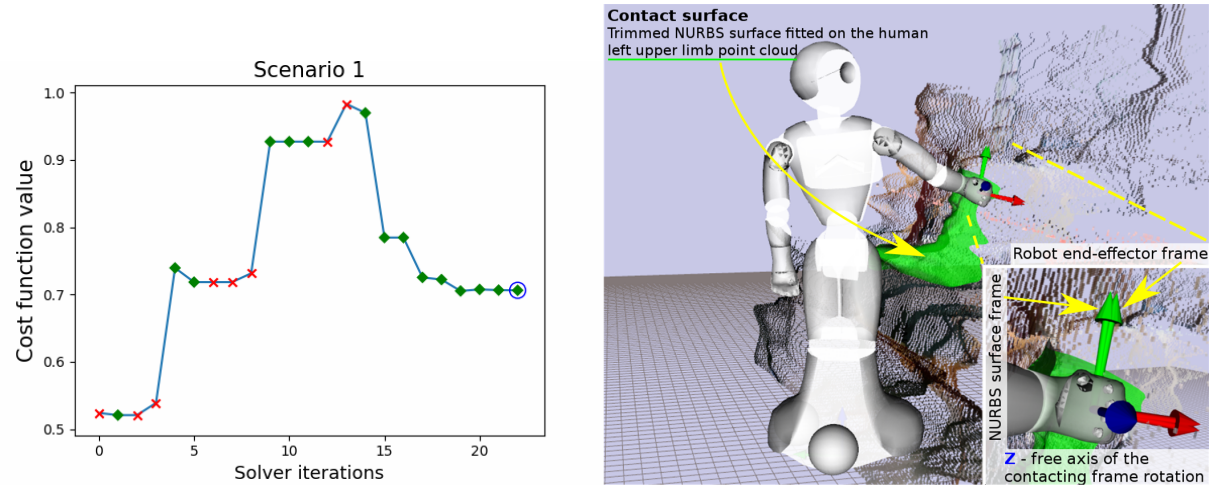


FIGURE 3.5 – *PG convergence plot (left), computed robot posture in contact with human point cloud (right) for experimental scenario 1.*

in our experimental scenarios. The final form of such PG is Eq. 3.5.

$$\min_{\boldsymbol{q},u,v,\boldsymbol{f}} \boldsymbol{w}_p \left\| \boldsymbol{q} - \boldsymbol{q}^{\text{pref}} \right\|^2 + \left\| \frac{\overrightarrow{\boldsymbol{p}^{\text{cam}}\boldsymbol{h}^{\text{avg}}}}{\left\| \overrightarrow{\boldsymbol{p}^{\text{cam}}\boldsymbol{h}^{\text{avg}}} \right\|} \cdot \boldsymbol{R}_z^{\text{cam}} - 1.0 \right\|^2 \tag{3.5a}$$

$$\text{s. t.} \quad \text{eq. 4.1b–eq. 3.1h} \quad \text{eq. 3.4c–eq. 3.4j} \tag{3.5b}$$

where $\boldsymbol{p}^{\text{cam}}$ and $\boldsymbol{R}_z^{\text{cam}}$ are the translation and Z-axis orientation of the robot camera optical frame, respectively, w.r.t. the world reference frame. The vector $\boldsymbol{w}_p$ contains the weights of the *preferred posture* objective for each robot joint. The elements of this vector are set to 1, except for the mobile base and neck joints, which are set to 0, to let the robot freely plan mobile base and head position and orientation.

The solution of Eq. 3.5, a safe and feasible robot posture for the pHRI task $\mathcal{T}_1$, is shown with annotations of all objectives and constraints in Fig. 3.1 at the beginning of this chapter. Another view of the same scene, that better illustrates the result of contact location planning on a trimmed surface fitted to point cloud, is shown on the right of Fig. 3.5.

3.4.4 Scenario 2: Initiating assistance for sit-to-stand transfer

The second scenario consists in initiating a process of assistance for sit-to-stand transfer. Note that a suitable strategy for assistance in such a scenario may vary from patient to patient. Here, we assume that a suitable strategy is to initiate two contacts. The first contact is closer to the patient's shoulder for applying a pushing force forward and upward. The second contact is closer to the hand of the patient, which would allow to control human's forward movement by resisting force applied on the robot end-effector by the patient. The same trimmed NURBS surface is used to formulate both PG contact constraints. However, different initial points in the surface parametric space are used for decision variables initial values, one closer to the shoulder and another closer to the hand. The command of the assistance task for this scenario given to the robot is $\mathcal{T}_2 = \{$Lightly contact the patient on the left upper limb near the shoulder

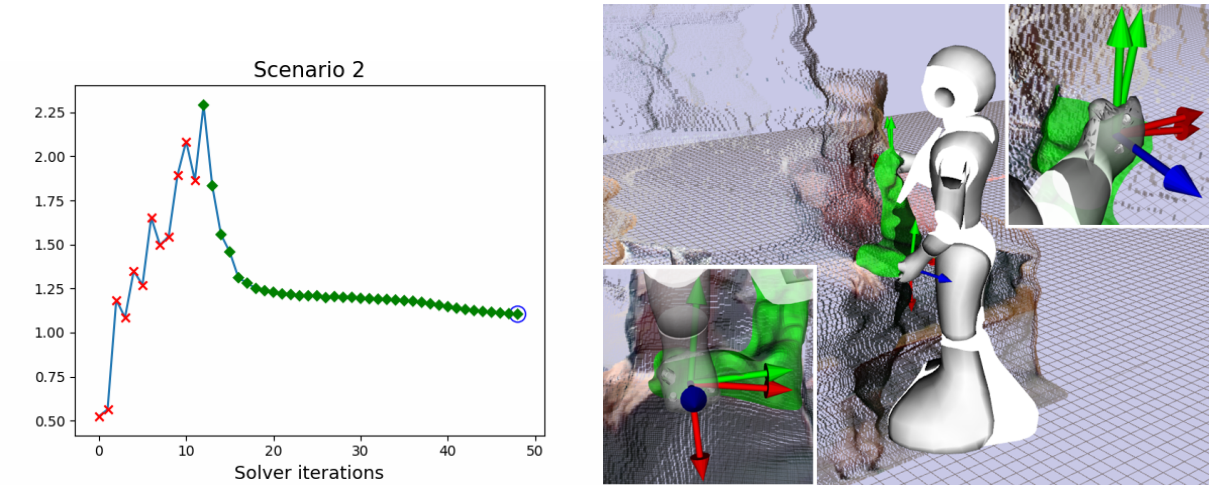


FIGURE 3.6 – *PG convergence plot (left), computed robot posture in contact with human point cloud (right) for experimental scenario 2.*

with your right hand, and near the patient's hand with your left hand, while looking at her/his face}. The objective function in this scenario has the same form as eq. 3.5a, with two sets of contact constraints with trimmed NURBS surface and a different task-specific $\boldsymbol{q}^{\text{pref}}$ (with both end effectors raised and turned in preparation for the contacts). The resulting PG convergence plot, computed robot posture and contact locations are shown in Fig. 3.6.

3.4.5 Scenario 3: Checking for responsiveness

In our last presented experimental scenario, the robot is required to check if a person, lying on a bed, is responsive. The task given to the robot is $\mathcal{T}_3 = \{$Lightly contact the patient on the right upper limb with your left hand and look at her/his face$\}$. The human right upper limb point cloud is segmented out and used for surface and curve fitting, which are then used to define the contact constraints between human right upper limb and robot left end-effector. We reuse the objective function of scenario 1, with a different value of $\boldsymbol{h}^{\text{avg}}$, as the human head is now in a different location in the scene. The robot left end-effector is slightly raised and turned in $\boldsymbol{q}^{\text{pref}}$. The PG convergence plot, resulting whole-body posture and optimal contact location are shown on Fig. 3.7.

3.4.6 Discussion, limitations and further developments

The task complexity, commutation times of each part of our method and the total number of PG solver iterations for each scenario are presented in Tab. 3.1. All computations were performed on the GeForce GTX 1050 Ti GPU.

The convex hull for collision avoidance between the robot and the bed, that a person is sitting or lying on, is defined and added manually to the simulation scene in all scenarios. Ideally, such convex hulls should be computed automatically. We also assume that a person is well detectable on a 2D image by OpenPose with the pixel probability threshold of 40%. Otherwise, the human point cloud cannot be well segmented and a safe robot posture cannot be computed using the proposed PG method.

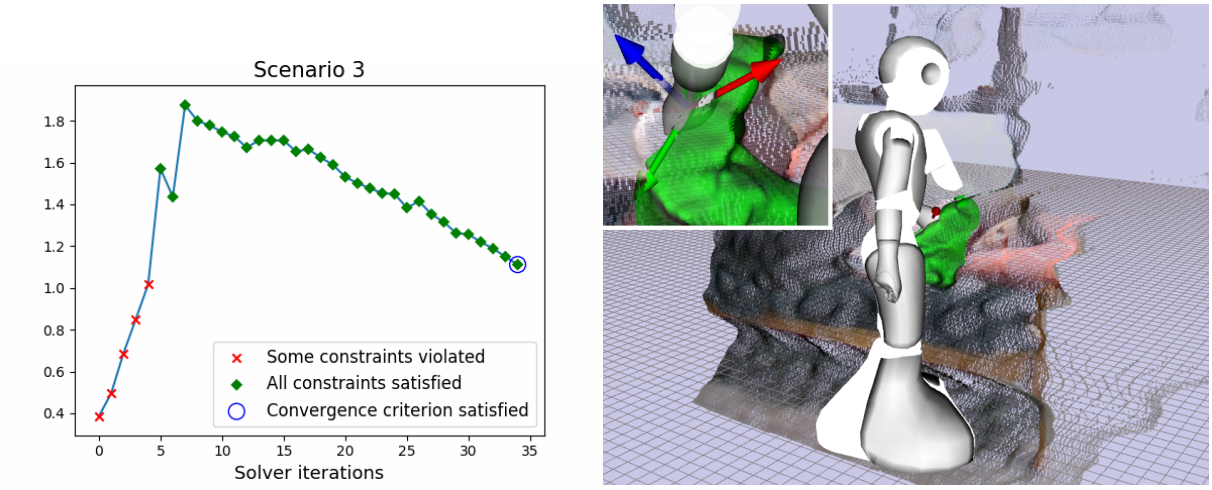


FIGURE 3.7 – *PG convergence plot (left), computed robot posture in contact with human point cloud (right) for experimental scenario 3.*

Scenario ID	1	2	3
Contact point cloud size	3049	3049	2862
Number of contacts	4	5	4
Point cloud segmentation / convex hulls creation time (s)	2.28	2.45	2.69
Trimmed NURBS surface fitting time (s)	0.717	0.734	0.926
Number of PG iterations	23	50	36
PG convergence time (s)	1.876	5.286	2.828

TABLE 3.1 – *The computation time of each phase of the method and the number of solver iterations for each test scenario.*

For the future work, the proposed method should be optimized for the online (re)planning of the robot motion in pHRI scenarios. The output of proposed PG could be used to control the robot in experiments, where the robot engages in physical contact with people for assistance in motion López et al. (2014). The user reaction to such interaction should be analyzed and used for method refinement in the following steps of development of motion planning methods and a 'robotiquette' for pHRI in close proximity Dautenhahn (2007). Lastly, the proposed method could be incorporated into the motion synthesis framework to compute robot trajectories accounting for the type of motion that human and robot must undergo while maintaining or switching contacts Escande et al. (2013). This must be done to guarantee that the computed postures outputted are indeed suitable for a particular a priori known assistance in motion task all along the motion path.

3.5 Conclusion

In this chapter, we have described in detail the proposed new constraints for generating safe and feasible multi-contact robot postures for pHRI tasks. We presented the details of the proposed pHRI specific contact constraints that allow to autonomously plan a feasible optimal contact location on human body parts for establishing a physical contact between a humanoid robot and a human subject. The implementation of the human point cloud processing, that generates the input for contact location plan-

ning and human-robot collision avoidance constraints, was presented and evaluated in three sample pHRI scenarios.

The next chapter is dedicated to the description, design and implementation of the whole-body humanoid robot QP controllers for pHRI applications in the caregiving context.

CHAPTER 4

AUTONOMOUS INITIATION OF HUMAN PHYSICAL ASSISTANCE

The goal of our work is to study the use of humanoid robot technology for physical assistance in motion for a frail person. A whole-body controller for a humanoid robot needs to be carefully designed in order to ensure efficient, intuitive and secure interaction between humanoid-assistant and human-patient. In this chapter, we present a design and implementation of a whole-body controller that enables a humanoid robot with a mobile base to autonomously reach a person, perform audiovisual communication of intent, and establish several physical contacts for initiating physical assistance. Our controller uses (i) visual human perception as a feedback for navigation and (ii) joint residual signal based contact detection for closed-loop physical contact creation. At the end of the chapter, we assess the developed controller on a healthy subject and report on the experiments achieved and discuss the results.

4.1 Introduction

In the work described in this chapter, we consider a general interaction scenario which may occur in everyday care. A humanoid robot is required to autonomously reach a person and establish several physical contacts on a human body to initiate physical assistance in motion. A whole-body task-space control framework [1] is used to develop a Finite State Machine (FSM) controller that enables a humanoid robot to autonomously participate in such an interaction scenario.

A visual human perception is utilized for closed loop navigation towards a human. The joint residual signal based contact detection, described earlier in Chapters 1 and 2, is used to determine time of robot-human contact events. Verbal, visual and body language communication is included in the controller design to enable a robot to autonomously communicate its intentions to a human. The contributions of the work presented in this chapter are following:

- we present the design of a whole-body controller architecture for the interaction scenario (Sec. 4.3.1);

1. https://jrl-umi3218.github.io/mc_rtc

- we describe the implementation of an autonomous visual feedback based navigation towards a human (Sec. 4.3.2);
- we detail the integration of a multi-modal communication of intent in the controller design (Sec. 4.3.3);
- finally, we present the results of using the controller for an interaction trial with a healthy subject (Sec. 4.4).

4.2 Background

In the field of research on pHRI for assistance in motion or power augmentation, a large majority of the works consider either the scenarios where human is creating a contact on the robot body surface Tirupachuri et al. (2019); Romano et al. (2017) or the application of exoskeletons Vaca Benitez et al. (2013). In our work, however, the roles are reversed, it is the robot who is responsible for autonomously establishing a physical contact. An interaction scenario where the robot itself is actively and autonomously creating contacts on a surface of a human participant body is rarely considered. In Yamada et al. (1997) a control of a robot establishing a contact on the human body was studied with consideration of safety limits in terms of the human pain tolerance limits. In López et al. (2014) a humanoid robot was used in a sit-to-stand assistance context to contact a human and perform a motion while maintaining the contact.

The human-aware navigation is a basic skill that a robot must have for operating in the same environment as humans Kruse et al. (2013). A human aware motion planner that takes safety, human comfort and social acceptability into account while planning the robot's path for HRI was proposed in Sisbot et al. (2007). An inverse reinforcement learning based navigation goal and path generation approach that also takes social norms into account was proposed in Ramírez et al. (2016). Such planners, if extended to account also for the physical interaction in the assistive context using conclusions from appropriate user studies, could be used in the controller presented in this chapter to design the path for a robot to follow when approaching the human subject prior to initiating the physical assistance process. For now, the position goal w.r.t. the human subject is defined manually in the controller presented in this chapter. The path to reach the navigation goal is the result of the visual servoing QP task error minimization. A sensor fusion approach for the human-following robot navigation was developed and tested in Tee Kit Tsun et al. (2018). Visual servoing was used to make a robot navigate towards a human, maintain constant distance between itself and a human, and follow a walking human. This visual servoing based control was presented in Claudio et al. (2016); Agravante et al. (2016). Although, this work does not include the details of the navigation method. However, a demo of this navigation approach is available online [2].

Yet, none of the previous works have considered a full autonomous interaction scenario with integration of all components of visual feedback based navigation towards a person, multi-modal communication of intent and closed-loop physical contact creation. In the work presented in this chapter, we describe the design and implementation of the whole-body QP FSM controller for the studied pHRI scenario (Sec. 4.3) and present the experimental results (Sec. 4.4).

2. https://youtu.be/QDmDY5koKIE

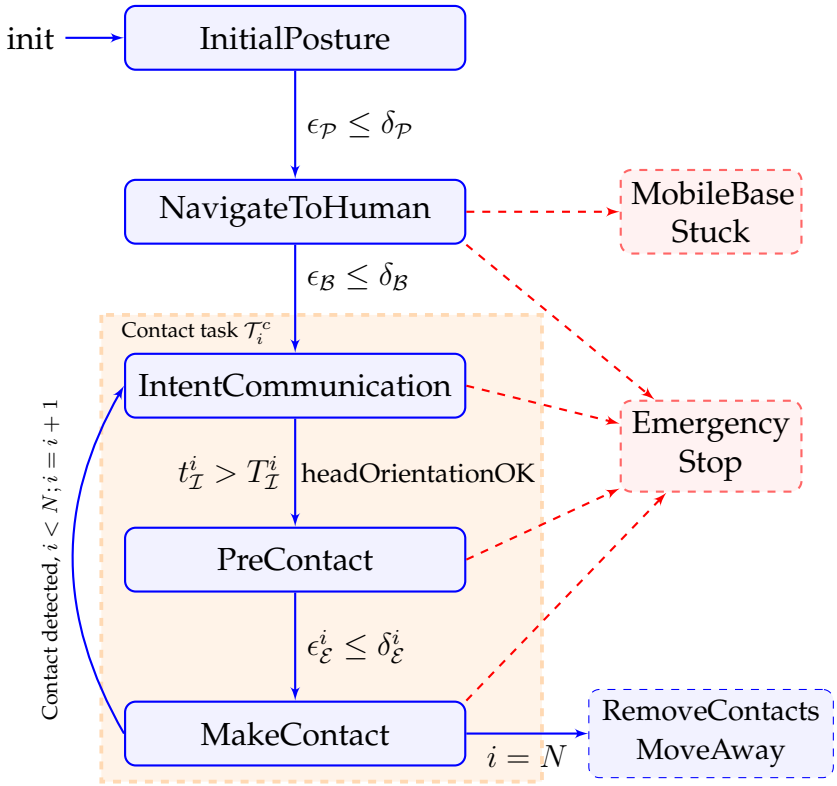


FIGURE 4.1 – *FSM based controller design for interaction scenario.*

4.3 Controller architecture

The whole-body controller for a Pepper humanoid robot is developed using the `mc_rtc` QP task-space control framework. We consider a complex interaction scenario which is comprised of several distinct stages, each of which consists in achieving multiple objectives. Therefore, the controller is implemented in a form of an FSM, where every state corresponds to a specific stage of the interaction process. Fig. 4.1 presents the general structure of the implemented FSM controller. In the following Sec. 4.3.1 we describe this controller implementation in detail.

4.3.1 FSM QP controller implementation

The robot actions are controlled by an acceleration-resolved QP controller. This means that an optimization problem is formulated with a quadratic objective function that consists of a weighted sum of *tasks* formulated as errors between actual and desired setpoints in task space, as well as first and second derivatives of these task errors. Robot joints acceleration are the decision variables of the problem. A set of linear *constraints* in the optimization problem formulation ensure that the solution is physically feasible and safe Bouyarmane et al. (2019).

As indicated in Fig. 4.1, the interaction starts with controller initialization (*init*). At this stage, the robot description module is used to build the base control problem as QP with default tasks and constraints (Eq. 4.1).

$$\min_{\ddot{q}, f} \quad \mathcal{P} + \mathcal{B} + \mathcal{C} \tag{4.1a}$$

$$\text{s. t.} \begin{cases} \text{joint position, velocity and torque limits} & (4.1b) \\ \text{self–collision avoidance} & (4.1c) \\ \text{sliding ground contact constraints} & (4.1d) \\ \text{bound mobile base velocity and acceleration} & (4.1e) \end{cases}$$

The objective function of the QP contains a default posture task $\mathcal{P}$, default mobile base position task $\mathcal{B}$ and a center of mass task $\mathcal{C}$. Interested readers can refer to Bouyarmane et al. (2019) for detailed definition of these common QP tasks and constraints.

After the controller initialization, transition to the *InitialPosture* state is triggered. At this stage, we assume that the robot is positioned far away from and facing the human participant. In this state, the default posture task $\mathcal{P}$ is given a new setpoint q_d, which is an upright standing posture. We define a threshold $\delta_{\mathcal{P}}$ and consider this state to be completed when the posture task error $\epsilon_{\mathcal{P}}$ is less than or equal to this threshold, which triggers transition to the *NavigateToHuman* state.

In the *NavigateToHuman* state, default mobile base task $\mathcal{B}$ is removed from the problem formulation and a Position Based Visual Servoing (PBVS) task $\mathcal{V}$, regulating mobile base motion, is added. The objective of this task is to minimize the error in the camera frame between mobile base position, computed from kinematics tree, and target mobile base position, which is defined w.r.t human frame detected in robot camera field of view (Fig. 4.2). We detail how this target position is computed and reached in closed-loop in Sec. 4.3.2. Termination of this state is triggered when $\mathcal{V}$ task error $\epsilon_{\mathcal{B}}$ reaches a predefined threshold $\delta_{\mathcal{B}}$.

Now, the robot executes a sequence of predefined contact tasks $\mathcal{T}_i^c, i = 0, \ldots, N$. For every contact task, the first state (of the sub-FSM) is *IntentCommunication*, where the robot explains and illustrates its further intentions specific to $\mathcal{T}_i^c$ to the human (Fig. 4.3). This state is implemented to ensure smooth and comprehensive transition to the states where the robot establishes physical contacts with the human. We detail the purpose and implementation of this state in Sec. 4.3.3. Human head orientation is monitored in this state to verify that a human has paid attention to the visual communication on the robot screen. The time for intent communication is predefined $T_{\mathcal{I}}$. Once state time $t_{\mathcal{I}}$ exceeds $T_{\mathcal{I}}$ intent is considered to be successfully communicated and transition to the *PreContact* state is triggered. After robot prepares for a contact, appropriate joint residual signal, described in Chapters 1 and 2, is monitored in the *MakeContact* state, as robot is moving its end-effector towards the human body, to detect the time of the contact event. When all contact tasks are finished the experiment ends with *RemoveContacts* state followed by the mobile base of the robot moving away.

4.3.2 Nearby surrounding navigation towards human

We define a fixed mobile base position and orientation target w.r.t human reference frame ${}^{h}X_{b}^{*}$. The pose of the human reference frame (*Pelvis* link), expressed in the robot camera frame, is obtained from human Body Tracking by Azure Kinect camera installed on the robot, we denote this transformation as ${}^{c}X_{h}$. From the joint encoder readings and known robot kinematics we compute transformation between camera frame and mobile base frame ${}^{b}X_{c}$, which at the start of the experiment is equal to the world frame

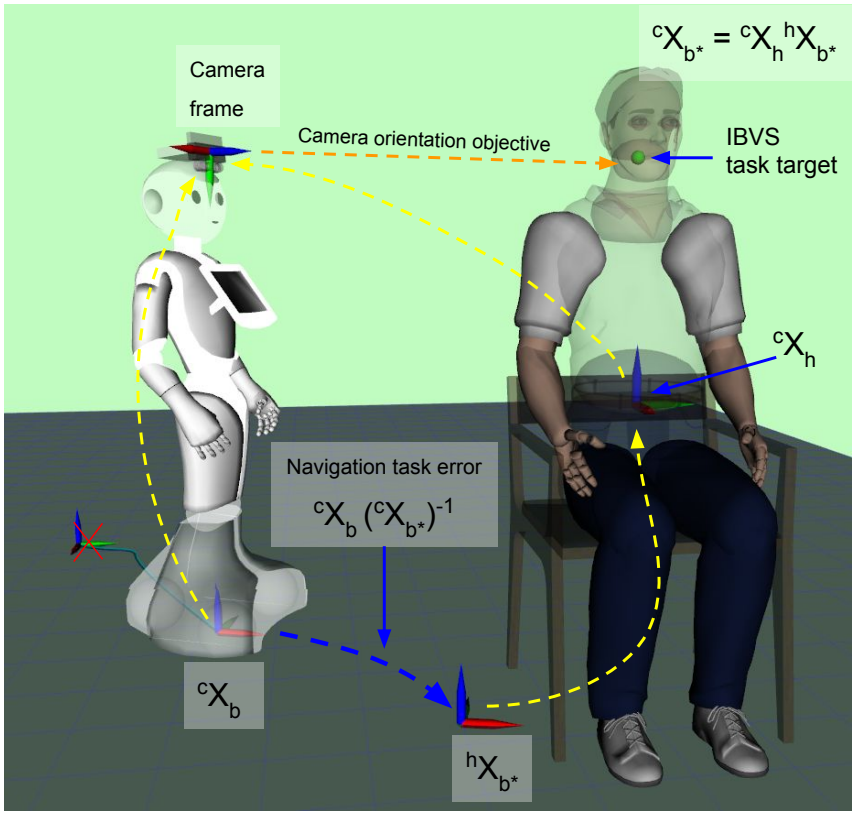


FIGURE 4.2 – NavigateToHuman *FSM state visual tasks.*

${}^{b}X_{c} = {}^{w}X_{c}$ at $t = 0$, where t is the time elapsed from the start of the experiment. The desired setpoint pose ${}^{h}X_{b}^{*}$ in the camera frame is (Eq. 4.2)

$$ {}^{c}X_{b}^{*} = {}^{c}X_{h}\,{}^{h}X_{b}^{*} \tag{4.2} $$

All frames involved in these computations are illustrated in Fig. 4.2.

This update is performed on every iteration of the controller and passed as a target to the PBVS QP task $\mathcal{V}$ associated with the motion of the mobile base. Due to the fact that a camera and the body tracking frame rate (30Hz and 15Hz respectively) is much lower than the control frame rate (83Hz in case of Pepper), new detection data is not available for every controller iteration. This may result in a sudden task error jumps and discontinuities. For a smoother PBVS task error evolution and convergence, in this work, we set lower limits for the mobile base speed and acceleration.

The error between the current and desired mobile base frame poses is used as a feedback for the PBVS task $\mathcal{V}$ to navigate in closed-loop to the desired position (Eq. 4.3)

$$ \epsilon_{\mathcal{B}} = {}^{c}X_{b}({}^{c}X_{b^*})^{-1} \tag{4.3} $$

Detected human head frame is used for a camera orientation target task implemented in the controller as an Image Based Visual Servoing (IBVS) task $\mathcal{O}$. With ${}^{\text{head}}X_{c}$ being a human head frame pose in the camera frame detected by Azure Kinect, the $\mathcal{O}$ task error which ensures that human head is kept as close as possible to the center of the field of view, (FoV) is $\epsilon_{\mathcal{O}} = {}^{\text{head}}X_{c}.\textit{translation}$. Based on our experience, keeping a human head in the center of FoV results in a better overall human body tracking results from the Azure Kinect, especially once some human body parts become occluded and are thus no longer in the FoV.

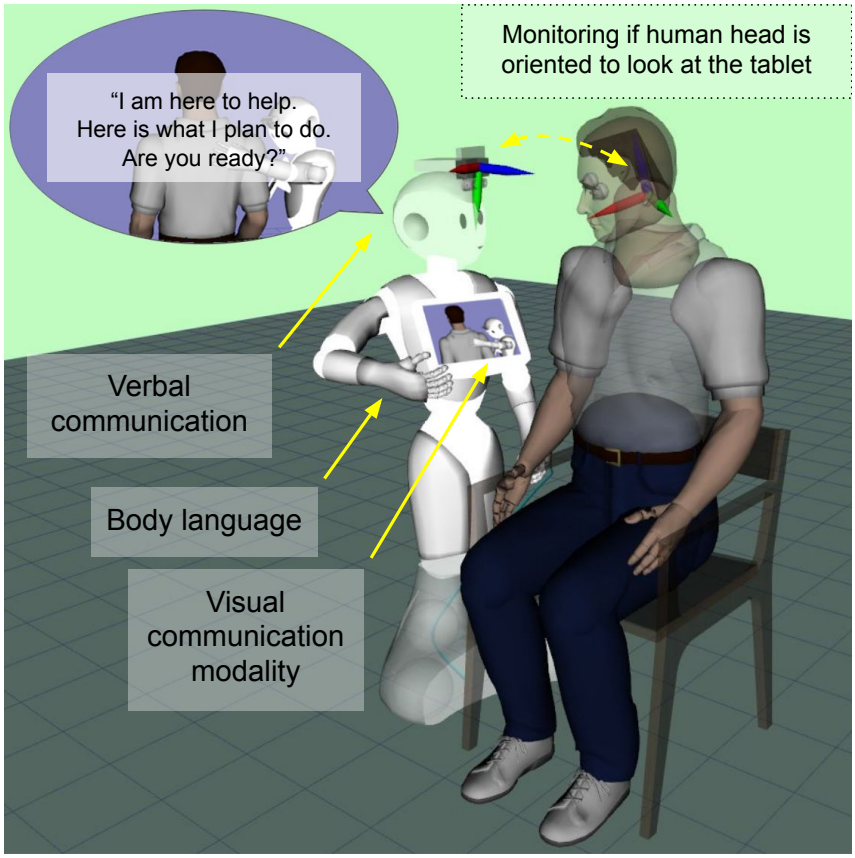


FIGURE 4.3 – *Multi-modal* IntentCommunication *FSM state.*

In the future, this method could be combined with V-SLAM [3] technology for highly robust performance. This is necessary for improved safety and for handling low frame rate and high frame latency of Azure Kinect depth sensor.

4.3.3 Intent communication for user comfort and safety

Humans usually express interest in touching a humanoid robot when they see one. However, when the roles are reversed and it is the robot that will establish physical contacts on a human body, the loss of control from a human side, the lack of understanding of the robot intentions, and the low communicative cognitive capabilities of the robot can cause discomfort and even fear during the interaction. The closer a robot moves to the person, the more considerations need to be taken into account for human comfort and safety. And in the case of direct physical contact, human participant safety and comfort (both physical and mental) are of extreme importance for successful human-robot interaction.

Indeed, a human participant needs to trust the robot to feel comfortable to allow it to establish physical contact. And this trust can be established only if the human can predict what the robot intentions are. Therefore, we integrate good user experience design considerations as parts of the FSM controller in order for the human to feel comfortable while the robot is in close proximity and is establishing physical contacts

3. We choose SLAM as it will certainly be a mature technology to navigate inside rooms, indoor senior citizens' homes, personal houses, hospitals, etc.

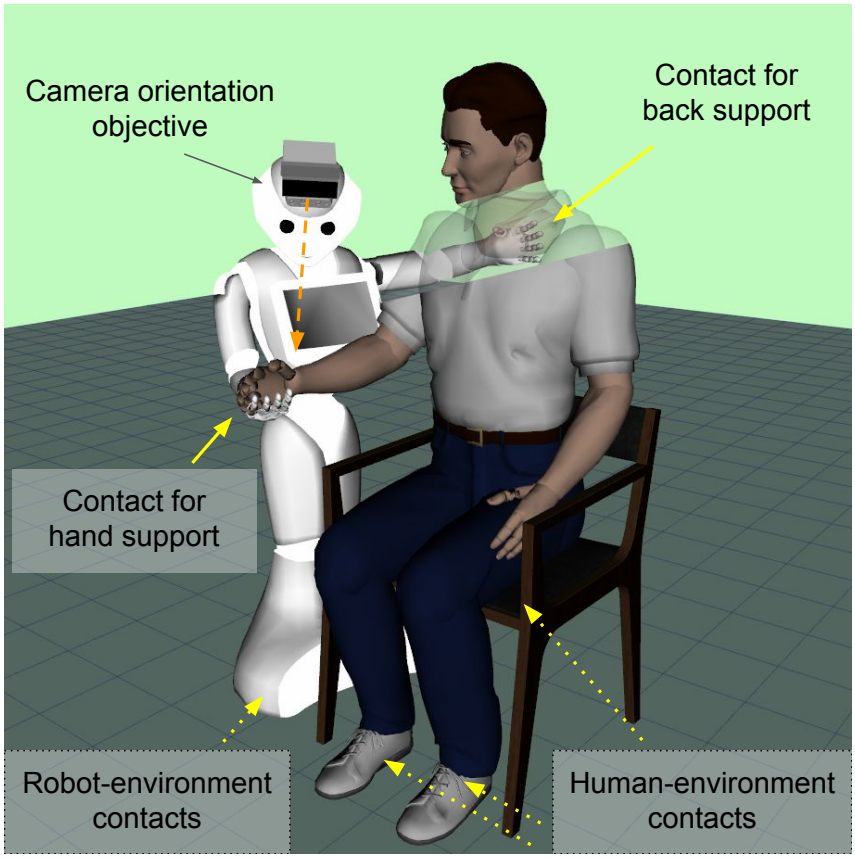


FIGURE 4.4 – MakeContact *FSM state, all contacts established.*

with the human. For that robot is programmed to clearly communicate its intentions to the user, using different communication modalities (verbal, visual, body language).

When the robot reaches the person close enough, prior to establishing physical contacts, the robot is communicating verbally what it intends to do next. At the same time, using body language the robot draws the attention of the human participant to its tablet screen where an illustration of the intended physical interaction is displayed on a schematic figure, as is shown in Fig. 4.3.

With the aforementioned tools for information transfer, that are familiar and intuitive for a human subject, we enrich the interaction with different communication modalities and enable the robot to make its intentions clear prior to establishing physical contact on a body part surface of the human subject.

4.3.4 Establishing physical contact

The physical contacts are established using posture QP tasks to move robot's right or left hand links towards the surface of the human body where the contact is supposed to be established (Fig. 4.4). Contacts are established in closed-loop via monitoring of residual signal between predicted (learned) and measured joint position tracking errors, as described in Chapters 1 and 2. Predefined contact locations on a human body in this work were inspired from observing the human caregiver's practices for similar interaction tasks. The exact position of the contacts can be adjusted to different human morphologies or chair heights. However, the feasible contact locations are limited by the robot's reachable workspace. The exact feasible placement of the contacts can

be planned by analyzing the human point cloud using the method presented in the previous Chapter 3.

4.4 Experimental Results

To test and evaluate the performance of the developed QP controller for pHRI, we have achieved preliminary experiments that could be enhanced further in the future work. [4] The autonomous initiation of physical assistance experiment is presented in Sec. 4.4.2. The software tools developed to achieve this experiment have been made publicly available and are thoroughly described in Appendix A.

4.4.1 Platform description

In the experiment described in this section, we are using the upgraded prototype of Pepper wheeled humanoid robot platform that has been customized for later in-situ experiments with real patients. With respect to the commercial version, this robot has an additional Azure Kinect camera mounted on the top of the head and used in our experiments for human body tracking. The robot is also equipped with the RealSense D434 camera for SLAM and additional source of IMU measurement that we were not using in the experiments. These additional hardware elements are added to the robot description module and used by `mc_rtc` framework to build the base controller QP formulation. Under the tablet, this upgraded prototype Pepper version has embedded Jetson TX2 that could be used for onboard image processing and other heavy computation. For the time being, we are running the controller on an external Personal Computer (PC) for easy debugging and programming purposes.

4.4.2 Results

At the start of thePersonal Computer experiment, the robot is placed about 1.2 meters away from a chair where the human participant is sitting. First, the robot navigates towards the person, then communicates its intentions, and proceeds to initiate several physical contacts. First contact robot establishes one on the right shoulder of the human participant. Then, the robot invites the participant to place their hand in the robot's right end-effector (as is being simultaneously demonstrated on the tablet screen of the robot). The experiment ends with all contacts being autonomously removed, robot thanking the participant and moving away.

Fig. 4.6 illustrates 4 main stages of the experimental controller run with the real human participant. Interested reader is invited to see the video [5] accompanying this work for the full experiment demonstration or to refer to the controller source code for implementation details [6].

4. For the time being, the experiments with more participants couldn't be achieved because of the restrictions of the Covid19 outbreak.

5. Video titled "Autonomous Initiation of Human Physical Assistance by a Humanoid" is hosted online at the IDH LIRMM YouTube channel: https://youtu.be/vDmEc1aBODA

6. https://github.com/anastasiabolotnikova/autonomous_phri_init

Plot on Fig. 4.5 shows evolution of the PBVS task errors in XY translational and Z rotational axes during the *NavigateToHuman* state (Fig. 4.6c). As can be seen from the plot, the task error evolution and convergence is mostly smooth. A slight sudden jump occurs at the end of the state for the Y axis, when the robot is very close to the human participant. Typically, at this stage the depth perception deteriorates due to the human being too close to the camera and, as a result, discontinuities in human body tracking are also likely to occur. Nevertheless, the *NavigateToHuman* state completion criteria is successfully reached and transition to the next FSM state is triggered.

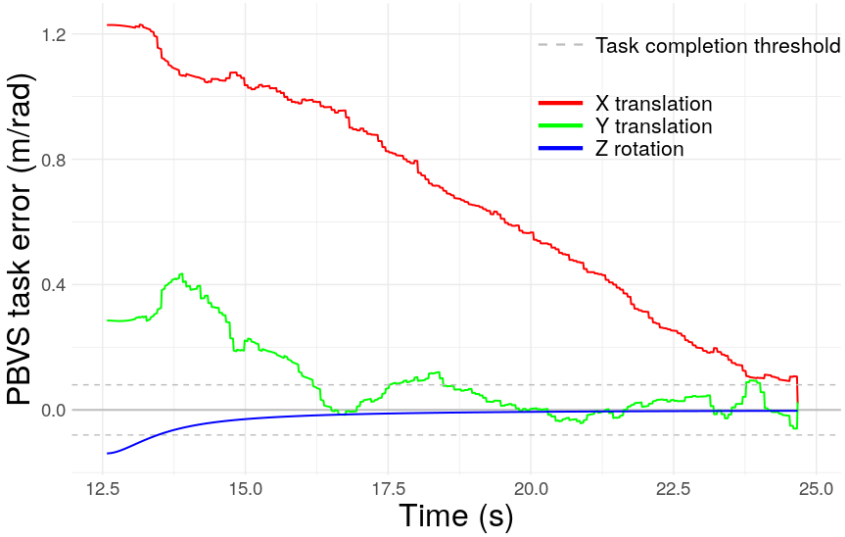
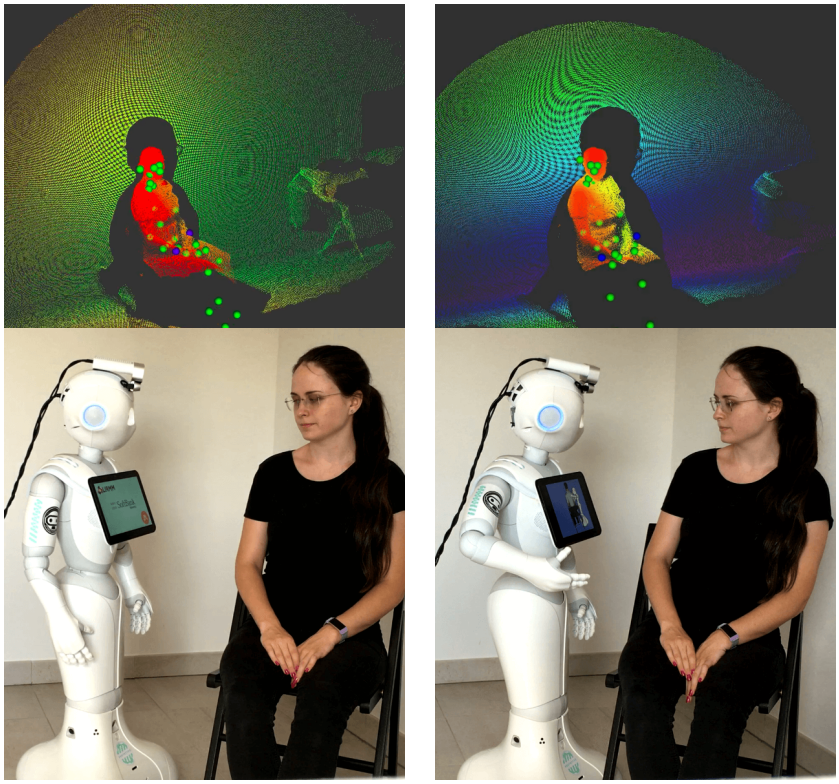


FIGURE 4.5 – *Evolution of navigation task errors.*

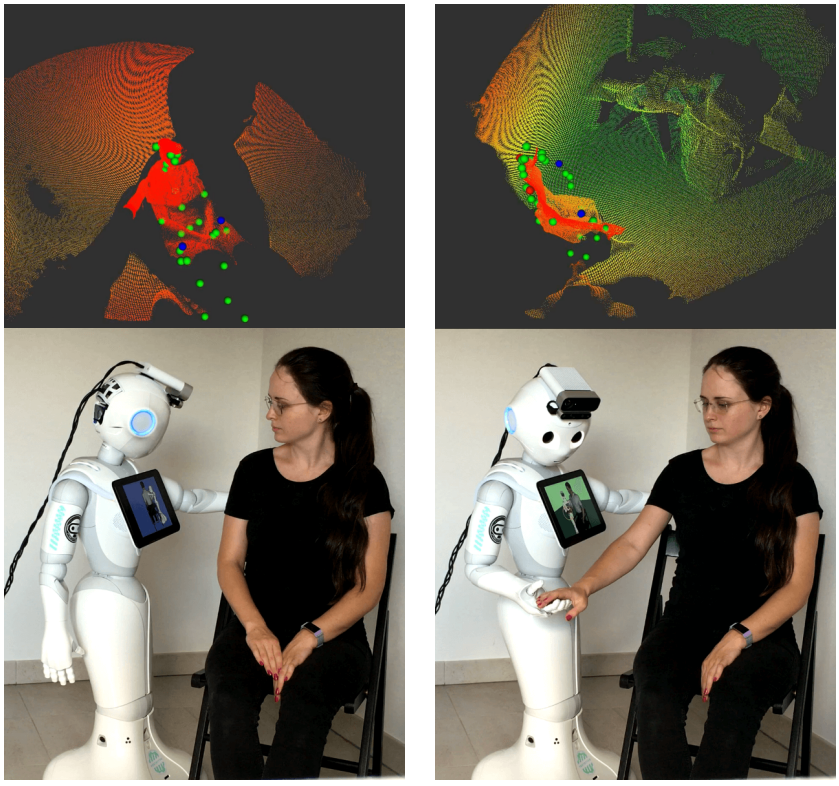
Next state in the controller FSM is *IntentCommunication*. This part of the experiment can be seen on Fig. 4.6d. While the robot is communicating its intentions, using multiple communication modalities, as described in Sec. 4.3.3, the IBVS QP task ensures that the human head is kept in the field of view. The orientation of the human head frame is monitored to verify that the human face was oriented to look at the robot's tablet at least once for paying attention to the figure displayed on the screen. This is an additional criteria for exiting *IntentCommunication* state. The plot of the monitored human head to robot tablet angle is shown on Fig. 4.7. This plot validates that, shortly after the start of *IntentCommunication* state, the human head is being oriented to look at the robot tablet.

Once robot intent is communicated, the transition to *PreContact* and then *MakeContact* states for the robot's left end-effector is triggered. The robot moves its left arm towards the human back to establish a contact on the right shoulder of the participant (Fig. 4.6g). The monitoring of the residual between measured left shoulder joint position tracking error and predicted expected position tracking error of this joint allows to detect the contact event (as mentioned in Sec. 4.3.4). Fig. 4.8 shows the residual signal for the left robot shoulder roll joint. As can be seen from the plot, the contact is detected (when residual reached a threshold) and maintained for several seconds after the detection occurs as is requested according to the *MakeContact* state design.

Similar process is repeated for the robot's right end-effector. However, in this case, the robot invites the user participant to place their hand into the robot's right end effector, which is brought up in front of the user as is shown in (Fig. 4.6h). The plot on the Fig. 4.9 shows the residual signal for the robot right elbow roll joint and indicates



(c) *Navigation* **(d)** *Intent communication*



(g) *First contact* **(h)** *Second contact*

FIGURE 4.6 – *Robot onboard camera view and video screenshots illustrating main parts of the experiment with human participant.*

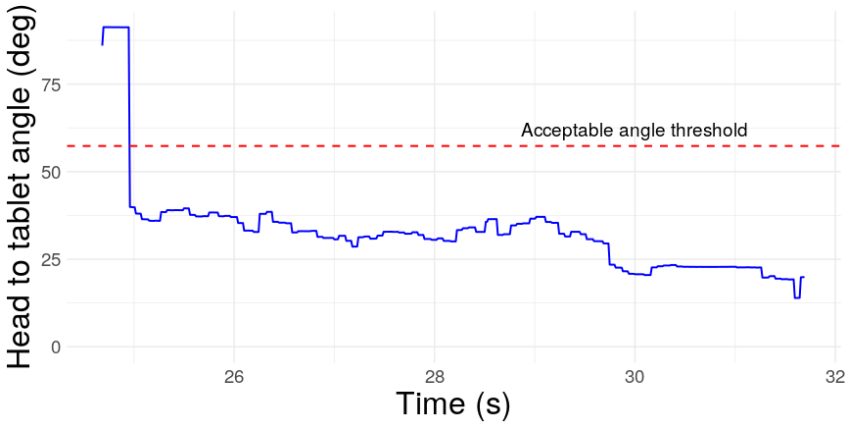


FIGURE 4.7 – *Human face to tablet angle.*

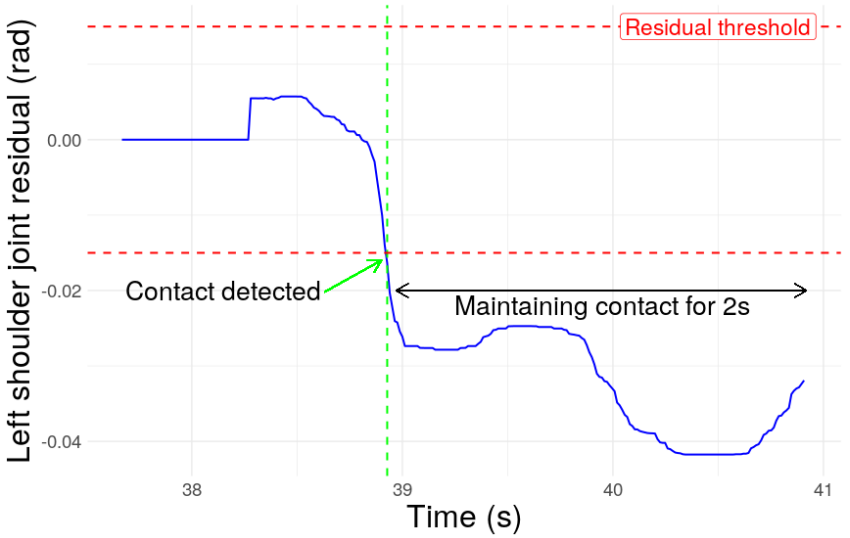


FIGURE 4.8 – *Left end-effector contact detection.*

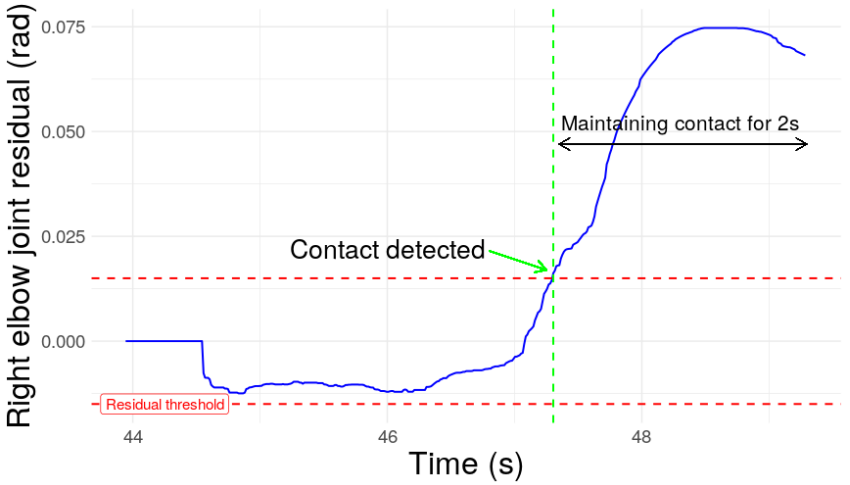


FIGURE 4.9 – *Right end-effector contact detection.*

successful detection of the contact event. Once the contact is detected, the robot closes its right gripper slightly, as a human would do in a similar interaction. This contact is also maintained for a few seconds before being removed.

The experiment ends with the robot autonomously and carefully removing all established physical contacts one by one. The robot thanks the human user for participation in the experiment and moves away.

4.5 Conclusion

In the work presented in this chapter, we integrated and enhanced different bricks to build an architecture that enables the humanoid robot Pepper to reach a person in need for physical assistance, communicate its intentions and establish several physical contacts to initiate the assistance process. We developed a task-space optimization controller instance derived from Bouyarmane et al. (2019), enhanced with visual servoing Paolillo et al. (2018), which allows reaching safely a person. In close contact interaction the robot initiates contacts to achieve compliant contact motion. The controller was enhanced with communication plug-ins.

In the next chapter, we propose an adaptive task-space force control strategy that could be used in the controller described in this chapter to regulate the robot force contribution to the human motion task while providing the physical assistance. We study the proposed methodology in simulation using a personalized human model and present the achieved results.

CHAPTER 5

ADAPTIVE HUMANOID-TO-HUMAN MOTION ASSISTANCE

We envision a humanoid robot to serve as a source of an additional force in motion assistance for frail persons. In this context, we present a control strategy for a humanoid to adaptively regulate its assistive force contribution. First, we identify a human model torque control for an optimal execution of a priori known motion task from sample recordings of this task performed by a healthy individual. We utilize the identified model in the proposed position discrepancy based observer of the human torque contribution, the unknown and unmeasurable variable. We propose an experience-based human contribution model learning strategy that allows improving the human contribution estimate from trial-to-trial. The target assistive torque contribution is then calculated as the difference between the optimal torque required for the motion task and the estimated human contribution. The target assistive torque is integrated into a multi-robot quadratic programming task-space controller to compute the desired interaction force required for the robot to supply the necessary assistive torque for the human model. We use the non-optimal recordings of the human motion task to emulate human frailty and apply our adaptive force control strategy to demonstrate the results of a humanoid successfully assisting the simulated human model to restore the optimal motion task performance.

5.1 Introduction

A promising area of the humanoid robots application is daily assistance for frail and elderly, e.g. Niemelä and Melkas (2019); Mitzner et al. (2014); Bolotnikova et al. (2020b). Humanoids could be specifically designed to be user-friendly, multi-functional and safe Pandey and Gelin (2018). These properties allow us to envision a humanoid providing companionship through social assistance Papadopoulos et al. (2020) and helping people in need to perform daily chores. Being a platform capable of physical interaction, one of the useful functionalities for humanoids would be to assist frail people with motion tasks that typically require assistance from a human caregiver. Enabling a humanoid to provide such assistance safely and efficiently can help to increase frail person autonomy.

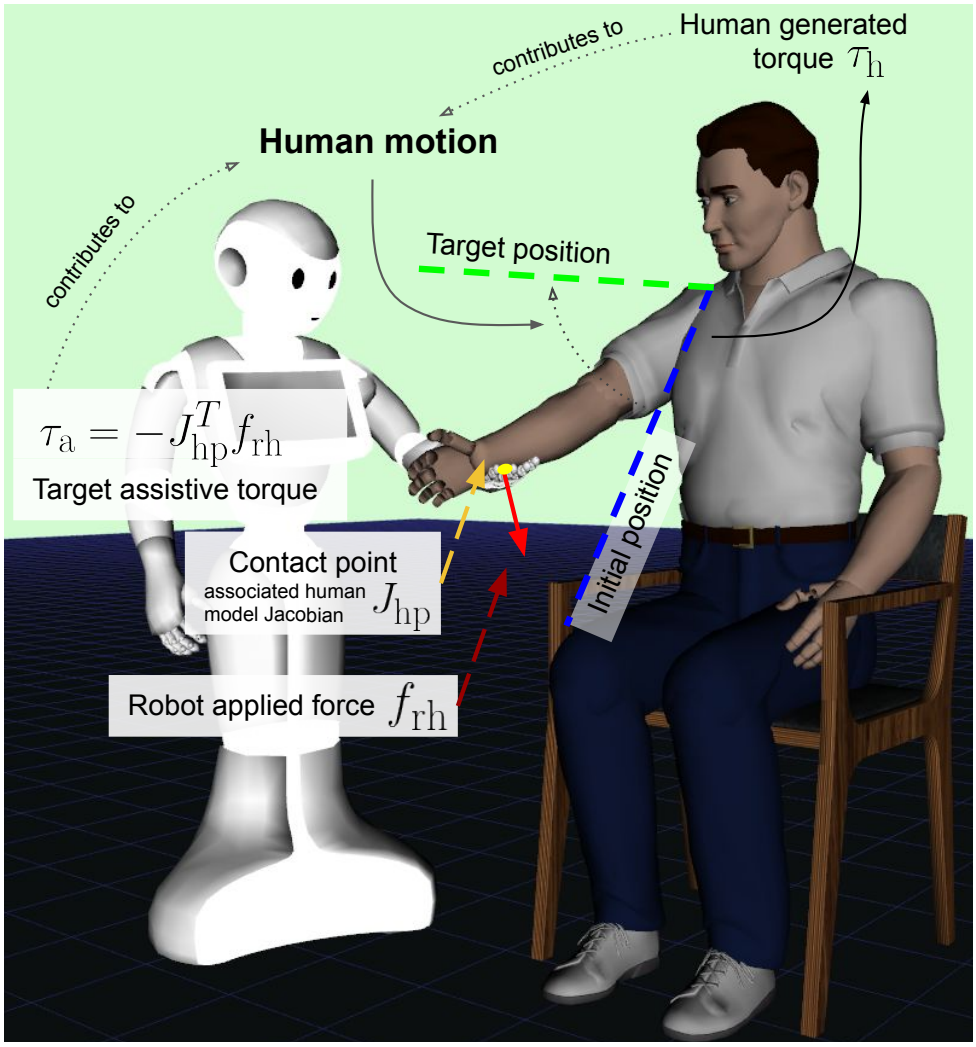


FIGURE 5.1 – *Humanoid-to-human physical assistance in motion.*

During physical assistance, two sources of force contribute to the human motion: forces that are generated by the human and the assistive forces supplied by the robot (Fig. 5.1). The control challenge in such a scenario is the fact that human force contribution is not known to the robot and cannot be directly measured. The exact intended human motion is not known either. Only if robot force contribution to the human motion task is adaptively regulated to account for the presence of another (unknown and unmeasurable) source of force, the motion can be performed correctly. Indeed, lower than required wouldn't provide the necessary assistance; more than required could engender fear, stress or deviate the motion greatly from the way it should be correctly executed, causing discomfort and potentially harm to the human.

Our contributions are as follows. First, as an extension to the multi-robot task-space quadratic programming controller (MQP) framework Bouyarmane et al. (2019), we integrate a human model as an additional 'robot'. Indeed, in the original work presenting the MQP Bouyarmane et al. (2019), this extension was left as future work that we partly address in this letter. By doing so, constraints inherent to frailty such as human reduced range of motion, muscles torques limits, absence of limb... are integrated straightforwardly to be accounted for during robot control. Second, based on this framework, we propose a control scheme to provide the strictly necessary assistive force for a priori

known human task that accounts for possible human force contribution to the task. By using knowledge on human expected motion for a given task performed without muscle strength limitations, an associate model of a human torque control is identified (Sec. 5.3.1). This model is used in a position discrepancy based human torque contribution observer (Sec. 5.3.2). An experience-based prediction strategy allows to improve a human contribution estimate from trial-to-trial (Sec. 5.3.3). The assistive torque required to support a motion task is then defined as a difference between the torque required for the correct motion execution and the estimate of the human contribution. Finally, the target assistive torque is integrated into the MQP to compute (and servo on) the desired force the robot applies to provide the assistive torque (Sec. 5.3.4). We use sample recordings of non-optimal human movement to emulate human frailty and demonstrate how our method enables humanoid assist (Sec. 5.4). To the best of the authors' knowledge, this work is the first to devise task-space optimal control –extensively used in multi-limbs redundant robots, to direct humanoid-human physical interaction to assist frail human motion.

5.2 Problem statement

For a common and well-defined human task (e.g. target reaching Todorov and Jordan (2002); Kratzer et al. (2018), sit-to-stand transfer Aissaoui and Dansereau (1999)...) human motion prediction can be obtained from theory computational models (simulation), or from sample recording of the motion executed correctly unassisted by individuals using motion capture systems. Human motion prediction is a very active research topic in itself, e.g. Kratzer et al. (2020); Corona et al. (2020). Such knowledge can be exploited and used in our context; however, it needs to be filtered in a one-run execution under the frailty constraints implemented in the MQP (e.g. human with reduced joint limits), which could induce a slightly different motion even in the availability of full muscle strength (no human torque limitation). The MQP task would implement a simple tracking of this ideal task-motion under all strict integrity-constraints. At the end, for a given task, we collect a time series of the human joint angles $q_{\mathrm{h}}^{\mathrm{task}}(t), t = 0, \ldots, T$, where T is the time when the target joint position q_{h}^{*} for the given task is reached, together with respective joint velocity $\dot{q}_{\mathrm{h}}^{\mathrm{task}}(t)$ and acceleration $\ddot{q}_{\mathrm{h}}^{\mathrm{task}}(t)$. The nominal torque $\tau^{\mathrm{task}}(t)$ required to perform the task motion can be obtained using a sample personalized model of a human body with dynamic link properties and inverse dynamics (eq. 5.1).

$$\tau^{\mathrm{task}}(t) = M_{\mathrm{h}}(q_{\mathrm{h}}^{\mathrm{task}}(t))\ddot{q}_{\mathrm{h}}^{\mathrm{task}}(t) + C_{\mathrm{h}}(q_{\mathrm{h}}^{\mathrm{task}}(t), \dot{q}_{\mathrm{h}}^{\mathrm{task}}(t)) \tag{5.1}$$

where M_{h} is a human model inertia matrix and vector C_{h} incorporates Coriolis, centrifugal and gravity terms. There are reliable methods to identify inertia parameters of healthy Herman (2007); Jovic et al. (2016) or frail Latella et al. (2019) persons.

In the physical assistance process, the total torque required to perform a task all along its motion is the sum of what human can possibly generate as torque τ_{h}, and the assistive torque τ_{a} provided by the robot by means of multi-contact physical interaction, i.e. by applying forces from robot to human f_{rh} at multiple points p (in practice, one or two) located on a human (eq. 5.2).

$$\tau^{\mathrm{task}} = \tau_{\mathrm{h}} + \tau_{\mathrm{a}} = \tau_{\mathrm{h}} + J_{\mathrm{hp}}^{T} f_{\mathrm{hr}} \tag{5.2}$$

where J_{hp} is a human body model stacked Jacobians that maps the forces ($f_{\text{hr}} = -f_{\text{rh}}$), applied at possibly multiple points p, into human model joint torques.

For a frail human $\tau_{\text{h}} < \tau^{\text{task}}$, i.e. human muscular strength is not sufficient for achieving the task motion correctly. The goal of the robotic assistance is to apply the contact forces that supplement the human generated torques such that the motion resembles as closely as possible to the expected one. Thus, roughly speaking, the amount of the required assistive torque is a difference between the total torque required to perform the task and the human generated torque (eq. 5.3).

$$\tau_{\text{a}} = \tau^{\text{task}} - \tau_{\text{h}} \tag{5.3}$$

The human muscle generated torque is not known to the robot *a priori* and cannot be directly measured. The main challenge in the assistance process is the adaptation of the robot force contribution to the unknown and unmeasurable variable τ_{h}. In this letter, we suggest that the robot contribution adaptation is achieved by observing the discrepancies between expected and measured human motion. This is because the latter can be estimated from vision, see e.g. Cao et al. (2021).

In the following we present the position discrepancy based human torque observer coupled with an experience based prediction model of τ_{h}. Then, we integrate the target assistive torque into the MQP.

5.3 Proposed method

5.3.1 Identifying reference task torque control model

Once we identify the task torque trajectory τ^{task} (eq. 5.1), the relation between task error e (as defined in Bouyarmane et al. (2019)) evolution data and the total torque required for the task to be performed can be identified. In this study, we use a neural network (NN) model for identifying this relation (eq. 5.4).

$$\tau^{\text{task}} = \text{NN}(e, \int e, \dot{e}) \tag{5.4}$$

Given the current task error state $(e, \int e, \dot{e})$, this model predicts what would be the total torque if a given human subject (represented by a personalized human model) would execute the task-motion unassisted ($\tau_{\text{a}} = 0$). Network structure in our study comprises 3 neurons in the input layer, 12 neurons in a single hidden layer and a single neuron in the output layer. Based on our experimentations with the model fitting, such a structure proved to be an optimal compromise between model complexity and prediction accuracy.

Next, we show what role τ^{task} plays in the estimation of τ_{h} and subsequent computation of the target assistive torque τ_{a}. In the Sec. 5.4, we demonstrate the use of eq. (5.4) for τ^{task} computation in the proposed control framework (Fig. 5.2).

5.3.2 Human torque contribution observer

We derive a position discrepancy based observer for estimating the human torque contribution τ_{h} to the task. At the very start of the interaction, we use an initial guess

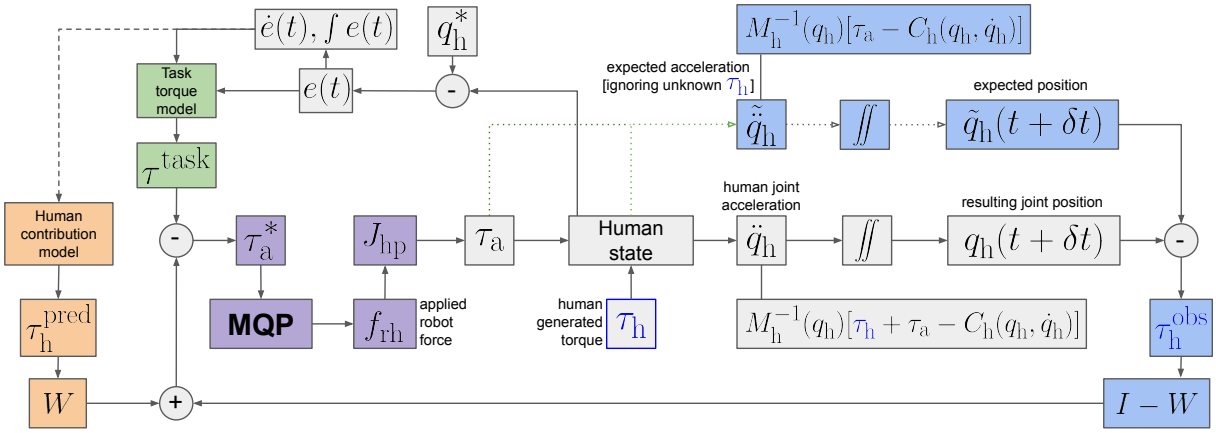


FIGURE 5.2 – *The proposed control scheme for adaptive humanoid-to-human assistance with a motion task.*

of the human torque contribution estimate, $\hat{\tau}_{\mathrm{h}}^{\mathrm{init}}$, to compute the target assistive torque $\tau_{\mathrm{a}}^{*} = \tau^{\mathrm{task}} - \hat{\tau}_{\mathrm{h}}^{\mathrm{init}}$.

As shown in Fig. 5.2, the target assistive torque is passed to the MQP block. Indeed, the MQP computes the desired robot motion together with consistent forces for the robot to apply at contact points so as to generate τ_{a}^{*} in the human model joints, through physical interaction. More details on the use of the human model in the MQP controller are provided later in Sec. 5.3.4. Such MQP based control of the humanoid results in f_{rh} forces being applied on the human body. These forces are mapped into human joint assistive torques, τ_{a}, via stacked contact Jacobian, J_{hp}, (eq. 5.2).

In order to observe yet unknown human contribution to the task τ_{h}, we assume τ_{a} to be the only joint torque acting on the human model. With this assumption in mind, we compute *expected acceleration* of the human model joints via forward dynamics (FD, eq. 5.5).

$$\tilde{\ddot{q}}_{\mathrm{h}} = M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})] \tag{5.5}$$

Then, we compute respective *expected velocity and position* for the following time step $t + \delta t$ (eq. 5.6-5.7)

$$\tilde{\dot{q}}_{\mathrm{h}}(t+\delta t) = \dot{q}_{\mathrm{h}}(t) + M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})]\delta t \tag{5.6}$$

$$\begin{aligned}\tilde{q}_{\mathrm{h}}(t+\delta t) &= q_{\mathrm{h}}(t) + \tilde{\dot{q}}_{\mathrm{h}}(t+\delta t)\delta t = \\ &= q_{\mathrm{h}}(t) + \dot{q}_{\mathrm{h}}(t)\delta t + M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})]\delta t^2\end{aligned} \tag{5.7}$$

After τ_{h} and τ_{a} are applied to the human model at time t, the resulting human joint position $q_{\mathrm{h}}(t+\delta t)$ is estimated (e.g. via any robot embedded motion tracking). This measured quantity can be expressed through integration of forward dynamics, this time taking both sources of torque, τ_{a} and τ_{h}, into account (eq. 5.8)

$$\begin{aligned}q_{\mathrm{h}}(t+\delta t) &= q_{\mathrm{h}}(t) + \dot{q}_{\mathrm{h}}(t)\delta t + \\ &+ M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{h}} + \tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})]\delta t^2\end{aligned} \tag{5.8}$$

Now, the difference between the *expected* and measured human model joints position is computed (eq. 5.9).

$$\begin{aligned}&q_{\mathrm{h}}(t+\delta t) - \tilde{q}_{\mathrm{h}}(t+\delta t) = q_{\mathrm{h}}(t) + \dot{q}_{\mathrm{h}}(t)\delta t + \\ &+ M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})]\delta t^2 - q_{\mathrm{h}}(t) - \\ &- \dot{q}_{\mathrm{h}}(t)\delta t - M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})[\tau_{\mathrm{h}} + \tau_{\mathrm{a}} - C_{\mathrm{h}}(q_{\mathrm{h}}, \dot{q}_{\mathrm{h}})]\delta t^2\end{aligned} \tag{5.9}$$

Simplifying this equation results in (eq. 5.10)

$$q_{\mathrm{h}}(t+\delta t) - \tilde{q}_{\mathrm{h}}(t+\delta t) = M_{\mathrm{h}}^{-1}(q_{\mathrm{h}})\tau_{\mathrm{h}}\delta t^2 \tag{5.10}$$

which allows us to write an expression for position discrepancy based observer of human contribution to the motion task τ_{h} (eq. 5.11)

$$\tau_{\mathrm{h}}^{\mathrm{obs}} = \frac{M_{\mathrm{h}}(q_{\mathrm{h}})[q_{\mathrm{h}}(t+\delta t) - \tilde{q}_{\mathrm{h}}(t+\delta t)]}{\delta t^2} \tag{5.11}$$

The computations involved in the τ_{h} observer are presented schematically in the Fig. 5.2.

5.3.3 Experience based human contribution prediction

The position discrepancy based observer for human contribution estimation presented in Sec. 5.3.2 only produces the estimate $\tau_{\mathrm{h}}^{\mathrm{obs}}$ after the motion is observed, i.e. after the human has actually applied its contribution τ_{h}. This results in a one time-step behind (lag) estimation. Assuming that between two consecutive time-steps human contribution does not changes significantly, i.e. $|\tau_{\mathrm{h}}(t) - \tau_{\mathrm{h}}(t-\delta t)| < \epsilon$; for small ϵ, the estimation of human torque contribution is likely to result in an overall good assistance performance.

As a strategy to compensate for the one step lag in τ_{h} observation, we propose to combine the observer with an *experience based* human contribution prediction. The idea is based on trial-to-trial learning of the human contribution. During the very first assistance trial, as there is no data to learn from yet, we fully rely on the $\tau_{\mathrm{h}}^{\mathrm{obs}}$ for computing τ_{a}^{*}. After the first trial, the experience gained –namely the observed human contribution and the task error evolution data computed during the assistance trial, can be used to learn the model for predicting human contribution Gribovskaya et al. (2011). We suggest task error evolution and assistive force contribution to be the features for learning such a model (eq. 5.12)

$$\tau_{\mathrm{h}}^{\mathrm{pred}} = f(e, \textstyle\int e, \dot{e}, \tau_a) \tag{5.12}$$

Thus, the training dataset is of the following structure: feature vector $(e(t), \int e(t)\,dt, \dot{e}(t), \tau_{\mathrm{a}}(t))$; label $(\tau_{\mathrm{h}}^{\mathrm{obs}}(t+\delta t))$.

During the next assistance trial (with the same human subject and for the same task), besides relying only on the observed human contribution $\tau_{\mathrm{h}}^{\mathrm{obs}}$, we can also make use of the ability to predict $\tau_{\mathrm{h}}^{\mathrm{pred}}$ and anticipate what the human contribution is likely to be at the upcoming time-step based on the model learned from previous assistance experiences.

The final τ_{h} estimate is then computed as a weighted sum of two terms: observer term and prediction term (eq. 5.13)

$$\hat{\tau}_{\mathrm{h}}^{\mathrm{fin}} = (I - W)\tau_{\mathrm{h}}^{\mathrm{obs}} + W\tau_{\mathrm{h}}^{\mathrm{pred}}, \tag{5.13}$$

where W is the diagonal matrix of prediction confidence weights and I is the identity matrix. W is updated online based on the evaluation of the human contribution prediction model test error. After each time-step, the difference between predicted human

contribution and the one observed will inform the system how accurate the experience based prediction model is (eq. 5.14).

$$E_{\text{test}} = |\tau_{\text{h}}^{\text{pred}}(t) - \tau_{\text{h}}^{\text{obs}}(t + \delta t)|. \tag{5.14}$$

If the test error E_{test} is large for a given human model joint, it is a sign that actual human contribution is significantly different from what was learned from previous experiences (e.g. the human is in a better shape and thus can contribute more than in the previous trials). In this case, the corresponding element on the diagonal of W is decreased. If the prediction matches closely the observed τ_h^{obs}, the W diagonal element is increased. For example, in Sec. 5.4, the W diagonal element corresponding to the shoulder joint is decreased by a fixed amount of 0.1 until it reaches the value of 0.0. For increasing this element of W the value of 0.01 is used until the value of the weight reaches 0.9. With such a strategy, the preference is given to the observer rather than the predictor for the computation of the final estimate of the human torque contribution to the motion task.

After every assistance, the data gathered during the process can be added to the training set and used to retrain and improve the experience based human contribution prediction model. If human performance does not improve or degrade significantly from one trial to another, then with every new assistance trial such a model becomes an increasingly more reliable source of human contribution estimation.

A supplementary benefit of iteratively and continuously training an experience based human contribution prediction model, is the ability to evaluate frailty performance during the assistance trial and report the progress compared to previous assistance experience. If human performance does improve or degrade significantly, this change in human performance can be detected by monitoring the magnitude and sign of the prediction error E_{test} recorded during the assistance process. For instance, if the observed human contribution is systematically higher than predicted one, it can be detected using the proposed system and subsequently reported to medical checks.

5.3.4 Force control for human assistance via MQP

In the previous sections, we explained the strategies for computing τ^{task} and estimating τ_{h}. With these quantities, we can compute the target amount of *assistive torque* (eq. 5.15).

$$\text{if } |\hat{\tau}_{\text{h}}^{\text{fin}}| < |\tau_{\text{task}}|\text{: } \tau_{\text{a}}^{*} = \tau^{\text{task}} - \hat{\tau}_{\text{h}}^{\text{fin}};\ \text{else: } \tau_{\text{a}}^{*} = 0 \tag{5.15}$$

Here, we explain how this value is used in the MQP to compute the amount of required interaction contact force for humanoid-to-human physical assistance.

Both humanoid and personalized human models are included in a single MQP formulation with all related typical MQP constraints and objectives (eq. 5.16), see details in Bouyarmane et al. (2019),

$$\min_{\ddot{\boldsymbol{q}}, \boldsymbol{f}} \quad \mathcal{P}_{\text{r}} + \mathcal{O}_{\text{r}} + \mathcal{M}_{\text{r}} + \mathcal{C}_{\text{r}} + \mathcal{P}_{\text{h}} \tag{5.16a}$$

$$\text{s. t.} \begin{cases} \text{robot joint position/velocity/torque bounds} \\ \text{robot (self-)collision avoidance} \\ \text{robot fixed environment contacts} \\ \text{robot non-sliding} \\ + \\ \text{human-frailty joint/velocity/torque bounds} \\ \text{human (self-)collision avoidance} \\ \text{human fixed environment contacts} \\ \text{human no-sliding} \\ \text{human-robot collision avoidance} \\ \text{robot-human assistive contacts (eq. 5.18)} \end{cases} \tag{5.16b}$$

where $\mathcal{P}_\text{r}$ and $\mathcal{P}_\text{h}$ are robot and human model posture tasks respectively, $\mathcal{O}_\text{r}$ is robot head orientation task, $\mathcal{M}_\text{r}$ is robot mobile base position task, $\mathcal{C}_\text{r}$ is robot CoM task, $\ddot{\boldsymbol{q}}$ and $\boldsymbol{f}$ are MQP decision variables accelerations of the joints of the models and interaction contact forces. A set of contact constraints between two models is defined. The contact point locations are planned w.r.t to the assistance task as in Bolotnikova et al. (2020b).

The feasibility of the physical interaction is ensured by including a combined robot-human equation of motion as a dynamics constraint in MQP, where robot-human interaction forces F_rh are part of the QP decision variables $\boldsymbol{f}$ (eq. 5.17)

$$\begin{aligned} \begin{pmatrix} M_\text{h}(q_\text{h}) & 0 \\ 0 & M_\text{r}(q_\text{r}) \end{pmatrix} \ddot{q} + \begin{pmatrix} C_\text{h}(q_\text{h}, \dot{q}_\text{h}) & 0 \\ 0 & C_\text{r}(q_\text{r}, \dot{q}_\text{r}) \end{pmatrix} = \\ = S\tau + \begin{pmatrix} J_\text{eh}^T & 0 \\ 0 & J_\text{er}^T \end{pmatrix} F_\text{e} + \begin{pmatrix} J_\text{rh}^T & 0 \\ 0 & -J_\text{hr}^T \end{pmatrix} F_\text{rh} \end{aligned} \tag{5.17}$$

Here $q = (q_\text{h}, q_\text{r}), \tau = (\tau_\text{h}, \tau_\text{r})$, where q_r is the vector of robot joint positions and τ_r is the vector of robot joint torques. $F_\text{e} = (F_\text{eh}, F_\text{er})$, and $F_\text{eh}, F_\text{rh}, F_\text{er}$ are stacked vectors of environment-human, robot-human and environment-robot exerted forces respectively. For instance, for m environment-human contacts $F_\text{eh} \in \mathbb{R}^{6m}$, with corresponding Jacobians stacked into $J_\text{eh} \in \mathbb{R}^{6m \times d_\text{h}}$, where d_h is number of DoF in the human body model. The diagonal selection matrix S indicates actuated DoFs, i.e. selection matrix diagonal elements are 1 for the actuated joints and 0 for the DoFs representing the floating bases or none-actuated joints of both humanoid robot and a human model.

Each contact between the humanoid and the human robot is represented in the MQP by the following constraint,

$$J_\text{rp}\dot{q}_\text{r} = J_\text{hp}\dot{q}_\text{h}, \tag{5.18}$$

where J_rp in a robot model link body Jacobian at the contact point p and J_hp is a corresponding body Jacobian at a contact point on a human model, see Bouyarmane et al. (2019) for more details.

Having τ_a^*, we know what assistive torque the robot needs to generate, which can be incorporated into MQP as a constraint $\tau_\text{a} = -J_\text{hp}^T f_\text{rh}$. Yet, considering many other constraints, it might be unfeasible for a robot to fulfill a strict equality constraint. In

such case, the target assistive torque could rather be incorporated into MPQ objective function as a task (eq. 5.19)

$$||\tau_a + J_{hp}^T f_{rh}|| \tag{5.19}$$

The amount of assistive force is computed by the MQP solver along with desired robot motion that fulfills all MQP constraints. In order for a position controlled robot to realize the desired assistive force f_{rh}, an admittance task is added into the MQP. This task takes the difference between desired force and actual one (sensor readings or contact force observer) and outputs the desired end-effector velocity to minimize this difference, see Bouyarmane et al. (2019).

5.4 Experimental results

5.4.1 Data description

Due to the pandemic situation, the hardware limitations of Pepper and other practical legal issues (ethical procedures, etc.) it is not yet possible to achieve experiments with actual frail patients. In order to assess our approach, we found real patient data that can gather both normal and deficient comparative motions for a given set of simple tasks. We borrow a data set [1] from a rehabilitation exercise Vakanski et al. (2018) and adapt it to our robotic assistance case-study. The data gathers recordings of both optimal tasks execution (that we considered to be our reference motion with full muscle strength) as well as non-optimal motions for similar tasks in case of muscle deficiency (that we consider to emulate human frailty). Note that the reduced version of the dataset Liao et al. (2020) is used in our work.

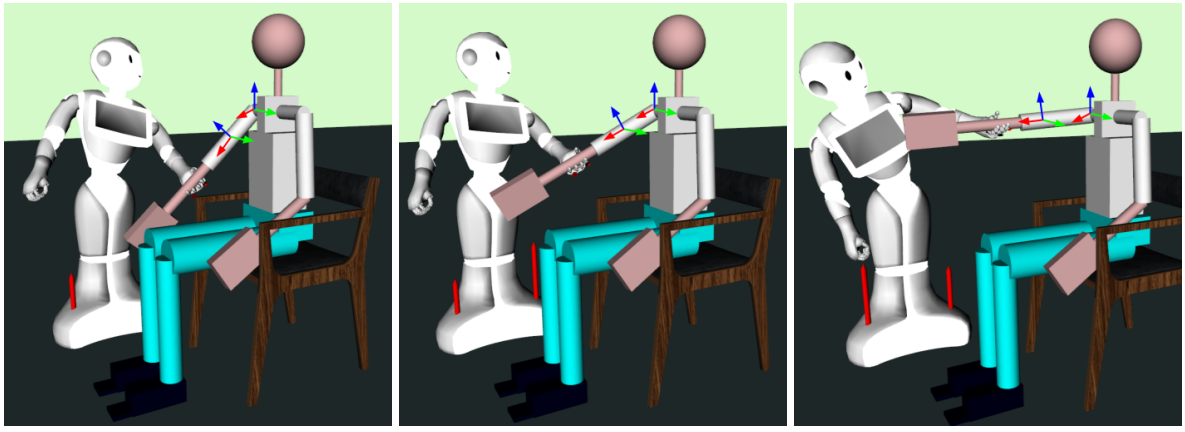


FIGURE 5.3 – *MQP controller scenes of the interaction wrench and robot motion computation for assistance under pHRI constraints.*

We validate our control scheme in simulation (one run) using the human motion recordings of the shoulder scaption rehabilitation exercise. This case is chosen for its achievability by Pepper. This task consists for a human to raise one arm in front of the chest until reaching the shoulders height, while other joints remain static, see Figs. 5.1 and 5.3. The data set contains 54 repetitions of the shoulder scaption task performed by 6 different subjects (9 repetitions per each subject).

1. https://webpages.uidaho.edu/ui-prmd/

From the whole-body motion recordings, we extract the right shoulder joint position around the Y-axis (green arrow in Fig. 5.3); it is the primary joint involved in the scaption task. The other joints of the human model are kept at a fixed position in our study, so the assistance is supplied primarily to the shoulder Y-axis joint. Moreover, the human model is configured to be in a sitting posture to enable Pepper to reach the right forearm and maintain the contact as the human arm is moving upwards during the scaption task.

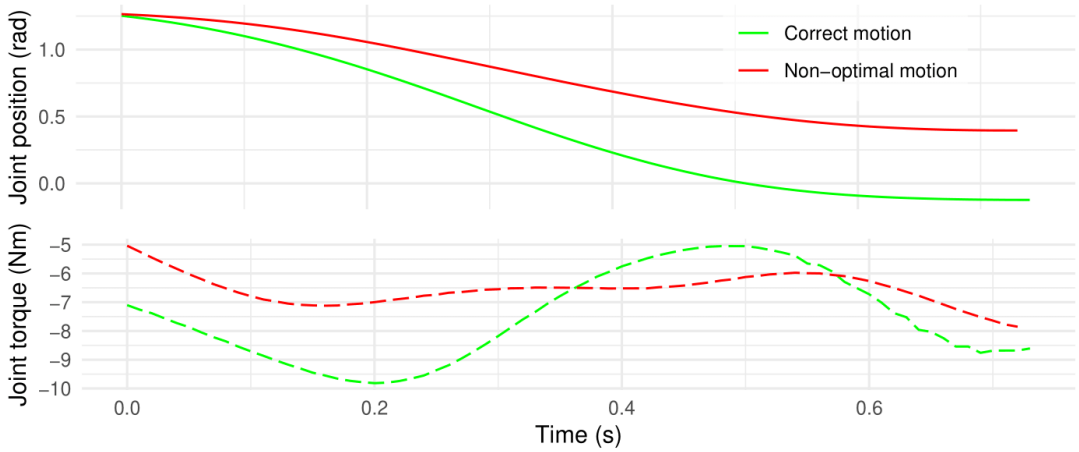


FIGURE 5.4 – *The correctly performed shoulder scaption (green), non-optimal performance of the same task (red). Position and torque plots for the right shoulder joint around Y-axis.*

The top plot in Fig. 5.4 shows that, compared to the correctly performed task, non-optimal (that we consider frail) human motion recording failed to reach the shoulders height level. The bottom plot shows that the range of torques for the frail motion recording is indeed narrower compared to the correct one. Therefore, this sample data is suitable for our study and emulates well the lack of human joint torque (frailty) to achieve the desired scaption task.

5.4.2 Computing torques required for the task

We use a single sample correctly executed scaption task motion recording to identify the task torque control model as described in Sec. 5.3.1. The Fig. 5.5 shows the performance of the NN model in computing task torque from the information of the task error evolution.

Using the NN model, it is hence possible to compute a task torque that matches closely the reference torque computed from the correct motion recording sample and a personalized human model.

5.4.3 Estimated human contribution

Now we have in hands the model to compute the τ^{task}. We assess our proposed human contribution observer $\tau_{\text{h}}^{\text{obs}}$. At this stage, we consider that no previous assistance trials took place, therefore no human contribution models exist yet. That is to say: $\tau_{\text{h}}^{\text{pred}} = \text{NA}; W = \mathbf{0}; \tau_{\text{h}}^{\text{fin}} = \tau_{\text{h}}^{\text{obs}}$). Fig. 5.6 shows the observed and true value (unknown to the controller) of the human torque contribution.

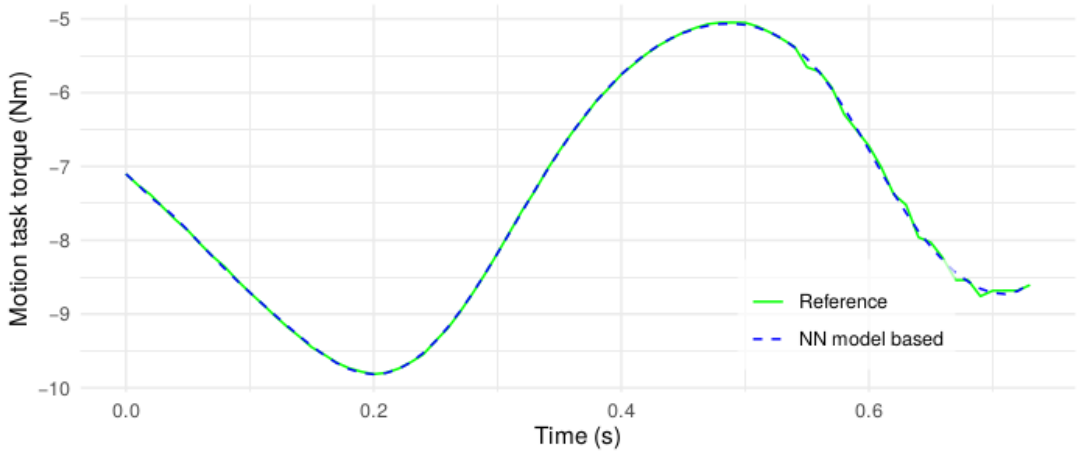


FIGURE 5.5 – *Identified model of the task torque.*

Assuming that the position and velocity of the human model joints $(q_{\mathrm{h}}(0), \dot{q}_{\mathrm{h}}(0))$ are measured before the start of the assistance process, the model based value is used as the initial guess of human contribution estimate (eq. 5.20).

$$\hat{\tau}_{\mathrm{h}}^{\mathrm{init}} = C_{\mathrm{h}}(q_{\mathrm{h}}(0), \dot{q}_{\mathrm{h}}(0)) \tag{5.20}$$

The observed human contribution data, collected in the previous (first) assistance trial, excluding the initial guess, is used to train the experience based human contribution prediction model as described in Sec. 5.3.3. Starting from the second assistance trial, this model is used in combination with the observer to improve the final human torque contribution estimate. The Fig. 5.6 shows that the initial guess of the human contribution is improved by 1.76 Nm using the experience based prediction model, reducing the estimation error by nearly 10 folds from 1.96 Nm to 0.2 Nm. This plot also

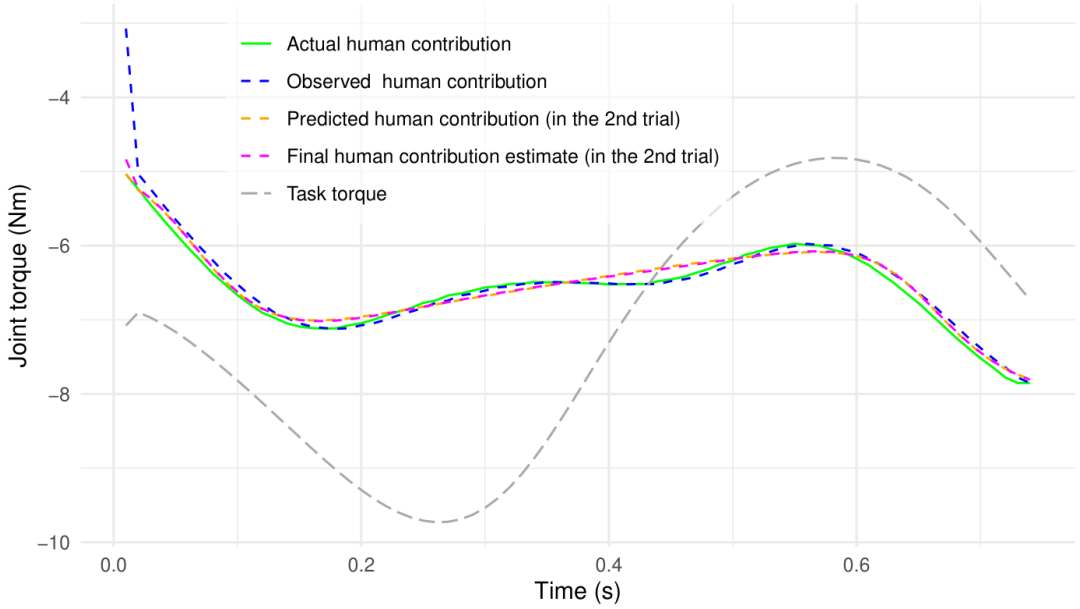


FIGURE 5.6 – *Estimated human contribution.*

shows that the human torque contribution starts exceeding the task torque after about 0.44 sec of the assistance process. After this point, according to eq. 5.15, the human

contribution to the motion alone is considered to be sufficient to achieve the desired task, the robot is thus moving along with the human, but is not required to generate any assistive force.

5.4.4 Assisted motion

Having both τ^{task} and τ_h^{fin}, the assistive torque τ_{a} can be computed (eq. 5.3). As described in Sec. 5.3.4, the human model state and the required amount of assistive torque are integrated into the MQP controller. The latter computes the interaction forces and robot motion to perform the assistance process while satisfying the human model-, robot- and contact constraints and minimizing the MQP tasks erros in the objective function (eq. 5.16).

The MQP computed robot-to-human assistive force for the first trial (with no prediction of the human contribution) is shown in Fig. 5.7. The same computation results for the second trial (this time, with prediction of the human contribution) are shown in Fig. 5.8. The MQP controller scenes during the force computation process at different times of the scaption task are shown in Fig. 5.3. In Fig. 5.6, it is shown how the use of the predictive model helps to reduce the estimation error for the initial guess of the human contribution by nearly 10 folds. This in turn results in lower τ_{a} being computed at the very start of the interaction, and consequently lower interaction forces being computed by MQP (Fig. 5.8). This results in lower (and closer to the reference motion) human joint acceleration at the start of the assistance process (Fig. 5.10).

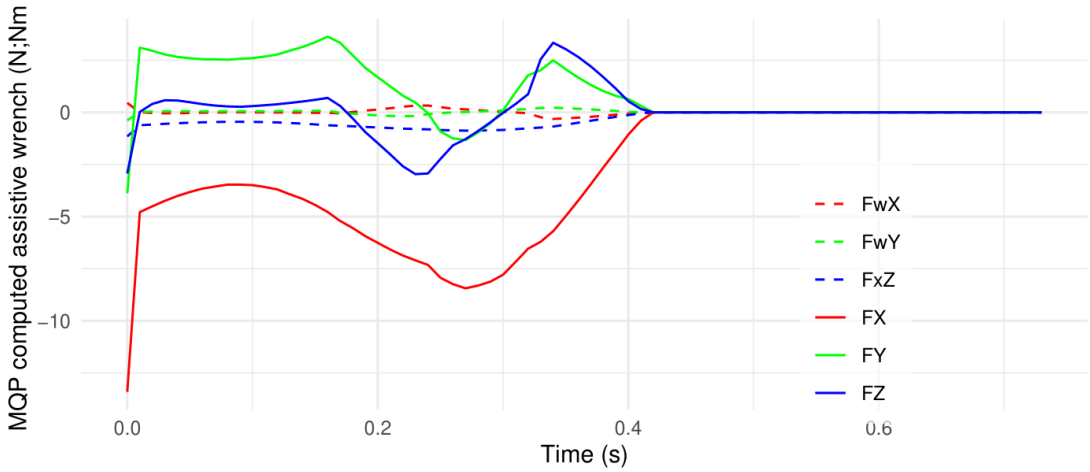


FIGURE 5.7 – *MQP computed assistive wrench.*

The MQP computed force is applied to the human model forearm link in `PyBullet` Coumans and Bai (2019) simulation along with the simulated insufficient human torque contribution, the result is the improved motion performance shown in Figs. 5.9 and 5.10. These plots demonstrate that supplied assistive forces help to achieve the scaption task motion that is closer to the correct full muscle strength reference motion.

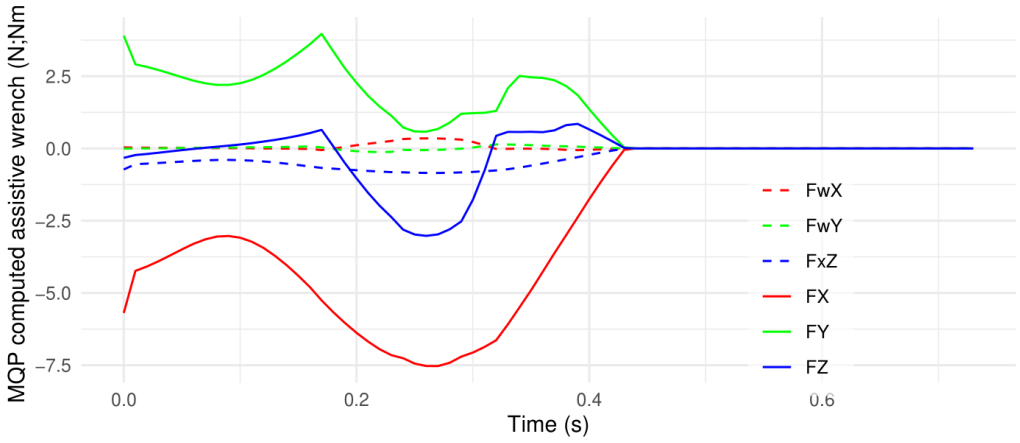


FIGURE 5.8 – *MQP assistive wrench with experience based human contribution prediction.*

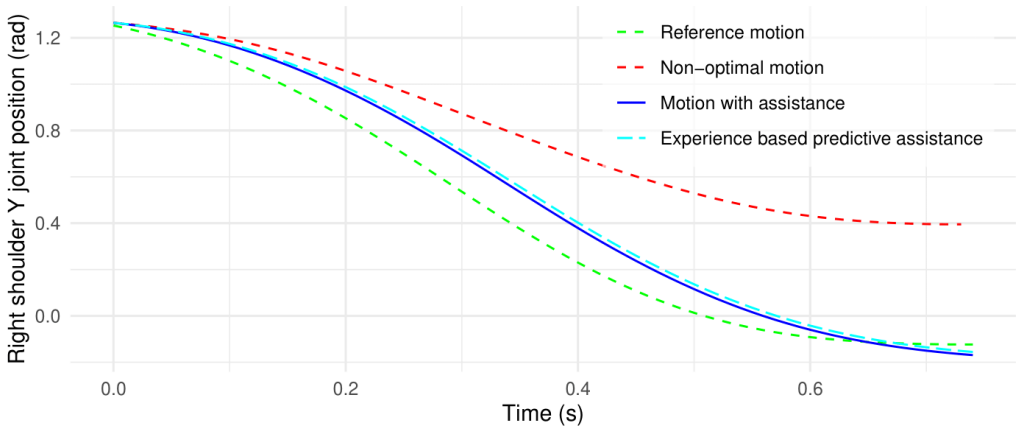


FIGURE 5.9 – *Joint position during correct motion, non-assisted non-optimal motion and non-optimal assisted motion.*

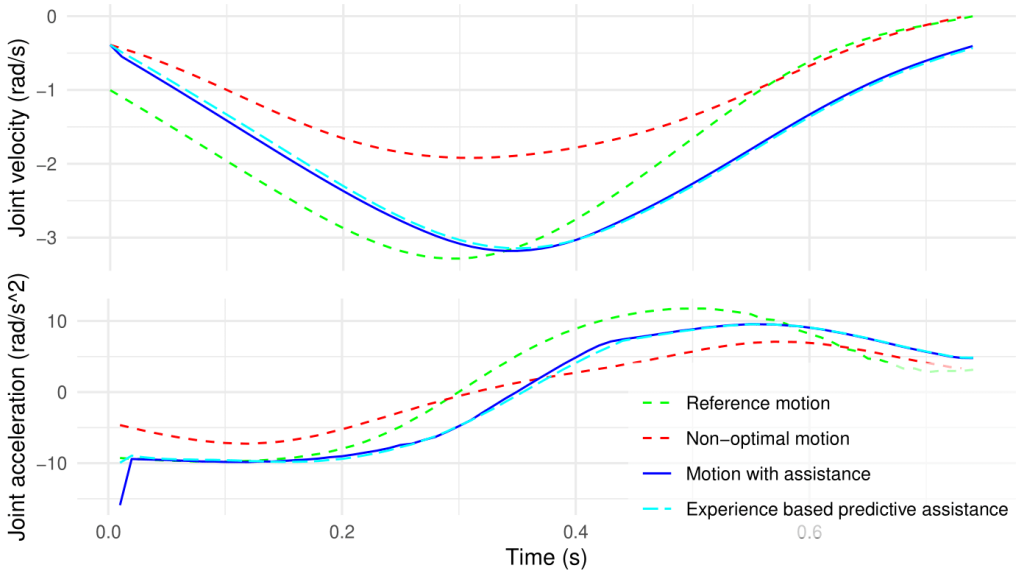


FIGURE 5.10 – *Joint velocity and acceleration during correct motion, non-assisted non-optimal motion and non-optimal assisted motion.*

5.4.5 Multi-contact assistance for multi-joint motion

To provide humanoid-to-human assistance in motion for multiple joints, several contacts can be established with a human model. For each contact an assistance requirement is defined as a constraint or a task on the contact force as described in Sec. 5.3.4.

Such a multi-contact assistive scenario is represented here by the following example. A contact between Pepper robot's left end-effector and human model upper arm is established to supply the required assistive torque at the human model shoulder joint, same as in the simulated experiment presented in the previous subsections. An additional contact, between the robot's right end-effector and the human model hand, is established to generate assistive torque at the human model elbow joint to support elbow bending motion (Fig. 5.11). The τ_{task} for the bending of the elbow is computed from simulating a human model performing the required motion unassisted. The human model frailty is then simulated by setting $\tau_a = 0.2\tau_{\text{task}}$ for the elbow joint.

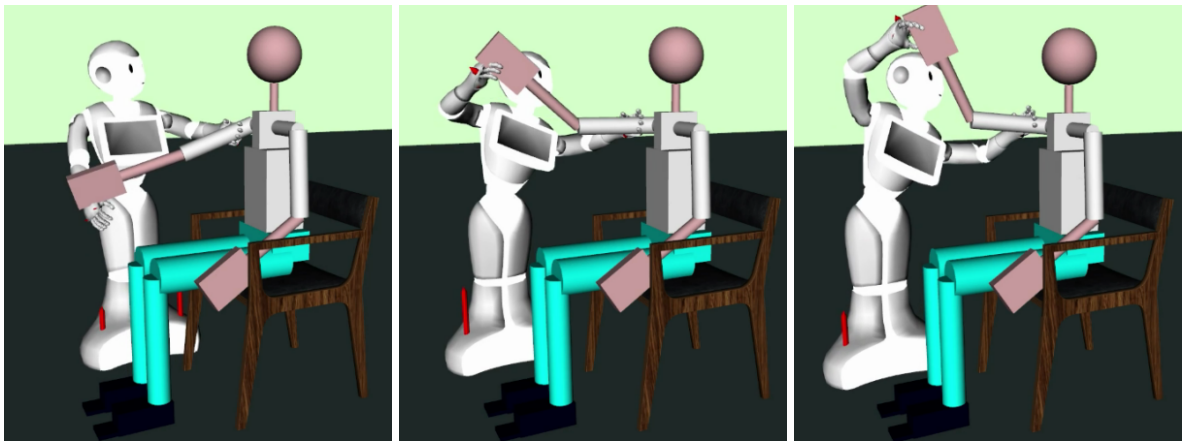


FIGURE 5.11 – *Humanoid-to-human assistance with two contacts.*

5.4.6 Discussion, limitations, and future work

Our proposed method and presented simulation results are based on full knowledge of the frail human model kinematics and dynamic parameters. The human state, namely position and velocity, is also assumed measurable at every time-step. Finally, in the presentation of the simulation results, it is assumed that the robot can apply the desired assistive forces f_{rh}, as computed by the MQP, to generate the required assistive torque τ_{a}.

Estimating human model inertia parameters might require an instrumented apparatus like in Jovic et al. (2016); Latella et al. (2019) or not, like in Pavol et al. (2002) for elderly. As the human torque contribution observer and the MQP parts of our method rely on human model knowledge, their performance correlates with the accuracy of such estimation techniques. Yet, the MQP is yet a single QP that operates in closed-loop and also in the task-space, which makes it tolerate uncertainties in such parameters Bouyarmane and Kheddar (2018). It is more critical to be very precise in the frailty parameters in terms of human joint range and muscle strength limitations (that can be set in a conservative way) than in the exact inertia parameters identification. Other human external contact forces are computed from the MQP, being a decision variable.

Yet, they can be guided from knowledge of the human dynamic parameters estimation and the knowledge of joint accelerations as in Pham et al. (2018). We assume that relying on frail patients physiological data knowledge is feasible because (i) collecting such data is critical in many applications and businesses driven by the Silver Economy boost; and (ii) health and strength conditions of frail patients are monitored through frequent medical checks. Such data can feed robotics and embedded AI algorithms.

An alternative pathway is to make the proposed framework robust to less critical uncertainties by means of domain randomization technique Tobin et al. (2017); Tremblay et al. (2018). The proposed method could be extended with an iterative reinforcement learning of the optimal assistive strategy while different variations of the human model are being tested against the method to help with better transfer of the efficient assistive strategy from simulation to real experiments.

In order for the position controlled Pepper robot to realize the assistive forces, a closed-loop admittance task based system needs to be implemented as part of the interaction MQP controller. This would require a real-time force sensing solution to be implemented on the robot, either as an additional sensor, or preferably as a proprioceptive sensor-based estimator Buondonno and De Luca (2016); Birjandi et al. (2020b).

Last, once the method is safely transferred from simulation to real experiments, user studies must be conducted to evaluate how the proposed robotic assistance is perceived by the users and answer such important questions as: (i) does the interaction process feel safe and intuitive?, (ii) does the assistance provided by the robot feel useful?, etc. Additionally, the benefits of using a humanoid robot technology in this context can be questioned. Maybe simpler robots specifically designed for a given task is a better solution Li et al. (2016); Lee et al. (2018)? Nevertheless, the use of the humanoid technology allows developers to provide a more user-friendly pHRI experience through the use of additional Pepper humanoid features, such as verbal, visual and body language communication Bolotnikova et al. (2020a). Besides providing physical assistance, a more interactive robot with a humanoid form can do more domestic tasks and provide encouragement via communication channels familiar to the human users Li et al. (2010); Torta et al. (2014). Although the base mobility plays an important role in repositioning, the lack of critical degrees of freedom in the Pepper wrists (one rotation only) is what generated the awkward postures seen in Fig 5.3. Because of this lack of dexterity, closed-kinematic chain subsequent to both arms contact manipulation of a human (arm) has very limited robot motions as can be seen in Fig 5.11. Therefore, dexterity and redundancy are important to consider in future versions of Pepper together with grasp/arm strength, to hold human limbs during assistance, and a less bulky mobile-base to avoid colliding bed and chairs in the vicinity of the human.

5.5 Conclusion

In this letter, we studied humanoid-to-human physical assistance for known task-motions. An adaptive force control framework is proposed for a humanoid to supply the required assistance. The proposed method consists of several interacting components. The first one is the model for computing human joint torque required to achieve a desired task; it is trained on a sample motion data. The second component, is the observer of the actual and potentially insufficient human contribution to the task. The

third component is the experience based human task contribution model training. The final component, includes frailty constraints and consists in the multi-robot whole-body task-space control that computes both the robot motion and the amount of assistive force to apply on the human body to generate the required amount of the assistive torque. We exemplify and discuss the performance of the proposed method on a sample humanoid-to-human assistance using the data of a human subject performing an exercise in a non-optimal way. Limitations and future venues of research and development are also thoroughly discussed.

CONCLUSION

Summary

In the frame of this thesis, we have studied several important aspects relevant to the frail human assistance by a domotic humanoid robot.

First, we focused on a topic of proprioceptive sensor based contact sensing methodology development, described in Chapter 1. We have thoroughly described the reasoning behind the proposed methodology. Several preliminary experiments for contact detection were presented.

Then, in Chapter 2, we have described in detail how a real-time compliant motion control was achieved using the robustified version of the contact observer methodology. The pHRI experiments were presented showcasing the use of the proposed and implemented control methodology for regulating the compliant robot motion.

An extension of the PG framework that allows to plan a contact location on a trimmed parametric NURBS surface fitted to the segmented human body part point cloud was presented in Chapter 3. We have detailed the human point cloud processing pipeline that supplies the input to the extended PG framework for defining contact location planning and human-robot collision avoidance constraints. Several pHRI scenarios were presented to demonstrate the results of the proposed whole-body posture and contact location planning method.

We have implemented a QP controller for a full pHRI interaction scenario to enable a humanoid robot to initiate the process of assistance in motion. We integrated in a single FSM controller several important functionality bricks for such a scenario, namely closed-loop navigation to the human, multi-modal intent communication, establishment of several physical contacts. A sample experiment with a healthy human subject was presented to demonstrate the performance of the developed controller.

Finally, the last part of the thesis, presented in Chapter 4, was dedicated to the development of the adaptive task-space force control methodology for partial humanoid-to-human assistance with a known motion task. The proposed methodology was described in detail and simulation results were presented for a sample assistance with a rehabilitation exercise.

The software tools developed over the course of this thesis were made publicly available to facilitate future developments of the task-space QP based controllers for interested users of the SoftBank Robotics Europe humanoid robots. The detailed description of these software tools is presented in Appendix A.

Future perspectives

From the experience gained over the course of this research work, we can outline several significant future research axes that, in our opinion, require attention of the research community and efforts of the industrial developments and innovations for enabling the affordable domotic humanoid robots, such as Pepper, to become a useful actor in the context of physical assistance for frail persons. Hereafter, we detail these future research axes:

- Further developments of the proprioceptive sensor based whole-body contact sensing, studied in this thesis (Chapters 1 and 2), could definitely help to achieve more complex and useful pHRI interactions. The exploitation of the joint electric current or torque sensing, that could not be used in this work, could allow to well estimate the forces applied on the robot structure. This in turn will enable closed-loop force control of the domotic humanoid robot in pHRI context.

- The multi-contact planning methodology, proposed in Chapter 3, could foremost benefit from an accelerated computations of the solution to the non-linear optimization PG problem. A faster computation of the PG solution would allow to use this method in the online experiments for a whole-body posture and contact location (re)planning.

- Several modifications of the hardware platform of the Pepper humanoid robot could greatly impact the use of such a platform in the assistive pHRI context. Namely, the increased onboard computing power via the use of modern portable GPUs is paramount for enabling the robot to be controlled without the use of an external PC. The use of onboard wide-angle depth perception for human body tracking is also a critical aspect for a pHRI in close proximity, as was demonstrated in the experiments presented in Chapter 4, where an additional camera was attached to the current Pepper robot platform for running the experiments.

- The open-source, well documented and maintained software for robot control in pHRI experiments is an essential tool for roboticists to share and reuse the experiences gained across various application areas in an easy and fast manner. From our side, we have made the developed software publicly available and described thoroughly in Appendix A. We hope that these open-source tools will prove to be useful for other roboticists and will help to foster the future developments that will contribute to the extension and continuous improvement of these tools.

- Finally, extensive user studies are required for refining the proposed methodologies taking user feedback into consideration. Unfortunately, the envisioned user studies could not be conducted in the frame of this thesis due to the Covid-19 pandemic. This is an extremely important part of the research work that would allow to reveal the benefits and limitations of the use of the domotic humanoid robot as an physically assistive agent for frail persons.

CHAPTER A

DEVELOPED SOFTWARE TOOLS

We present an open-source software interface, called `mc_naoqi`, that allows to perform whole-body task-space QP based control, implemented in `mc_rtc` framework, on the SBRE humanoid robots.

QP task-space control has become a golden standard in humanoid robot control Escande et al. (2014a); Cisneros et al. (2019); Bouyarmane et al. (2019). One of the most powerful software control frameworks that implements QP task-space control, called `mc_rtc` [1], is now publicly available. This framework provides developers with useful tools for writing complex controllers for either individual or multiple interacting robots to perform wide variety of experiments, e.g. in aircraft automation Kheddar et al. (2019) or in physically interacting with humans Otani et al. (2018); Bolotnikova et al. (2020a) that we presented in Chapter 4.

In order to control any given complex robots with this framework, an interface must be developed to allow communication between the control framework and the robot's low-level controllers, sensors and devices. For different types of robots a specific interface, as well as robot description and robot module, need to be developed in order to adapt to particularities of robot brand and low-level control strategies, devices and onboard operating system (OS). The idea of task-space control is to lie exactly between low-level control and high-level planning with somewhat adjustable frontiers.

In this appendix, we present the open-source software interface, called `mc_naoqi` [2], that enables running the controllers implemented with `mc_rtc` framework on widely used SBRE humanoid robots Gouaillier et al. (2009); Pandey and Gelin (2018), that are running NAOqi OS and a customly developed local module `mc_naoqi_dcm` (Sec. A.1). We also provide robot modules and description packages that serve as a representation of SBRE robots in the `mc_rtc` framework (Sec. A.2). Additionally, we release a basic sample FSM `mc_rtc` controller for Pepper robot and its interaction with a human model in simulation (Sec. A.3).

Fig. A.1 shows an overview of the developed tools and their interconnection, described in detail in the rest of the appendix. The video presentation of the developed tools and their use in pHRI experiments is available online [3].

1. https://jrl-umi3218.github.io/mc_rtc
2. https://github.com/jrl-umi3218/mc_naoqi
3. Video "Task-Space Control Interface for SoftBank Humanoids and its Human-Robot Interaction Applications" is hosted at the IDH LIRMM YouTube channel: https://youtu.be/qzEnCGlT93s

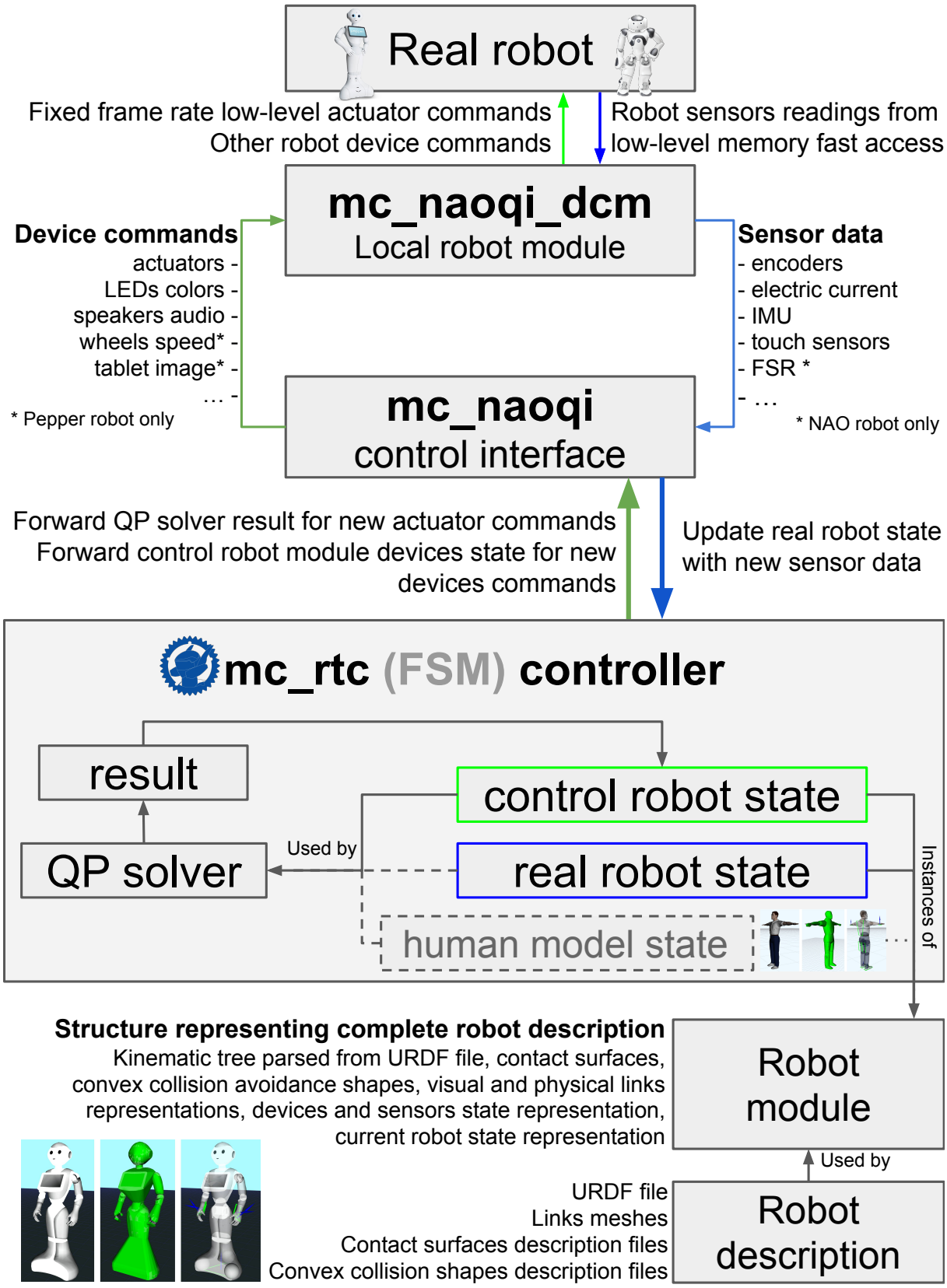


FIGURE A.1 – `mc_naoqi` *interface enables communication between SBRE humanoid robots and* `mc_rtc` *control framework. It can be used to steer the robot behaviour in HRI experiments.*

A.1 Control Interface

Fig. A.2 gives a schematic overview of the `mc_rtc` control framework and its connection to the simulation and control interfaces, such as `mc_naoqi`. Interested readers are invited to visit `mc_rtc` project website for more information, installation instructions and tutorials.

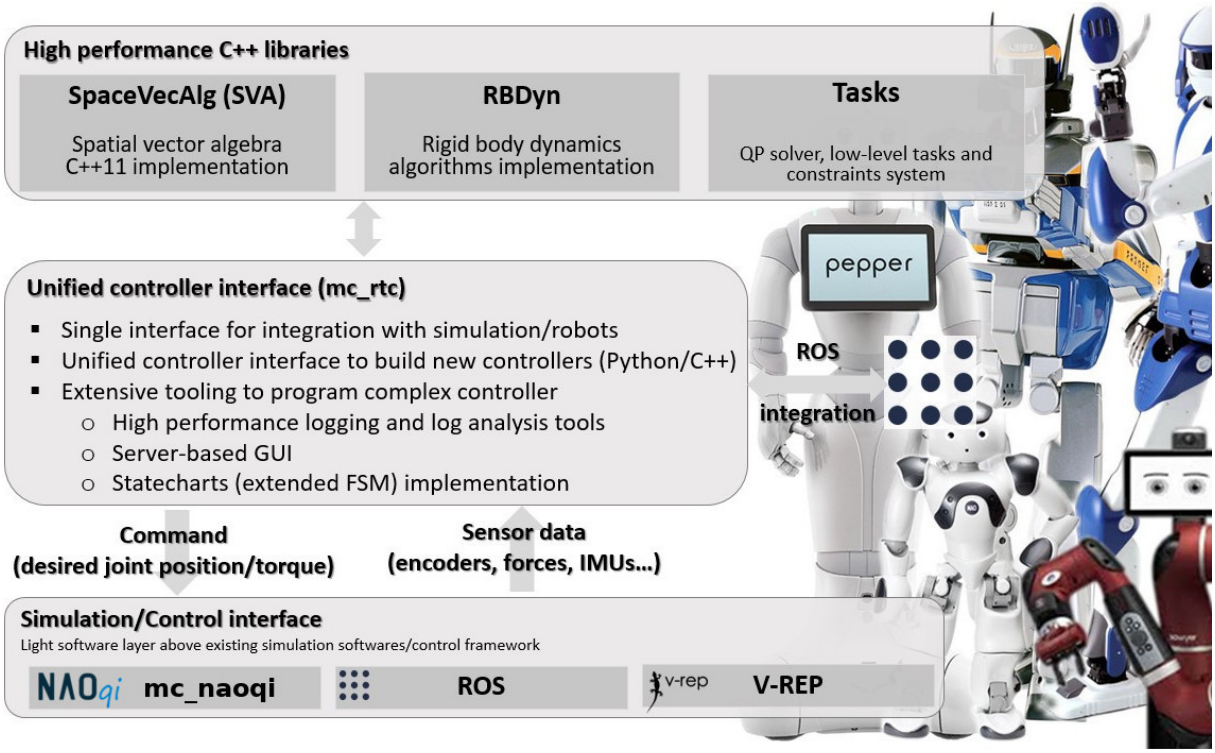


FIGURE A.2 – `mc_rtc` *control framework architecture.*

Fig. A.1 illustrates schematically the role of the `mc_naoqi` interface as a communication layer between `mc_rtc` control framework and NAOqi OS running onboard SBRE humanoid robots, such as NAO, Pepper or Romeo. In the following subsections, we describe the entire system in detail.

A.1.1 Forwarding device commands from controller to the robot

The main role of the `mc_naoqi` interface is to forward the fixed frame rate control commands computed by the QP solver of `mc_rtc` whole-body QP controller to the onboard low-level robot actuators control. The decision variable of the `mc_rtc` QP controller is robot joint accelerations, which is integrated once to get desired velocity (e.g. for Pepper mobile base command), and then once more to get desired robot joint angles for joint actuator commands.

For social, user-friendly and interacting humanoid robots, such as Pepper or NAO, it is highly beneficial to endow `mc_rtc` controller with the functionality to also forward other device commands, such as sentence to play from the speakers, desired tablet screen image or eye led color, from the `mc_rtc` controller to the robot devices via `mc_naoqi` interface. This functionality allows `mc_rtc` framework users to develop controllers which can provide a richer interaction experience for HRI applications.

For instance, when a certain `mc_rtc` controller FSM state is terminated, the robot can indicate this event (and that it is going to transition to the next FSM state) with a comprehensive verbal message, illustrative tablet image and/or a specific LED color.

The `mc_naoqi` interface is thus developed with rich HRI consideration in mind, and allows `mc_rtc` controller to forward commands to all the robot devices, not only the joint actuators. We describe how it is implemented on the robot module side in detail in Sec A.2.

A.1.2 Forwarding sensor data from the robot to the controller

The `mc_naoqi` interface is also responsible for getting the most up-to-date sensor readings from a robot low-level memory in real-time and forwarding sensor measurements in a suitable form to the `mc_rtc` controller real robot state representation as a feedback. This way, the task-space QP controller keeps track of the real robot state and can use it to perform closed-loop QP control computations.

Besides common sensor readings, such as encoder values, force sensors or Inertial measurement unit (IMU) measurements, `mc_naoqi` interface also allows to forward from the robot to the `mc_rtc` controller such sensor readings as electric motor current and touch sensor readings (from tactile or bumper sensors). We describe how custom robot sensors are implemented on the robot module side in detail in Sec A.2.

For HRI applications, the touch sensor readings are especially beneficial to be forwarded to the controller, as they allow to detect when a human touches the robot. This signal can then be used inside the `mc_rtc` FSM controller, for instance, to trigger an appropriate reaction of the robot to the touch or to trigger a transition to a specific FSM state of the controller.

A.1.3 Local low-level robot module

A customized local robot low-level module, called `mc_naoqi_dcm`[4], is cross-compiled for NAOqi OS and is set to run onboard the robot to read sensor values and set device commands synchronized with a robot control loop via Device Communication Manager (DCM) every 12 ms, fastest update rate currently available for SBRE robots.

When a user connects to the robot via `mc_naoqi` interface, and commands to turn on robot motors, a preprocess function is connected to DCM loop, to start setting the actuator commands at a fixed time rate, synchronized with the other DCM operations. After the user commands to turn off robot motors via `mc_naoqi` interface, the preprocess actuator command update function is disconnected from the DCM loop. This way, the user can safely use other control applications, such as *Choregraphe*, with the robot right after using `mc_naoqi` interface, even though `mc_naoqi_dcm` module is still active on the robot.

In order to prevent the default high-level robot behaviours to interfere with the commands forwarded to the robot via `mc_naoqi` from the `mc_rtc` controller, the default robot safety reflexes are disabled when robot motors are turned on via `mc_naoqi`.

4. https://github.com/jrl-umi3218/mc_naoqi_dcm

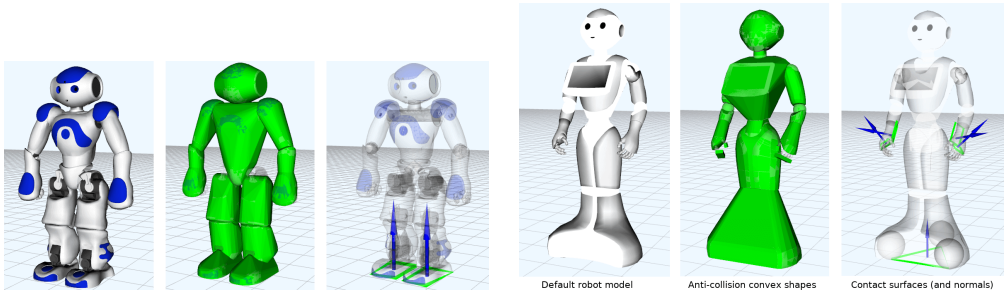
Once the motors are turned off via `mc_naoqi`, the default robot safety reflexes are re-enabled. This has to be taken into account by an `mc_rtc` controller designer, to ensure that the developed controller is safe to be run on the real robot, through extensive testing in simulation.

A fast access to the low-level robot memory is initialized when `mc_naoqi_dcm` starts to run on the robot. This allows to read a predefined set of sensor values from robot memory in the fastest way.

A.2 Robot representation in `mc_rtc` framework

A.2.1 Robot description packages

To control any robot with `mc_rtc` framework, a basic description of this robot needs to be provided. Such robot description includes robot kinematic tree, dynamic properties of the links, description of robot contact surfaces (i.e. covers), and convex (eventually strictly convex Escande et al. (2014b)) anti-collision shapes (these shapes can be automatically generated from existing robot links graphic representation). Examples of these robot description elements for SBRE humanoids are shown in Fig. A.3. With this work, we release these robot description projects, implemented as Robot Operating System (ROS) packages, for NAO [5] and Pepper [6] robots.



(a) *First contact* **(b)** *Second contact*

FIGURE A.3 – *NAO and Pepper robot descriptions for the robot representation in the robot module of the* `mc_rtc` *framework.*

A.2.2 Robot `mc_rtc` modules

As illustrated in Fig. A.1, the robot description packages are used by a *robot module* software layer to create a structure that provides a complete description of the robot: kinematic tree parsed from URDF files, visual and physical representations of robot links, surfaces attached to the robot bodies, sensors and other devices, strictly convex hulls and primitive shapes for collision avoidance, etc. The instance of this structure is used by the `mc_rtc` framework, as control robot state representation, to formulate the QP control problem (objectives and constraints), that is then passed to the solver to compute the next desired robot state.

5. https://github.com/jrl-umi3218/nao_description
6. https://github.com/jrl-umi3218/pepper_description

We make the robot modules for NAO[7] and Pepper[8] publicly available with this work. For fast prototyping and experiments, the robot description package and module can easily be augmented with any new robot hardware elements, e.g. new onboard camera, which we demonstrate in Sec. A.4.

The Pepper robot module exploits a recently introduced new `mc_rtc` framework feature - generic robot devices. This feature enables developers to implement any kind of robot custom device representation as part of the robot module, which can then be used in the `mc_rtc` controller. Currently implemented devices in Pepper robot module are:

- *loud speaker*: to forward speech commands to the robot directly from the `mc_rtc` controller via `mc_naoqi`;
- *visual display tablet*: to set robot tablet image from the `mc_rtc` controller via `mc_naoqi` interface;
- *touch sensor*: to forward tactile sensor readings from the robot to the `mc_rtc` controller via `mc_naoqi`

Wheeled mobile base of Pepper is modeled as a floating base, constrained to move on a plane (i.e. the room ground), with limits imposed to the mobile base body maximum speed and acceleration according to physical hardware limitations.

In `mc_rtc` framework, two main hierarchies for the robot control are the following:

- *Tasks*: that are objectives 'describing' what the robot should do at the best it can;
- *Constraints*: that are limits under which the tasks are to be performed, i.e. what the robot should always fulfill strictly.

Many tasks and constraints are already implemented as robot-independent templates in `mc_rtc`. For instance *PostureTask*, *CoMTask*, *EndEffectorTask*, *KinematicsConstraint*, *ContactConstraint*, etc. However, in some cases it might be desirable to design and implement new custom tasks or constraints, not yet implemented in `mc_rtc`. Such new tasks and constraints might be specific to a robot, use-case or research topic. In the Pepper robot module `mc_pepper`, we provide an example of the implementation of a custom *CoMRelativeBodyTask* QP control objective. This task allows to specify the desired Pepper center of mass (CoM) target relative to the robot mobile base frame (as opposed to world frame in `mc_rtc` *CoMTask*). The implementation of this Pepper specific objective was necessary to allow a controller to simultaneously compute a new mobile base position and keep the CoM objective as part of QP computations. An example of a Pepper specific QP constraint implementation is a custom *BoundedAccelerationConstr* constraint, included in the Pepper robot module. This constraint allows to impose acceleration bounds for Pepper mobile base link.

An example of how these custom Pepper robot specific tasks and constraints are loaded and used in a sample `mc_rtc` controller can be seen in *PepperFSMController* open-source project, which we describe in detail in Sec. A.3. In an analogous way, many other novel tasks and constraints can easily be implemented and tested.

7. https://github.com/jrl-umi3218/mc_nao
8. https://github.com/jrl-umi3218/mc_pepper

A.3 Sample Pepper `mc_rtc` FSM controller

A.3.1 Individual robot controller

To facilitate the process of writing a new `mc_rtc` controller, especially for new potential users of the framework, we provide a basic sample *PepperFSMController* [9]. It can be used as a starting point for new controller development or as an example of how similar projects should be implemented.

The sample controller includes Pepper as the main controller robot, associated default posture task, kinematics and dynamics constraints, self-collision avoidance constraints, robot-ground contact constraint. The controller also includes implementation of few sample FSM states that allow to control robot posture, mobile base, camera orientation and right and left hand end-effector positions, either through predesined setpoints or interactively through RViz Kam et al. (2015) (Fig. A.4).

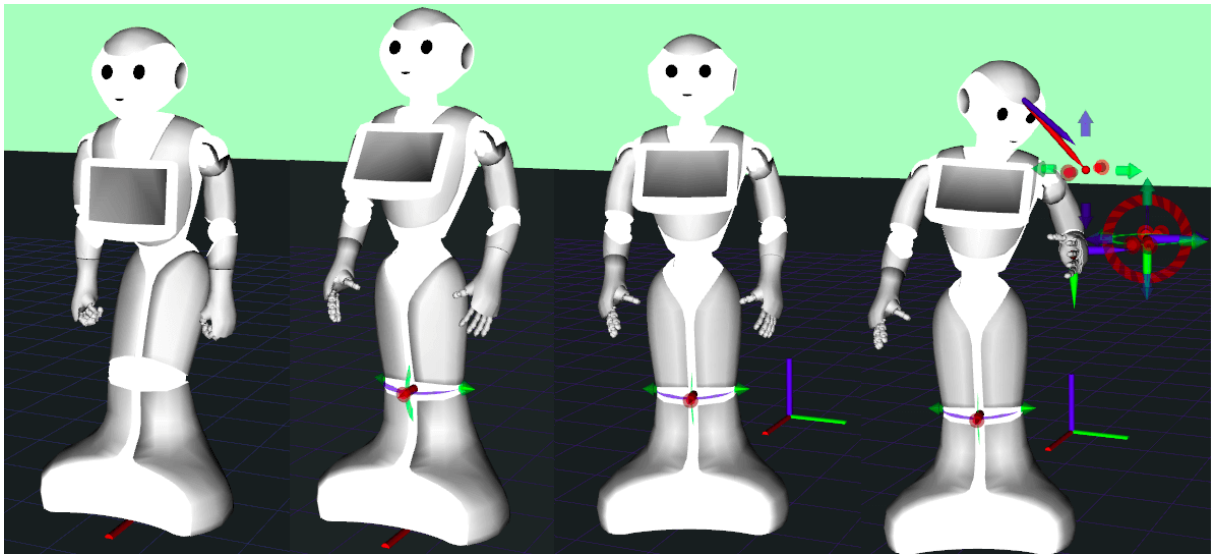


FIGURE A.4 – *RViz scenes of sample FSM Pepper controller states.*

A.3.2 Controller for HRI including a human model

A direct extension of the `master` branch of the sample controller project, is a branch called `topic/withHumanModel`. On this branch, a multi-robot QP (MQP) control feature of the `mc_rtc` framework is exploited by adding a human model and its state as part of the `mc_rtc` controller (recall Fig. A.1). A human model is integrated into `mc_rtc` exactly the same way that any other robot model, by providing a description ROS package [10] and implementing a corresponding robot module [11]. Fig. A.5 shows the human model description used in our projects.

This MQP controller can be used to develop and simulate a wide variety of HRI scenarios including pHRI ones, e.g. using human model and robot contact surfaces description to define contact tasks and constraints.

9. https://github.com/jrl-umi3218/pepper-fsm-controller
10. https://github.com/jrl-umi3218/human_description
11. https://github.com/jrl-umi3218/mc_human

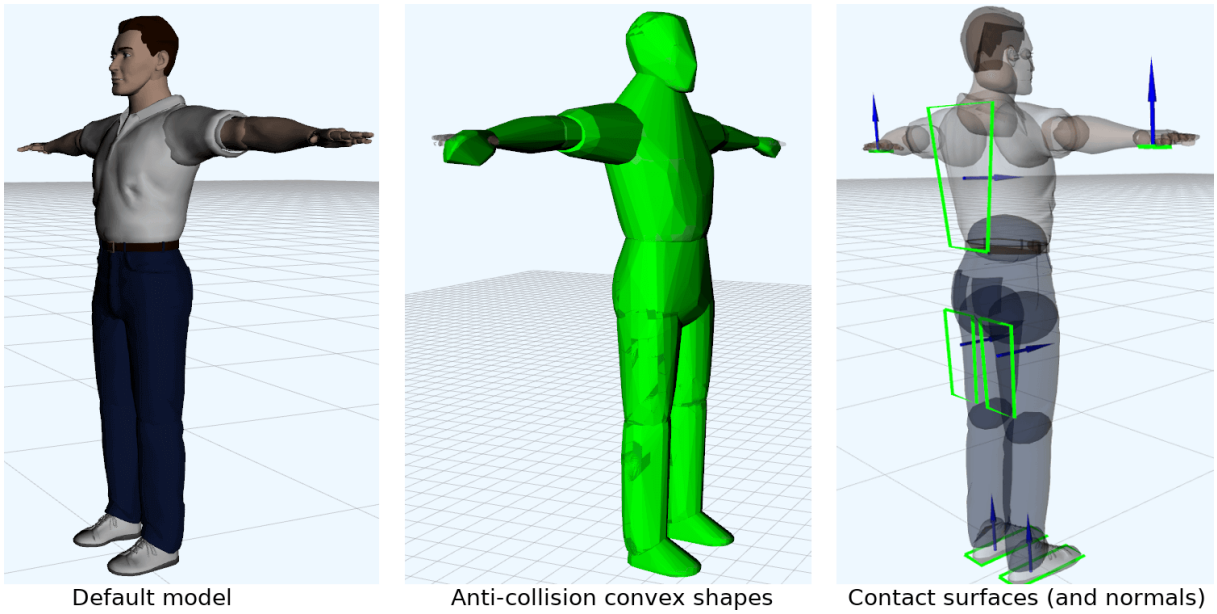


FIGURE A.5 – *Simulation human model used in* `mc_rtc` *framework.*

In the open-source sample MQP controller project, we provide an example of an HRI controller state called *NavigateToHuman*. In this state, a Position Based Visual Servoing (PBVS) task of `mc_rtc` framework is used to control Pepper robot mobile base to navigate in closed-loop to a set-point defined w.r.t human model torso frame (it can be any other human model frame), while using simulation data as a virtual visual feedback signal. At the same time, Image Based Visual Servoing (IBVS) task is used to control robot camera orientation to look at the human head. Fig. A.6 illustrates the simulation of the *NavigateToHuman* FSM state at the start, middle and the end of this state.

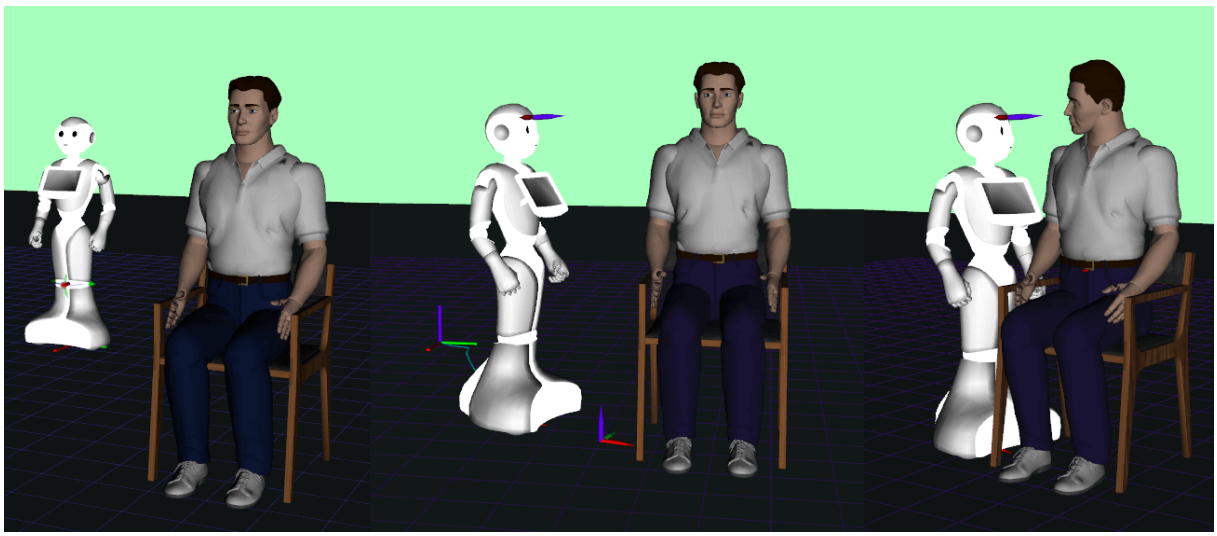
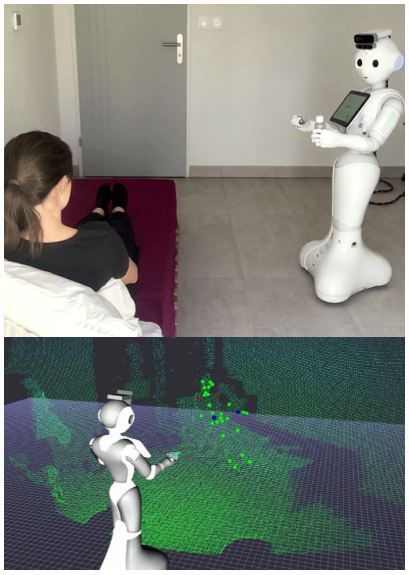


FIGURE A.6 – *Simulated* NavigateToHuman *state of a sample MQP* `mc_rtc` *FSM controller with a human model included.*

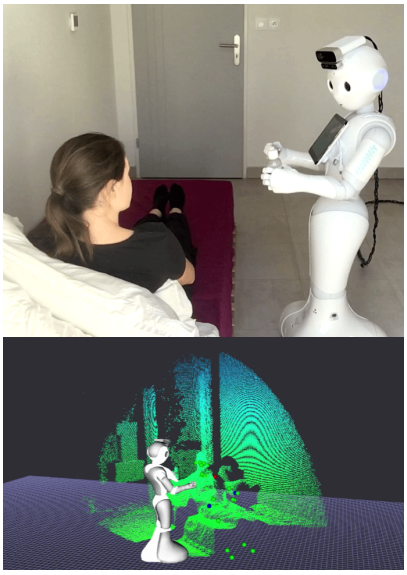
The sample controller project can also efficiently serve as a starting point for developing controllers for various real HRI applications, which we showcased in Sec. 4.4 for instance.

A.4 Medicine delivery experiment

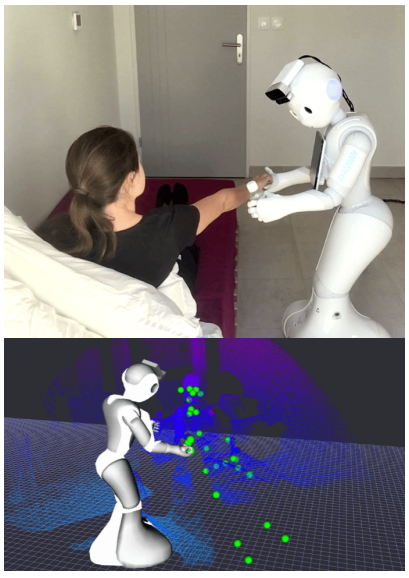
Recently, the Covid-19 pandemic has put attention into robotics, which can help to deal with the problematics of human virus transmission, by transferring some risky labours and non-added value tasks of care-givers to robotic systems. Examples come from different companies, like disinfecting ultraviolet (UV) rovers and ground drones for hospitals, or even beaming videos to connect patients to their relatives. There are more examples in other fields, like flagging patients with suspected pneumonia during their hospital admission, or using robots and AI in logistics to transport daily groceries.



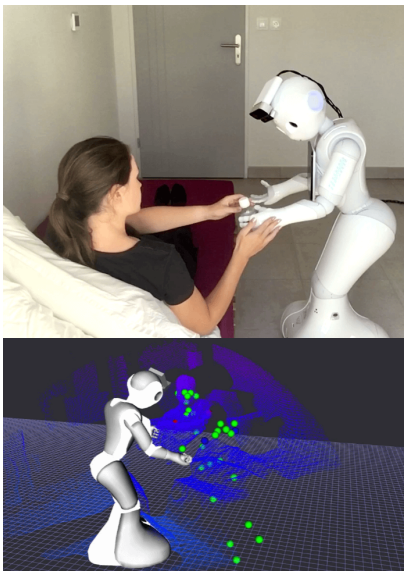
(c) *Navigation to the human*



(d) *Verbal communication*



(g) *Human takes the pills*



(h) *Human takes the bottle*

FIGURE A.7 – *Experiment screenshots (top rows), perceived robot state, Azure Kinect point cloud and human body markers (bottom rows).*

We have already demonstrated, in Chapter 4, how the developed tools, presented in this appendix, can be used to enable Pepper robot to perform autonomous initiation

of human physical assistance. We also made the controller code [12] publicly available to serve as a software reference to our work and as an advanced FSM `mc_rtc` HRI controller example. This controller shows how our developed software components allow to easily create complex controllers for rich, intuitive and efficient HRI. This includes closed-loop navigation toward human, verbal, visual and body language communication, and physical interaction with a human for an assistance process initiation.

In this section, we present a controller for another HRI application –inspired from the Covid-19 outbreak, where Pepper robot behaviour is regulated, with the help of the `mc_naoqi` and the developed robot modules, to autonomously deliver medication to a human lying on the bed. In this scenario a bed height conforms that of Pepper for the given tasks.

The aim here is also to demonstrate how the controllers for new HRI scenarios can be easily prepared using our developed tools and re-using states and elements of the controllers written for other HRI scenarios. As such, for instance, the *NavigetaToHuman* state could be re-used from our previous work, for this new scenario with almost no modifications, despite the person lying in the bed instead of sitting on a chair. Other parts of this new controller FSM also needed only slight adjustments w.r.t our existing work (states present in the other HRI controller) to create a controller for this new HRI application.

Fig. A.7 shows excerpts screenshots from the experiment video. In the bottom rows images, the scenes are visualized in RViz. Note, that additional hardware, namely Azure Kinect, which is used for human state feedback, and RealSense camera, which is included in the robot prototype, but not used in this experiment, are easily included in the robot description (Fig. A.8) and processed by the robot module, and therefore are also included in the QP problem formulation (e.g. for more accurate robot center of mass computation). Full experiment can be seen in the accompanying video [13].

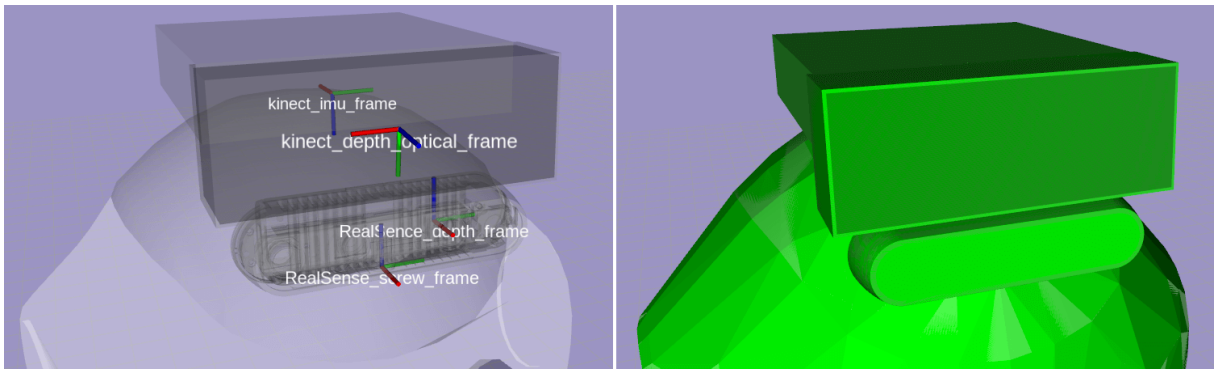


FIGURE A.8 – *Extra hardware included in the prototype Pepper robot*

Once the robot reaches a position nearby the person, it communicates verbally its intention to pass the medicine (Fig. A.7d). Then it proceeds to open its right gripper for the person to take the pills (Fig. A.7g). The passing of the bottle with liquid is arranged with the help of the robot module feature, described in Sec. A.2, that allows to forward robot tactile sensor data to the `mc_rtc` controller, which then triggers robot left hand gripper to open slightly to allow the person to get out the bottle more easily (Fig. A.7h).

12. https://github.com/anastasiabolotnikova/autonomous_phri_init
13. Video "Task-Space Control Interface for SoftBank Humanoids and its Human-Robot Interaction Applications" is hosted at the IDH LIRMM YouTube channel: https://youtu.be/qzEnCGlT93s

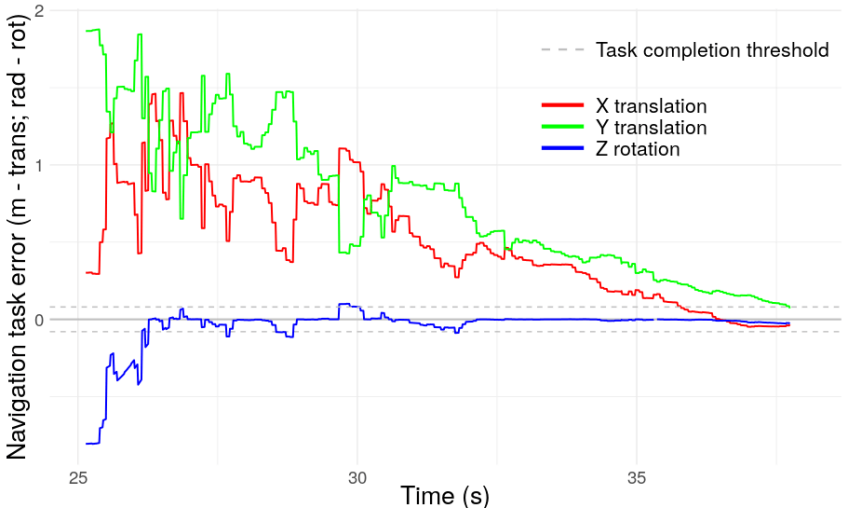


FIGURE A.9 – *Evolution of Position Based Visual Servoing closed-loop Pepper mobile base navigation to human task errors*

Fig. A.9 shows the progress of the closed-loop PBVS task, that uses Azure Kinect body tracking for feedback to make the robot navigate to the person (Fig. A.7c). Task errors eventually converge near zero, although in a not very smooth way. This is due to the low quality of human detection (that prohibits constant usage in a continuous closed-loop way), and the low detection frame rate and latency issues (that limits the speed in reaching the person). Indeed, the existing human motion trackers are not robust enough for robotic usage. It is important to underline some shortcomings that prohibit current spreading of HRI in real practice:

- reliable human posture and body detection algorithms (even those advanced) often fail in robotics because of the usage perspective. In assistive robotics scenarios, it is the robot which moves toward a human that can be static (i.e. not moving much), e.g. lying in a home or hospital bed or sitting on a chair. Most of the human pose detection algorithms are trained with rather static camera and moving human and not with a moving camera and static (or moving) human. We have witnessed considerable effects on the robustness of the human pose acquisition by a moving robot with all the algorithms we tried. As a consequence, it is difficult to use them continuously in closed-loop control.

- lack of ground-truth: most of the human pose detection matches well in an augmented-reality, their image or video counterpart, but no one provides measurement ground-truth concerning the pose results that are returned. That is to say, it is difficult to assess the precision of the 6D pose returned by most algorithms with ground-truth 6D measurements. In robotics, it is important to know precisely and in real-time the exact joint and floating base values of the human posture and configuration, namely when it comes to contact a person or to manipulate a given limb of a person.

- last but not least, as it is the most critical issue: in close-contact interaction with humans (i.e. when the robot reaches a person), tracking may be lost even when a wide fish-eye camera is used. This is even more critical when physical interaction causes an obstructed view of the person. This means that human pose estimation

approaches in the framework of human-robot close-contact interactions have to be deeply reviewed.

This being said, the approach we adopt is rather modular enough to live with such shortcoming without recalling into question the controller. Any improvement in human pose estimation w.r.t the performances mentioned previously will straightforwardly result in better robustness and performance of the controller, because of its conceptual simplicity.

A.5 Concluding notes

For robotics to gain more insight, trust and meet the challenge of future human societal stakes, such as home daily assistance for frail or elderly persons, or to be efficiently used in future outbreaks, research efforts shall be paired with important integration development ones. Any advances made in critical human-robot interaction technological bricks such as human perception, artificial intelligence chatbots, 5G, advanced SLAM... should be integrated in readily available and sustained task-space control frameworks. This is the very purpose of the tools described in this appendix: we do not only aim at sharing the open-source code for the robotic community in general and the HRI in particular, but also sharing experiences that led to existing developments which are made to be further used, improved and sustained. Our methodology in terms of control can be seen now under a different philosophy. In computer science, algorithms, simple instructions such as if-then-else, while-for loops, variables... together with basic well-agreed routines and functions constitute the basic bricks of any modern algorithm that solve increasingly more complex problems. Our robot control framework should be seen under this spectrum: we shall provide elementary controllers and controller "routines", templates that form the common "instructions", and functions of more complex controllers that solve more and more complicated tasks. We exemplify what `mr_rtc` may bring under a unified framework in terms of task specification, embedding straightforwardly the constraints... how a multi-(sensory,objectives,robots) task-space controller can be used to build sustainable controllers, and show that what we propose is consistent, as all these tasks are defined exactly the same way for any robot or multi-robot system. We hope that users of SBRE robots could assess, enrich and use our developed software tools in even more challenging scenarios.

BIBLIOGRAPHY

(2019). World population ageing 2019. highlights. report ST/ESA/SER.A/430, United Nations, Department of Economic and Social Affairs, Population Division, New York, US. 7

Agravante, D. J., Claudio, G., Spindler, F., and Chaumette, F. (2016). Visual servoing in an optimization framework for the whole-body control of humanoid robots. *IEEE Robotics and Automation Letters*, 2(2):608–615. 54

Aissaoui, R. and Dansereau, J. (1999). Biomechanical analysis and modelling of sit to stand task: a literature review. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 141–146. 67

Arreguit, J., Faraji, S., and Ijspeert, A. J. (2018). Fast multi-contact whole-body motion planning with limb dynamics. In *IEEE-RAS International Conference on Humanoid Robots*, pages 1–9, Beijing, China. 39

Barros, J. M. D., Garcia, F., and Sidibé, D. (2015). Real-time human pose estimation from body-scanned point clouds. In *International Conference on Computer Vision Theory and Applications*, Berlin, Germany. 40

Birjandi, S. A. B. and Haddadin, S. (2020). Model-adaptive high-speed collision detection for serial-chain robot manipulators. *IEEE Robotics and Automation Letters*, 5(4):6544–6551. 14

Birjandi, S. A. B., Kühn, J., and Haddadin, S. (2020a). Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing. *IEEE Robotics and Automation Letters*, 5(2):954–961. 14

Birjandi, S. A. B., Kühn, J., and Haddadin, S. (2020b). Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing. *IEEE Robotics and Automation Letters*, 5(2):954–961. 79

Bolotnikova, A., Courtois, S., and Kheddar, A. (2018a). Compliant robot motion regulated via proprioceptive sensor based contact observer. In *IEEE-RAS International Conference on Humanoid Robots*, pages 854–859, Beijing, China. 10, 48

Bolotnikova, A., Courtois, S., and Kheddar, A. (2018b). Contact observer for humanoid robot Pepper based on tracking joint position discrepancies. In *IEEE International Conference on Robot and Human Interactive Communication*, pages 29–34, Nanjing, China. 10, 48

Bolotnikova, A., Courtois, S., and Kheddar, A. (2020a). Autonomous initiation of human physical assistance by a humanoid. In *IEEE International Conference on Robot and Human Interactive Communication*, Naples, Italy. 10, 79, 83

Bolotnikova, A., Courtois, S., and Kheddar, A. (2020b). Multi-contact planning on humans for physical assistance by humanoid. In *IEEE Robotics and Automation Letters*, volume 5, pages 135–142. 10, 65, 72

Bolotnikova, A., Courtois, S., and Kheddar, A. (2021a). Adaptive task-space force control for humanoid-to-human assistance. Submitted, under review. 10

Bolotnikova, A., Gergondet, P., Tanguy, A., Courtois, S., and Kheddar, A. (2021b). Task-space control interface for softbank humanoid robots and its human-robot interaction applications. In *2021 IEEE/SICE International Symposium on System Integration*. 10

Bouyarmane, K. and Kheddar, A. (2011). Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4414–4419. 15

Bouyarmane, K. and Kheddar, A. (2018). On weight-prioritized multitask control of humanoid robots. *IEEE Transactions on Automatic Control*, 63(6):1632–1647. 78

Bouyarmane, K., Vaillant, J., Chappellet, K., and Kheddar, A. (2019). Quadratic programming for multirobot and task-space force control. *IEEE Transactions on Robotics*, 35(1):64–77. 15, 48, 55, 56, 64, 66, 68, 71, 72, 73, 83

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Chapman and Hall/CRC. 18

Brossette, S. (2016). *Viable multi-contact posture computation for humanoid robots using nonlinear optimization on manifolds*. PhD thesis, University of Montpellier. 41, 48

Brossette, S., Escande, A., and Kheddar, A. (2018). Multicontact postures computation on manifolds. *IEEE Transactions on Robotics*, 34(5):1252–1265. 38, 41, 47

Brossette, S., Vaillant, J., Keith, F., Escande, A., and Kheddar, A. (2013). Point-cloud multi-contact planning for humanoids: Preliminary results. In *IEEE Conference on Robotics, Automation and Mechatronics*, pages 19–24, Manila, Philippines. 39

Buondonno, G. and De Luca, A. (2016). Combining real and virtual sensors for measuring interaction forces and moments acting on a robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 794–800. 13, 79

Buys, K., Cagniart, C., Baksheev, A., De Laet, T., De Schutter, J., and Pantofaru, C. (2014). An adaptable system for rgb-d based human body detection and pose estimation. *Journal of Visual Communication and Image Representation*, 25(1):39–52. 40

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2021). OpenPose: realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186. 68

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, Honolulu, Hawaii, US. 39, 45

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *International conference on knowledge discovery and data mining*, pages 785–794. 28

Cheng, G., Dean-Leon, E., Bergner, F., Olvera, J. R. G., Leboutet, Q., and Mittendorfer, P. (2019). A comprehensive realization of robot skin: Sensors, sensing, control, and applications. *Proceedings of the IEEE*, 107(10):2034–2051. 26

Cisneros, R., Benallegue, M., Morisawa, M., and Kanehiro, F. (2019). QP-based task-space hybrid/parallel control for multi-contact motion in a torque-controlled humanoid robot. In *IEEE-RAS 19th International Conference on Humanoid Robots*, pages 663–670. IEEE. 83

Claudio, G., Spindler, F., and Chaumette, F. (2016). Vision-based manipulation with the humanoid robot Romeo. In *IEEE-RAS 16th International Conference on Humanoid Robots*, pages 286–293. 54

Corona, E., Pumarola, A., and Moreno-Noguer, G. A. F. (2020). Context-aware human motion prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6990–6999. 67

Coumans, E. and Bai, Y. (2016–2019). PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org. 76

Dai, H., Valenzuela, A., and Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, Madrid, Spain. 39

Dautenhahn, K. (2007). Socially intelligent robots: Dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):679–704. 51

De Luca, A. and Ferrajoli, L. (2008). Exploiting robot redundancy in collision detection and reaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3299–3305. 26

De Luca, A. and Mattone, R. (2005). Sensorless robot collision detection and hybrid force/motion control. In *IEEE International Conference on Robotics and Automation*, pages 999–1004. 13, 26

Deits, R. and Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, Madrid, Spain. 39

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. 28

Dimitrov, A., Gu, R., and Golparvar-Fard, M. (2016). Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling. *Computer-Aided Civil and Infrastructure Engineering*, 31(7):483–498. 42

Drucker, H. and Cortes, C. (1996). Boosting decision trees. In *Advances in neural information processing systems*, pages 479–485. 28

Escande, A., Kheddar, A., and Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442. 39, 51

Escande, A., Mansard, N., and Wieber, P.-B. (2014a). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028. 83

Escande, A., Miossec, S., Benallegue, M., and Kheddar, A. (2014b). A strictly convex hull for computing proximity distances with continuous gradients. *IEEE Transactions on Robotics*, 30(3):666–678. 47, 87

Flacco, F. and Kheddar, A. (2017). Contact detection and physical interaction for low cost personal robots. In *IEEE International Conference on Robot and Human Interactive Communication*, pages 495–501, Lisbon, Portugal. 13, 14, 16, 25, 26

Flacco, F., Paolillo, A., and Kheddar, A. (2016). Residual-based contacts estimation for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, pages 409–415. 13, 25, 26

Flöry, S. (2009). Fitting curves and surfaces to point clouds in the presence of obstacles. *Computer Aided Geometric Design*, 26(2):192–202. 42

Flöry, S. and Hofer, M. (2008). Constrained curve fitting on manifolds. *Computer-Aided Design*, 40(1):25–34. 42

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., and Maisonnier, B. (2009). Mechatronic design of NAO humanoid. In *IEEE International Conference on Robotics and Automation*, pages 769–774. IEEE. 83

Gribovskaya, E., Kheddar, A., and Billard, A. (2011). Motion learning and adaptive impedance for robot control during physical interaction with humans. In *IEEE International Conference on Robotics and Automation*, pages 4326–4332. 70

Haddadin, S., Albu-Schaffer, A., De Luca, A., and Hirzinger, G. (2008). Collision detection and reaction: A contribution to safe physical human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363. 26

Haddadin, S., De Luca, A., and Albu-Schäffer, A. (2017). Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312. 13, 25, 26, 105

Heo, Y. J., Kim, D., Lee, W., Kim, H., Park, J., and Chung, W. K. (2019). Collision detection for industrial collaborative robots: A deep learning approach. *IEEE Robotics and Automation Letters*, 4(2):740–746. 15

Herman, I. (2007). *Physics of the human body*. Springer Science & Business Media. 67

Hyon, S.-H., Hale, J. G., and Cheng, G. (2007). Full-body compliant human–humanoid interaction: Balancing in the presence of unknown external forces. *IEEE Transactions on Robotics*, 23(5):884–898. 27

Ibanez, A., Bidaud, P., and Padois, V. (2014). Automatic optimal biped walking as a mixed-integer quadratic program. In *Advances in Robot Kinematics*, pages 505–516. 39

Jovic, J., Escande, A., Ayusawa, K., Yoshida, E., Kheddar, A., and Venture, G. (2016). Humanoid and human inertia parameter identification using hierarchical optimization. *IEEE Transactions on Robotics*, 32(3):726–735. 67, 78

Kam, H. R., Lee, S.-H., Park, T., and Kim, C.-H. (2015). RViz: A toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345. 89

Khatib, O., Yokoi, K., Brock, O., Chang, K., and Casal, A. (1999). Robots in human environments: Basic autonomous capabilities. *The International Journal of Robotics Research*, 18(7):684–696. 40

Kheddar, A. and Billard, A. (2011). A tactile matrix for whole-body humanoid haptic sensing and safe interaction. In *IEEE International Conference on Robotics and Biomimetics*, pages 1433–1438, Phuket, Thailand. 26

Kheddar, A., Caron, S., Gergondet, P., Comport, A., Tanguy, A., Ott, C., Henze, B., Mesesan, G., Englsberger, J., Roa, M. A., et al. (2019). Humanoid robots in aircraft manufacturing: The airbus use cases. *IEEE Robotics & Automation Magazine*, 26(4):30–45. 83

Kim, M., Kim, J. H., Kim, S., Sim, J., , and Park, J. (2018). Disturbance observer based linear feedback controller for compliant motion of humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 403–410. 27

Kratzer, P., Toussaint, M., and Mainprice, J. (2018). Towards combining motion optimization and data driven dynamical models for human motion prediction. In *IEEE-RAS International Conference on Humanoid Robots*, pages 202–208. 67

Kratzer, P., Toussaint, M., and Mainprice, J. (2020). Prediction of human full-body movements with motion optimization and recurrent neural networks. In *IEEE International Conference on Robotics and Automation*, pages 1792–1798. 67

Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743. 54

Kwok, T.-H., Yeung, K.-Y., and Wang, C. C. (2014). Volumetric template fitting for human body reconstruction from incomplete data. *Journal of Manufacturing Systems*, 33(4):678–689. 39

Latella, C., Lorenzini, M., Lazzaroni, M., Romano, F., Traversaro, S., Akhras, M. A., Pucci, D., and Nori, F. (2019). Towards real-time whole-body human dynamics estimation through probabilistic sensor fusion algorithms. *Autonomous Robots*, 43(6):1591–1603. 67, 78

Lee, C., Kim, J.-Y., Kim, S.-Y., and Oh, S. (2018). Human force observation and assistance for lower limb rehabilitation using wire-driven series elastic actuator. *Mechatronics*, 55:13–26. 79

Li, D., Rau, P. P., and Li, Y. (2010). A cross-cultural study: Effect of robot appearance and task. *International Journal of Social Robotics*, 2(2):175–186. 37, 79

Li, X., Pan, Y., Chen, G., and Yu, H. (2016). Adaptive human–robot interaction control for robots driven by series elastic actuators. *IEEE Transactions on Robotics*, 33(1):169–182. 79

Liao, Y., Vakanski, A., and Xian, M. (2020). A deep learning framework for assessing physical rehabilitation exercises. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(2):468–477. 73

López, A. M., Vaillant, J., Keith, F., Fraisse, P., and Kheddar, A. (2014). Compliant control of a humanoid robot helping a person stand up from a seated position. In *IEEE-RAS International Conference on Humanoid Robots*, pages 817–822, Madrid, Spain. 51, 54

Manuelli, L. and Tedrake, R. (2016). Localizing external contact using proprioceptive sensors: The Contact Particle Filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5062–5069. 13

Mattioli, T. and Vendittelli, M. (2017). Interaction force reconstruction for humanoid robots. *IEEE Robotics and Automation Letters*, 2(1):282–289. 13, 26

Mittendorfer, P., Yoshida, E., and Cheng, G. (2015). Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot. *Advanced Robotics*, 29(1):51–67. 26

Mitzner, T. L., Chen, T. L., Kemp, C. C., and Rogers, W. A. (2014). Identifying the potential for robotics to assist older adults in different living environments. *International Journal of Social Robotics*, 6(2):213–227. 37, 65

Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(43):43. 39

Niemelä, M. and Melkas, H. (2019). Robots as social and physical assistants in elderly care. In *Human-Centered Digitalization and Services*, volume 19, pages 177–197. 7, 40, 65

Oßwald, S., Gutmann, J.-S., Hornung, A., and Bennewitz, M. (2011). From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. In *IEEE-RAS International Conference on Humanoid Robots*, pages 93–98, Bled, Slovenia. 39

Otani, K., Bouyarmane, K., and Ivaldi, S. (2018). Generating assistive humanoid motions for co-manipulation tasks with a multi-robot quadratic program controller. In *IEEE International Conference on Robotics and Automation*, pages 3107–3113. IEEE. 83

Ott, C., Henze, B., and Lee, D. (2013). Kinesthetic teaching of humanoid motion based on whole-body compliance control with interaction-aware balancing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4615–4621. 27

Pandey, A. K. and Gelin, R. (2018). A mass-produced sociable humanoid robot: Pepper: the first machine of its kind. *IEEE Robotics & Automation Magazine*, 25(3):40–48. 8, 65, 83

Pandey, A. K., Gelin, R., Alami, R., Viry, R., Buendia, A., Meertens, R., Chetouani, M., Devilliers, L., Tahon, M., Filliat, D., Grenier, Y., Maazaoui, M., Kheddar, A., Lerasle, F., and Fitte Duval, L. (2014). Romeo2 Project: Humanoid robot assistant and companion for everyday life: I. Situation assessment for social intelligence. In *International Workshop on Artificial Intelligence and Cognition*, pages 140–147. 12

Paolillo, A., Chappellet, K., Bolotnikova, A., and Kheddar, A. (2018). Interlinked visual tracking and robotic manipulation of articulated objects. *IEEE Robotics and Automation Letters*, 3(4):2746–2753. 64

Papadopoulos, I., Koulouglioti, C., Lazzarino, R., and Ali, S. (2020). Enablers and barriers to the implementation of socially assistive humanoid robots in health and social care: A systematic review. *BMJ open*, 10(1). 65

Parusel, S., Haddadin, S., and Albu-Schäffer, A. (2011). Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot. In *IEEE International Conference on Robotics and Automation*, pages 4298–4305. 26

Pavol, M. J., Owings, T. M., and Grabiner, M. D. (2002). Body segment inertial parameter estimation for the general population of older adults. *Journal of biomechanics*, 35(5):707–712. 78

Pham, T.-H., Caron, S., and Kheddar, A. (2018). Multicontact interaction force sensing from whole-body motion capture. *IEEE Transactions on Industrial Informatics*, 14(6):2343–2352. 79

Piegl, L. and Tiller, W. (2012). *The NURBS book*. 42, 47

Ponton, B., Herzog, A., Schaal, S., and Righetti, L. (2016). A convex model of momentum dynamics for multi-contact motion generation. pages 842–849. 39

Ramírez, O. A. I., Khambhaita, H., Chatila, R., Chetouani, M., and Alami, R. (2016). Robots learning how and where to approach people. In *25th IEEE international symposium on robot and human interactive communication (RO-MAN)*, pages 347–353. 54

Romano, F., Nava, G., Azad, M., Čamernik, J., Dafarra, S., Dermy, O., Latella, C., Lazzaroni, M., Lober, R., Lorenzini, M., et al. (2017). The CoDyCo project achievements and beyond: Toward human aware whole-body controllers for physical human robot interaction. *IEEE Robotics and Automation Letters*, 3(1):516–523. 54

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation*, pages 1–4, Shanghai, China. 47

Sisbot, E. A., Marin-Urias, L. F., Alami, R., and Simeon, T. (2007). A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883. 54

Staffa, M. and Rossi, S. (2016). Recommender interfaces: The more human-like, the more humans like. In *International Conference on Social Robotics*, pages 200–210, Kansas City, US. 37

Tee Kit Tsun, M., Lau, B. T., and Siswoyo Jo, H. (2018). An improved indoor robot human-following navigation model using depth camera, active IR marker and proximity sensors fusion. *Robotics*, 7(1):4. 54

Tirupachuri, Y., Nava, G., Rapetti, L., Latella, C., and Pucci, D. (2019). Trajectory advancement during human-robot collaboration. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–8, New Delhi, India. 54

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30. 79

Todorov, E. and Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235. 67

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics*, 34(3):586–601. 39

Torta, E., Werner, F., Johnson, D. O., Juola, J. F., Cuijpers, R. H., Bazzani, M., Oberzaucher, J., Lemberger, J., Lewy, H., and Bregman, J. (2014). Evaluation of a small socially-assistive humanoid robot in intelligent homes for the care of the elderly. *Journal of Intelligent & Robotic Systems*, 76(1):57–71. 37, 79

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977. 79

Vaca Benitez, L. M., Tabie, M., Will, N., Schmidt, S., Jordan, M., and Kirchner, E. A. (2013). Exoskeleton technology in rehabilitation: Towards an EMG-based orthosis system for upper limb neuromotor rehabilitation. *Journal of Robotics*. 54

Vakanski, A., Jun, H.-p., Paul, D., and Baker, R. (2018). A data set of human body movements for physical rehabilitation exercises. *Data*, 3(1):2. 73

Vorndamme, J., Schappler, M., and Haddadin, S. (2017). Collision detection, isolation and identification for humanoids. In *IEEE International Conference on Robotics and Automation*, pages 4754–4761. 14, 26

Xu, W., Chatterjee, A., Zollhöfer, M., Rhodin, H., Mehta, D., Seidel, H.-P., and Theobalt, C. (2018). MonoPerfCap: Human performance capture from monocular video. *ACM Transactions on Graphics*, 37(2):27. 47

Yamada, Y., Hirasawa, Y., Huang, S., Umetani, Y., and Suita, K. (1997). Human-robot contact in the safeguarding space. *IEEE/ASME transactions on mechatronics*, 2(4):230–236. 54

Yogeswaran, N., Dang, W., Navaraj, W. T., Shakthivel, D., Khan, S., Polat, E. O., Gupta, S., Heidari, H., Kaboli, M., Lorenzelli, L., et al. (2015). New materials and advances in making electronic skin for interactive robots. *Advanced Robotics*, 29(21):1359–1373. 26

Zhang, Q. and Benveniste, A. (1992). Wavelet networks. *IEEE Transactions on Neural Networks*, 3(6):889–898. 18

Zwiener, A., Geckeler, C., and Zell, A. (2018). Contact point localization for articulated manipulators with proprioceptive sensors and machine learning. In *2018 IEEE International Conference on Robotics and Automation*, pages 323–329. 15

List of Figures

List of Tables