NEUROADAPTIVE HUMAN-MACHINE INTERFACES FOR COLLABORATIVE ROBOTS

by

SVEN CREMER

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2017

In memory of my father Dieter (1944-2017).

To my mother Elfi.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Dan O. Popa for giving me the opportunity to join and participate in the Next Generation Systems (NGS) research group these past 4½ years. I am grateful for his continuous support and mentorship during my research and graduate education. I would also like to thank Dr. Frank L. Lewis for his guidance during my academic journey. Thank you both for giving me the opportunity to work with and learn from you.

I would like to acknowledge and thank Dr. George Kondraske, Dr. Michael T. Manry, Dr. Wendell A. Davis, and Dr. William E. Dillon for taking interest in my research and serving on my comprehensive and dissertation committees.

Many thanks to the members of the NGS research group for providing a stimulating and enjoyable working environment. I would like to thank Dr. Isura Ranatunga and Dr. Indika Wijayasinghe as mentors and friends. I also thank Dr. Joe Sanford, Sumit Das, Larry Mastromoro, Rommel Alonzo, Yathartha Tuladhar, Fahad Mirza, Joshua Baptist, Emrys Maier, Drew Waller, Sandesh Gowda, Dr. Oguz Yetkin, and Anthony Hingeley for their support and help with my research.

A big thank you to the faculty and staff at the University of Texas at Arlington Research Institute (UTARI), including Dr. Jeongsik Shin, Cody Lundberg, Matt Middleton, Kris Doelling, Priscila Walsh, and Norm Spayd.

I would also like to acknowledge the following grants and institutions that have supported my research: US National Science Foundation grants NRI-1208623, 1643989, IIP-1534124, ECCS-1405173, and IIS-1208623. Qinetiq-North America Corporation, BIC-1261005720, ONR grant N00014-13-1-0562, AFOSR EOARD Grant

13-3055, and ARO grant W911NF-11-D-0001. China NNSF grant 61120106011 and China Education Ministry Project 111 (No.B08015).

August 2, 2017

ABSTRACT

NEUROADAPTIVE HUMAN-MACHINE INTERFACES FOR COLLABORATIVE ROBOTS

Sven Cremer, Ph.D.

The University of Texas at Arlington, 2017

Supervising Professor: Dan O. Popa

With an increasing number of collaborative robots or “co-robots” entering human environments, there is a growing need for *safe*, *intuitive*, and *efficient* (physical) Human-Machine Interfaces. Unlike industrial robots, co-robots operate in cluttered and dynamic working spaces, where accidental collisions are more likely to occur. To minimize interaction forces, co-robots are usually lightweight and compliant. However, this makes the robot dynamics highly nonlinear and therefore difficult to model and control. In addition, the control loop must incorporate feedback from integrated sensors. Future systems under development are covered with force-sensing robot skin comprised of thousands of multi-modal sensors, creating the need for efficient robot sensor calibration and processing.

During this thesis work, a neuroadaptive (NA) controller was developed and validated for *safe* and stable physical interaction. In order to achieve *intuitive* physical Human-Robot Interaction (pHRI), the robot error dynamics were modified to behave equivalent to a simple admittance model by expanding the NA controller with prescribed error dynamics (PED). Another new approach for modifying the robot error dynamics was an inner/outer-loop structure consisting of an admittance model in the outer-loop, which generates a model trajectory that the inner-loop follows. This admittance model was implemented with an autoregressive moving average (ARMA) filter, which was tuned with recursive least squares and with the help of a prescribed task model. Experiments conducted during this thesis showed that the developed two-loop framework allows a high degree of generality and adaptability to different human preferences, tasks, robots, and sensors. It is also offers a novel algorithm for adaptive calibration of robot skins by directly tuning admittance models that map sensor voltages into desired robot motion. Finally, it was suggested that the pHRI can be made more *efficient* by reducing the human effort during a collaborative task. The human force exerted on the robot to achieve a desired pose can be minimized by predicting and then executing the desired human motion. Different human intent estimators (HIEs) were proposed, including a neural network based estimator.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Background and Motivation

There is a general sentiment that mankind is entering a new industrial age. Across the globe, industrial nations are anticipating a paradigm shift to more flexible and intelligent autonomous manufacturing, and are making major investments to remain competitive. In the past decade, Japan has launched several smart factory initiatives [1] to connect real-word machines with virtual information and networked sensors resulting in so-called "cyber-physical systems" (see Fig. 1.1). In 2011, the White House announced the Advanced Manufacturing Partnership, which represents a "national effort bringing together industry, universities, and the federal government to invest in emerging technologies that will create high quality manufacturing jobs and enhance our global competitiveness" [2]. In the late 2000s, Germany's vision of a new smart factory started to take shape and resulted in the Industry 4.0 platform. It focuses on standardization and modularization, decentralized self-organization and optimization, to make "production systems as much as 30 percent faster and 25 percent more efficient" [3]. Industry 4.0 research areas include individualized, modularized, and flexible production, which is autonomous, networked, and real-time optimized, but still energy and resource efficient. In this fourth industrial revolution, human-machine interaction plays a crucial role. The goal is to further reduce the need for manual labor and instead allow workers to "adopt the role of strategic decision-makers and flexible problem-solvers" [4]. During manufacturing and service tasks, workers will be assisted by collaborative robots or "co-robots." Through advance-

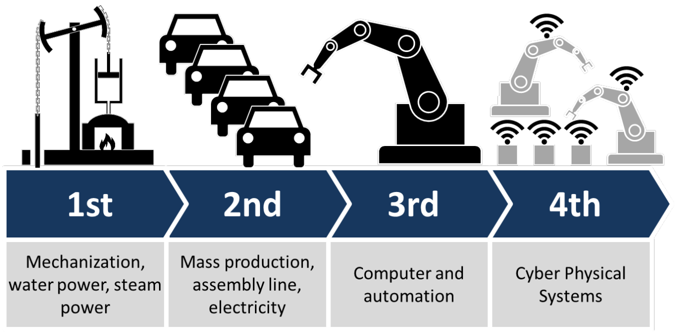

1st
2nd
3rd
4th
Mechanization, water power, steam power
Mass production, assembly line, electricity
Computer and automation
Cyber Physical Systems

Figure 1.1. Industrial revolutions leading up to Industry 4.0 [6].

ments in areas such as sensing, communication, and computing, robots are starting to be deployed in close proximity to humans in industrial as well as daily living environments [5]. Hence, co-robots will not only play a crucial role in manufacturing, but also heavily influence the service sector.

The growth of assistive and collaborative robotics has substantiated the need for safe and reliable interaction between humans and robots. User-friendly Human-Machine Interfaces (HMIs) are crucial for assistive technology deployed in manufacturing or service industries such as healthcare [5,7,8]. The interfaces must be intuitive (i.e. easy-to-use) and reconfigurable, so that they can adapt to the user's preferences and needs, as well as different robot configurations. HMIs can range from a simple device to a software application that allows operators to control a complex robotic system with multiple degrees of freedom (DoF). The interface can also be physical in nature. For example, co-robots could guide humans in co-manipulation tasks such as welding [9], surgical procedures [10], or parts assembly [11]. The physical interface can also be used during Programming by Demonstration (PbD), in which an operator teaches a robot new tasks by manually moving its limbs through the desired motions [12,13]. Vice versa, a human can also be taught by the robot during physical

Human-Robot Interaction (pHRI), for example to improve motor skills in surgical procedures [10] and rehabilitation exercises [14].

Thus, current macro-scale developments in industry and the service sector strongly motivate the need for pHRI. However, human-robot collaborative tasks present difficult challenges related to safety, intuitiveness, and efficiency of the physical interaction.

1.2 Challenges imposed by Co-robots

Compared to non-collaborative robots, co-robots are specifically designed for safe pHRI by being lightweight and compliant [15, 16]. Recent advancements in hardware-based safety features include flexible and composite materials, gravity compensation, joint torque limiting, and bio-inspired hybrid actuation [17]. These methods have reduced contact forces and the risk of high energy collisions, however they make the robot dynamics highly nonlinear and difficult to model. In addition, the dynamics of the robot changes during physical contact, for example when picking up objects or interacting with a human. During such tasks, the robot controller has to continuously compute the necessary motor inputs or torques to achieve the desired motion. Typically, robot controllers are designed for trajectory control where the goal is to track a desired reference trajectory with the robot manipulator or end-effector over time. For better tracking, the systems are closed-loop and utilize feedback in form of a tracking error, e.g. the Euclidian distance between the reference or desired pose and the actual pose of the end-effector. However, most classical control techniques are not suited for complex collaborative tasks.

1.2.1 Robot Control for Nonlinear Systems

There are several groups or families of control schemes in robotics [18]. Inverse dynamics control computes a torque that cancels the nonlinear robot dynamics, hence it is also commonly known as computed-torque control. The nonlinearities are fed back into the control loop, thereby linearizing the system which allows tracking a reference trajectory with a simple Proportional-Derivative (PD) control law. However, this type of feedback linearization requires a fairly accurate model of the robot dynamics. Model-based controllers work well for rigid (industrial) robots, but are not adequate for flexible and compliant co-robots that have unknown dynamics. Robust control methods have better performance in terms of stability and tracking error, but can only deal with parameter uncertainty or disturbances within a certain range. Adaptive controllers estimate model parameters online and decrease tracking errors over time using runtime data. However, they generally require linear-in-the-parameters assumptions. Neural network (NN) control overcomes the linear parameterization property by approximating the unmodeled dynamics with nonlinear functions [19–24].

Feedback control with NNs was first proposed by Werbos [25] and Narendra et al. [26] in 1989 and 1990, respectively. In control theory, stability is of great interest. A system with finite inputs should not produce infinite outputs. For example, an unstable controller could cause a robot to move in a dangerous way resulting in physical damage. In 1995, Lewis et al. [27] provided initial proofs for internal stability, bounded errors, guaranteed tracking performance, and robustness. Later, the work was expanded for discrete-time control using multilayer NN [28]. Additional stability proofs were provided by Ge et al. [29], Chen [30], Polycarpou [31], Rovithakis and Christodoulou [32], Poznyak [33], Rovithakis [34] and others. Simulations of a 2-link robot manipulator have verified the NN controller's ability to successfully track a joint space trajectory [27]. However, due to the relative high computational complexity, the implementation and testing of NN controllers on real-time systems with several degrees of freedom (DoF) has been a major challenge. Therefore, these methods have lacked real-world validation and it was uncertain whether NN control was feasible for today's complex co-robots, which often have 5 or 6 links. Also, the NN controller with stability proof [27] had originally been developed for joint space and needed to be extended to Cartesian space. Since co-robots also interact with humans, the NN control approach also required extensions for pHRI.

1.2.2 Robot Control for pHRI

Another challenge arises when the robot physically interacts with its environment (or human operator). Control schemes used for tracking motion trajectories are best suited for free-space motion and often result in jitter and high interaction forces during contact. Thus, it is usually more important to control the interaction forces applied by the robot during physical interaction [35]. Robot force control is generally grouped into two categories: direct and indirect [36]. During direct force control, the controller explicitly closes the force feedback loop using measurements from Force/Torque sensors. In hybrid force/position control, the robot controls both motion and force, which requires a defined model of the robot and its task [37]. If the task details are unknown, parallel force/position control is more practical since it emphasizes the force over motion action. It utilizes an inner/outer-loop structure, where the outer force control loop closes around the inner motion control loop [36]. However, in general explicit force control methods can become unstable due to imperfect models and changes in the contact environment [18, 38, 39]. Indirect force control methods are easier to implement and force constraints are only applied when the robot deviates from a target position. The most popular form of indirect force control is impedance control, pioneered by Hogan [40]. It introduces a virtual mass-spring-damper between the actual robot and target position. By controlling the force applied to the environment, the interaction becomes stable and safe. A drawback with impedance control is the need for identifying the robot model and the environmental contact dynamics [38, 41, 42]. The reciprocal of impedance control is admittance control, which measures forces needed for realizing a desired position. It generates a so-called model trajectory, which allows for compliant HRI (without following a reference trajectory). The duality principle between admittance and impedance control is illustrated in Fig. 1.2. Since co-robots are deployed in many different environments for a variety of tasks, it is impractical to pre-program every possible scenario and limits the use of classical force control methods.

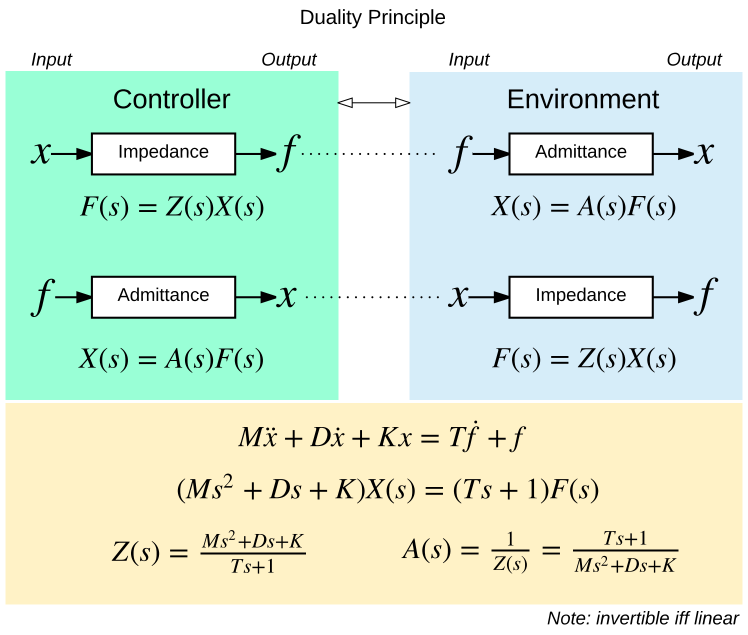


Figure 1.2. The duality principle between admittance and impedance control. The relationship between the input and output (X, F) can be represented by a second order transfer function (Z, A) with mass M, damping D, spring constant K, and time constant T.

Assuming that a model-free controller is able to track a desired motion and limit the interaction forces, there is still the challenge of making the pHRI intuitive and efficient. A co-robot should assist the user and minimize the operator's effort needed to complete a certain task. In addition, the interaction dynamics should behave in a predictable, consistent manner that is suitable for the task at hand. This presents several challenges. Firstly, since the human becomes part of the control loop during pHRI, the robot must be able to adapt to different human dynamics. Ideally, the robot should adapt to different operator characteristics and preferences without any extensive training. Secondly, the robot must be able to adjust its own dynamics to allow for task-specific behaviors. In summary, NN control of co-robots is promising, but requires additional features and must be incorporated into a larger controller framework to achieve safe, intuitive, and efficient pHRI.

1.2.3 Robot Sensors for Human Environments

During pHRI co-robots must be able to anticipate and sense interactions with users. The force applied by a human can be measured with conventional end-effector Force/Torque sensors, but this limits the contact area. Robot skin provides the robot with the sense of touch and allows multiple contact points. The data from tactile sensors or tactels, such as force location and magnitude during collisions and interactions, can be used with reactive controllers for safe interaction. Feedback from the tactile sensors can also be used for human intent prediction and execution of behaviors such as robot guidance during a collaborative task. A challenge with robot skin is the calibration required to achieve reliable measurements necessary for safe pHRI. To this end, the traditional method of calibrating each sensor prior to its use is a tedious task, especially with inexpensive, miniaturized hardware that can experience material degradation with time. In addition, the advancement of robot skin requires addressing several design problems for whole-body tactile sensor arrays. Since it is generally expensive and time consuming to develop and test new hardware, realistic simulators are important tools for tackling and solving such issues in an efficient manner. Hence, there is a need for scalable modeling approaches for simulating pressure-sensitive skin patches under the consideration of sensing element geometry and mechanical structure, signal quality, data processing, and force controller.

1.3 Objectives and Approach

The objectives of my graduate work were to address the challenges and problems described in the previous section and to offer a contribution to their solution. Since physical interaction is crucial for co-robots, the main research object was to develop a controller framework for safe, intuitive, and efficient pHRI. The following design principles were proposed:

- Principle 1: A pHRI controller must be safe, stable, and be able to adapt to different types of users and tasks.
- Principle 2: A highly nonlinear machine becomes easier to operate when behaving equivalent to a simple admittance model. This makes the interaction more intuitive and requires less effort by the user.
- Principle 3: The pHRI can be improved if the robot assists the user by minimizing the human force. This adds a safety factor in that the robot moves so as to reduce human effort.

Because co-robots are highly nonlinear systems, the NN controller with stability proof introduced by Lewis et al. [27] was used as a starting point. This method utilizes NN function approximation techniques to learn unknown dynamics online, and can be grouped under neuroadaptive (NA) control. In my thesis work, the NA controller was expanded to solve current challenges related to physical HMIs for co-robots. In particular, it was employed in an inner/outer-loop control architecture. In control theory, it is common to use multi-loop feedback control for complex systems [43]. If a process depends on multiple variables, the control can be simplified by treating each separately with nested feedback loops around specific sub-systems. Thus, the pHRI control problem was approached with a two-loop framework, comprised of a robot-specific inner-loop and a task-specific outer-loop (first published in [44] and then re-used by others in [45, 46]). In the controller inner-loop, the unknown robot dynamics is identified online and linearized through dynamic compensation. The outer-loop determines and adjusts an admittance based on task requirements and user skill. Furthermore, the outer-loop can be used for identifying human intent and generating a reference trajectory that the inner-loop follows, thereby proactively helping the operator complete the task.

Human-Robot Interaction involves a broad range of research areas including topics from robot mechanics to human modeling, requiring both theoretical and practical knowledge. To become familiar with robot hardware and software, a co-robot (Baxter) was compared with an industrial robot and a robotic platform (youBot) was sensorized to evaluate several HMIs. The PR2 robot was used for mobile manipulation and developing real-time controllers for pHRI. As my graduate work progressed, my focus shifted from more technical work to control theory and algorithm development. Since validation of the different approaches was necessary, new software packages and tools still had to be implemented and tested. Some programming statistics are summarized in Table 1.1.

1.4 Research Contributions

The contributions of this thesis are associated with the following four topics: (*i*) Investigation and evaluation of state-of-the-art co-robots and their interfaces, (*ii*)

Table 1.1. Programming statistics for the main software repositories. A single commit records several modifications made to the code base along with a message from the user describing the changes. The code base includes C++, Python, and ROS files.

Software tool	Commits
SkinSim	
Version 2.0	259
SkinLearn	
General robot functionality	567
Neuroadaptive controllers	596

validation and expansion of an online, model-free NN controller with stability proof, (*iii*) improvements of pHRI utilizing a two-loop neuroadaptive control architecture, and (*iv*) the development of new open-source software tools for HMIs and pHRI. My research accomplishments include 12 peer-reviewed articles. In addition, 2 are ready for submission and 1 in preparation. The research contributions and the corresponding publications are summarized below:

I. Co-robots and their Interfaces

- A Programming by Demonstration framework for kinesthetic teaching was developed and tested on a PR2 robot [13].
- Physical and non-physical interfaces were compared in a co-manipulation task with a PR2 robot [47].
- The Baxter co-robot was evaluated and compared to a traditional industrial manipulator in several experiments [48].
- A youBot robotic platform was sensorized for additional HMIs and their performance was investigated in a household environment [7, 8].

- A survey was conducted of state-of-the-art co-robots and application requirements for robotic nursing assistants [5].

II. Neuroadapative Control

- To achieve safe and stable pHRI, a NN controller was improved and extended beyond its original formulation (Lewis et al. [27]). A safety evaluation of neuroadaptive trajectory following and case study was published in [49].
- To achieve intuitive pHRI, the neuroadaptive controller was extended and tested with prescribed error dynamics [50].

III. Inner/Outer-loop Control Structure

- A two-loop architecture was developed to improve pHRI with co-robots. Work related to prescribed task model following and adaptive inverse filtering was published in [44], including experimental validation on a PR2 robot.
- The outer-loop controller was enhanced with a NN human intent estimator. Derived stability proofs and results from pHRI experiments will be published in [50].
- A major contribution of this thesis is the formulation of sensor-invariant outer-loop calibration of robot skin, which automatically tunes admittances of sensor arrays already mounted on a robot [51].

IV. Developed Software Tools

- SkinSim: A simulation environment for prototyping and testing robot skin originally conceptualized in [52]. I performed a major overhaul with new tactile modeling techniques and a completely new architecture. This work and a case study investigating the effect of pressure-sensitive skin array densities will be published in [53]. My code is available at `https://bitbucket.org/nextgensystems/skinsim`.

- SkinLearn: A unifying ROS framework for HMI and pHRI that was developed and tested on several robotic platforms (Baxter, youBot, and PR2). My code is available at `https://bitbucket.org/nextgensystems/skinlearn`.

1.5 Thesis Outline

The thesis outline and the relation between the different chapters is illustrated in Fig. 1.3. After the introduction, Part I provides background on co-robots and HMIs, while demonstrating the importance of pHRI. Robots that were evaluated and utilized in experiments include Baxter (Chapter 2), youBot (Chapter 3), and PR2 (Chapter 4). In Part II, the inner-loop neuroadaptive controller (NA) is presented. Chapter 5 describes the basic NA controller for reference trajectory following, and Chapter 6 its expansion with prescribed error dynamics. Part III highlights various outer-loop control schemes: The prescribed task and adaptive admittance model in Chapter 7, neural network human intent estimation in Chapter 8, and robot skin calibration employing adaptive inverse filtering in Chapter 9. Software developed during my thesis work is described in Part IV: SkinSim in Chapter 10 and SkinLearn in Chapter 11. Finally, Chapter 12 provides a summary and future research directions.

1.6 List of Publications

1.6.1 Related Contributions

1. **S. Cremer**, M. Middleton, and D. O. Popa, "Implementation of advanced manipulation tasks on humanoids through kinesthetic teaching," in *7th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '14*. ACM Press, 2014, pp. 1–6. [13]
2. **S. Cremer**, I. Ranatunga, and D. O. Popa, "Robotic waiter with physical co-manipulation capabilities," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 1153–1158. [47]

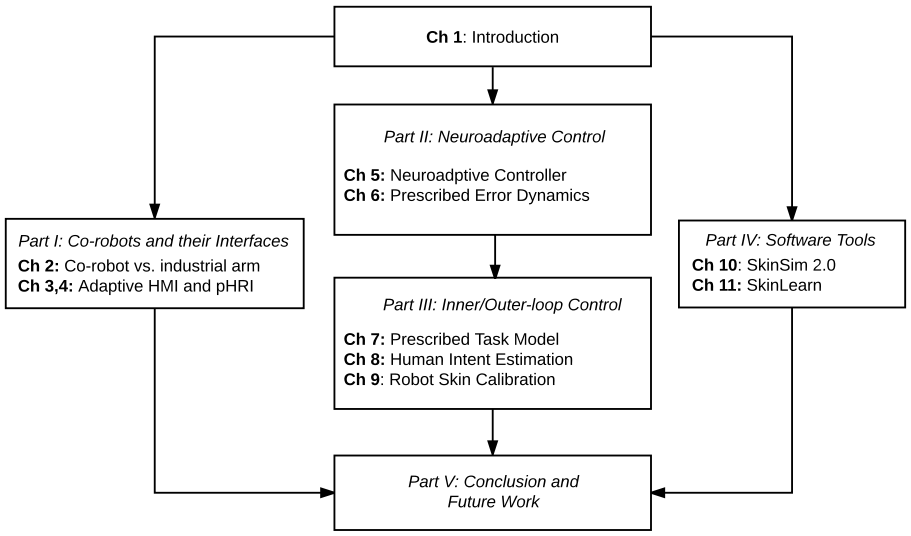

Ch 1: Introduction
Part II: Neuroadaptive Control
Ch 5: Neuroadptive Controller
Ch 6: Prescribed Error Dynamics
Part I: Co-robots and their Interfaces
Ch 2: Co-robot vs. industrial arm
Ch 3,4: Adaptive HMI and pHRI
Part IV: Software Tools
Ch 10: SkinSim 2.0
Ch 11: SkinLearn
Part III: Inner/Outer-loop Control
Ch 7: Prescribed Task Model
Ch 8: Human Intent Estimation
Ch 9: Robot Skin Calibration
Part V: Conclusion and Future Work

Figure 1.3. Thesis outline.

3. I. Ranatunga, **S. Cremer**, F. L. Lewis, and D. O. Popa, "Neuroadaptive control for safe robots in human environments: A case study," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 322–327. [49]
4. R. Alonzo, **S. Cremer**, F. Mirza, S. Gowda, L. Mastromoro, and D. O. Popa, "Multi-modal sensor and HMI integration with applications in personal robotics," in *Proc. SPIE*, 2015, p. 8. [7]
5. I. Ranatunga, **S. Cremer**, D. O. Popa, and F. L. Lewis, "Intent aware adaptive admittance control for physical Human-Robot Interaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5635–5640. [44]
6. **S. Cremer**, L. Mastromoro, and D. O. Popa, "On the performance of the Baxter research robot," in *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 2016, pp. 106–111. [48]
7. **S. Cremer**, I. Ranatunga, S. K. Das, I. B. Wijayasinghe, and D. O. Popa, "Neuroadaptive Calibration of Tactile Sensors for Robot Skin," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1079–1085. [51]
8. **S. Cremer**, K. Doelling, C. L. Lundberg, M. McNair, J. Shin, and D. Popa, "Application requirements for Robotic Nursing Assistants in hospital environments," in *Proc. SPIE*, vol. 9859, 2016, p. 10. [5]

9. **S. Cremer**, F. Mirza, Y. Tuladhar, R. Alonzo, A. Hingeley, and D. O. Popa, "Investigation of human-robot interface performance in household environments," in *Proc. SPIE*, vol. 9859, 2016, p. 13. [8]
10. I. B. Wijayasinghe, S. Peetha, S. Abubakar, M. N. Saadatzi, **S. Cremer**, and D. O. Popa, "Experimental setup for evaluating an adaptive user interface for teleoperation control," in *Proc. SPIE*, vol. 10216, 2017, p. 10. [54]

1.6.2 To Be Submitted

1. **S. Cremer**, D. O. Popa, and F. L. Lewis, "Model-Free Online Neuroadaptive Control with Intent Estimation for Physical Human-Robot Interaction," *To Be Submitted: IEEE Transactions on Robotics (T-RO)*, 2017. [50]
2. **S. Cremer**, I. B. Wijayasinghe, S. K. Das, and D. O. Popa, "SkinSim 2.0: A Design and Simulation Tool for Robot Skin with Closed-loop pHRI Controllers," *To Be Submitted: IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2017. [53]

1.6.3 In Preparation

1. **S. Cremer**, D. O. Popa, and N. Bugnariu, "Neuroadaptive pHRI for stroke rehabilitation," in *TBD*, 2017. [55]

1.6.4 Other Contributions

1. J. Sanford, O. Yetkin, **S. Cremer**, and D. Popa, "A novel EMG-free prosthetic interface system using intra-socket force measurement and pinch gestures," in *8th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '15.* ACM Press, 2015, pp. 1–8. [56]
2. J. Sanford, C. Young, **S. Cremer**, D. Popa, N. Bugnariu, and R. Patterson, "Grip Pressure and Wrist Joint Angle Measurement during Activities of Daily Life," *Procedia Manufacturing*, vol. 3, pp. 1450–1457, 2015. [57]

Part I

Co-robots and their Interfaces

CHAPTER 2

On the Performance of the Baxter Robot

ON THE PERFORMANCE OF THE BAXTER ROBOT

S. Cremer, L. Mastromoro, and D. O. Popa, “On the performance of the Baxter research robot,” in *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 2016, pp. 106–111.

2.1 Abstract

The rise of collaborative industrial robotics in the past few years is poised to have a significant impact on manufacturing. Many have joined the movement toward developing a more sustainable and affordable collaborative robotic workforce that can safely work alongside humans; perhaps, none more so than Rethink Robotics. This paper provides a performance assessment of Rethink's Baxter research robot: its kinematic precision was compared experimentally to that of a conventional industrial arm utilizing a point-to-point motion test and writing task. In addition, the Denavit-Hartenberg (DH) parameters were determined and verified in order to model the kinematic chains of the robot arms. Finally, the yield of pick and place tasks consistent with the 2015 IEEE Amazon's Picking Challenge were assessed. Results show that while Baxter's precision is limited, its ability to handle common household-size items in semi-structured environments is a great asset.

2.2 Introduction

In 2008, Heartland Robotics was co-founded by Rodney Brooks and Ann Whittaker with a clear view to develop a new class of robotic platforms that would keep manufacturing jobs in America [58]. These robots were to have an intuitive user interface as well as being "capable of autonomously sensing and adapting to their environment". The vision was to design a manufacturing robot that was easy to train, and "much less expensive than traditional industrial robots" [58]. After nearly 5 years of development and renaming the company to Rethink Robotics, the Baxter robot was the first realization of this vision [59].

With a 360-degree head sonar suite, two 7 degrees of freedom (DOF) arms capable of using interchangeable end-effectors, and an intuitive more behavior-based user interface, Baxter was designed to work alongside factory workers [60]. Six months after release of the Baxter industrial robot, in April 2013, a research version of the robot was released to promote the evolution of Baxter with a Software Development Kit (SDK) and interface developed around the open source Robot Operating System (ROS) [61].

Early testing by the industrial community confirmed expectations [59], and there seemed to be agreement that natural interaction and ease of programming were some of Baxter's biggest assets. A great deal of information about Baxter has been published over the last two years as more researchers have used the robot and shared their results [62]. In 2016, a single-arm counterpart called Sawyer was released for smaller, high-precision tasks. Other companies have also joined the race and built similar co-robots. Right around the time Baxter was released, the Swedish based company ABB introduced a dual-arm concept robot named FRIDA (Friendly Robot for Industrial Dual-Arm). The key properties of the robot were safe, accurate, and agile movements, flexibility, and ease of deployment and reconfiguration. Unlike Baxter though, the user interface relied on a programming controller and the robot was physically smaller and portable [63].

Another competitor to Baxter called Nextage was developed by the Japanese firm Kawada Industries [64]. The robot's design included a head and two 6-DOF arms attached to a torso mounted on a mobile base. Stereo vision was implemented and torso LEDs were used to assure visibility of the robot's status. Cameras were placed on each end effector that could capture 3D information [65]. To ensure safety next to factory workers, Nextage employed another feature: the robot's axle structure, which guaranteed that the elbows would never move beyond its working environment. The robot's "15 operational axes (6 per arm, 2 for the head, and 1 for the torso) use low-power motors of 80 watts to move", which also prevent harmful forces to other factory workers or fellow robots [66].

The Danish company Universal Robots entered the collaborative robotic market with six-axis arms that allowed ample flexibility for a variety of tasks. Two variants of the robot, the UR5 and UR10, were designed to handle different amounts of weight, 5 and 10 kg respectively [67]. Unlike Baxter (with an advertised position accuracy of ± 5 mm) [60], the UR was very precise (± 0.1 mm) [67]. To program a trajectory, the operator had to move the arm and record points using a 12 inch touch-screen tablet. To ensure safety in case of collision, the robot delivered a force of less than 150 N (33.72 lbs). The robot could also be customized for the client's needs [67].

Robotiq Corporation has developed a useful comparison of a select number of collaborative robots now being used in industry, which compares Baxter's specifications and performance to those of other competitors [66]. Interestingly, all have features designed to better ensure collaboration with a human workforce.

The primary contribution of this paper is to provide an objective set of experiments to evaluate Baxter's performance in part handling tasks and more precision-demanding operations. During this work, a kinematic model was validated and used to emulate Baxter in simulation.

An experiment based upon an industry accepted test method [68] was carried out to validate the repeatability of Baxter's arm positioning while performing redundant point-to-point movements. A similar point-to-point experiment using a Denso industrial robot was conducted to compare results with Baxter. Baxter's ability to write a simple rectangle on a flat surface was also evaluated and compared to the same Denso robot. Further testing was conducted while using the Baxter Research Robot in the 2015 IEEE Amazon Picking Challenge, where pick-and-place operations of common household items were performed.

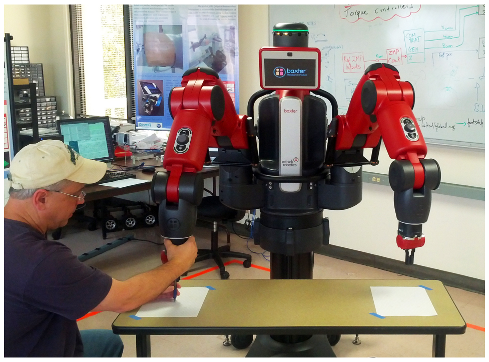


Figure 2.1. The Baxter robot being setup for a writing precision test.

In this work, one of the objectives was to determine whether replacing non-collaborative industrial robots with closely priced co-robots, like Baxter, is feasible for certain types of tasks. The results indicated that a strong case can be made for using collaborative robotic platforms like Baxter in less precise pick-and-place operations where position accuracy of a few millimeters is adequate.

This paper is organized as follows: in Section 2.3 the initial modeling of Baxter is detailed, in Section 2.4 the repeatability experiments on Baxter are discussed, in Section 2.5 the experimental setup and results of testing the Denso industrial robot are outlined, and in Section 2.6 the work completed for participation in the Amazon Picking Challenge is shared. Finally, the conclusions and suggested future work are delineated in Section 2.7.

2.3 Modeling Baxter

A kinematic model for Baxter was developed and compared to real-time data captured in several experiments. The approach was to create a Denavit-Hartenberg (DH) parameter table for Baxter's arms and to subsequently use Peter Corke's open-source robotic toolbox for MATLAB [69] to analyze the forward kinematics, inverse kinematics, Jacobian, and singularities associated with Baxter's arms. The parameters were determined from Baxter's XML-based Universal Robot Definition File (URDF) [60] and physical measurements as shown in Fig. 2.2. The DH parameters listed in Table 2.1 are similar for both arms since the joint variables have identical signs for the same direction of motion.

In MATLAB, the DH parameters were used to create a 7-DOF model resembling Baxter's arm. To validate the kinematic model, the joint angles and associated end-effector pose data were recorded in a 10-point pose experiment, described in Section 2.4. The positions saved were compared to those predicted by the model, which were calculated using the robotic toolbox *fkine* function. It performs the forward kinematics and computes the end-effector pose given a joint angle configuration.

The performance was evaluated by computing the 2-norm distance between the calculated pose (x_c, y_c, z_c) and the observed pose (x_o, y_o, z_o) with

$$e = \sqrt{(x_c - x_o)^2 + (y_c - y_o)^2 + (z_c - z_o)^2} \tag{2.1}$$

for several data points. The average error was within 1.5 mm as shown in Table 2.2. Note that the pose orientations were not considered in the calculations. The results provided confidence that Baxter's kinematics could be modeled and animated with a reasonable measure of fidelity. Related work [70] produced similar results when modeling Baxter.

2.4 Baxter's Repeatability Experiments

A set of experiments was carried out to determine how well Baxter repeatedly follows instructions for a fine pick-and-place operation. A 10-point dual-plane task was developed with respect to details provided in ISO standard 9283:1998 [68], as depicted in Fig. 2.4. For the lower plane, a grid was drawn with a black pen on

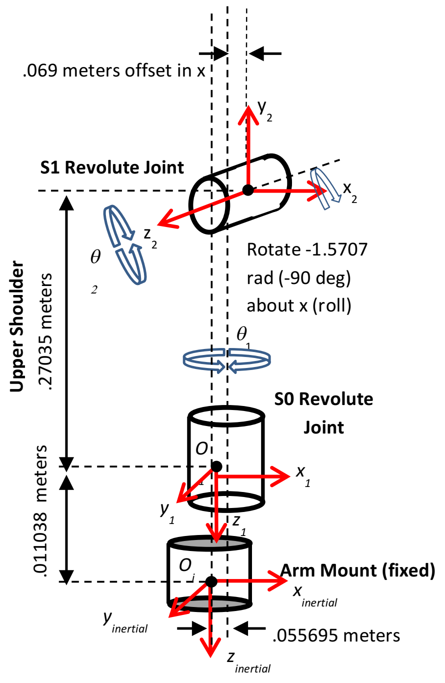
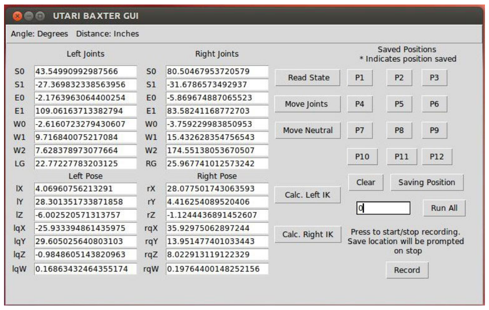


Figure 2.2. Physical translation of Baxter's joints.

Table 2.1. DH parameters for the baxter robot arm extracted from the URDF and physical measurements.

Link	a_i	α_i	d_i	θ_i
1 (S_0)	0.069	-1.571	0.2703	θ_1
2 (S_1)	0	1.571	0	θ_2
3 (E_0)	0.069	-1.571	0.3644	θ_3
4 (E_1)	0	1.571	0	θ_4
5 (W_0)	0.01	-1.571	0.3743	θ_5
6 (W_1)	0	1.571	0	θ_6
7 (W_2)	0	0	0.2295	θ_7



Angle: Degrees   Distance: Inches

	Left Joints		Right Joints
S0	43.54990992987566	S0	80.50467953720579
S1	-27.369832338563956	S1	-31.6786573492937
E0	-2.1763963064400254	E0	-5.869674887065523
E1	109.06163713382794	E1	83.58241168772703
W0	-2.6160723279430607	W0	-3.759229983850953
W1	9.716840075217084	W1	15.432628354756543
W2	7.628378973077664	W2	174.55138053670507
LG	22.77227783203125	RG	25.967741012573242

	Left Pose		Right Pose
lX	4.06960756213291	rX	28.077501743063593
lY	28.301351733871858	rY	4.416254089520406
lZ	-6.002520571313757	rZ	-1.1244436891452607
lqX	-25.933394861435975	rqX	35.92975062897244
lqY	29.605025640803103	rqY	13.951477401033443
lqZ	-0.9848605143820963	rqZ	8.022913119122329
lqW	0.16863432464355174	rqW	0.19764400148252156

Saved Positions
* Indicates position saved

0

Press to start/stop recording.
Save location will be prompted on stop

Figure 2.3. The Python graphical user interface (GUI) for testing.

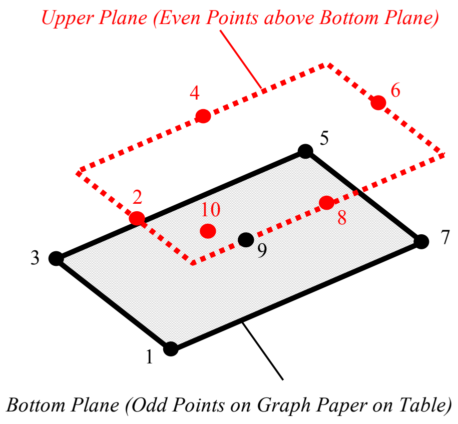

Upper Plane (Even Points above Bottom Plane)
4
6
5
2
10
8
3
9
7
1
Bottom Plane (Odd Points on Graph Paper on Table)

Figure 2.4. Layout of the 10-point test

standard graph paper with 5 mm by 5 mm squares. A 100 mm by 100 mm square with corner points was used with a center point. The grid points were numbered 1,3,5,7, and 9 (clockwise). The graph paper was affixed to a leveled table in front of Baxter for the test (Fig. 2.1). The upper plane was created when Baxter was manually trained. For the upper plane, intermediate points above and between each odd point on the bottom grid were created approximately 6 inches above the bottom plane. The points were designated 2,4,6,8, and 10, respectively (Fig. 2.4). The test sequence moved Baxter's arm sequentially from point to point in numeric order, beginning with point one and ending on point one for each iteration.

The Python developed graphical user interface (GUI) was used to collect data relevant to this experiment (Fig. 2.3). Test boxes displaying the current joint angles and pose data were included as well as provisions to save points. The interface was developed to include the ability to clear the saved points, to enter the number of loops the test sequence would perform before stopping, and to record the test sequence data to a CSV file. As Baxter's arms were manually moved to each of the ten positions the points were saved.

After all the points were saved, the number of iterations was entered in and the 'Record' button was depressed to ensure recording started prior to running the test sequence. Data was captured at a 10 Hz rate. Recording was stopped after the test sequence was completed by simply depressing the 'Record' button again.

In keeping with ISO 9283:1998 [68], a 30 iteration loop count was used for testing (thirty 10-point movements for both of Baxter's arms was performed for each test trial). Multiple test trails were performed and compared. For testing, a worst case scenario was implemented with Baxter not having been calibrated for over a month and without the use of the arm's cameras to aide in visual servoing. Baxter's precise joint sensors were used as the measuring stick for repeatability. The resolution of each of Baxter's "joint sensors is 14 bits over 360 degrees," equating to 360/(214) degrees per tick resolution (0.021972656 degrees/tick) [60]. Each joint has sinusoidal non-linearities which may contribute to the sensor's ability to output the correct position at a worse case accuracy of ±0.25 degrees (which will vary from joint to joint). An absolute zero-offset of up to 0.10 degrees may also be resident in the joint sensor readout which can be minimized through calibration [60]. The Baxter RSDK *settings.py* file JOINT_ANGLE_TOLERANCE parameter defines a threshold of 0.00872664626 radians (0.5 degrees) by default. The value was not changed for the experiment.

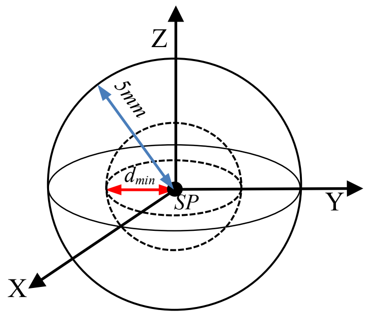


Figure 2.5. Threshold region for measuring 3D position accuracy.

Table 2.2. Calculated to observed position differences (meters).

Point	Left arm error (m)	Right arm error (m)
1	0.0014	0.0015
3	0.0008	0.0014
5	0.0011	0.0014
7	0.0012	0.0011
9	0.0008	0.0014

A ± 5 mm threshold window illustrated in Fig. 2.5 was used for the analysis of each approach to a saved point (SP) based upon the robot's advertised position accuracy [60]. The total time spent within the threshold window for each pass was considered for a set of data (each of the 10 saved points had 30 values per test trial).

The minimum distance (d_{min}) between Baxter's measured position and the original saved position (the 2-norm distance) was then extracted from the 10 Hz samples in the set. The time it took Baxter to find the minimum position once the end-effector entered the threshold window (t_{find}) was also calculated to better observe the robot's ability to finely resolve its position once it was close to the saved point. The total time the arm spent within the threshold window was also derived (t_{total}). The time it took Baxter to travel between successive minimum positions (t_{f2f}) for the same saved

Table 2.3. 10-point test results mean and standard deviation.

MEASURE	d_{min} (mm)	t_{find} (sec)	t_{f2f} (sec)	t_{total} (sec)
Mean	1.4	1.73	53	3.6
Standard Deviation	0.6	0.54	0.7	2.8

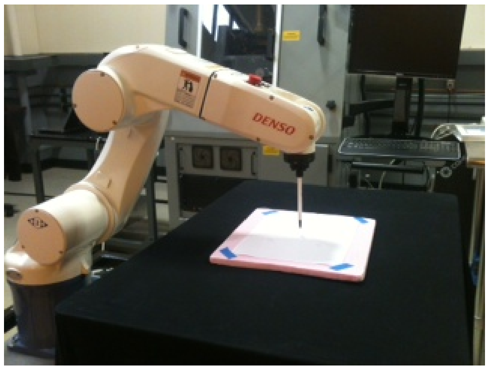


Figure 2.6. Denso VS-6577GM robot performing a writing test.

point was calculated to observe how repeatable Baxter traveled around the 10-point test grid. The same process was employed for both of Baxter's arms. The d_{min}, t_{find}, t_{f2f} , and t_{total} values were finally averaged and a standard deviation was calculated for each of the values from each set (Table 2.3).

2.5 Comparison with a Standard Industrial Arm

To compare Baxter's performance with another non-collaborative industrial robot, a 10-point test grid was set up for a Denso VS-6577GM industrial robot. The Denso robot has a 30 µm advertised pose position accuracy [71]. A special fixture was 3D printed to hold a pen at the end effector to ensure the end-effector was lined up with the points as shown in Fig. 2.6. The test pattern was trained into the robot by means of a teaching pendant. Once the setup process was complete, the 10-point test was performed at half speed (the Denso is capable of moving up to 7 m/s) [71].

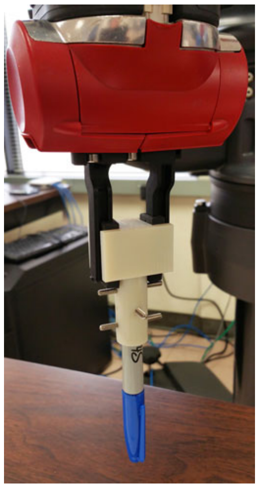


Figure 2.7. The 3D printed pen mount for the Baxter gripper.

The Denso was very accurate (less than 1 mm repeatable accuracy as measured on the graph paper) and notably faster than Baxter.

Adequate space around the manipulator was roped off to ensure safety, since the Denso is not a collaborative robot (it will not stop if someone gets in the way of the arm while moving). Typical stop times to position the robot at each test point while throttled to half speed were observed to be less than one second.

A rectangle write experiment was also performed using Denso and Baxter. The same Denso manipulator was used to write a rectangle on a piece of graph paper using a permanent marker and the same 3D printed pen fixture.

For the rectangle write test, a special 3D printed pen fixture was designed and printed for Baxter. The fixture was specifically designed to allow Baxter's grippers to more firmly hold the same permanent marker (Fig. 2.7).

The Denso industrial robot was programed via a teaching pendant to draw a rectangle on a table placed in front of it. The same rectangle write operation was performed on Baxter using both Baxter's demo mode Record and Playback feature and another specially developed C++ based interface which utilizes the *MoveIt!* [72]

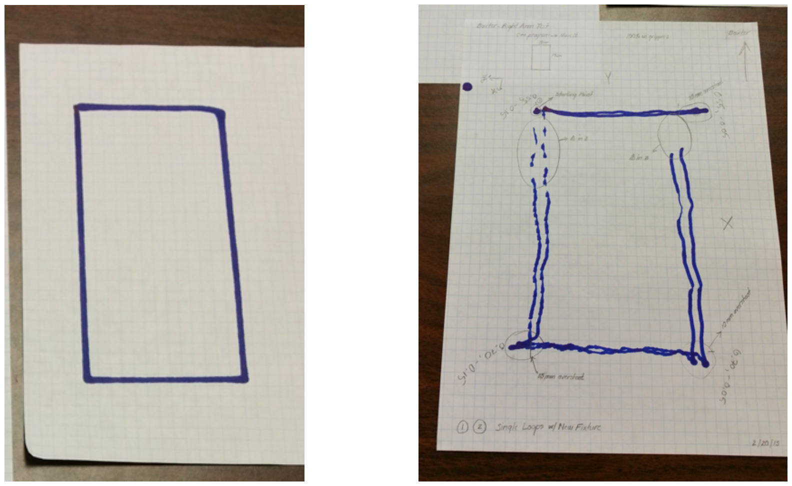

(a) (b)

Figure 2.8. Write test results (Denso left and Baxter on the right).

motion planner application. Both rectangles were drawn on graph paper taped to a level table placed in front of the robot.

The Denso drawn rectangle was very precise with no observable overshoot on the corners (67 mm × 138 mm). The Baxter rectangles (100 mm × 150 mm) were somewhat repeatable in iterative drawing attempts, but clearly could not match the Denso results as shown in Fig. 2.8. Overshoot on the corners and deviations to follow a straight line were observed. Using Baxter's default control parameter settings, up to 10-20 mm of overshoot consistently occurred as Baxter's arm attempted to stop and start at the rectangle corners. Baxter's control parameters were not modified for the test. The manual training mode produced similar results. It should be noted that visual servoing was not utilized on either Denso or Baxter for the test.

Perhaps, the most important fact obtained from the comparison was the repeatable way Baxter drew the rectangle. Each pass was almost identical to the previous one, down to the amount of overshoot and breaks in the lines where the pen was vertically lifted off of the paper. Variations in Baxter's end-effector z-position (vertical) either caused no writing on the paper (pen position too high) or an aberration to the line drawn from excessive friction of the pen pressing too hard on the paper (pen position too low). Repeated runs produced very similar deviations (nearly on top of each other) with respect to these effects.

2.6 Amazon Picking Challenge

Several teams competed with the Baxter robot in the 2015 International Conference in Robotics and Automation (ICRA) Amazon Picking Challenge (APC) [73]. In the competition, participants were tasked to build hardware and/or develop software that autonomously picks a subset of items from a stationary shelf and places them in an order bin. This required solving problems related to perception, planning, manipulation, error detection, and error correction. At the event, companies such as Rethink Robotics, Clearpath, Universal Robots, Yaskawa, and Fanuc provided platforms for the contestants [74].

In 2012, Amazon acquired Kiva Systems, a mobile robotic company for material handling [75]. Currently, their robots are used to move shelves from storage locations to a human worker at a picking station. Typically, a shelf has several bins that are tightly packed with up to 10 items. The worker has to identify and retrieve the ordered item, scan it for verification, and then place it in an order bin. The goal is to automate the process [74].

Retrofitting a warehouse with robots can be costly. Collaborative robots, such as Baxter, can easily be deployed and are already designed for the human workspace [59]. Since they can work side by side to humans, the robotic workforce can slowly be expanded as funding permits. However, as stated previously, there is still a question as to whether collaborative robots like Baxter are accurate and fast enough to compete with more precise non-collaborative industrial robots in every situation. To further answer this question, Baxter's performance within the APC environment, a simple pick and place task scenario, was evaluated.

The APC designed picking station was a 2×2 meter area containing the robot and the order bin. The shelf was located in front of the robot and had 12 bins that were packed with 1 or more items [73]. The items were placed such that they did not occlude each other. The objects were chosen from a set of 25 common items pre-selected and placed randomly in bins. During our first experiments, the item to be picked was placed in the same location for each trial in order to remove errors introduced by the perception pipeline. In our trials, the item was localized using LINE-MOD [76], which is part of the Object Recognition Kitchen [77]. The algorithm uses template matching and identifies the pose in real-time with the help of an object database containing trained 3D models. The location was sent to the MoveIt! motion planner application, which was also used in the rectangle write test [72]. A Cartesian waypoint trajectory was computed from the starting location to the object, and from the object to the order bin. This path was stored and then executed 10 times with enough time in between to allow an operator to place back the removed item.

Performance measures for the experiment included the success rate and completion time. A trial was determined to be successful whenever the item was autonomously transferred from the shelf to the order bin (Fig. 2.9). The time measured included Baxter starting from a neutral position, moving the item, and then moving back to neutral. A total of 4 different items were picked 10 times. The size of the electric gripper was selected to accommodate the width of the object. The objects were placed at the (x, y, z) locations $(1.0, 0.0, 0.2)$ and moved to $(0.4, 0.8, 0.0)$, viewed from Baxter's base frame and measured in meters.

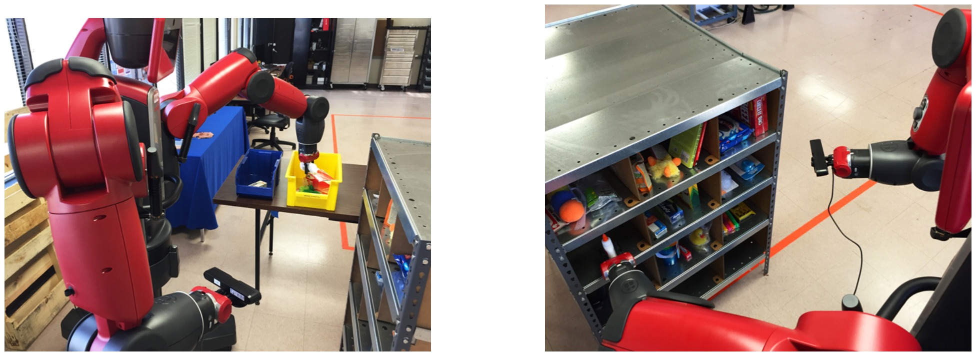

(a) (b)

Figure 2.9. Baxter picking household items from a shelf during APC test trials.

Table 2.4. Baxter pick and place testing within the APC environment.

Object	Success	Mean (sec)	STD (sec)
Elmers Glue	10/10	26.963	1.667
Book	10/10	27.000	2.540
Crayons	9/10	26.766	1.406
Soda can	10/10	26.569	1.674
Total	39/40 =97.5%	26.825	1.806

It was observed that one out of the 40 trials was unsuccessful (Table 2.4); the crayon box was dropped once because it slipped out of the grippers. An average time of 26.8 seconds with a standard deviation of 1.8 seconds was noted.

Comparatively, Baxter performed slower than a human worker, who would typically take just a few seconds for the same task. However, to Baxter's credit, a human would require breaks and the robot does not. Consequently, since Baxter repeatedly operated with enough accuracy when given the correct location of an item, it was considered a viable option for this task set in a factory setting

2.7 Conclusion

Based upon our experiments, we conclude that applications where higher position accuracy ($< 1\,\mathrm{mm}$) and speed are a factor, closely priced, non-collaborative robots like Denso may still be the best choice, though considerations will have to be made for a more complex setup and safety. Our results also demonstrated however, that while Baxter's position accuracy may be limited compared to other comparatively priced collaborative robots, its ability to safely handle common household-size items in semi-structured environments is a great asset which would make it suitable for many material handling operations native to factories and similar settings.

Further work includes comparing Baxter's performance against other collaborative robots and improving Baxter's positioning capabilities, such as its dynamics arm controller and trajectory planning.

2.8 Acknowledgment

This research was supported by NSF Grant NRI-1208623. Discussions with Dr. Mark Killpack at BYU, and Dr. Woo Ho Lee at UTARI were very helpful. The authors wish to thank Joshua Kornblum for help with Python scripting, and Kris Doelling at UTARI for assistance with the Baxter experiments presented in this paper.

CHAPTER 3

Human-Machine Interfaces

3.1 Introductory comments

This chapter describes the sensorization of a youBot robotic platform and provides an overview of different HMIs, which was published in [8]. Section 3.5 describing sensors for HMI has been supplemented with material from [5], which is now presented in 3.5.1 Navigation, 3.5.2 Manipulation, 3.5.3 Detection of Human Activity, 3.5.4 Human-Robot Communication, and 3.5.5 Robot Skin. To limit the scope of this thesis, descriptions of hospital environments and nursing tasks have not been included from [5].

INVESTIGATION OF HUMAN-ROBOT INTERFACE PERFORMANCE IN HOUSEHOLD ENVIRONMENTS & APPLICATION REQUIREMENTS FOR ROBOTIC NURSING ASSISTANTS IN HOSPITAL ENVIRONMENTS

S. Cremer, F. Mirza, Y. Tuladhar, R. Alonzo, A. Hingeley, and D. O. Popa, "Investigation of human-robot interface performance in household environments," in *Proc. SPIE*, vol. 9859, 2016, p. 13.

S. Cremer, K. Doelling, C. L. Lundberg, M. McNair, J. Shin, and D. Popa, "Application requirements for Robotic Nursing Assistants in hospital environments," in *Proc. SPIE*, vol. 9859, 2016, p. 10.

3.2 Abstract

Today, assistive robots are being introduced into human environments at an increasing rate. Human environments are highly cluttered and dynamic, making it difficult to foresee all necessary capabilities and pre-program all desirable future skills of the robot. One approach to increase robot performance is semi-autonomous operation, allowing users to intervene and guide the robot through difficult tasks. To this end, robots need intuitive Human-Machine Interfaces (HMIs) that support fine motion control without overwhelming the operator. In this study we evaluate the performance of several interfaces that balance autonomy and teleoperation of a mobile manipulator for accomplishing several household tasks.

Our proposed HMI framework includes teleoperation devices such as a tablet, as well as physical interfaces in the form of piezoresistive pressure sensor arrays. Mobile manipulation experiments were performed with a sensorized KUKA youBot, an omnidirectional platform with a 5 degrees of freedom (DOF) arm. The pick and place tasks involved navigation and manipulation of objects in household environments. Performance metrics included time for task completion and position accuracy.

3.3 Introduction

Robotics is currently in its 3rd phase of development (Fig. 3.1). The late 1970s saw the rise of unintelligent, stationary industrial robots. During the 1990s, progress was made in mobility, intelligence, and cooperation to develop "personal" robots in areas such as research, education, and entertainment [78]. Today is the generation of "ubiquitous" robots that will support humans in everyday life. Unlike industrial robots, future co-robots will share their working space with humans and work in household environments. These assistive devices do not necessarily have to be fully

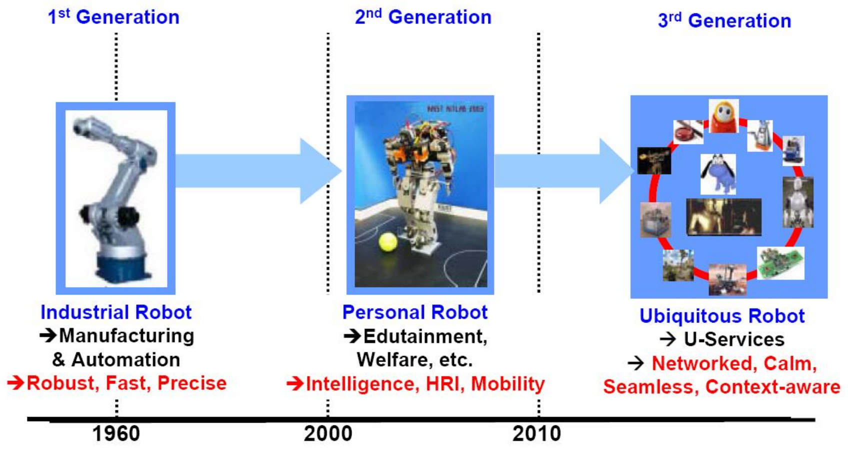


Figure 3.1. Development trends in robotics [78].

autonomous. It has been shown that a human-robot pair can outperform either a human or robot working alone [79].

Therefore, intuitive human-robot interfaces will play a crucial role. Non-expert users must be able to interact and communicate with the robot, which may involve speech, gesture, haptic displays, etc. [80]. This interaction can also be physical. For example, a robot can learn a new task from a novice operator via kinesthetic teaching, where the user manually pushes and pulls the robot's manipulator to complete a task [13].

Since physical contact may occur, human-robot interfaces will also play a key role in safety. Safety can be divided into a physical and behavior aspect [81]. When physical contact occurs, the control architecture can limit joint torques and velocities, and take advantage of passive compliance. There are adaptive schemes that can compensate for unknown parameters and disturbances while guaranteeing robust and stable control [82]. In our recent work, we validated a novel neuroadaptive framework that improved physical HRI [44, 83]. These methods use single F/T (Force/Torque) sensors to estimate the force applied by a human on the robot end-effector. A promising technology is robot skin, which equips the robot with touch and allows precise localization of multiple contact forces [16]. While whole-body, human-like skin has yet to be realized, pressure sensitive piezoresistive “taxel” arrays encapsulated in flexible silicone substrates already exist [84].

For behavior safety, the robot takes multi-modal cues such as facial expression and body pose to determine human intent. This allows the robot to align its goals with the operator, plan ahead, and adjust its behavior. For example, if a collision is anticipated or a child interacts with the robot, the control scheme could increase its compliance. Feedback from the operator could also allow the robot to behave more human-like, making the interface simpler to understand and more intuitive for the operator [15].

The intuitive and safe human-robot interfaces must be facilitated by multi-modal sensor data. The robot can perceive and understand its environment through camera images that are then processed by vision algorithms. For example, the OpenCV library implements several real-time computer vision functionalities, including people detection and face tracking [85]. Additional information can be gained from depth: RGB-D cameras such as the Microsoft Kinect or Asus Xtion have become the de-facto standard in robotics [86]. Laser scanners usually provide 2D depth information and are commonly used for mobile navigation. Sonar or ultrasonic sensors are less accurate but relatively inexpensive range-finders, and have been deployed on co-robots such as Baxter to detect human in the workspace and surrounding areas [87]. Another alternative is thermal sensors, which can detect radiated heat or far-infrared rays from nearby humans. In addition to low cost, processing infrared data is fast compared to image processing for human detection.

In this chapter, a standard robot platform was modified to investigate the performance of several HMIs in a household environment, continuing the work described in [7]. The KUKA youBot, which is commonly used in academia, was transitioned from the "personal" to "ubiquitous" generation of robots by performing several hardware upgrades. The platform was sensorized by installing an RGB-D camera for object and human detection, a laser scanner for fully and semi-autonomous navigation as well as collision detection, robot skin patches consisting of piezoresistive pressure sensors for physical interaction, and thermal sensors to detect a human's presence. A National Instruments roboRIO was added to provide additional input/output ports and software was developed to process incoming data. The network infrastructure was improved to allow remote control via a tablet interface and assisted joystick teleoperation. As such, the contribution of this chapter is how to sensorize a standard robot platform (both hardware and software) for human-robot interfaces and their expected performance in a household environment.

The following section (3.4), describes the robot platform and how it was upgraded to meet the HMI requirements. In Section 3.5, we discuss the sensors used for the multi-modal interfaces and the software architecture in Section 3.6. Section 3.7 describes the experimental setup for evaluating the HMIs performance in a household environments and the results.

3.4 The youBot Hardware Platform

This section describes the robot platform and the necessary hardware sensors and interfaces needed for teleoperation. The automation company KUKA specifically developed the youBot as a research and application platform for mobile robotics [88]. It has a five degrees of freedoms (DoF) manipulator with a height of 655 mm and a workspace of 0.513 m^3. The arm can lift a payload up to 0.5 kg and has a position repeatability of 0.1 mm. Grasping can be performed with a two-finger gripper that is being powered by 2 independent stepper motors and has a range of 70 mm.

The arm is mounted on an omnidirectional base of dimension 580 mm (length) by 376 mm (width). The four Mecanum wheels allow movement in any direction (x, y) at any orientation (θ). It can reach velocities up to 0.8 m/s and carry a payload of 20 kg. The mobile base houses a rechargeable battery which allows a runtime of approximately 90 minutes. An onboard mini PC (Intel Atom Dual Core CPU, 2GB RAM) runs the Linux-based operating system Ubuntu. The platform components rely on EtherCAT communication with a 1 ms real-time cycle.

A remote workstation can be connected to the robot via Ethernet cable. This is suitable for running computationally demanding algorithms and heavy graphics processing that would overload the onboard PC. To create an untethered setup, our first upgrade consisted of mounting a wireless router (ASUS RT-N66U Dual-Band Wireless-N900 Gigabit Router) on top of the base. This router is connected to the onboard PC via Ethernet connection, and acts as a bridge to transmit data to a remote workstation (Fig. 3.2). This allows an operator to run and troubleshoot processes remotely, view and collect live data, and visualize the robot state in a graphics program. In addition, the youBot WiFi network can be used to connect interface devices such as tablets, phones, or laptops.

The additional sensors for the HMI require Input/Output (I/O) ports for signal acquisition, conditioning, and networking. The National Instrument roboRIO is an advanced robotics controller that features several built-in ports for "I^{2}C, SPI, RS232, USB, Ethernet, PWM, and relays" [89]. Released in 2015, it is part of the reconfigurable I/O (RIO) family and use the Xilinx Zynq chipset. It has a reconfigurable FPGA and is powered by a 667 MHz dual-core Real-Time processor.

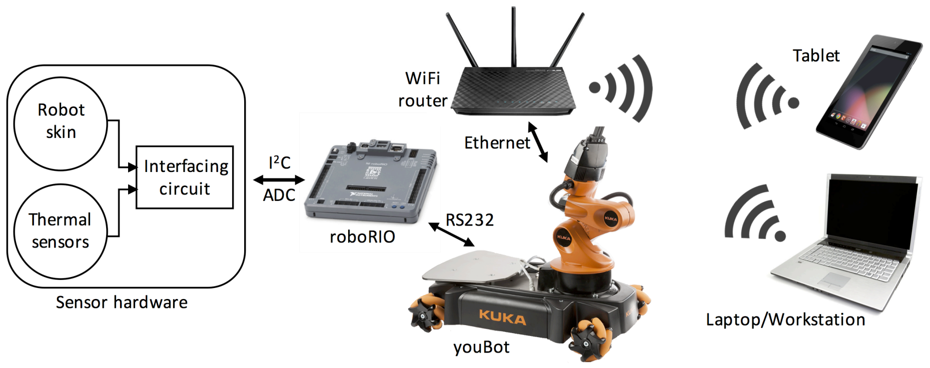


Figure 3.2. Communication diagram for sensor data acquisition and wireless setup between youBot and remote workstation.

3.5 Sensors for HMI

Several sensors were mounted and integrated with the youBot platform to facilitate human-machine interaction (HMI), including a laser scanner, robot skin, and thermal sensors. The goal was to solve problems related to navigation, perception, and manipulation. Various sensors are necessary to support these functionalities and several algorithms are necessary to process the sensor data.

3.5.1 Navigation

Human environments are inherently dynamic and may be cluttered, which may present several challenges for robot navigation. Autonomous navigation with minimum guidance may relieve burden for people that may otherwise be required to teleoperate and guide the robot. Safety is an important aspect, especially since the robot shares the environment with humans and could injure people. It must avoid collisions and move in a predictable manner. Simply limiting the velocity may not be a solution, because a close to human walking speed is essential for completing tasks in a timely fashion and assisting patients during walking exercises.

In order to move autonomously, the robot must sense its surroundings. A basic hardware requirement is a planar laser-scanner mounted at the robot's base for obstacle detection during 2D navigation. To this end, a laser scanning rangefinder was installed at the front side of the youBot base to detect obstacles. The sensor data can be used for autonomous navigation and assistive teleoperation. A Hokuyo URG-04LX-UG-01 model was used, which has a 240° field of view and a measurement distance of 4 m. A tilting laser-scanner or RGB-D camera mounted on the robot's body or head could be used for obstacle detection in 3D space. In addition, wheel encoders are necessary for computing odometry data, which would help estimating the robot's location more accurately.

The ROS navigation stack provides utilities for mapping, localization, path planning, and obstacle avoidance [90]. In order to use the navigation stack, the robotic platform must be able to publish and receive ROS messages as detailed in the configuration tutorial [91]. Additionally, there are several ROS packages for depth cameras which publish the sensor data and perform the transformations between the camera and the robot frames. The ROS simultaneous localization and mapping (SLAM) package is a wrapper for OpenSLAM's GMapping and builds a map while the robot moves around the world [92]. During that process, the robot can either be teleoperated or autonomously explore the environment with the help of an exploration stack [92]. The results are published to a map server node, which can be saved to file and re-used for autonomous navigation. If an unexpected obstacle is detected, the robot plans a local path that avoids the obstacle. Once the obstacle is cleared, the robot resumes its previously planned global path. Finally, ROS includes the visualization tool RViz which provides a graphical user interface (GUI) for specifying goal commands and viewing maps, sensor data, planned paths, and detected obstacles.

3.5.2 Manipulation

In order to physically interact with its environment, a co-robot must be able to identify objects and detect their poses. This could involve processing color images and depth data from RGB-D cameras, training classifiers, building 3D models, creating databases, etc. The ROS Object Recognition Kitchen (ORK) is a comprehensive tool for solving computer vision and perception tasks [77]. It takes care of all non-vision aspects such as robot integration and database management. There is a ROS wrapper which provides infrastructure for storing and retrieving model and training data via ROS messages.

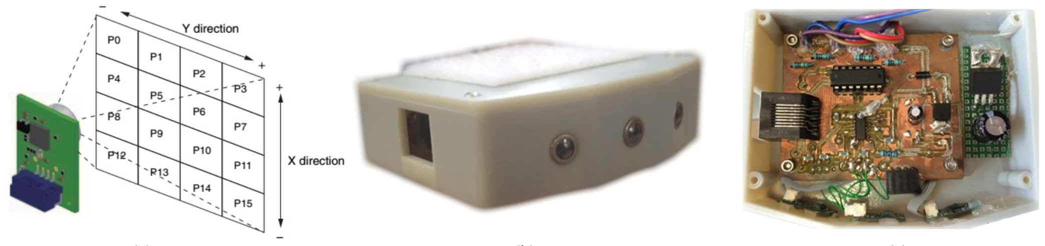
The detection itself is built on Ecto, a C++/Python based framework that organizes computations according to acyclic graphs with qualities such as synchronous execution and efficient scheduling routines [93]. This enables simultaneous detection of multiple objects at a high rate. In addition, several detection pipelines can run in parallel. This is useful since each has strengths and weaknesses. For example, the TableTop pipeline tries to match segmented object point clouds with object meshes stored in a database [33]. Detection requires no training but fails if the orientation of the mesh and actual item differ too much. The LineMod method is more robust since it does not require the object to be on a planar surface and generally works with any object orientation [76]. It utilizes template matching to identify an object pose with the help of a database containing trained 3D models. However, this requires pre-processing 3D models and is computationally intensive compared to the TableTop method.

ORK is a powerful idea but from our experience it is not ready for real-world, commercial applications. It can be difficult to setup and results can be mixed. Its underlying framework (Ecto) may be difficult to understand, and it is not straightforward to modify functionality and adding new features. If the environment and task are known, it might be reasonable to implement a simple, tailored object recognition framework using the Point Cloud Library (PCL) [94]. For example, the number of objects can be limited and placed in locations with reduced clutter and occlusions. The open source library has plenty of examples for processing 3D images, such as geometric segmentation, object recognition, scene registration, etc., and can be easily adapted into ROS.

The object detection can be simplified further by making the environment robot friendly. The ar_track_alvar package is a ROS wrapper for ALVAR, an open source library for marker based tracking [36]. It is a robust tool for accurately determining the position and orientation of QR codes. The QR codes could be placed on objects for fast and accurate detection, which would allow focusing on other aspects and challenges in the described scenarios. Once the object has been correctly identified and the robot knows its pose, the robot may try to grab the object. To do so, it needs to utilize 3D depth data to generate collision free arm motions. *MoveIt!* is a software package for motion planning and can be easily configured for different ROS platforms [72]. To detect object contact and applied forces, the gripper could be equipped with pressure sensitive finger tips.

3.5.3 Detection of Human Activity

To assist and interact with humans the co-robot must have some form of situational awareness, which requires the ability to detect humans as well as their current activity. This type of detection is often performed by a skeletal- and/or face-tracking algorithm utilizing a video feed from a regular or depth camera. However, visual observation might be hampered by different lighting conditions and occlusions. Future movements of the patient could be predicted based on current and past activity, with detailed information about pose and velocity of the face.



(a) (b) (c)

Figure 3.3. (a) The detection area of the Omron D6T-44L [95]. Three sensors were mounted in a 3D printed enclosure (b), including circuitry for robot skin and roboRIO interfacing shown from the top in (c).

Using a mobile robot and actuating the camera could make it easier to track a person's head or body. In addition, a mobile robot with an actuated "neck" can maintain "eye contact" during conversations. This may enhance the interaction between the robot and human and be used, for example, to get a human's attention during a greeting or introduction phase. Head nods and shakes could be detected after a robot asks a verbal question to quickly estimate intent and confirm recognized voice commands.

A simpler and more cost effective approach for detecting human presence involves thermal sensors, which detect radiated heat or far-infrared rays of an object. Hence, they are not affected by different lightning conditions like conventional cameras and simple thresholding can be used for detection. The Omron's MEMS thermal sensor (D6T-44L) consists of a MEMS thermopile sensor chip covered with a silicon lens [95]. The chip measures an electromotive force and an embedded circuit converts the analog signals to digital temperature values. In contrast to conventional pyroelectric sensors, Omron's sensor does not measure a change in signal and continually detects the far-infrared ray of an object. Hence, the sensor is able to catch a signal of a moving or stationary person. A custom box with three thermal sensors was mounted at the end-effector just below the two finger-gripper. The sensor itself outputs a 4 by 4 pixel array (Fig. 3.3a) and has a viewing angle of 45° [95]. Utilizing three sensors, 4 by 12 pixels cover approximately 120° including some overlap. The hardware with thermal sensors and wiring were encased in a 3D printed enclosure as shown in Fig. 3.3b and 3.3c. The values measured are transmitted through an I^2C bus to the roboRIO.

3.5.4 Human-Robot Communication

Commands and information can be exchanged verbally, via gestures, or through computer as well as tablet interfaces. Since each HRI has its advantages and disadvantages, it could be beneficial to allow users to choose from several input methods.

Verbal communication requires the robot to be equipped with a speaker and microphone. To talk to the user, the robot needs a text-to-speech program. There are several free programs available with natural sounding voices, for example the Festival Speech Synthesis System [96]. Speech recognition is more challenging and requires sophisticated language processing software. An alternative could be cloud services such as the Google voice API, however this would require uninterrupted internet access and might introduce small delays. Pocketsphinx from CMU is an offline, lightweight speech-to-text program and has been integrated with ROS [97]. The language processing is simplified by using a pre-determined list of words, which could allow the robot to understand straightforward commands. Ideally, a dialog system with advanced artificial intelligence would handle the interaction but this would require significant software development. Instead, it could be advantageous to keep the interaction simple and focus on reliability.

Correct speech recognition heavily depends on the sound quality and user pronunciation. A user could wear and talk into a wireless microphone, however this might not practical. Another possible mode of communication is gesture. To support gesture recognition, the ARNA robot would need to identify the user and capture different poses. ROS includes a skeleton tracker that determines the human pose from RGB-D camera images [98]. A classifier could then be used to detect various gestures and commands. The challenge would be training the classifier such that it is reliable with several different users.

A computer or tablet may perhaps be the most reliable interface. Tablets have the advantage of being portable and can directly communicate with the robot using Bluetooth or WiFi. Simple layouts can be tailored for a multitude of tasks such as video monitoring, joystick teleoperation, or object fetching.

3.5.5 Robot Skin

To physically collaborate with humans and be safe to operate, co-robots need to detect human contact and motion intent. Multiple human detection modalities can be employed using sensors that can detect pressure, temperature, and proximity to humans, thus leading to enhancements in safety and the ability to adapt to preferences of users and improve performance. Robot skin is a key type of heteroceptive sensor inspired by nature that could eventually enable co-robots to share their workspace with humans.

The Electro-Hydro-Dynamic (EHD) printing process can be used to create multi-modal pressure and temperature sensors on flexible substrates [84]. The key advantages of EHD include its ability to accommodate a wide-range of materials and substrates with different sizes, shapes, and topographies. EHD can be exploited for sensor integration as it might allow both sensors and interconnects to be directly printed into the silicone rubber skin and attached to robot surfaces or to prostheses (Figure 3.4). Currently, 1 mm spatial resolutions, 50 N force ranges, and measurement sensitivities in the order of 10 mN can be obtained using EHD printing methods.

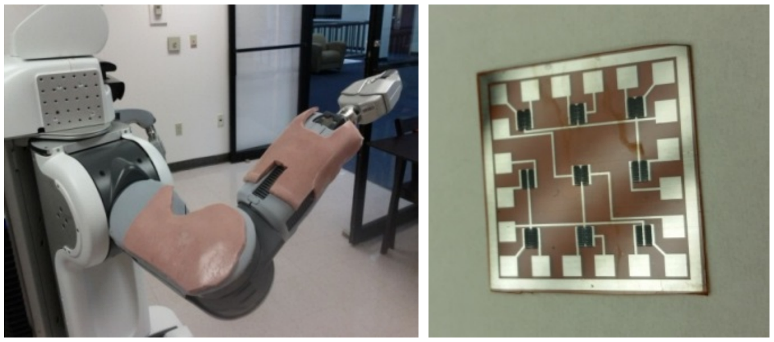


Figure 3.4. Robot Skin on PR2 (left), realized using EHD printing technology of pressure sensors on flexible substrates encapsulated in silicone skins (right).

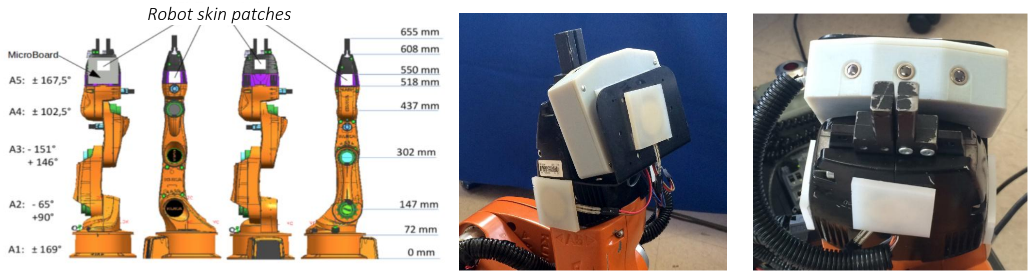


Figure 3.5. Robot skin patch placement on the KUKA youBot manipulator (left image modified from [88]).

For physical interaction, the youBot end-effector link was outfitted with four skin patches consisting of pressure sensors embedded in P10 RTV silicone rubber as shown in Fig. 3.5. The Tekscan Flexiforce thin-film sensors are ideal for measuring the force between two surfaces. They are cost effective and durable with good sensor characteristics: linearity error within 3%, hysteresis less than 4.5%, drift less than 5%, and low temperature sensitivity (0.36% per °C). The maximum response time is 5 µs and they handle up to 445 N (100 lbs). The active sensing area is 9.53 mm in diameter. The particular sensors used are piezoresistive in nature: As the force increases, the resistance decreases from infinity to approximately 300 kΩ. A voltage divider circuit with an emitter buffer was designed to measure human applied forces up to 222 N (50 lbs). The circuit was implemented on a custom data acquisition board (or MicroBoard), which conditions the pressure data from several sensors (or taxels). The results are read by the roboRIO using its analog input ports.

3.6 Software Architecture

A modern co-robot has to solve problems related to navigation, perception, and manipulation. Various sensors are necessary to support these functionalities and several algorithms are necessary to process the sensor data. These capabilities could run in parallel while a high level planner controls the task flow and handles error detection and correction. Several robotics software solutions exist such as Microsoft Robotics Developer Studio (MRDS) [99], Mobile Robot Programming Toolkit [100], and Rock [101]. Perhaps a more popular and comprehensive ecosystem is the open-source Robot Operating System (ROS) originally developed by Stanford Artificial Intelligence Laboratory in 2007 [102]. ROS is a framework for robot software development and includes a large collection of libraries (stacks) that facilitate implementation of common robotic functionalities (packages). ROS provides an infrastructure where these functionalities may run in separate nodes that communicate with each other via messages. It is language independent (C++ and Python are most commonly used) and runs on Unix systems such as Ubuntu. Recently, there has been increased support for Microsoft Windows and Mac OS X. Most importantly, ROS is hardware agnostic and focuses on code reusability. The same ROS packages could be used on different robotic platforms with only minor modifications Hence, ROS appears well suited for operating robots in complex environments such as hospitals. The following sections will discuss sensor requirements and state-of-the-art ROS stacks and packages that could provide some of the basic capabilities for the ARNA robot.

The previous sections describe the youBot hardware upgrades. This section describes the software architecture that was developed to process the sensor data and

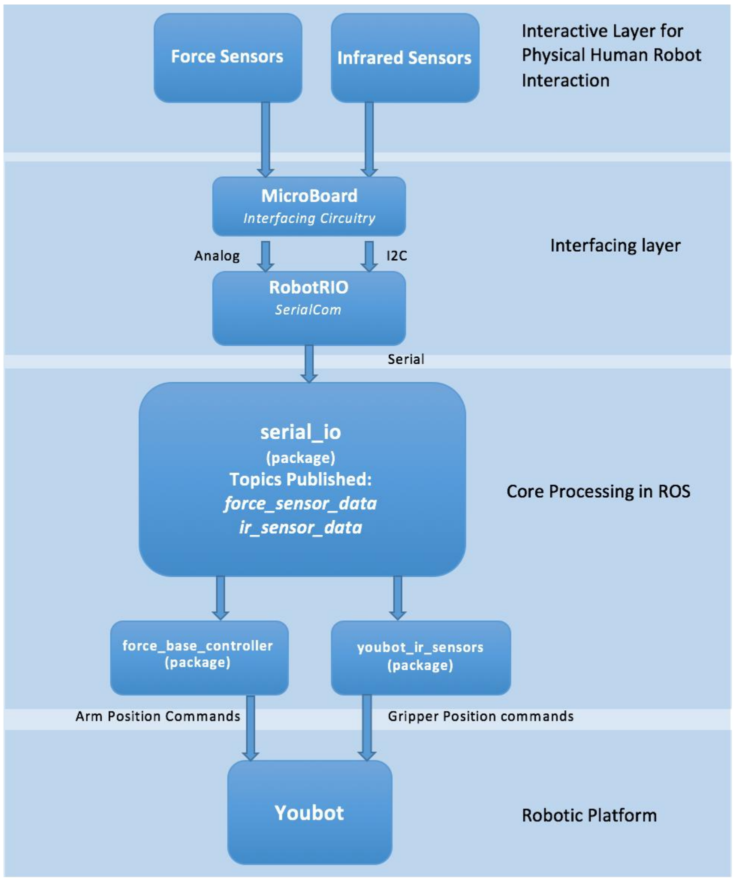


Figure 3.6. Software architecture showing the flow of data. The sensors are being read by the roboRIO, which then sends the data to ROS. The arm and gripper commands are then executed by the low-level hardware controllers in the youBot.

enable robot control with the various interfaces consisting of ROS, an Android tablet application, and LabVIEW.

3.6.1 Robot Operation System

The KUKA youBot runs the linux-based operating system Ubuntu 12.04. The platform is controlled by the open-source Robot Operating System (ROS), a software framework originally developed by Stanford Artificial Intelligence Laboratory in 2007 [102]. ROS has become a de-facto standard in robotics and there is a large collection of software packages developed by the community. Since ROS is both programming

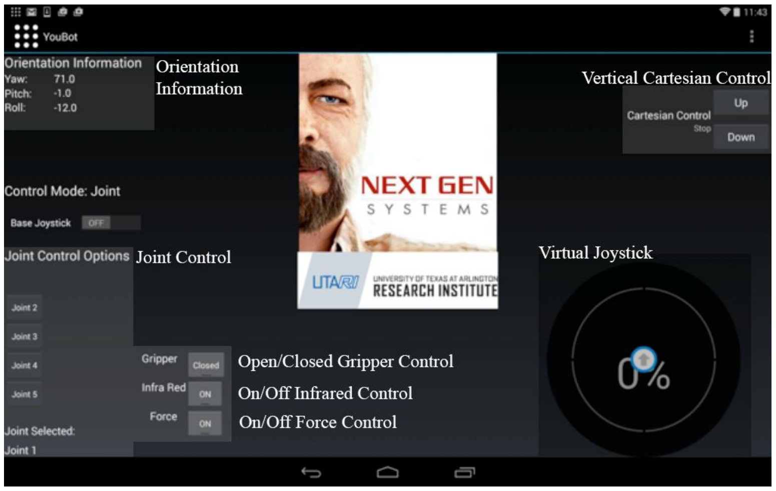


Figure 3.7. Tablet interface for controlling the youBot.

language and hardware agnostic, these packages can easily be re-used on different platforms. Programs are executed as independent ROS nodes and data is transmitted via ROS messages. This distributed architecture allows several components to run simultaneously, for example sensor data processing, navigation, and perception.

In this spirit, we developed a multi-layered, ROS-based architecture to allow robot autonomy as well as user intervention via several HMIs such as tablet apps or pressure sensors (Fig. 3.6). Several stand-alone ROS packages were developed, with different functionalities for navigation and manipulation as described in Table 3.1. The program flow is determined by a cortex node which acts as layer between the interfaces and control modules. It contains a state machine which utilize ROS topics or services to call other ROS nodes that then execute the necessary code. If anything fails, the cortex node runs contingency plans and is able to restart faulty ROS nodes. This also simplifies data flow making it easier to debug and capture data.

Table 3.1. Overview of the developed ROS packages.

ROS package	Description
youbot_cortex	Contains a state machine node which acts as a layer between the data received from the tablet app and the robot controllers. It sends service requests to enable/disable different control modes, joints, and sensors.
cortex_msgs	Defines custom ROS messages and services.
serial_io	Reads data from roboRIO via a serial communication channel and publishes it into ROS topics.
youbot_force_sensors	Subscribes to force data and computes the command velocities for the base movement. KDL (Kinematics and Dynamics library) is used for 3D frame and vector transformations.
youbot_ir_sensors	Processes thermal sensor data (filtering and thresholding) to control the gripper.
youbot_description	Contains the youBot robot model.
youbot_navigation	Utilities for mapping, localization, and path planning.
youbot_arm_controllers	Sends position commands to the youBot motors using different control algorithms (individual and combined joint control).
youbot_cartesian	Cartesian arm controller using the motion planning software MoveIt! [72].

3.6.2 Tablet Interface

A youBot tablet application was developed using ROSJava [103] to allow users to control the platform with an easy-to-use graphical user interface (GUI). The app is split up into 3 different classes: ViewController, NodePublisher and the VirtualDivet. The ViewController connects the GUI interface (buttons, toggle buttons, switches, text) to backend code. Using ROS messages, the NodePublisher communicates user intent to the cortex program which then initiates the robot movement. The VirtualDivet class implements the touch-screen joystick in the lower right hand corner. It allows the user to move around a divet inside a circular area, where the off-center distance produces velocity commands between 0 and 100%.

Figure 3.7 shows the layout on a Nexus 10 Android tablet. The top left corner displays the tablet orientation, i.e. the pitch and roll from the gyroscope and the yaw relative to earth from the magnetic compass. The active control mode can either be individual or combined joint control. Joint control mode allows the selection of individual joints which then can be moved by tilting the tablet. The joint velocity is proportional to the pitch of the tablet. In combined joint control, several joints are moved together to move the end-effector in an arc. Joint limits prevent the user from hitting the floor or robot. The Cartesian position of the end-effector can be controlled with buttons in the upper right corner. Vertical Cartesian control moves the end-effector along the y-axis of the base frame. Finally, there are switches for activating different interfaces such as the thermal and pressure sensors. The gripper switch opens or closes the two-finger pincher. The base joystick toggle enables the virtual joystick, which moves omnidirectional base of the robot platform.

3.6.3 RoboRIO Software

The roboRIO was programmed using the National Instruments' LabVIEW. The code is executed in a real-time loop, where pressure sensors are read every 1 ms using the analog input ports. The infrared sensors generate data every 30 ms and use the I^{2}C bus interface. After being captured, the data is relayed via USB to the youBot using the RS232 communication protocol. Collecting sensor information and communicating with ROS are parallel tasks, made possible by the multicore processors on the roboRIO. This way the ROS nodes can access the sensor data at approximately the same rate: 1000 Hz for pressure sensor and 33 Hz for thermal data.

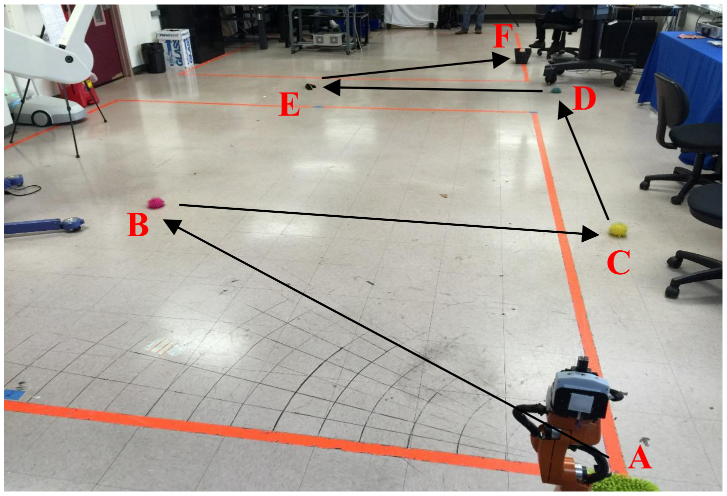


Figure 3.8. Pick and place experiment on the laboratory floor showing waypoints A through F.

3.7 Performance Evaluation of HMIs in Household Environments

Experiments were conducted to test the functionality of the implemented hardware and performance of the proposed HMI schemes. The tasks involved object pick-and-place using the tablet interface and pressure sensors, as well as autonomous navigation. The primary objective was to measure the difference in completion time and accuracy of path trajectory for expert and novice users. Novice or non-expert users were defined as those who had no prior experience with the tablet application, the robot, and its functionalities. Both type of users were given a brief overview of the system and the tasks to be completed. Measurements included completion time and trajectory error, which was determined by localizing the youBot with the help of odometry and laser scanner data.

3.7.1 Pick and Place

In the pick and place task, a user was asked to move several objects between waypoints with the youBot. To create a rigorous method of testing, the waypoints were marked on the floor as shown in Fig. 3.8. The path involved straight and diagonal movements of various lengths: 3.05 m from point A to point B, 2.76 m from B to C, 3.68 m from C to D, 2.80 m from D to E, and 3.07 m from point E to point F. Different interfaces were tested to compare the accuracy and completion time of physical guidance and tablet teleoperation. Both expert and novice users were asked to pick up 5 objects in 6 trials. This was repeated for two types of interfaces: teleoperation via tablet (tablet mode) and direct physical interaction (mannequin mode) via the skin patches. In tablet mode, the user was required to accomplish the following tasks:

1. Pick up object at current location
2. Transport object to next waypoint using the virtual joystick
3. Place object 2-3 inches within waypoint marker
4. Repeat until final waypoint has been reached (point F)
5. Place final item into a bin

The base was operated using the virtual joystick and the robot arm was controlled in joint or Cartesian mode depending on user preference (see [7] for further details). The gripper was opened and closed by pressing the corresponding button inside the tablet interface. In mannequin mode the tasks were as follows:

1. Pick up object at current location
2. Guide robot to next waypoint using the robot skin
3. Place object 2-3 inches within waypoint marker
4. Repeat until final waypoint has been reached (point F)
5. Place final item into the bin

The user guided the youBot along the outlined path by pushing on the robot skin mounted on the manipulator. Once a waypoint was reached, the user moved the arm into position and operated the gripper with a simple hand-wave in front of the thermal sensors.

The completion time for the pick and place experiments are listed in Table 3.2. Six trials were conducted with two expert and two novice users. The mean completion time for an experienced user was approximately 2 minutes and 45 seconds. As expected, the non-experts performed slower and needed more than double the time to complete an identical task. However, the time difference between the two interfaces for each user was minimal as shown in Fig. 3.9. This seems to indicate that both the tablet and mannequin mode provided the same level of control. However, this was not the case as shown further down by the position data. It should also be noted that the physical interface required almost no training, while it took a few minutes for the novice users to get used to the tablet interface. In addition, when the youBot was facing the operator, there was some confusion since the left and right tablet commands would be flipped from the operator's point of view. The disparity in time between the users may be attributed to different individual skill levels.

In addition to completion times, the youBot's location in the map frame was recorded during the experiments. The position was estimated from internal odometry and laser scanner data. The two expert and two novice users were asked to follow the path marked on the floor as closely as possible. Fig. 3.10 and 3.11 depict the trajectories obtained via tablet and mannequin control, respectively. Due to slippage of the Mecanum wheels on the hard floor, the odometry position estimate drifted over time. Therefore, the position measurements were not accurate enough to quantitatively compare the trajectory errors. In future work, the youBot position estimate could be recalibrated between trials or a motion capture system could localize the robot more accurately. However, by general inspection of the graphs in Fig. 3.10 and 3.11, one can infer that the mannequin mode produced smoother and more ac-

Table 3.2. Task completion time (in seconds) for expert and novice users for the tablet and mannequin mode. The mean and standard deviation (STD) is computed for 6 trials.

Mode	Expert user 1	Expert user 2	Novice user 1	Novice user 2
Tablet	175	188	395	368
	158	174	388	343
	162	159	382	340
	153	141	370	348
	148	139	372	330
	178	181	409	398
Mean	162	164	386	355
STD	12	20.7	14.7	24.7
Mannequin	169	176	401	382
	171	159	402	389
	173	170	393	381
	166	162	399	395
	159	160	383	346
	171	169	401	389
Mean	168	166	397	380
STD	5.1	6.7	7.4	17.6

curate trajectories compared to the tablet mode. The mannequin trajectories in Fig. 3.11 show less spread and there are fewer path deviations. Hence, similar completion times does not equate to identical performance. For both interfaces there was a clear difference between expert and non-expert users. From a qualitative assessment, the expert users produced smoother paths and performed fewer corrections.

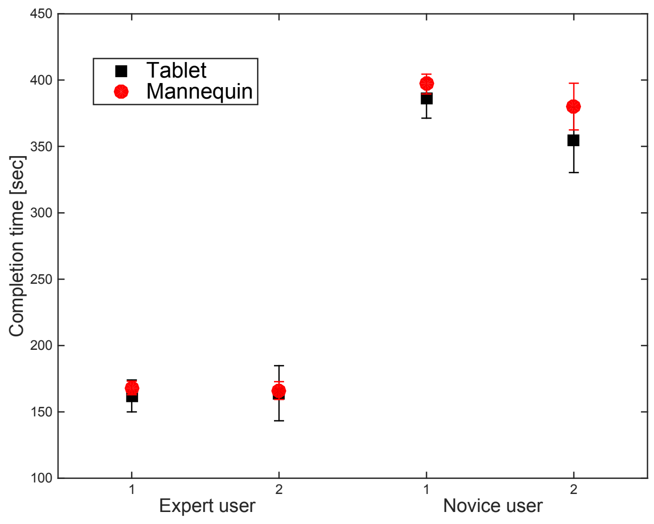


Figure 3.9. Error plot showing mean completion times (in seconds) for two expert and two novice users.

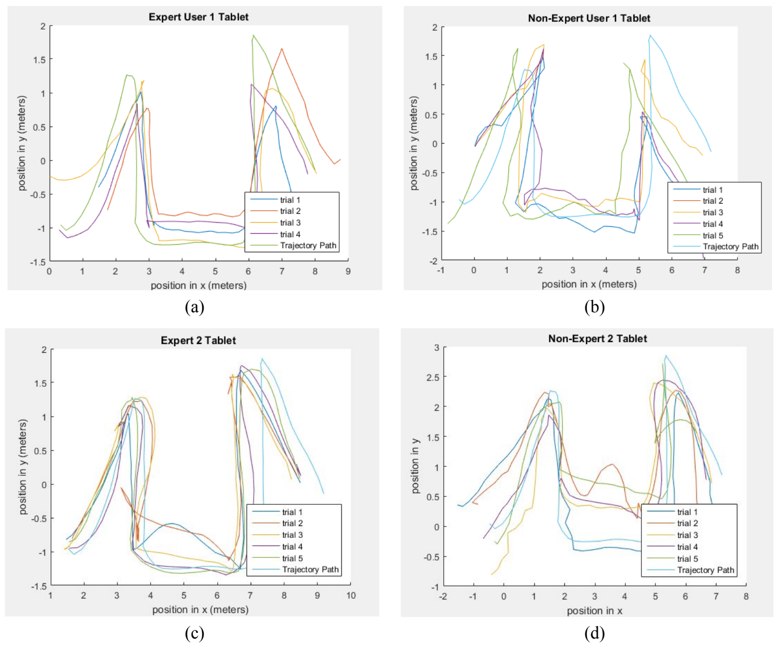


Figure 3.10. The youBot map coordinates in meters during tablet mode operation for 6 trials with (a), (c) expert and (b), (d) non-expert users.

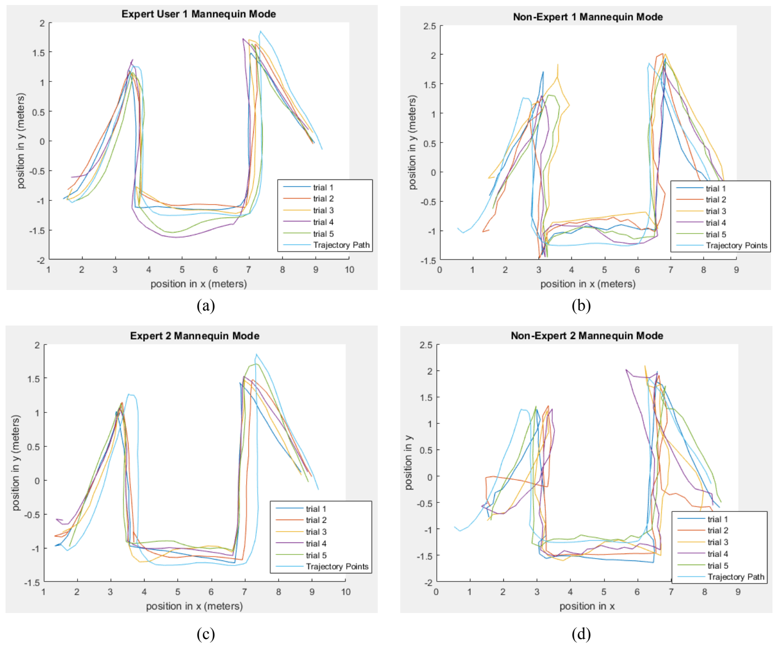
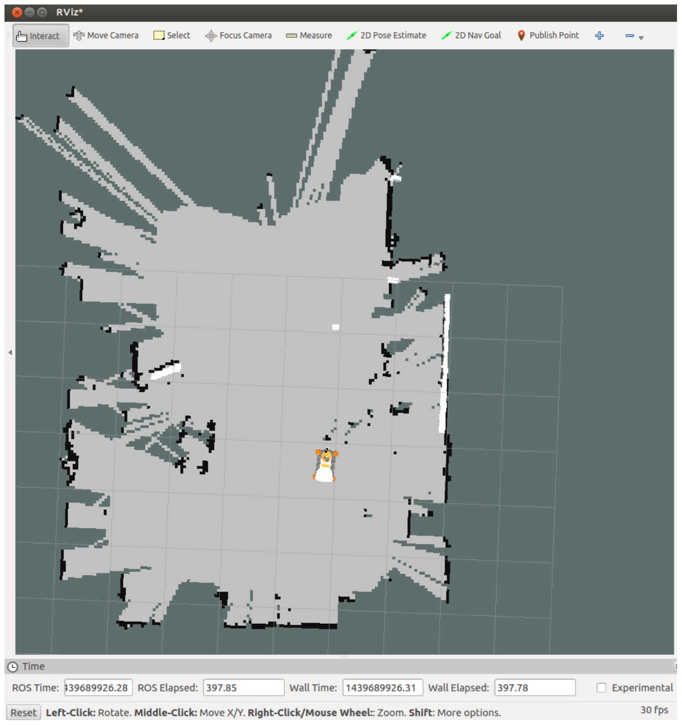


Figure 3.11. The youBot map coordinates in meters during mannequin mode operation for 6 trials with (a), (c) expert and (b), (d) non-expert users.

3.7.2 Autonomous Navigation

In addition to pick and place experiments, in which the user actively guided the robot, experiments were conducted to evaluate the youBot's autonomous navigation and obstacle avoidance capabilities. To navigate, the youBot received goal positions via point-and-click mouse commands in the ROS Visualization (RViz) interface shown in Fig. 3.12. The software provides a graphical user interface (GUI) showing a 3D model of the robot located in a 2D map outline in light grey color. Potential obstacles such as walls and furniture are marked in black and the real-time laser scan data is represented by white pixels. The youBot navigation program utilizes robot odometry and laser scan data to localize the robot. After a user provides a goal location in RViz, the global planner generates a collision free path using algorithms such as A* search. If an unexpected obstacle appears during motion, for example a human stepping in



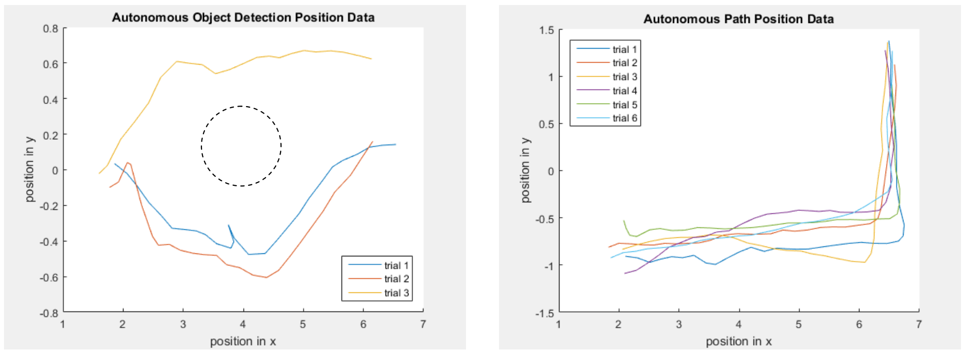
ROS Time: 439689926.28 ROS Elapsed: 397.85 Wall Time: 1439689926.31 Wall Elapsed: 397.78

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

Figure 3.12. RViz interface showing a top-view of the youBot inside a mapped area (light grey region). The black pixels represent obstacles and white pixels the real-time laser scanner data.

front of the robot, the local planner takes over and creates a modified path around the obstacle.

Two different setups were utilized to evaluate autonomous navigation using the RViz interface. First, the youBot had to avoid a circular obstacle to reach the same waypoint from three different initial positions. Moving right to left in Fig. 3.13(a), the youBot successfully avoided any collisions and chose the shortest trajectory around the obstacle. In the second setup, the robot moved along two successive waypoints requiring a 90 degree turn. The results depicted in Fig. 3.13(b) show that the position accuracy decreased over time, which again was caused by slippage of the wheels. Qualitatively, the accuracy is similar to mannequin mode operation and better than tablet control.



(a) (b)

Figure 3.13. The youBot map coordinates in meters during autonomous navigation from right to left. The setup in (a) required avoiding a circular obstacle and (b) involved two waypoints resulting in a 90 degree turn.

3.8 Conclusion

In this paper the KUKA youBot mobile manipulator was sensorized for user operation via a tablet, physical interaction, and simple hand gestures. Sensor upgrades included a laser scanner, pressure sensitive robot skin, and thermal sensors. System integration was accomplished through the addition of a roboRIO controller and the software architecture was based on the Robot Operating System (ROS). After the HMI and sensor upgrades, the robot can be guided by users via teleoperation with a tablet, direct physical guidance, as well as through a point-and-click GUI. We demonstrated basic capabilities and acquired data from a few users, which will pave the way for experimental studies including a larger number of subjects in near future. Several pick and place tasks were completed by two users familiar with the system (experts), and two users not familiar with the system (non-experts). The results indicate that: (*i*) Physical interaction (or mannequin) guidance results in the most accurate robot trajectories for both expert and non-expert users. The interface is intuitive and might require less training time. (*ii*) Tablet and mannequin control result in similar task completion times. Expert users can complete pick and place manipulation and mobility tasks significantly faster than non-expert users. (*iii*) The autonomous point-and-click interface resulted in shortest path for traversing free and obstacle-filled environments, and had similar accuracy to mannequin guidance. There is minimal effort for the user, however the robot requires a map of the environment.

ACKNOWLEDGEMENTS

This research was supported by National Science Foundation Grants NRI-1208623 and BIC-1261005720, and Qinetiq-North America Corporation.

CHAPTER 4

A Robotic Waiter with Physical Co-manipulation Capabilities

ROBOTIC WAITER WITH PHYSICAL CO-MANIPULATION CAPABILITIES

S. Cremer, I. Ranatunga, and D. O. Popa, “Robotic waiter with physical co-manipulation capabilities,” in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 1153–1158.

4.1 Abstract

In this paper, we compare the performance of physical and non-physical interfaces for the behavior of a personal robot. The PR2 robot was programmed as a waiter with co-manipulation capabilities and experimentally tested with respect to a trajectory following task. For physical interaction (pushing/pulling), we implemented a compliance controller for compliant, stable arm positioning and a velocity based position controller for moving the robot base. Experiments were conducted to assess the effectiveness and accuracy of single and dual arm control compared to joystick teleoperation. Results indicate that the PR2 in physical collaboration with a human performs better than in teleoperation mode, as measured by task completion time while maintaining a comparable task accuracy. A result of our work is the open source, robot operating system (ROS) package *pr2_cartPull*, which will be shared with the robotics community.

4.2 Introduction

The introduction of robots into human environments has motivated a need for safe and reliable physical interaction [104]. There are many challenges to moving a robot and accomplishing tasks in environments that are inherently cluttered and dynamic. Static obstacles such as furniture and dynamic obstacles such as humans have to be detected and avoided. Dynamic situations require fast and real-time reaction to avoid damage, variations in pose requires good situational awareness and maneuvering skills [105].

Current research in robotics has focused on the extensive use of vision, planning, and optimization to solve these problems [106]. Tasks are performed by systems that are increasingly complex and depend on vision including 3D sensors [107]. Although these techniques are powerful, they are not mature enough to deploy in a home environment where reliability and repeatability are important [108]. The challenge with using vision is the time required to process and plan, occlusions, and unsuitable lighting conditions. In this paper, we study the interaction between a human and a robotic waiter operating in a dynamic environment with low, variable lighting, in which reliability, repeatability, and speed are crucial. Therefore, we rely on physical co-manipulation to allow a human to guide the robot through the environment. In general, physical Human-Robot Interaction (pHRI) requires coordination and control of the forces being applied by the human onto the robot. The problem consists of two parts: 1) safe, stable contact with the human and 2) identifying the human intent [105, 109]. During the physical interaction, the robot should not exert any large forces on the human and have relative low joint velocities. The robot's limbs should move in a smooth, predictable manner and no oscillations should occur. In this paper, the problem of safe and stable contact is addressed through compliance control [110, 111].

In a task the robot was required to follow along with a cart. The task definition allowed us to restrict the problem space such that the human pulling force applied through the cart was the input and the output was position regulation of the robot base with respect to the cart. Such a configuration enabled a natural cart-following behavior by the robot, and allowed the human operator to lead the robot to desired locations. The research contribution of our work is in evaluating the performance of the controller during a cart-following task with respect to its parameters. In addition, we report on the performance of the co-manipulation interface with two arms of the PR2 robot, and compare it with a one-hand interface, and a traditional joystick. Not surprisingly, results clearly indicate that a two-handed natural interface of the robot to the cart yields the best performance, as defined by the task completion time.

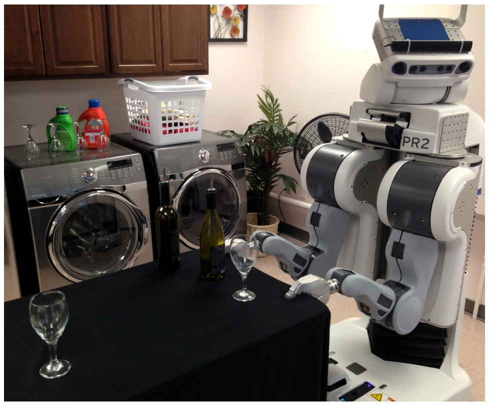


Figure 4.1. Personal Robot 2 (PR2) manipulating a cart at the UTA Research Institute (UTARI) Living Laboratory.

The paper is organized as follows: we first describe the system used to follow human commands using a cart and a PR2 robot shown in Fig. 4.1. In particular, the compliance controller is described in Sec. 4.3.1 and the velocity based position controller in Sec. 4.3.2. We discuss the experiments conducted to assess the effectiveness of the proposed method in Sec. 4.4 and the results in Sec. 4.5. Finally, in Sec. 4.6 the conclusion and future work are presented.

4.3 Co-manipulation System

In this section, we describe the system used to follow human commands using a cart and the PR2 robot, consisting of a mobile base, two robotic arms, and a pan

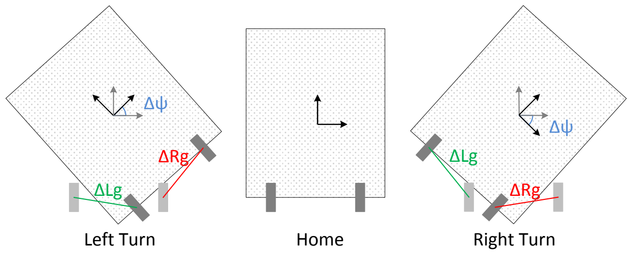


Figure 4.2. PR2 grippers location, and frames of the cart in top view.

and tilt head unit (Fig. 4.1). Each arm is gravity compensated and offers 7 degrees of freedom (DOF). The arms extend up to 81 cm (32 inches) and the omnidirectional base can reach velocities up to 1.0 m/s (3.3 ft/s).

We first describe the controller used for compliant, stable interaction with the cart. Then, we describe the velocity based position controller that moves the robot to the home condition seen in Fig. 4.2, which leads to the cart-following behavior. A human will push/pull the cart or the robot, and the robot will initiate base and arm movements to follow guidance from the human.

4.3.1 Compliance Controller for the PR2 Arms

In classical explicit force control the objective is to maintain a desired interaction force f_d, using schemes such as proportional-integral-derivative controllers for regulation. This does not work in all interaction scenarios, especially not in cases where chattering occurs. It is well known that PID explicit force controllers can become unstable due to changes in contact environment [38, 39, 110, 112].

Impedance control was developed by Hogan to achieve stable environmental contact [113]. It controls the dynamics of the physical interaction with the environment instead of achieving an explicit force objective. We use compliance control as a simplified version of this scheme [110, 111].

The general robot dynamic equation with actuator dynamics and external interaction with the environment is

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau + \tau_e \tag{4.1}$$

where $q \in \mathbb{R}^n$ are the joint positions (n is the DOF), $M(q)$ is the inertia matrix, $V(q,\dot{q})$ is the Coriolis/centripetal vector, $G(q)$ is the gravity vector, and $F(\dot{q})$ is the friction term. We also add the disturbance torque $\tau_d \in \mathbb{R}^n$ and the environmental torque $\tau_e = J^T(q)f_e$, where f_e are interaction forces exerted on the robot in task space, $J^T(q)$ is the Jacobian and $\tau \in \mathbb{R}^n$ the control torque. Neglecting friction and disturbance torques, the task space pose $x = [p_e, \phi_e]^\intercal \in \mathbb{R}^6$, where $p_e \in \mathbb{R}^3$ is the task space position and $\phi_e \in \mathbb{R}^3$ is the orientation. The dynamics in task space is

$$M_x(q)\ddot{x} + V_x(q, \dot{q})\dot{x} + G_x(q) = J^{-T}(q)\tau + f_e \tag{4.2}$$

where,

$$M_x(q) = J^{-T}(q)M(q)J^{-1}(q),$$
$$V_x(q, \dot{q}) = J^{-T}(q)V(q, \dot{q})J^{-1}(q) - M_x(q)\dot{J}(q)J^{-1}(q),$$
$$G_x(q) = J^{-T}(q)G(q).$$

The desired impedance model to balance the interaction force f_e is chosen as

$$M_m(\ddot{x} - \ddot{x}_d) + D_m(\dot{x} - \dot{x}_d) + K_m(x - x_d) = f_e \tag{4.3}$$

where x_d is the desired trajectory, M_m is the inertia, D_m is the damping, and K_m is the stiffness of the impedance model. For the PR2 robot, the mass matrix and the Coriolis term are not readily available, while the gravity term can be neglected because of the robot's gravity compensated design. Therefore, replacing $M_m = M_x(q)$ in (4.3) and substituting into (4.2) the control torque leads to

$$\begin{aligned} \tau =& J^T(q)(M_x(q)\ddot{x}_d + V_x(q, \dot{q})\dot{x} + G_x(q) \\ &+ D_m(\dot{x}_d - \dot{x}) + K_m(x_d - x)). \end{aligned} \tag{4.4}$$

Assuming $\dot{x}_d = 0$ and for low joint velocities $\dot{q} \approx 0$ we obtain the compliance controller

$$\tau = J^T(q)(K_m(x_d - x) - D_m\dot{x} + G_x(q)). \tag{4.5}$$

Since the PR2 arms are gravity compensated, the gravity compensation term $G_x(q)$ can be removed. Replacing $\dot{x}$ with $J(q)\dot{q}$, the final desired torque is:

$$\tau = J^T(q)(K_m(x_d - x) - D_m J(q)\dot{q}). \tag{4.6}$$

This does not require interaction force measurements and avoids the need for an estimator to isolate the interaction forces from the PR2 gripper dynamics.

Because the controller in (4.6) only sets a desired gripper pose, a resorting torque

$$\tau_r = K_{P,r}\epsilon + K_{D,r}\dot{\epsilon} \tag{4.7}$$

was used to keep the arm joints close to their original home position q_d. The joint error ϵ is defined as the difference between the desired and current joint angles

$$\epsilon = q_d - q. \tag{4.8}$$

We consider the following reference frames as seen in Fig. 4.3: $\{w\}$ attached to the world, $\{b\}$ attached to the robot base, $\{t\}$ attached to the robot torso, $\{g_r\}$ attached to the right gripper, $\{g_l\}$ attached to the left gripper, and $\{m\}$ attached to the cart being used. The right and left gripper pose in task space is $x_r \in \mathbb{R}^6$ and

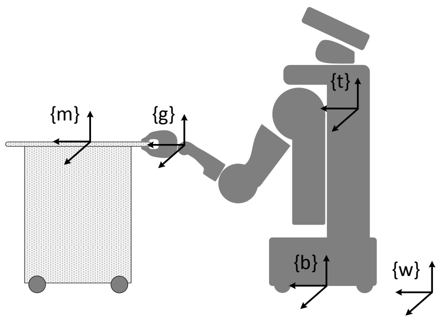


Figure 4.3. PR2 robot and cart transfer frames.

$x_l \in \mathbb{R}^6$ in frame $\{t\}$. The pose of the robot base in frame $\{w\}$ is $x_b \in \mathbb{R}^6$. The generalized forces at the grippers are $f_r \in \mathbb{R}^6$ and $f_l \in \mathbb{R}^6$ in frames $\{g_r\}$ and $\{g_l\}$.

The cart, pulled by a human, will apply forces f_r and f_l on the grippers of the PR2. These interaction forces will cause changes in the task space states of the grippers $x_r = [r_x, r_y, r_z, r_\phi, r_\theta, r_\psi]$, $\dot{x}_r$, $\ddot{x}_r$, $x_l = [l_x, l_y, l_z, l_\phi, l_\theta, l_\psi]$, $\dot{x}_l$ and $\ddot{x}_l$ according to the dynamics of the defined impedance model (4.3) . The desired task space position of the compliance controller is set to some point in front of the PR2, wheras the reference velocity and acceleration are set to zero. When the cart is grasped, the grippers lose 3 DOF and the motion of the grippers will lie in the plane defined by the constraints due to the cart. The remaining DOF are translation in x, y and rotation about the z axis as seen in Fig. 4.2. The gains are set low in these directions for smooth interaction.

4.3.2 Velocity Based Controller for the PR2 base

In this section, we describe a velocity based position controller to move the PR2 base. When the force applied by the cart on the robot exceeds the stiffness force of the compliance controller, task space pose errors are observed in the grippers. These errors are used by a velocity controller to generate a twist for the PR2 base when they exceed a certain threshold. This will reduce the errors in the gripper pose and drive the grippers and base to their home pose with respect to the robot base as seen in Fig. 4.2. Let the pose error be $e_p \in \mathbb{R}^6$. The translation errors in the unconstrained directions are given as ΔRg and ΔLg, and the orientation error about the z axis is given as $\Delta\psi$. We define the Cartesian space position error as

$$e_p = x_d - x \tag{4.9}$$

where e_p is $[\Delta x, \Delta y, 0, 0, 0, \Delta\psi]$. Now we can define the base velocity to be generated as

$$\dot{x}_b = K_P e_p + K_D \dot{e}_p. \tag{4.10}$$

If the pose of the right gripper is used to drive the base, $\Delta x = \Delta Rg_x$, $\Delta y = \Delta Rg_y$, and $\Delta\psi$ will be the angular pose error of the right gripper. If two grippers are used

$$\begin{aligned} \Delta x &= \tfrac{1}{2}\left(\Delta Rg_x + \Delta Lg_x\right) \\ \Delta y &= \tfrac{1}{2}\left(\Delta Rg_y + \Delta Lg_y\right) \\ \Delta\psi &= -\tan^{-1}\left(\frac{r_x - l_x}{r_y - l_y}\right). \end{aligned} \tag{4.11}$$

The combined twist error $\dot{e}_p$ is found from

$$\begin{aligned} \Delta\dot{x} &= \tfrac{1}{2}\left(\Delta\dot{R}g_x + \Delta\dot{L}g_x\right) \\ \Delta\dot{y} &= \tfrac{1}{2}\left(\Delta\dot{R}g_y + \Delta\dot{L}g_y\right) \\ \Delta\dot{\psi} &= \frac{\dot{u}}{1+u^2}, \end{aligned} \tag{4.12}$$

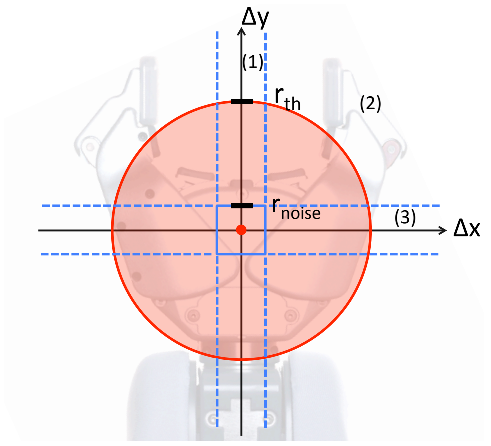


Figure 4.4. Threshold regions for the velocity controller. The velocity command $\dot{x}_b$ has components in y-direction (1), in x- and y-direction (2), or in x-direction (3).

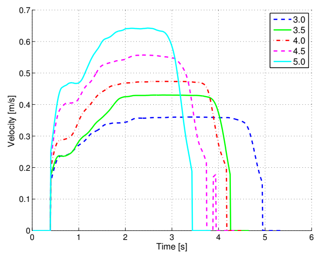
where

$$\begin{aligned} u &= \frac{r_x - l_x}{r_y - l_y} \\ \dot{u} &= \frac{(\dot{r}_x - \dot{l}_x)(r_y - l_y) - (r_x - l_x)(\dot{r}_y - \dot{l}_y)}{(r_y - l_y)^2}. \end{aligned} \tag{4.13}$$

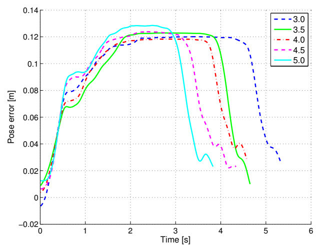
The computed velocity $\dot{x}_b$ will move the PR2 base to be aligned with the home position as depicted in Fig. 4.2. A switching logic is used to activate the velocity controller when the pose error is larger than a circular threshold radius in the xy-plane:

$$\sqrt{\Delta x^2 + \Delta y^2} > r_{\text{th}} \tag{4.14}$$

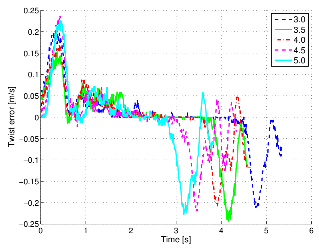
Due to noise, Δx or Δy are non-zero even when moving purely in the y- or x-direction. Therefore, a smaller, square threshold region was used to achieve stable lateral and longitudinal movements. When (4.14) is valid but $\Delta x < r_{\text{noise}}$ the base is only moved



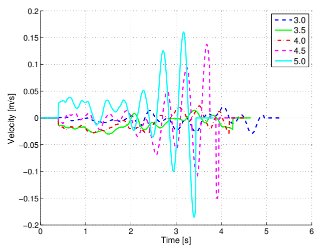
(a) x base command velocity from $\dot{x}_b$



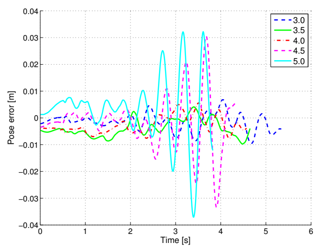
(b) Δx pose error from e_p



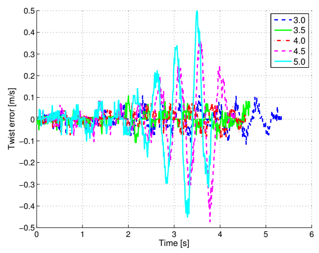
(c) $\Delta\dot{x}$ twist error from $\dot{e}_p$



(d) y base command velocity from $\dot{x}_b$



(e) Δy pose error from e_p



(f) $\Delta\dot{y}$ twist error from $\dot{e}_p$

Figure 4.5. Recorded base command velocities and errors during gain tuning of the dual gripper velocity controller. The PR2 was pulled forward 1.5 m (4.9ft) in a straight line for several K_p values.

in the y-direction and vice versa (Fig. 4.4). The base is rotated when the angular pose error exceeds the threshold angle, i.e.

$$\Delta\psi > \psi_{\text{th}}. \tag{4.15}$$

4.4 Experiments

In this section, we describe the experiments conducted to assess the effectiveness of the proposed method, and compare results of interaction between robot (with and without cart) and humans. The 7 DOF arms of the PR2 run under compliance control, whereas the base runs under the velocity controller described in the previous section. Both controllers were implemented using the real-time control manager framework of the PR2. The real-time loop on the PR2 runs at 1000 Hz and communicates with the sensors and actuators on an EtherCAT network. ROS Groovy [114] was used for all the experiments.

4.4.1 Parameter Tuning

The initial compliance and velocity gains for the single and dual controller were set using the Ziegler-Nichols tuning method [115]. Additional tuning was conducted as described below:

(i) The compliance controller gains were chosen such that the robot arms had low stiffness and could follow the cart movements before reaching the pose error thresholds. This lagging behavior resulted in smooth transitions between changes in velocity and direction. In contrast, high stiffness resulted in abrupt and jerky base movements. The linear gain in the z direction was set low since the arms were already restricted to the xy-plane by holding onto the cart.

(ii) The velocity controller was tuned by moving the PR2 in longitudinal and lateral directions while increasing K_P with $K_D = 0$ from (4.10). Fig. 4.5(a) and 4.5(b) show the resulting base velocity commands and computed pose errors while moving the PR2 robot 1.5m (4.9ft) in the x-direction. The proportional gain was set to 80% of the ultimate gain, at which the movements became unstable and oscillatory as depicted in Fig. 4.5(d) and 4.5(e). The derivative gains proved to reduce rather than increase the stability, and therefore K_D was set to a small value. This was caused by noisy velocity measurements of the error term $\dot{e}_p$ in (4.10) as shown in Fig. 4.5(c) and 4.5(f).

(iii) Small r_{th} and ψ_{th} values resulted in a sensitive response to user movements whereas large values led to a sluggish response. The values were set manually in a region that was most comfortable for the user. A minor issue was encountered

Table 4.1. Controller configurations for single and dual arm experiments. Each compliance controller gain has a separate value for translation and rotation.

Compliance controller		
	trans.	rot.
$K_{m,x}$	100	100
$K_{m,y}$	100	100
$K_{m,z}$	10	10
$D_{m,x}$	1.0	1.0
$D_{m,y}$	1.0	1.0
$D_{m,z}$	0.1	0.1

Torque controller	
$K_{P,r}$	3.0
$K_{D,r}$	0.1
Velocity controller	
K_P	3.4
K_D	0.1
r_{th}	0.05 [m]
ψ_{th}	0.10 [m]

when moving at slow velocities such that the pose error stayed close to the threshold values. The switching logic in (4.14) and (4.15) resulted in oscillations and abrupt movements.

(iv) The restoring gains in (4.7) were set such that the arm joints would slowly drift to their initial home position when moved. As expected, the restoring torque did not interfere with the velocity nor the compliance controller.

Once set, the parameters were left unchanged. See Table 4.1 for configurations used during testing.

4.4.2 Experimental Environment

To test the performance of the system, we conducted experiments comparing several methods of guiding the PR2 in a well-defined environment. The objective was to follow a figure eight pattern consisting of two adjacent circles of 1.22m (4ft) in radius. This simulated the turns and directional changes potentially needed for avoiding obstacles in a human environment. A user was asked to guide the robot 3

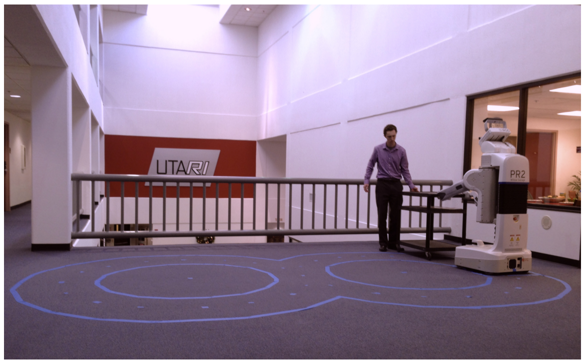


Figure 4.6. PR2 cart co-manipulation around a figure eight

.

times around the pattern as fast as possible without leaving the path. An inner and outer line on the ground marked a corridor that had the same width as the base of the PR2 (Fig. 4.6).

First, the operator followed the pattern using a joystick while walking in front of the robot. Next, the PR2 was pulled from the grippers and physically led around the figure eight by the user. Then a program was executed in which the PR2 would grasp the cart and the operator used the joystick again to teleoperate. Finally, the cart was pulled by a human operator to perform the same task. The pulling and cart following was tested with both the single and dual arm controllers.

Besides timing each completed loop, Adaptive Monte Carlo localization (AMCL) was used to track the pose of a robot against a known map [116]. The PR2 navigation apps stack provided the necessary utilities for mapping and localization. The *pr2_2dnav_slam* package uses the OpenSLAM's GMapping to build a map while the robot moves in the environment, which then can be used by the *pr2_2dnav* package for localization [117]. During the experiment, the PR2 torso was raised to a height of 0.3m. This provided enough clearance above the cart for the tilt laser to successfully perform measurements. A baseline was established by slowly driving the PR2 around

Table 4.2. Mean path deviation from baseline and standard deviation (STD) in meters.

Interface	No cart		With cart	
	Mean	STD	Mean	STD
Joystick	0.0433	0.0280	0.0413	0.0324
Single gripper	0.0442	0.0363	0.0562	0.0451
Dual gripper	0.0382	0.0302	0.0420	0.0314

Table 4.3. Mean completion time for guiding the PR2 around the figure eight pattern after 3 trials.

Interface	No cart (sec)	With cart (sec)	Percent. Difference (%)
Joystick	32.6	51.0	56.4
Single gripper	31.2	35.4	13.5
Dual gripper	32.4	33.3	2.78

the figure eight pattern with the joystick. This was the only time the operator walked behind the robot to avoid any interference with the laser scanner and perform the most accurate measurements as possible.

4.5 Evaluation of Interface Performance

The objective of the experiments was to measure the performance of the human-machine interface used for co-manipulating the cart. The inner and outer lines allowed the user to consistently guide the PR2 along the same path, making the completion time the main performance measure.

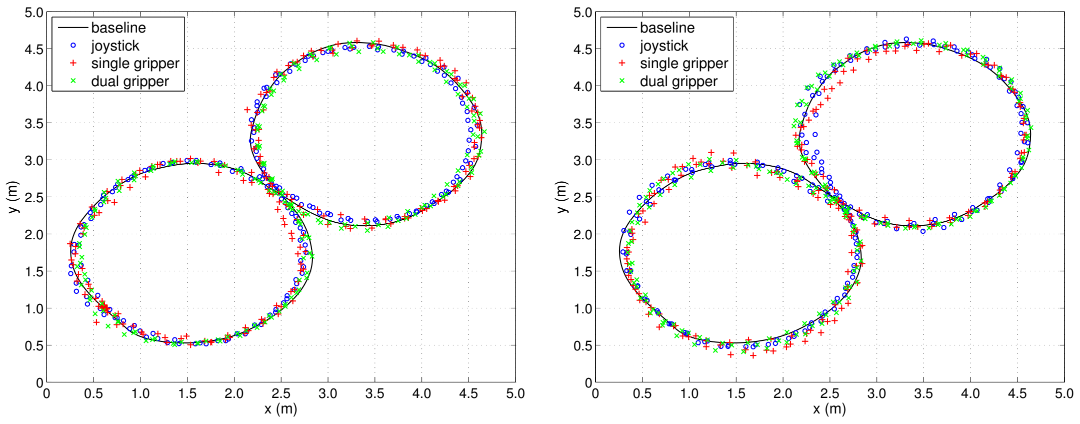
A finer spaced baseline was made by using cubic spline interpolation on the AMCL data points. For each interface position measurement k, a path deviation error e_k was computed by finding the minimum Euclidean distance between positions p_k and the baseline of n data points b_i:

$$e_k = \min\left(\sum_{i=1}^{n} \sqrt{(p_k - b_i)^2}\right). \tag{4.16}$$

The 3 trials for each interface produced approximately 200 data points (Fig. 4.7) and corresponding path errors e_k. The means and standard deviations are listed in Table 4.2, which are less than 5 cm except for the single gripper. The paths taken are similar and the performance is quantified by the completion time. When the interface performed poorly, the operator had to adopt a slower pace to stay within the figure eight boundaries. If the robot moved outside the lines, the user had to stop and re-adjust the position, which resulted in a larger time penalty.

First, the user navigated the robot without a cart along the pattern. There was no significant difference in the average completion time for the three different interfaces without the cart (Table 4.3). Since the operator walked backwards in front of the PR2, the joystick control was counterintuitive. The left and right directions were inverted, which could cause problems for less experienced joystick users. While turning with the single gripper controller, the PR2 would move sideways instead of rotating. The user had to use on arm to hold the gripper in place and the other to rotate or twist the gripper. Since the right gripper was used for velocity control, counter-clockwise turns were easier when the gripper was on the outside of the circle.

Guiding the PR2 together with a cart increased the average completion time, especially for joystick control (Table 4.3). The inertia of the cart made the joystick less responsive and higher velocities resulted in overcorrections. The lag and jerky



(a) without cart (b) with cart

Figure 4.7. AMCL position measurements for guiding the PR2 around a figure eight with different human-robot interfaces.

movements added additional confusion to the joystick being in the robot's and not the user's reference frame, making the operation difficult even for an experienced user. Similar to the previous results, the single gripper performed poorly when turning. To follow the path the user had to focus on the angle of the gripper rather than the cart. The dual arm interface performed best as reflected by a completion time similar to the one without the cart. These results show that co-manipulating the cart rather than indirectly moving it via joystick provides the most agile and intuitive interface.

4.6 Conclusions and Future work

A compliance and velocity controller was successfully implemented on a PR2 robot for compliant, stable arm positioning and base movement. They provide a simple, intuitive way for operators to reliably guide the robot or the robot-cart system. Co-manipulation through physical interaction proved to be better than the joystick teleoperation, as measured through completion times. This is perhaps not surprising in co-manipulation tasks where motions are relative to the robot rather than the user.

Furthermore, the user had difficulties turning the robot instead of moving it sideways with the single arm controller, whereas a dual arm controller had better performance.

This system was demonstrated at a live event called "Sky Ball XI", which is an annual fundraising event for US military veterans [118]. Specifically, the PR2 was used to serve wine to VIP guests seated at different tables. A cart with wine glasses and bottles had to be moved from table to table in a crowded, dynamic environment. The use of perception was limited due to uncontrolled lightning conditions that included many sources of potential interference with the robot's sensor. The proposed compliance and velocity controller performed well. Together with a human waiter, the PR2 successfully navigated through the crowded environment without any spills or other incidents.

Future work will focus on automatic and adaptive tuning of the controller gains. Improvements can be made by including force measurements into the system model. For example, a force-torque sensor installed between the gripper and forearm can provide feedback for an impedance controller. A dynamic threshold instead of switching logic can result in smoother transitions and filtering the noisy twist error can improve the derivative portion of the controller. Future testing could be done using actual obstacles in real environments that require more diverse paths.

APPENDIX

The source code for the ROS package *pr2_cartPull* is available at `uta.edu/ee/ngs/papers/2014case/`. It consists of a real-time plugin and a manager for arm controller switching, obtaining the controller status, as well as service calls for setting gain and threshold values. The manager utilizes motion clients from our *pr2_motion_clients* package for initializing the torso, arm, and gripper positions. A keyboard interface is provided for executing the different manager functionalities.

ACKNOWLEDGMENT

This research was supported in part by the NSF Grant NRI-1208623. We wish to thank the UTA Research Institute (UTARI) Living Laboratory staff members for their assistance with the experiments conducted with the PR2.

Part II

Neuroadaptive Control

CHAPTER 5

Neuroadaptive Control for Safe Robots in Human Environments

NEUROADAPTIVE CONTROL FOR SAFE ROBOTS IN HUMAN ENVIRONMENTS: A CASE STUDY

I. Ranatunga, **S. Cremer**, F. L. Lewis, and D. O. Popa, "Neuroadaptive control for safe robots in human environments: A case study," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 322–327.

5.1 Abstract

Safety is an important consideration during physical Human-Robot Interaction (pHRI). Recently the community has tested numerous new safety features for robots, including accurate joint torque sensing, gravity compensation, reduced robot mass, and joint torque limits. Although these methods have reduced the risk of high energy collisions, they rely on reduced speed or accuracy of robot manipulators. Indeed, because lightweight robots are capable of higher velocities, knowledge of dynamical models is required for precise control. However, feed-forward compensation is difficult to implement on lightweight robots with flexible and nonlinear joints, links, cables, and so on. Furthermore, unknown objects picked up by the robot will significantly alter the dynamics, leading to deterioration in performance unless high controller gains are used. This paper presents an online learning controller with convergence guarantees, that is able to learn the robot dynamics on the fly and provide feed-forward compensation. The resulting joint torques are significantly lower than conventional independent joint control efforts, thus improving the safety of the robot. Experiments on a PR2 robot arm are conducted to validate the effectiveness of the neuroadaptive controller to reduce control torques during high speed free-motion, lifting unknown objects, and collisions with the environment.

5.2 Introduction

For robots to work in collaboration with humans good physical Human-Robot Interaction (pHRI) is vital. According to De Santis et al. [15] safe and dependable pHRI systems should be developed before introducing robots into human environments. Safety in an industrial context was explored by Haddadin et al. in several studies which measured and characterized the results of robot collisions with hu-

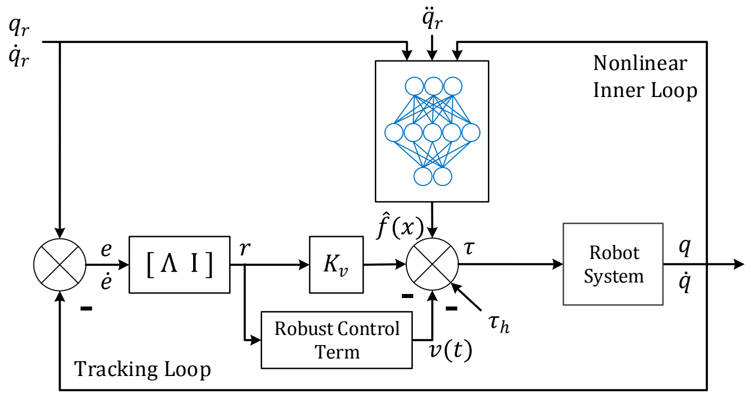


Figure 5.1. Model Reference Neuroadaptive Controller.

mans [104, 119–121]. In their work, novel hardware capable of accurate joint torque sensing was developed to reduce impact forces.

Interest in safe pHRI has motivated hardware modifications including gravity compensation using a counter balance such as the PR2 [122]. Recent developments in hardware based safety features include accurate joint torque sensing, gravity compensations, reduced robot weight, joint torque limiting, and bio-inspired hybrid actuation [17]. Although these methods have reduced the risk of high energy collisions, they have also significantly reduced the speed and accuracy of robot manipulators.

In household robot applications the robots are generally compliant and safe with changing often imprecise dynamic models. The tasks involved also change from pick and place tasks requiring accurate trajectory tracking, to physical interaction with humans. Model-based computed torque controllers can achieve good results but they depend on known dynamic models. Even if a robot model is known, the presence of a payload comparable with the mass of a lightweight robot will alter the performance of such compensation schemes. Furthermore, nonlinearities due to the inherent flexibility of lightweight transmission systems will also increase the uncertainty of such models.

Established techniques that overcome the need for precise models include: adaptive control, which estimates the model parameters, and robust control, which makes the controller resistant to unknown parameters [110]. Several methods have been proposed to avoid linear-in-the-parameters assumptions of adaptive controllers, including neural networks, support vector machines, Gaussian mixture models, and reinforcement learning. These are designed to learn the full nonlinear dynamics of the robot [22, 29, 123, 124]. Thus, learning methods are a great asset to overcome the need for modelling, and some of these methods offer stability and performance guarantees.

The use of neural networks (NN) in feedback control systems was first proposed by [25] and [26]. The properties of interest for trajectory tracking using NN based controllers are the tracking error and NN estimation errors. Some of the first results that included internal stability, weight bounds, tracking performance guarantees, and controller robustness was provided in 1995 [125, 126].

However there has also been a lack of rigorous testing and real world validation of these controllers on complex robots and tasks until recently. Neuroadaptive controllers were implemented on the Atlas humanoid robot [127] as part of the DARPA Virtual Robotics Challenge, while adaptive admittance pHRI schemes have been experimentally validated [44].

In this paper the impact of such neuroadaptive controllers to the safety of robots is studied. The neuroadaptive control method [22, 125] was applied to lifting tasks on a PR2 robot [122], which comes with default factory-tuned PID control gains for independent joint control. Experiments were conducted to comparatively test the controllers under different manipulator loading conditions. Results demonstrate the effectiveness of the neuroadaptive controllers in three types of tasks from a safety perspective.

This paper is organized as follows: Section 5.3 outlines the joint space controller formulation. Results from experiments on a PR2 robot with trajectory tracking under different loading conditions is presented in Section 5.4. Finally, Section 5.5 concludes the paper.

5.3 Controller Formulation

In this section the neuroadaptive controller formulation developed by Lewis et. al [22, 125] is briefly described (Fig. 5.1). The controller formulation used in this paper is in joint space assuming desired joint space trajectories.

The general robot dynamic equation with actuator dynamics is [110]

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau \tag{5.1}$$

where n is the number of degrees of freedom (DOF) of the robot, $q \in \mathbb{R}^n$ are the joint positions, $M(q)$ is the inertia matrix, $V(q,\dot{q})$ is the Coriolis/centripetal vector, $G(q)$ is the gravity vector, and $F(\dot{q})$ is the friction term. The disturbance torque is $\tau_d \in \mathbb{R}^n$ and $\tau \in \mathbb{R}^n$ is the control torque.

Given a desired reference trajectory q_r, define the trajectory-following error as

$$e = q_r - q \tag{5.2}$$

and the sliding mode error is

$$r = \dot{e} + \Lambda e \tag{5.3}$$

where Λ is a positive definite design parameter matrix.

Using (9.1), (5.2) and (5.3) yields the robot error dynamics

$$M(q)\dot{r} = -V(q,\dot{q})r + f(x) + \tau_d - \tau \tag{5.4}$$

where

$$f(x) = M(q)(\ddot{q}_r + \Lambda\dot{e}) + V(q,\dot{q})(\dot{q}_r + \Lambda e) + F(\dot{q}) + G(q) \tag{5.5}$$

is a nonlinear function of unmodeled robot parameters.

Let an approximation based controller be

$$\tau = \hat{f}(x) + K_v r - v(t) \tag{5.6}$$

where $\hat{f}(x)$ is the approximation of the robot function $f(x)$ in (5.5), $K_v r$ is the gain of the outer PD tracking loop, $K_v = K_v^T > 0$ is a diagonal outer-loop gain matrix, and $v(t)$ is a robustifying signal that compensates for unmodelled and unstructured disturbances.

Putting (5.6) in (5.4) and simplifying yields the closed loop error dynamics

$$M(q)\dot{r} = -V(q,\dot{q})r - K_v r + \tilde{f}(x) + \tau_d + v(t) \tag{5.7}$$

where $\tilde{f}(x) = f(x) - \hat{f}(x)$ is the function approximation error.

The learning loop performance and stability proof is based on prior works [22, 29, 125].

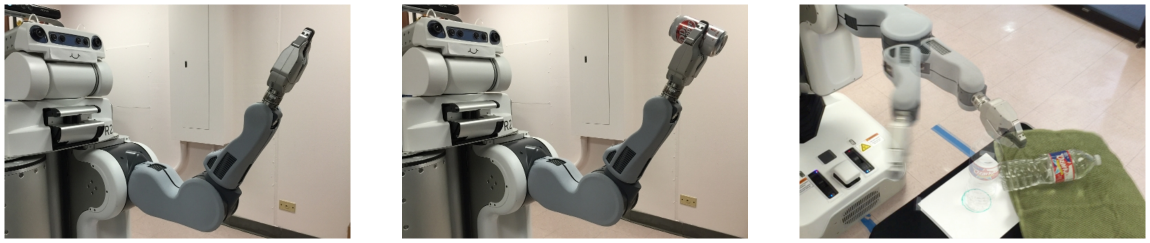

(a) No payload (b) With payload (c) Collision

Figure 5.2. Experimental setups for comparing NN and PID controllers.

The nonlinear function $f(x)$ in (5.5) is unknown. This function can be approximated by a neural network (NN)

$$f(x) = W^T \sigma(V^T x) + \varepsilon \tag{5.8}$$

where W and V are ideal unknown weights and $\sigma(.)$ is the activation function. Let the neural network approximation property given by (7.10) hold for the function $f(x)$, specified by (5.5) with a given accuracy $\|\varepsilon\| \leq \varepsilon_N$ on a compact set [22, 29].

The ideal weights for the NN are unknown, therefore the weight tuning algorithms of [22, 29, 125] are used to update the approximate NN weights $\hat{W}$ and $\hat{V}$. The input to the NN is $x = \left[\begin{array}{ccccc} e^T & \dot{e}^T & q_r^T & \dot{q}_r^T & \ddot{q}_r^T \end{array}\right]^T$. Then the control input is

$$\tau = \hat{W}^T \sigma(\hat{V}^T x) + K_v r - v \tag{5.9}$$

The robustifying signal $v(t)$ is

$$v(t) = -K_z(\|\hat{Z}\|_F + Z_B) r \tag{5.10}$$

where K_z is the gain of the robustifying term

$$\hat{Z} = \begin{bmatrix} \hat{W} & 0 \\ 0 & \hat{V} \end{bmatrix}$$

$\|.\|_F$ is the Frobenius norm, and Z_B is a bound on the NN weights.

The following are the NN weight update equations

$$\dot{\hat{W}} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^T x r^T - \kappa F\|r\|\hat{W} \tag{5.11}$$

$$\dot{\hat{V}} = Gx(\hat{\sigma}'^T\hat{W}r)^T - \kappa G\|r\|\hat{V} \tag{5.12}$$

$$\hat{\sigma}' = diag\left\{\sigma(\hat{V}^T x)\right\}\left[I - diag\left\{\sigma(\hat{V}^T x)\right\}\right] \tag{5.13}$$

where F and G are positive definite matrices, and $\kappa > 0$ is a small design parameter. (5.13) assumes that $\sigma(.)$ is a sigmoid activation function.

Define

$$S_r \equiv \{r \mid \|r\| < \frac{b_x - q_B}{c_0 + c_2}\} \tag{5.14}$$

where c_0, c_2 are computable positive constants. If $r(0) \in S_r$, then the approximation property holds. Details on Uniform Ultimate Bounds (UUB) of both $\|r\|$ and $\|\hat{Z}\|_F$ so that the approximation property holds throughout, can be found in [22, 29].

5.4 Experimental Results

In this section, we present results from experiments on a PR2 robot equipped with ATI Mini45 force/torque sensors. The neuroadaptive controller was implemented in ROS Groovy [102] using the real-time controller manager framework of the PR2. Both the neuroadpative and the standard PID controller were executed at 1000Hz in hard real-time. The joint positions, joint torques, and end-effector force data was collected at the same rate. The 7 degrees of freedom (DOF) left arm was used to conduct the experiments.

The controller parameters used were $K_v = 5I_7$, $\Lambda = 5I_7$, $F = 100I_7$, $G = 50I_7$, $\kappa = 0.07$, $K_z = 0.001$, and $Z_b = 100$, where I_7 is the 7×7 identity matrix. A two-layer Neural Network with 36 inputs including the bias input, 10 hidden layer neurons, and 7 outputs was used. The sigmoid function $\sigma(x)$ was used for the activation functions. The weights $\hat{W}$ and $\hat{V}$ of the network were initialized to small random values.

Three different types of experiments were conducted to demonstrate the effectiveness of the neuroadaptive controller as compared to the standard robot PID joint controllers (Fig. 5.2). The experiments conducted were:

A. Free space motion without payload, in which the arm is following a desired joint trajectory.

B. Free space motion with payload, in which the end-effector is carrying an object of unknown mass.

C. Collision experiments, in which an unknown obstacle is encountered during robot motion.

In all experiments a sinusoidal trajectory was applied to joint 3, i.e. the elbow joint of the PR2 arm. In the first two experiments, the amplitude of the joint motion was set to 0.5 radians. Joints 0 through 6 were positioned at $q = [0, 0, 0, 1.0, 0, 0, 0]^\intercal$, corresponding to the shoulder pan, shoulder lift, upper arm roll, elbow flex, forearm roll, wrist flex, and wrist roll joints. Five different rates or angular frequencies were tested in experiment *A*. In experiment *B*, a soda can weighing 355 grams was used as a payload. The highest rate was not tested because the motion became too distorted due to torque saturation and the desired reference trajectory could no longer be followed. In experiment *C*, the amplitude was changed to 0.75 radians and the arm joints were set to $q = [0, 0, 1.57, 0.75, 0, 0, 0]^\intercal$. A 1 liter water bottle was placed in the path of the gripper directly in front of the PR2 (Fig. 5.2c). Ten collision were performed with each controller by executing the sinusoidal trajectory with a rate of 3 radians per second.

5.4.1 Free Space Motion

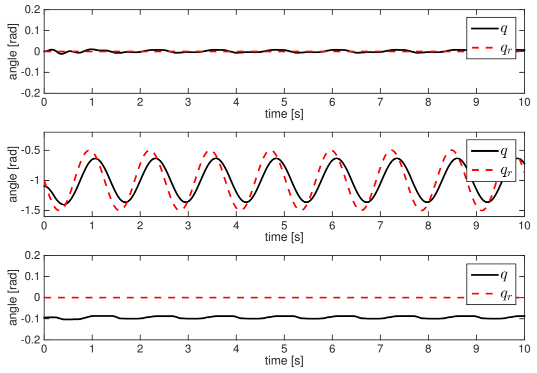
The experiments without a payload was carried out to compare the performance of the controllers in free space. To quantify the joint tracking performance, the error (5.2) at each time step $\Delta t = 0.001$sec was computed over a period of 10 seconds. The 2-norm was computed for each joint and then summed:

$$\sum\nolimits_{i=0}^{6} \|q_r - q\|_2 \tag{5.15}$$

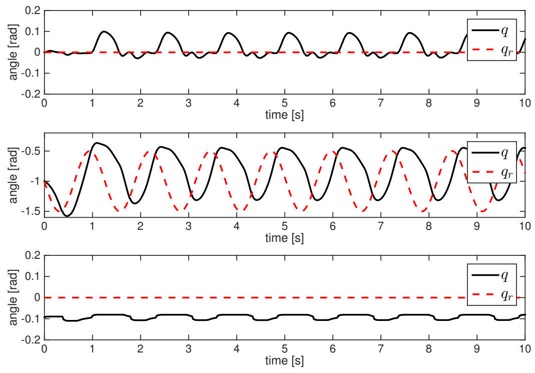
where i is the joint number. Similarly, the norm of the torques was computed for each joint and then combined into a torque performance value:

$$\sum\nolimits_{i=0}^{6} \|\tau\|_2 \tag{5.16}$$

Figs. 5.3a and 5.3b show the executed trajectory q and desired reference trajectory q_r for joints 1, 3, and 5, which correspond to the shoulder lift, elbow flex, and wrist flex joints. As expected, the shoulder pan and the roll joints displayed little to no error and are therefore not depicted. At 5 radians per second both controllers have degraded performance, with the PID controller performing worse. Fig. 5.7a shows the the total error of the two controllers at five different rates. The tracking error is initially lower for the PID controller, but increases more dramatically with the joint velocities. The joint tracking is more consistent for the neuroadaptive controller because it can compensate for changes in the robot dynamics. The standard PID

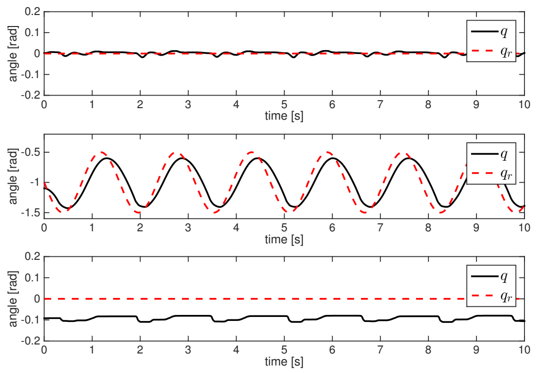


(a) Neuroadaptive

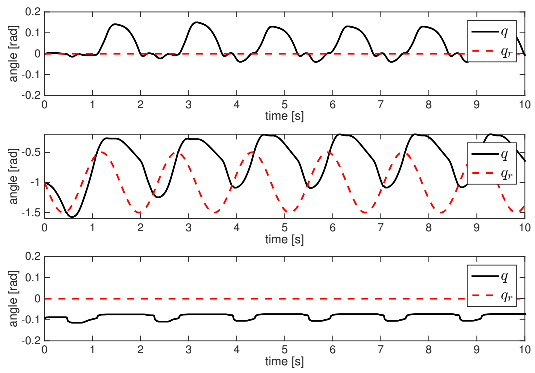


(b) PID

Figure 5.3. Tracking performance of joints 1, 3, and 5 without a payload at 5 radians per second.



(a) Neuroadaptive

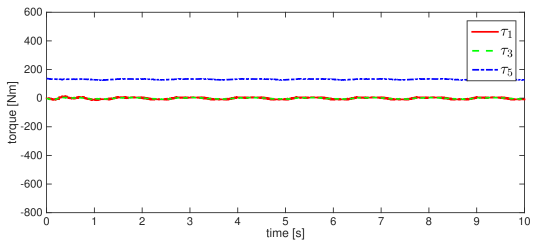


(b) PID

Figure 5.4. Tracking performance of joints 1, 3, and 5 with a 355gram payload at 4 radians per second.

controller can only be tuned for a limited range of joint velocities and fails when those are exceeded.

The neuroadpative controller also performs better from a safety point of view when considering the lower torque values clearly shown in Figs. 5.5a and 5.5b. Interestingly, joint 5 (wrist flex) has the largest control effort while the PID controller generates the highest torques for joint 3 (elbow flex). The total joint torque is depicted in Fig. 5.8a. At 1 radians per second the performance is comparable. As the


(a) Neuroadaptive

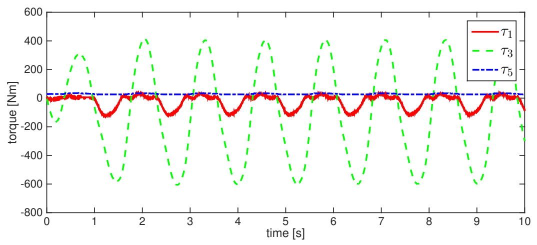
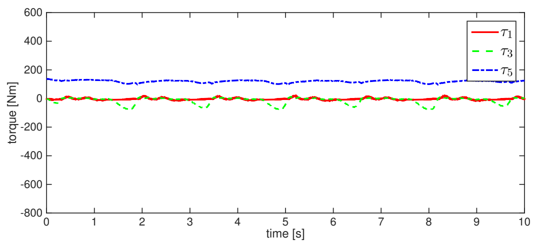

(b) PID

Figure 5.5. Control torque without payload at 5 radians per second.


(a) Neuroadaptive

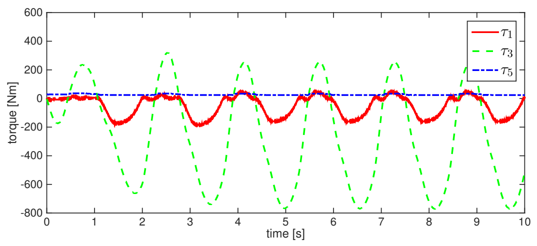
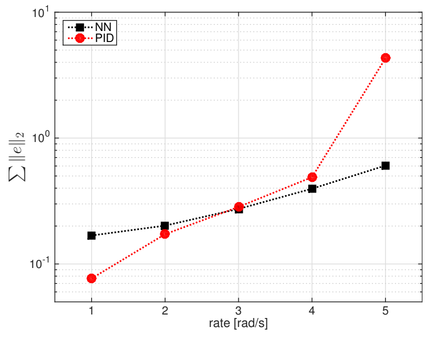

(b) PID

Figure 5.6. Control torque with a 355gram payload at 4 radians per second.


(a) No payload

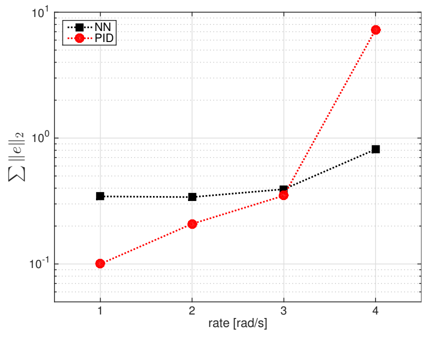
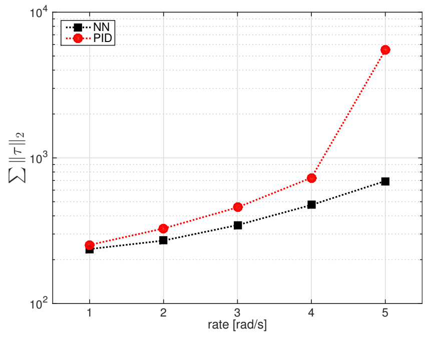

(b) With payload

Figure 5.7. Total joint tracking error for different movement frequencies (log scale).


(a) No payload

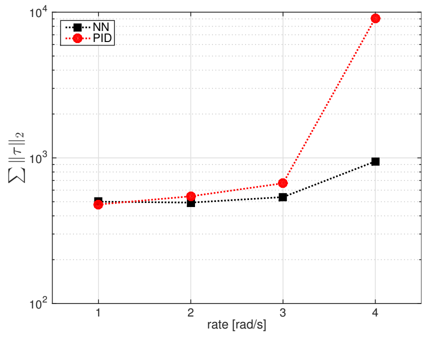

(b) With payload

Figure 5.8. Total control torque for different movement frequencies (log scale).

joint velocities increase, the PID control torque increases rapidly and diverges faster than the neuroadaptive controller.

5.4.2 Free Space Motion with Payload

The added payload changes the manipulator dynamics by increasing the end-effector inertia. In traditional computed torque or inverse dynamics controllers, this change in manipulator dynamics has to be detected and explicitly added to the controller. Since the weight of the object is unknown to the robot in this experiment, the tracking performance is worse even at lower rates.

Similarly to the previous section, the joint tracking performance of the neuroadaptive controller outperforms the PID controller at the highest tested rate as shown in Figs. 5.4a and 5.4b. In addition, the control torques generated by the PID controller are much higher than those generated by the neuroadaptive controller (Figs. 5.6a and 5.6b). However, at lower rates the difference is less profound as depicted in Figs. 5.7b and 5.8b.

The results for Sections 5.4.1 and 5.4.2 are summarized in Table 5.1.

5.4.3 Collision with Obstacle

In this section the contact forces experienced at the end effector of the robot during collisions is presented. The force tangential to the circular motion of the elbow flex joint (joint 3) was measured during a 2.5 second time interval. Typical results for the neuroadaptive and PID controller are shown in Fig. 5.9.

The results of running ten trials are represented in Table 5.3. The maximum force $\|F\|_\infty$ and impulse $\|F\|_1 \Delta t$ was computed for each trial. The neuroadaptive controller produces an average maximum contact force of 4.85N, which is lower than the PID value of 6.99N. Furthermore, the average impulse is lower at 0.56Ns compared

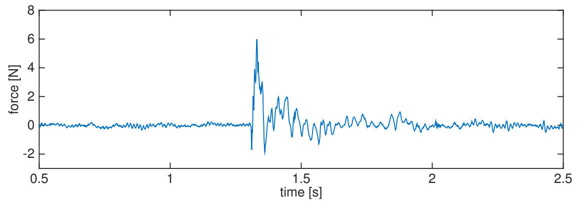
Table 5.1. Neuroadaptive (NA) and PID controller joint error and torque performance results for experiment A and B.

Rate	Payload	$\sum \lVert e \rVert_2$		$\sum \lVert \tau \rVert_2$	
(rad/s)	(g)	NA	PID	NA	PID
1	0	0.17	0.08	236.62	252.54
2		0.20	0.17	270.49	327.73
3		0.27	0.28	345.46	457.11
4		0.40	0.49	474.83	730.07
5		0.60	4.31	693.16	5492.36
1	355	0.34	0.10	500.32	478.82
2		0.34	0.21	493.11	544.68
3		0.39	0.35	537.95	669.06
4		0.81	7.28	945.80	9119.96

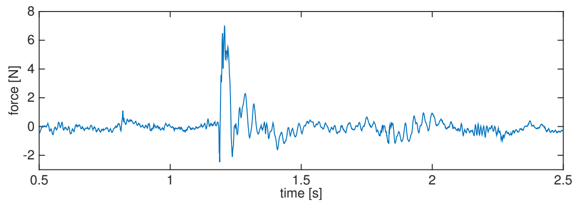
to 0.89Ns. Hence, the neuroadaptive controller results in lower collision energies and could therefore be considered a safer alternative to the standard PID controller. This is especially important in human environments, which are highly dynamic and unpredictable. In physical HRI interaction scenarios, the controller has to be accurate and responsive, while minimizing the risk of human injury.

Table 5.2. Collision performance results for 10 trails.

Controller	$\lVert F \rVert_\infty$ (N)		$\lVert F \rVert_1 \Delta t$ (N s)	
	Mean	STD	Mean	STD
NA	4.8532	0.7982	0.5577	0.0456
PID	6.9937	0.5683	0.8867	0.0474



(a) Neuroadaptive



(b) PID

Figure 5.9. Contact force versus time during the collision experiment.

5.5 Conclusions and Future Work

In this paper the performance of the neuroadaptive controller was compared against independent joint control via experiments on a PR2 robot. Results demonstrated the effectiveness of the neuroadaptive controllers in joint trajectory following tasks, and during handling of objects with unknown mass. In particular, the neuroadaptive controller had superior tracking performance at high joint rates, and much lower joint torques while lifting payloads. Tests were also conducted to demonstrate the performance of the neuroadaptive controller compared to a PID controller during impact. This test was conducted to demonstrate the inherent safety afforded by the neuroadaptive controller by reducing the impact forces.

Future work will involve further testing the controllers with different weight update laws, neural network size, and activation functions. The use of neuroadaptive controllers in conjunction with impedance and admittance control will also be studied.

Table 5.3. Collision performance results for 10 trails.

Controller	$\|F\|_\infty$ (N)		$\|F\|_1 \Delta t$ (N s)	
	Mean	STD	Mean	STD
NA	4.8532	0.7982	0.5577	0.0456
PID	6.9937	0.5683	0.8867	0.0474

ACKNOWLEDGMENT

This research was supported by NSF Grant NRI-1208623, NSF grant ECCS-1405173, NSF grant IIS-1208623, ONR grant N00014-13-1-0562, AFOSR EOARD Grant 13-3055, and ARO grant W911NF-11-D-0001. We wish to thank University of Texas Arlington Research Institute (UTARI) Living Laboratory staff for their assistance with the experiments conducted with the PR2.

CHAPTER 6

Neuroadaptive Control with Prescribed Error Dynamics

The crossover aircraft pilot model developed by McRuer et al. [128] in 1974 was one of the first mathematical models of the human operator and is experimentally verified. It states that the combined human-machine dynamics remain unchanged for a broad range of operating frequencies centered at w_c. If Y_p is the pilot and Y_c the aircraft transfer function, then the equivalent transfer function

$$Y_p Y_c = \frac{w_c e^{-\tau s}}{s} \tag{6.1}$$

where w_c is the crossover frequency and τ a time delay introduced by the neuromuscular system. When the dynamics of the controlled element Y_c changes, the human adjust their control characteristics Y_p such that the total system remains unchanged. The human intuitively learns to compensate for the new dynamics in Y_c which might require several trials and training exercises. Hence, a highly nonlinear machine becomes easier to operate if it behaves according to a fixed, linear admittance model. This is also applicable for physical Human-Robot Interaction. For example Suzuki et al. [129] utilized this principle to tune a virtual internal model of an interface system, resulting in an improved manipulation performance.

Suppose a human operator tries to guide a robot end-effector with pose $\mathbf{x}$. The desired motion trajectory $\mathbf{x}_d$ can be achieved by applying human forces $\mathbf{f}_h$ as shown

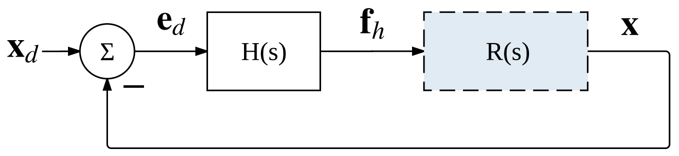


Figure 6.1. The human control loop with human transfer function $H(s)$ and robot dynamics $R(s)$.

in Fig. 6.1. Given the human transfer model $H(s)$ and robot transfer model $R(s)$, the combined task model

$$H(s)R(s) = D(s) \tag{6.2}$$

is a fixed first order system with a time delay. Assume the desired or prescribed error dynamics (PED) for $R(s)$ is

$$\ddot{\mathbf{e}} + D_d\dot{\mathbf{e}} + K_d\mathbf{e} = \mathbf{f}_h \tag{6.3}$$

where D_d, K_d can be fixed or adjusted according to some performance metric. Given the tracking error

$$\mathbf{e} = \mathbf{x}_r - \mathbf{x} \tag{6.4}$$

modify the neuroadapative (NA) controller described in Chapter 5 with a new novel sliding mode error

$$r = \dot{\mathbf{e}} + \Lambda\mathbf{e} - \mathbf{f}_l \tag{6.5}$$

$$\dot{r} = \ddot{\mathbf{e}} + \Lambda\dot{\mathbf{e}} + \dot{\Lambda}\mathbf{e} - \dot{\mathbf{f}}_l \tag{6.6}$$

where matrix $\Lambda(t)$ is a function of time. The filtered force f_l is determined from

$$\mathbf{f}_h = \dot{\mathbf{f}}_l + \Gamma \mathbf{f}_l \tag{6.7}$$

The following theorem is a formalization of the method in [130].

Theorem 1 *Given the filtered force (6.7) and the sliding mode error (6.5), define*

$$D_d = \Lambda + \Gamma \tag{6.8}$$

$$K_d = \dot{\Lambda} + \Gamma\Lambda \tag{6.9}$$

If $r \to 0$, then the system behaves according to the error dynamics in (6.3).

Proof: Define an error signal

$$w = \ddot{\mathbf{e}} + D_d\dot{\mathbf{e}} + K_d\mathbf{e} - \mathbf{f}_h \tag{6.10}$$

Substituting the definitions into the error signal gives

$$\begin{aligned} w &= \ddot{\mathbf{e}} + (\Lambda + \Gamma)\dot{\mathbf{e}} + (\dot{\Lambda} + \Gamma\Lambda)\mathbf{e} - (\dot{f}_l + \Gamma f_l) \\ &= \ddot{e} + \Lambda\dot{e} + \dot{\Lambda}e - \dot{f}_l + \Gamma(\dot{e} + \Lambda\mathbf{e} - f_l) \\ &= \dot{r} + \Gamma r \end{aligned} \tag{6.11}$$

If $r \to 0$ then $w \to 0$ and the system behaves according to (6.3). ■

Hence, introducing the auxiliary parameters $\mathbf{f}_l$, Γ, and Λ makes it is possible for the NA controller to behave like a simple linear admittance model with gains K_d and D_d. This simplifies the robot error dynamics and makes the interaction more intuitive. Less effort is required by the user by not having to learn the robot model and constantly adjust their own control dynamics. The modified control structure

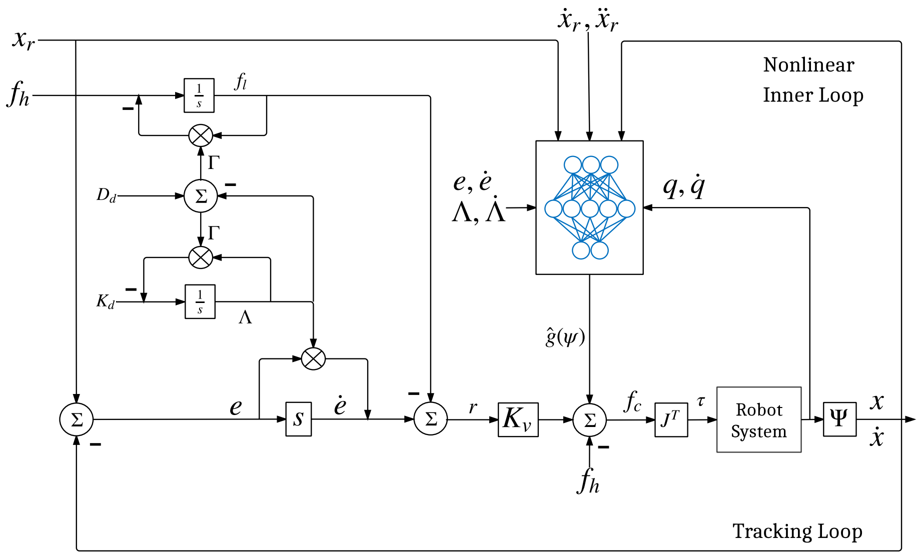


Figure 6.2. The neuroadaptive controller expanded with prescribed error dynamics (PED) specified by the gains K_d and D_d.

is shown in Fig. 6.2. The NA with PED was implemented on a PR2 robot and is experimentally validated in Chapter 8.

Part III

Inner/Outer-loop Control Structure

CHAPTER 7

Intent Aware Adaptive Admittance Control for Physical Human-Robot Interaction

INTENT AWARE ADAPTIVE ADMITTANCE CONTROL FOR PHYSICAL HUMAN-ROBOT INTERACTION

I. Ranatunga, **S. Cremer**, D. O. Popa, and F. L. Lewis, “Intent aware adaptive admittance control for physical Human-Robot Interaction,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5635–5640.

7.1 Abstract

Effective physical Human-Robot Interaction (pHRI) needs to account for variable human dynamics and also predict human intent. Recently, there has been a lot of progress in adaptive impedance and admittance control for human-robot interaction. Not as many contributions have been reported on online adaptation schemes that can accommodate users with varying physical strength and skill level during interaction with a robot. The goal of this paper is to present and evaluate a novel adaptive admittance controller that can incorporate human intent, nominal task models, as well as variations in the robot dynamics. An outer-loop controller is developed using an ARMA model which is tuned using an adaptive inverse control technique. An inner-loop neuroadaptive controller linearizes the robot dynamics. Working in conjunction and online, this two-loop technique offers an elegant way to decouple the pHRI problem. Experimental results are presented comparing the performance of different types of admittance controllers. The results show that efficient online adaptation of the robot admittance model for different human subjects can be achieved. Specifically, the adaptive admittance controller reduces jerk which results in a smooth human-robot interaction.

7.2 Introduction

Physical interaction between humans and robots has become a viable option with the development of multi degree of freedom (DOF) human scale robots, large scale tactile sensors, and development of full body multi-contact force control theory. In the application domain, human-robot interaction in industrial and household settings have moved into the physical Human-Robot Interaction (pHRI) domain.

The application of humans and robots working in collaboration has promising results in terms of efficiency and collaborative performance. Work by ABB provides evidence that a hybrid human-robot cell can improve efficiency as compared to either a human or robot working alone [79]. Erden et al. present results on impedance measurement in a collaborative human-robot welding task [131]. Recent developments in wearable robotic limbs for manufacturing assistance involves close coordination between human and robotic limbs [132].

Direct and indirect force control has been extensively studied in terms of performance and stability [35]. Impedance control pioneered by Hogan [40] is the most popular form of indirect force control. It allows stable contact by the end effector of the robot and overcomes many instability issues associated with explicit force control. However, impedance control requires identifying the robot model as well as the environmental contact dynamics [38, 41, 42]. Admittance control, the dual of impedance control, has also been applied in various robotic applications [133, 134]. Recent work by Ficuciello et al. [135] exploits redundancy to improve the Cartesian space inertia decoupling of a 7 DOF robot arm.

Adaptive impedance control has been studied by numerous others [38, 136, 137]. Human intent has been used to adapt the impedance parameters of robot systems [138, 139]. For instance, Dimeas et al. [140] implements a Fuzzy logic controller for admittance model adaptation considering human intent. Their method involves offline tuning of the parameters. Lecours et al. [141] reports on performance tuning of an assist device based on separation of the position and velocity space of the robot.

The contribution of this paper is a new framework for online adaptive admittance control that takes human intent into account. The proposed method does not require offline model tuning prior to use, can apply to users of varying degrees of skill and force capabilities, and is also robust to changes in the robot dynamics. The

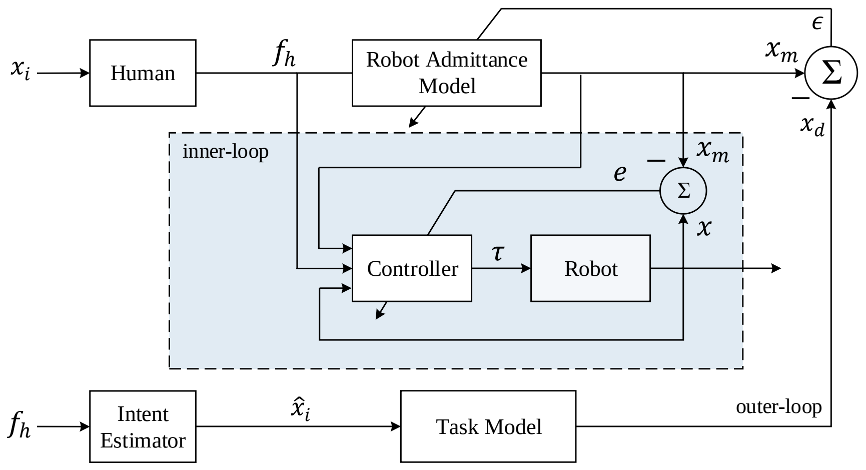


Figure 7.1. Inner-loop and outer-loop control system for physical HRI.

framework utilizes two loops: an inner neuroadaptive loop that feedback linearizes the robot, and an outer model reference adaptive loop that tunes the admittance model based on the intent of the human (Fig. 9.1). Past work has demonstrated that the inner-loop neuroadaptive controller linearizes the robot dynamics. Therefore it is an effective way to ensure consistent performance of the outer-loop across the entire robot workspace. It can also compensate for varying mass and stiffness characteristics of unknown humans in direct physical contact with the robot [22, 125]. Tuning of the outer-loop is based on an autoregressive inverse model that can encapsulate information about more specific human intent and task models. In this paper, the effectiveness of the outer-loop is illustrated against conventional admittance control in conjunction with simple point-to-point guided motions and human intent estimators. The advantage of our proposed control scheme is its generality, namely the ability to incorporate poorly known robots, sensors, tasks, and human intent models.

This paper is organized as follows: Section 7.3 first provides an overview of the control method proposed. Details of the outer-loop adaptive admittance controller with human intent estimation and the inner-loop neuroadaptive controller are provided next. In Section 7.4, the performed experiments are described and the results are presented. Finally, Section 7.5 presents the conclusions and discusses future work.

7.3 Adaptive Physical HRI Scheme

The proposed controller structure can be seen in Fig. 7.1, and consists of: (a) an inner-loop controller utilizing online learning with neural networks to estimate and cancel the nonlinear robot dynamics; (b) an adaptive admittance model that can be programmed to follow known robot Cartesian trajectories $\boldsymbol{x}_i$ or estimated trajectories $\hat{\boldsymbol{x}}_i$ resulting from a human intent estimator.

For the pHRI problem, the general dynamics equation in Cartesian space is written as follows [142]

$$\boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{x}} + \boldsymbol{J}^{\dagger T}\boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}_x(\boldsymbol{q}) = \boldsymbol{f}_c + \boldsymbol{f}_h \tag{7.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ are the joint positions, n is the DOF of the robot, $\boldsymbol{x} = \binom{\boldsymbol{p}_e}{\boldsymbol{\phi}_e} \in \mathbb{R}^6$ is the Cartesian space pose, where $\boldsymbol{p}_e \in \mathbb{R}^3$ is the task space position and $\boldsymbol{\phi}_e \in \mathbb{R}^3$ the orientation. $\boldsymbol{\Lambda}(\boldsymbol{q})$ is the Cartesian space inertia matrix, $\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Cartesian space Coriolis/centripetal vector, $\boldsymbol{g}_x(\boldsymbol{q})$ is the Cartesian space gravity vector, and $\boldsymbol{F}(\dot{\boldsymbol{q}})$ is the joint space friction term. $\boldsymbol{J}$ is the Jacobian and $\boldsymbol{J}^\dagger = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + k^2\boldsymbol{I})^{-1}$ is the damped least-squares pseudoinverse of the Jacobian with damping factor k. The force applied by a human operator in Cartesian space is $\boldsymbol{f}_h \in \mathbb{R}^6$. The Cartesian space control force $\boldsymbol{f}_c \in \mathbb{R}^6$ will be computed via a neuroadaptive scheme and can be used to compute the joint control torque $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{f}_c$.

Assume the following robot admittance model

$$\boldsymbol{M}_m\ddot{\boldsymbol{x}}_m + \boldsymbol{D}_m\dot{\boldsymbol{x}}_m = \boldsymbol{f}_h \tag{7.2}$$

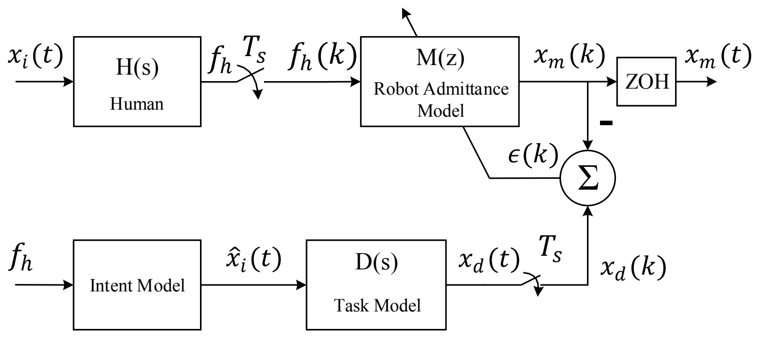


Figure 7.2. Human intent aware robot admittance model adaptation using adaptive inverse filtering.

where $\boldsymbol{x}_m$ is the admittance model output. The matrices $\boldsymbol{M}_m$ and $\boldsymbol{D}_m$ are the mass and damping matrices respectively. In this chapter we assume zero stiffness $\boldsymbol{K}_m = 0$, and no virtual trajectory. Therefore the admittance model is a simple mass-damper system in Cartesian space.

In Section 7.3.1, an adaptive admittance controller with human intent is proposed to generate desired trajectories for the robot to follow. This is shown in Fig. 7.2. If the ideal intent trajectory $\boldsymbol{x}_i$ is known, as it is usually the case in rehabilitation exercises, then it is sent to the human subject as an audiovisual cue and to a robot task model $D(s)$ which generates $\boldsymbol{x}_d$. An adaptive filter with human force $\boldsymbol{f}_h$ as an input and output $\boldsymbol{x}_m$ will then be employed to minimize the error between $\boldsymbol{x}_m$ and $\boldsymbol{x}_d$.

If the intent trajectory is not known, a human intent estimator can estimate it from interaction forces and/or other sensor measurements. In Figs. 7.1 and 7.2, the estimated human intent trajectory is denoted by $\hat{\boldsymbol{x}}_i \approx \boldsymbol{x}_i$. The human model can generally be defined as a transfer function $H(s)$ assumed to be unknown, and representing the dynamics of thinking and completing a trajectory tracking task by physically guiding the robot.

7.3.1 Outer-Loop: Adaptive Admittance with Human Intent

The outer-loop adaptive admittance controller is inspired by the adaptive inverse control approach pioneered by Widrow et al. [143]. The objective is to tune the admittance model in (7.2), denoted here by the discrete transfer function $M(z)$, such that the overall human-robot transfer characteristics $H(z)M(z)$ equals the task model $D(z)$. This outer-loop design uses sampled versions of $\boldsymbol{f}_h$ and the desired model output $\boldsymbol{x}_d(t)$ with a sampling period of T_s, namely $\boldsymbol{f}_h(k) = \boldsymbol{f}_h(kT_s)$ and $\boldsymbol{x}_d(k) = \boldsymbol{x}_d(kT_s)$.

Let $M(z)$ be given in ARMA form by

$$\begin{aligned} \boldsymbol{x}_m(k) = &-a_1\boldsymbol{x}_m(k-1) \;...\; - a_{n_\theta}\boldsymbol{x}_m(k-n_\theta) \\ &+ b_0\boldsymbol{f}_h(k) + b_1\boldsymbol{f}_h(k-1) + ... + b_{m_\theta}\boldsymbol{f}_h(k-m_\theta) \end{aligned} \tag{7.3}$$

where n_θ is the degree of the denominator of $M(z)$ and m_θ the degree of its numerator. Then $M(z)$ is implemented as

$$\boldsymbol{x}_m(k) \equiv \boldsymbol{h}^T(k)\boldsymbol{\theta} \tag{7.4}$$

where the measured regression vector is

$$\begin{aligned} \boldsymbol{h}^T(k) = [&-\boldsymbol{x}_m(k-1),\; ...\; , -\boldsymbol{x}_m(k-n_\theta), \\ &\boldsymbol{f}_h(k), \boldsymbol{f}_h(k-1),\; ...\; , \boldsymbol{f}_h(k-m_\theta)] \end{aligned} \tag{7.5}$$

and the ideal ARMA parameter vector is

$$\boldsymbol{\theta} = [a_1,\; ...\; , a_{n_\theta}, b_0, b_1,\; ...\; , b_{m_\theta}] \tag{7.6}$$

The estimated ARMA parameter vector $\hat{\boldsymbol{\theta}}(k)$ can be updated based on newly observed data $\boldsymbol{x}_m(k+1)$, $\boldsymbol{f}_h(k+1)$ by using a Recursive Least Squares (RLS) algorithm. This effectively converges to the Wiener solution by driving the error, $\boldsymbol{\epsilon}(k) = \boldsymbol{x}_d(k) - \boldsymbol{x}_m(k)$, to zero which makes the combined human and prescribed robot admittance model behave like the task model by solving $H(z)M(z) = D(z)$. The derivatives $\dot{\boldsymbol{x}}_m(t)$ and $\ddot{\boldsymbol{x}}_m(t)$ are obtained by applying zero-order hold (ZOH) and backward difference to $\boldsymbol{x}_m(k)$.

The above controller formulation assumes that the ideal human intent trajectory $\boldsymbol{x}_i(t)$ is known by the robot. In reality this signal is not available a priori. If the human intent is not taken into account by the outer-loop controller, the goal position is fixed and communicated to the human. This makes it impossible for the human subject to change the motion of the robot at will, and applications are restricted to rehabilitation exercises, or following known trajectories. In this case, if the user changes the goal in his/her mind, the controller will have no information about the new intent $\hat{\boldsymbol{x}}_i(t)$ and will fight against the user's movements.

Therefore, in a second formulation, we add a human intent estimator to generate approximations of $\boldsymbol{x}_i$. In this chapter, a simple model was tested which converts the user applied force on the robot to a desired future position propagated by some time interval. The following admittance model output is propagated into the future and it was assumed that this roughly approximates human intent $\boldsymbol{x}_i(t)$

$$\boldsymbol{M}_i \ddot{\hat{\boldsymbol{x}}}_i = \boldsymbol{f}_h, \qquad \hat{\boldsymbol{x}}_i(t) = \iint_t^{t+\Delta t} \ddot{\hat{\boldsymbol{x}}}_i(t)dt \tag{7.7}$$

where $\boldsymbol{M}_i$ is a fixed mass matrix, and $\boldsymbol{f}_h$ is the measured human force, and Δt is the propagation time. Thus, the human intent is obtained by double integration of the estimated intent acceleration $\ddot{\hat{\boldsymbol{x}}}_i(t)$. This simple model has been utilized for human walking path predictions in social situations by Luber et al. [144]. In this work, the future motion paths of humans were predicted using environmental constraint forces and intent forces towards a known goal.

7.3.2 Inner-Loop: Neuroadaptive Control

Given the model output $\boldsymbol{x}_m(t)$ and robot state $\boldsymbol{x}(t)$, the model following error will be $\boldsymbol{e} = \boldsymbol{x}_m - \boldsymbol{x}$. To drive $\boldsymbol{e}$ to zero, define a sliding mode error $\boldsymbol{r} = \dot{\boldsymbol{e}} + \boldsymbol{\Gamma}\boldsymbol{e}$ where $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^T > 0$ is a positive definite matrix.

From (9.1), the robot model following error dynamics can be written as

$$\boldsymbol{\Lambda}(\boldsymbol{q})\dot{\boldsymbol{r}} = -\boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{r} + f(\boldsymbol{\varphi}) - \boldsymbol{f}_c - \boldsymbol{f}_h \tag{7.8}$$

where

$$\begin{aligned} f(\boldsymbol{\varphi}) =& \boldsymbol{\Lambda}(\boldsymbol{q})(\ddot{\boldsymbol{x}}_m + \boldsymbol{\Gamma}\dot{\boldsymbol{e}}) + \boldsymbol{\mu}(\boldsymbol{q}, \dot{\boldsymbol{q}})(\dot{\boldsymbol{x}}_m + \boldsymbol{\Gamma}\boldsymbol{e}) \\ &+ \boldsymbol{J}^{\dagger T}\boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}_x(\boldsymbol{q}) \end{aligned} \tag{7.9}$$

is a nonlinear function of unmodeled robot parameters and argument $\boldsymbol{\varphi} = \left[\ \boldsymbol{e}^T\ \dot{\boldsymbol{e}}^T\ \boldsymbol{x}_m^T\ \dot{\boldsymbol{x}}_m^T\ \ddot{\boldsymbol{x}}_m^T\ \boldsymbol{q}^T\ \dot{\boldsymbol{q}}^T\ \right]^T$.

This function can be approximated by a neural network

$$f(\boldsymbol{\varphi}) = \boldsymbol{W}^T\sigma(\boldsymbol{V}^T\boldsymbol{\varphi}) + \boldsymbol{\varepsilon} \tag{7.10}$$

where $\boldsymbol{W}$ and $\boldsymbol{V}$ are ideal unknown weights, $\sigma(.)$ is an activation function, and $\boldsymbol{\varepsilon}$ is the approximation error, as detailed in [22].

The ideal weights $\boldsymbol{W}$ and $\boldsymbol{V}$ for the neural network (NN) are unknown. Therefore the weight tuning algorithms of [22, 125] are used to update approximate NN weights $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{V}}$.

Take the control input as

$$\boldsymbol{f}_c = \hat{\boldsymbol{W}}^T \sigma(\hat{\boldsymbol{V}}^T \boldsymbol{\varphi}) + \boldsymbol{K}_v \boldsymbol{r} - \boldsymbol{v}(t) - \boldsymbol{f}_h \tag{7.11}$$

where $\boldsymbol{K}_v = \boldsymbol{K}_v^T > 0$ is a diagonal outer-loop gain matrix,

$$\boldsymbol{v}(t) = -K_z(\|\hat{\boldsymbol{Z}}\|_F + Z_B)\boldsymbol{r} \tag{7.12}$$

is the robustifying signal as detailed in [22], K_z is the gain of the robustifying term, $\hat{\boldsymbol{Z}} = \begin{bmatrix} \hat{\boldsymbol{W}} & 0 \\ 0 & \hat{\boldsymbol{V}} \end{bmatrix}$, $\|.\|_F$ is the Frobenius norm, and Z_B is a scalar bound on the NN weights such that $\|\hat{\boldsymbol{Z}}\|_F < Z_B$.

The NN weight update equations are

$$\begin{aligned} \dot{\hat{\boldsymbol{W}}} = \boldsymbol{F}\sigma(\hat{\boldsymbol{V}}^T\boldsymbol{\varphi})\boldsymbol{r}^T - \boldsymbol{F}\sigma'(\hat{\boldsymbol{V}}^T\boldsymbol{\varphi})\hat{\boldsymbol{V}}^T\boldsymbol{\varphi}\boldsymbol{r}^T \\ -\kappa\boldsymbol{F}\|\boldsymbol{r}\|\hat{\boldsymbol{W}} \end{aligned} \tag{7.13}$$

$$\dot{\hat{\boldsymbol{V}}} = \boldsymbol{G}\boldsymbol{\varphi}(\sigma'(\hat{\boldsymbol{V}}^T\boldsymbol{\varphi})^T\hat{\boldsymbol{W}}\boldsymbol{r})^T - \kappa\boldsymbol{G}\|\boldsymbol{r}\|\hat{\boldsymbol{V}} \tag{7.14}$$

where $\boldsymbol{F}$ and $\boldsymbol{G}$ are positive definite matrices, $\sigma'(\boldsymbol{\xi}) = \frac{d\sigma(\boldsymbol{\xi})}{d\boldsymbol{\xi}}$, and $\kappa > 0$ is a small design parameter. It can be formally shown using a Lyapunov argument, that under reasonable assumptions, the error signal $\boldsymbol{e}$ will converge to zero. Therefore this inner-loop control scheme tracks Cartesian space trajectories $\boldsymbol{x}_m$ generated by the human and the outer-loop controller [22, 125].

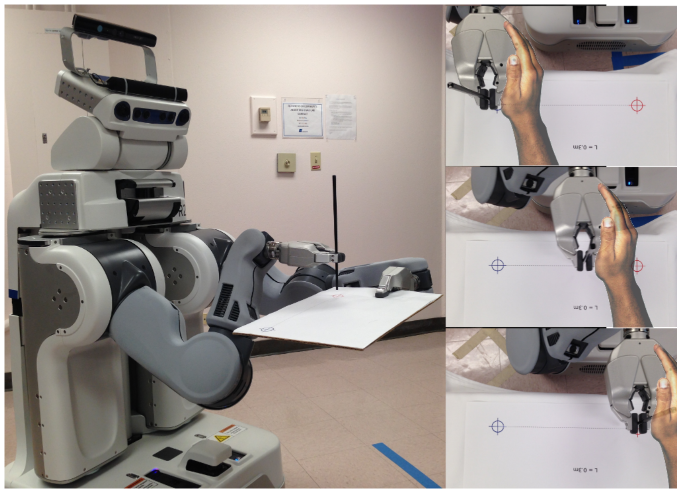


Figure 7.3. Experimental setup with the PR2 at the UTARI Living Lab.

7.4 Experiments

Validation experiments were conducted on a PR2 robot which has an omnidirectional mobile base, two 7-DOF gravity compensated arms with parallel grippers, a pan tilt head, and several sensor modules. The controller was implemented using the real-time controller manager framework of the PR2 available via the Robot Operating System (ROS) [102]. The real-time loop on the PR2 runs at 1000 Hz and is implemented using the linux-rt kernel. The communication with the sensors and actuators is via an EtherCAT network. Interaction forces and torques are measured using an ATI Mini40 Force/Torque (FT) sensor attached between the gripper and forearm of the PR2.

7.4.1 Experimental Setup

The experiments performed involved a point-to-point motion task. This task corresponds to a step function input to the system and enables clear analysis of the system performance. The PR2 robot was setup as shown in Fig. 7.3, where the subjects were asked to sit in front of the robot and grasp the right gripper with

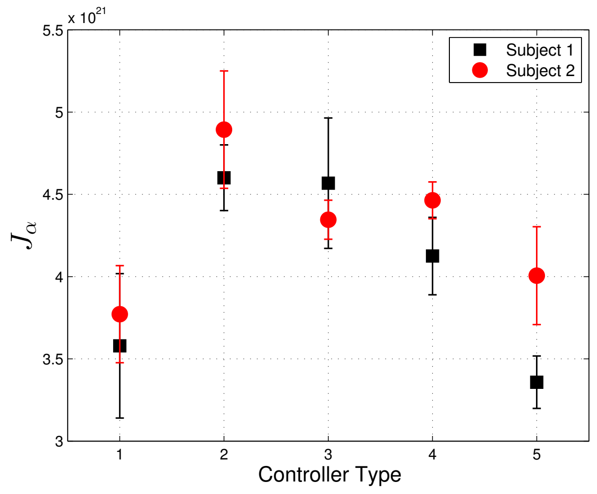


Figure 7.4. Error plot of J_α for controller Types 1 through 5.

their dominant hand. The start of the experiment was automatically indicated by the system via a text-to-speech program. The reference point-to-point motion was indicated by a "RED" or "BLUE" audiovisual cue to indicate the next movement point. Human factors studies [145] suggest that the human brain learns a closed-loop controller that behaves like a first order model with high bandwidth in closed-loop. Motivated by this a first-order transfer function is used as the task model

$$D(s) = \frac{a_d}{s + b_d} \tag{7.15}$$

where the model parameters depend on the specific task.

Experiments were conducted with the following five different types of controllers:

Type 1 *Adaptive admittance controller with human intent estimation*: In this experiment the admittance model in (7.4) is utilized. The admittance model in (7.7) is used to generate a time propagated signal corresponding to the human intent $\hat{\boldsymbol{x}}_i(t)$, where $\Delta t = 0.1$s. The matrix $\boldsymbol{M}_i$ is the 6×6 identity matrix. The parameters of $D(s)$ in (7.15) are selected as $a_d = b_d = 1.5$.

Type 2 *Adaptive admittance controller with fixed task reference trajectory*: In this experiment the admittance model in (7.4) is utilized with a fixed task reference trajectory $\boldsymbol{x}_i(t)$. The parameters of $D(s)$ are the same as for Type 1.

Type 3 *Admittance controller with fixed ARMA parameters*: In this experiment the ARMA parameter weights (7.6) were automatically tuned to a human subject and fixed as

$$\begin{aligned} \boldsymbol{\theta} = [\ &-1.8775 \;\; 1.2729 \;\; -0.0578 \;\; -0.3269 \\ &\;\;\,0.0018 \;\; 0.0001 \;\; -0.0013 \;\;\;\; 0.0002\ ] \end{aligned} \tag{7.16}$$

Type 4 *Fixed mass-damper admittance controller*: In this experiment the fixed mass-damper admittance model in (7.2) is used with the parameters tuned for Subject 1 as $\boldsymbol{M}_m = 20\boldsymbol{I}_6$, and $\boldsymbol{D}_m = 50\boldsymbol{I}_6$, where $\boldsymbol{I}_6$ is the 6×6 identity matrix. This model converts the user applied force $\boldsymbol{f}_h$ to Cartesian positions $\boldsymbol{x}_m(t)$ and velocities $\dot{\boldsymbol{x}}_m(t)$.

Type 5 *Direct task model controller*: In this experiment the output of the task reference model is sent directly to the inner-loop controller. The user applied force $\boldsymbol{f}_h$ is converted to Cartesian positions $\boldsymbol{x}_d(t)$ via the intent estimator.

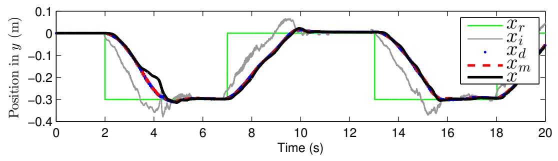
Three experimental trials were conducted for each type of controller with two male human subjects of ages 26 and 27. For all experiments, the inner-loop rate was 1000 Hz and the outer-loop rate was 20 Hz. The performance tuned inner-loop neuroadaptive controller parameters were $\boldsymbol{\Gamma} = 20\boldsymbol{I}_6$, $\boldsymbol{K}_v = 5\boldsymbol{I}_6$, $K_z = 0.001$, $Z_B = 100$, $\boldsymbol{F} = 100\boldsymbol{I}_6$, $\boldsymbol{G} = 200\boldsymbol{I}_6$, and $\kappa = 0.3$, where $\boldsymbol{I}_6$ is the 6×6 identity matrix. A two-layer NN with 44 inputs (including the bias input), 10 hidden layer neurons, and 6 outputs was implemented. The sigmoid function was used as the activation function $\sigma(\boldsymbol{\xi})$. The weights $\hat{\boldsymbol{V}}$ of the neural network were initialized to random values and the weights $\hat{\boldsymbol{W}}$ were initialized to zero.

7.4.2 Performance Measure

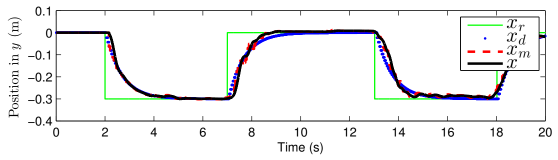
To compare the performance of the different controllers a performance measure J_α is utilized. It is the dimensionless squared jerk of the gripper motion in Cartesian space $\dddot{\boldsymbol{x}}(t)$. The work by Flash and Hogan [146] provides experimental evidence for a human motion model involving the minimization of jerk. Among the different jerk based performance measures proposed, the dimensionless squared jerk has been shown to be the most effective [146]. It is defined as

$$J_\alpha = \int_{t_1}^{t_2} \dddot{\boldsymbol{x}}(t)^2 dt \, \frac{(t_2 - t_1)^5}{A^2} \tag{7.17}$$

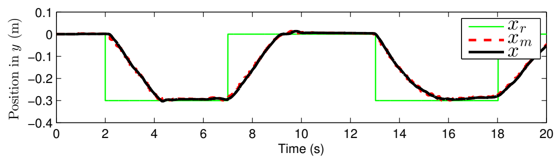
where A is the maximum amplitude of $\boldsymbol{x}(t)$. Smaller values of J_α indicate better human-robot interaction performance.



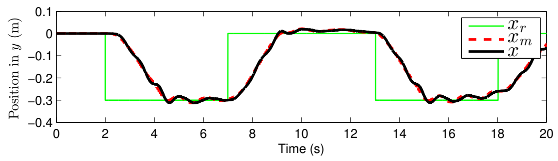
(a) Type 1: Adaptive admittance controller with human intent.



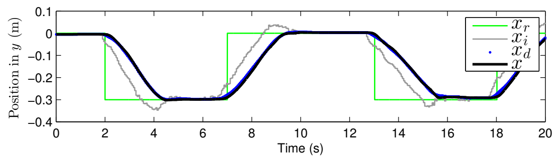
(b) Type 2: Adaptive admittance controller with fixed task reference.



(c) Type 3: Admittance controller with fixed ARMA parameters.

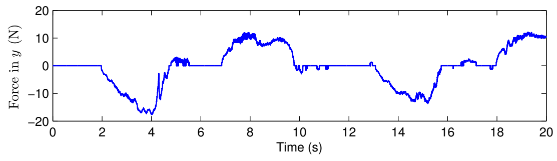


(d) Type 4: Fixed mass-damper admittance controller.

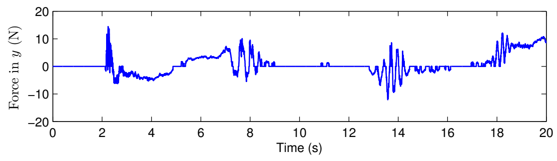


(e) Type 5: Direct task model controller.

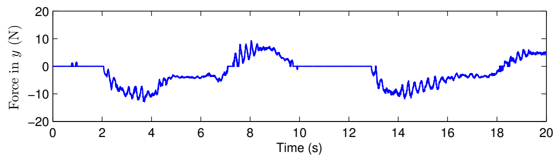
Figure 7.5. Postion in y (m) vs. time (s).



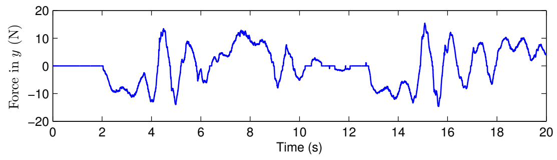
(a) Type 1: Adaptive admittance controller with human intent.



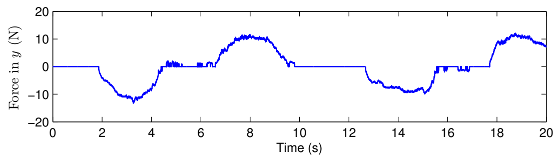
(b) Type 2: Adaptive admittance controller with fixed task reference.



(c) Type 3: Admittance controller with fixed ARMA parameters.



(d) Type 4: Fixed mass-damper admittance controller.



(e) Type 5: Direct task model controller.

Figure 7.6. Force in y (N) vs. time (s).

Table 7.1. Mean μ and standard deviation σ of J_α for controller types 1 through 5. In units of 10^{21}.

	Subject 1		Subject 2	
	Mean μ	σ	Mean μ	σ
T_1	3.5791	0.4388	3.7717	0.2948
T_2	4.6010	0.1999	4.8935	0.3570
T_3	4.5680	0.3962	4.3461	0.1187
T_4	4.1249	0.2351	4.4638	0.1117
T_5	3.3580	0.1593	4.0056	0.2973

7.4.3 Results

The best performance in terms of trajectory smoothness is achieved by the adaptive admittance controller with human intent estimation (Type 1) and the direct task model controller (Type 5). This is shown by the dimensionless squared jerk performance measure J_α in Fig. 7.4 and Table 7.1, and is consistent for both human subjects. The outer-loop in the Type 1 controller successfully tunes the admittance model based on the estimated human intent $\hat{\boldsymbol{x}}_i(t)$, resulting in admittance model trajectories $\boldsymbol{x}_m(t)$ that closely follow the task model trajectories $\boldsymbol{x}_d(t)$. The inner-loop controller enables accurate tracking of the admittance model trajectories. This results in robot trajectories $\boldsymbol{x}(t)$ that closely follow the admittance model trajectories $\boldsymbol{x}_m(t)$ shown in Fig. 7.5a. The controller also exhibits minimal force jitter as seen in Fig. 7.6a, resulting in a smooth force profile which indicates good interaction performance. The direct task model controller (Type 5) controller does not include an adaptive component, but due to the structure chosen for this system results in smooth trajectories. The task model then generates the appropriate response.

The controller of Type 2 with a fixed trajectory reference $\boldsymbol{x}_i(t)$ does not perform as well as the Type 1 and Type 5 controllers. This is indicated by higher values of J_α for Type 2. The minimization of the error between the task model trajectory $\boldsymbol{x}_i(t)$ and the admittance model trajectory $\boldsymbol{x}_m(t)$ tunes the outer-loop admittance model $M(z)$. This could result in a conflict between the robot trajectory and the human intent since priority is given to task model following. Evidence of this conflict is seen in the interaction force $\boldsymbol{f}_h$ oscillations in Fig. 7.6b. Also, Fig. 7.5b shows poor tracking performance of the task model $\boldsymbol{x}_d(t)$, admittance model $\boldsymbol{x}_m(t)$, and robot trajectories $\boldsymbol{x}(t)$.

The controllers of Types 3 and 4 with fixed ARMA weights and mass-damper do not include a desired model trajectory $\boldsymbol{x}_d(t)$ that could result in the controller competing with the human as in Type 2. These controllers perform marginally better than the Type 2 controller indicated by lower values of J_α. Although these controllers achieve relatively smooth position tracking as seen in Figs. 7.5c and 7.5d, the force profiles in Figs. 7.6c and 7.6d exhibit considerable jitter and oscillations.

The Type 3 controller utilizes the fixed ARMA model (7.16) that was automatically tuned to the human subject. The a priori tuning of the Type 3 controller ARMA weights is more efficient and precise than manually tuning a mass-damper system based on subjective human preferences. The manually tuned Type 4 controller still exhibits oscillations at steady state as shown in Fig. 7.5d.

Overall, the results provide evidence of the viability of the Type 1 adaptive admittance controller with human intent estimation in pHRI applications. The direct model reference controller which included no adaptation (Type 5) performed similar to the adaptive admittance controller (Type 1). The results for subject 2 showed better performance with the Type 1 controller while the results for subject 1 favoured the Type 5 controller. This requires further study by performing tests with different human subjects, intent estimators, and task reference models.

Both the Type 1 and Type 5 controllers resulted in minimized jerk in the coupled human-robot motion as compared to the other controller types. In addition, they are easier to deploy since parameter tuning is not needed. The Type 1 controller can adapt online in real-time to changes in human dynamics and intent. It can further be used to learn the optimal admittance using more complex intent estimator models. These complex models can then be replaced by the learnt admittance model further reducing final system complexity. The Type 2 controller can be utilized for automated rehabilitation systems where the human intent, i.e. the desired rehabilitation motion, is known. Here, the admittance model can be automatically tuned to the strength and ability of the patient. The Type 3 controller is a static tuned admittance model, such a controller can be used for instances where a one time admittance optimization is required, such as calibrating the system to different human operators.

7.5 Conclusions and Future Work

In this paper a two-loop adaptive admittance controller framework is presented that includes human intent estimation. An inner neuroadaptive loop feedback linearizes the robot, and an outer model reference adaptive loop tunes the admittance model based on the intent of the human. The inner-loop neuroadaptive controller linearizes the robot dynamics and ensures consistent performance of the outer-loop across the entire robot workspace.

Experiments were conducted with five different types of admittance controllers. The results demonstrated the performance advantages of the adaptive admittance controller with human intent estimation. The proposed method resulted in lower jerk in the combined human-robot motion, does not need any offline tuning, is robust to changes in human dynamics, and also compensates for changing robot dynamics.

Future work will include testing with more human subjects, integrating different task models, testing new intent models, and performing more complex Cartesian tasks. Another extension of this work is the use of multi-sensory data from whole body robot skin for adaptive admittance control.

ACKNOWLEDGMENT

This research was supported by NSF Grant NRI-1208623, NSF grant ECCS-1405173, NSF grant IIS-1208623, ONR grant N00014-13-1-0562, AFOSR EOARD Grant 13-3055, ARO grant W911NF-11-D-0001, China NNSF grant 61120106011, and China Education Ministry Project 111 (No.B08015). We wish to thank University of Texas Arlington Research Institute (UTARI) Living Laboratory staff for their assistance with the experiments conducted with the PR2.

CHAPTER 8

Model-Free Online Neuroadaptive Control with Intent Estimation for Physical Human-Robot Interaction

MODEL-FREE ONLINE NEUROADAPTIVE CONTROL WITH INTENT ESTIMATION FOR PHYSICAL HUMAN-ROBOT INTERACTION

S. Cremer, D. O. Popa, and F. L. Lewis, "Model-Free Online Neuroadaptive Control with Intent Estimation for Physical Human-Robot Interaction," *To Be Submitted: IEEE Transactions on Robotics (T-RO)*, 2017.

8.1 Abstract

With the rise of collaborative robots, the need for safe, reliable, and efficient physical Human-Robot Interaction (pHRI) has grown. High performance pHRI requires robust and stable controllers suitable for multiple degrees of freedom (DoF) and highly nonlinear robots. In this paper we describe an online, local cascade loop pHRI controller which relies on human force and pose measurements and can adapt to varying robot dynamics. It can also adapt to different user preferences and simplifies the interaction by making the robot behave according to a prescribed dynamic model. In our controller formulation, two neural networks in the "outer-loop" predicts human motion intent and estimates a reference trajectory for the robot that the "inner-loop" controller follows. The inner-loop imposes a prescribed error dynamics with the help of a model-free neuroadaptive controller, which uses a neural network to feedback linearize the robot dynamics. Lyapunov stability analysis gives weight tuning laws that guarantee that the error signals are bounded and the desired reference trajectory is achieved. Our control scheme was implemented on a PR2 robot and experimentally validated. Results confirmed that the motion jerk and human control effort was reduced similar to a tuned admittance controller, indicating that the two-loop controller achieves efficient and intuitive human-robot collaboration.

8.2 Introduction

An increasing number of collaborative robots are being introduced into human environments. Some have started relying on physical Human-Robot Interaction (pHRI) for assisting humans in cooperative tasks such as welding [9], parts assembly [11], and surgical procedures [10]. Physical interaction is also crucial during Learning from Demonstration (LfD), in which the user teaches the robot new tasks by manually moving its limbs through the motions [12]. On the other hand, a human can also be taught by the robot during pHRI, for example in skill improvement for surgical procedures [10] and in rehabilitation exercises [14]. Compared to non-collaborative robots, co-robots are inherently safer by being lightweight and compliant [15, 16], however, they may also be highly nonlinear and more difficult to model. In addition, torque joint sensors monitor and minimize interaction forces between the robot and people or the environment.

There are several established strategies for controlling the interaction behavior between humans, the environment, and a robot. Force, impedance, and admittance control methods are generally suitable for stable and compliant interactions [147]. Adaptive methods have traditionally been proposed to overcome variations in the robot dynamics [82]. More recently, researches have proposed combining the two methods into pHRI schemes, such as those in [129, 148, 149], which aim to adjust impedance parameters to achieve safe and stable contact with a human.

Human intent estimation has been utilized by several others to improve pHRI. In [150], a Hidden Markov Model was used to predict impedance parameter values in a hand-shaking task for realistic haptic human-robot interaction. Medina et al. [151] utilize impedance-based Gaussian Processes (GP) and a neuromechanical model of the human arm [152] to predict behavior during pHRI. Results demonstrate superior performance compared to a naive GP model, however the approach was not scalable due to computational complexity.

In [129], the human brain was modeled in the control loop as a simple PD controller and the neuromuscular dynamics were approximated with a first-order lag. The human model parameters were identified online and then used to tune a virtual internal model of the interface system, resulting in an improved manipulation performance. This work was inspired by the "crossover model," which states that a human adjusts their control characteristics to the dynamics of the controlled element such that the total system remains unchanged [129]. Thus, a highly nonlinear machine becomes easier to operate when behaving like a simple linear admittance model.

In this paper, a controller framework is presented for safe, intuitive, and efficient pHRI using a neuro-inspired two-loop structure in which both the robot dynamics and the human intent during interaction are being estimated online. Our work is based on neural network (NN) control methods that first appeared in the 1990's and can offer guaranteed tracking performance, stability, and robustness [**?**, **?**, 27, 154]. These algorithms provided the foundation for our recent work [44, 49, 153], in which a controller learns a combined human-robot model using a neuroadaptive "inner-loop" and adaptive inverse filter in an "outer-loop". Experiments showed that a neuroadaptive controller combined with a simple human intent estimator reduces motion jerk during interaction. In our previous work, the human intent model was approximated by a simple double integrator of acceleration push from the human operator, and could not capture the whole spectrum of complex physical interaction cues.

Therefore, in this paper, we add a more sophisticated human intent estimator (HIE) model into the outer-loop and a prescribed error dynamics (PED) component into the inner-loop. The HIE allows online adjustment of trajectory commands for the robot to follow during a nominal task based on input forces exerted on the robot by a human operator. The PED turns the robot into a linear admittance achieving the crossover model condition, and also reduces the number of controller parameters that need to be tuned. Compared to many other methods, our framework is easy to generalize and implement on robots with poorly known dynamics, and offers stability and performance guarantees via Lyapunov analysis. No offline training is required and only a couple parameters have to be adjusted when optimizing performance, i.e. the NN size and learning rates.

Our proposed method involves two control loops. In the inner control loop, the unknown robot dynamics is identified online and feedback linearized through dynamic compensation. A second outer-loop determines the human intent based on the control effort by the user. A combined stability proof rigorously verifies the performance of these two interacting loops. The contributions of this paper are as follows.

1. *Inner-loop*: A model-free online neuroadaptive controller is designed by a new sliding mode method that guarantees a prescribed error dynamics.
2. *Outer-loop*: A new sliding mode approach is given to estimate human intent and minimize the effort required by the user.
3. New techniques based on filtered errors and forces are given for estimating the human intent and further ensuring that the robot follows this intent with a prescribed error dynamics.
4. A combined Lyapunov stability analysis is provided for the two-loop controller framework, leading to the NN tuning laws of the inner and outer-loop.
5. Controller validation was implemented on a PR2 robot in point-to-point motion experiments, producing low position errors, jerk, and human effort. This indicates that our controller achieves efficient and intuitive pHRI.

The paper is organized as follows: Section 8.3 describes the outer-loop controller capable of estimating human intent. The inner-loop controller for prescribed error dynamics is described in Section 8.4. Stability analysis of the two-loop controller is performed in Section 8.5. The approach is validated in Section 8.6 by providing experiment results. Finally, Section 8.7 presents the conclusion and future work.

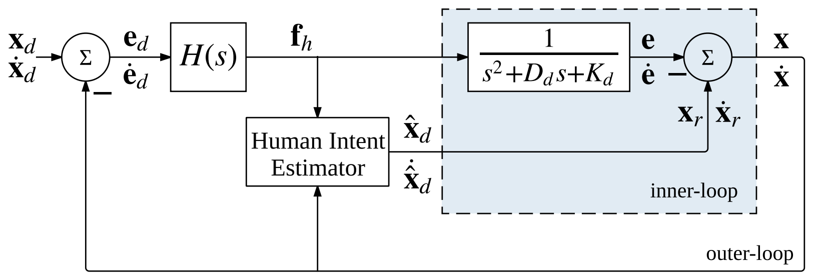


Figure 8.1. The neuroadaptive control system with human intent estimation for pHRI. The inner-loop simplifies the robot dynamics experience by the human controller $H(s)$ in the outer-loop.

8.3 Outer-loop HRI Control Structure

This paper develops a two-loop control structure for physical HRI shown in Fig. 8.1. In the outer-loop, the desired human motion trajectory x_d is estimated based on human force measurements f_h and robot end-effector position x. The human intent estimator (Fig. 8.2) provides $\hat{x}_d$ as an input for an inner-loop, designed to make the robot end-effector follow a desired reference trajectory $x_r = \hat{x}_d$. In addition, the inner-loop (shown in Fig. 8.3) imposes prescribed error dynamics, making the robot behave like a second-order linear system from the perspective of the human operator. In this section, we describe the human transfer function $H(s)$. Based on this impedance model, we develop a human intent estimator and derive the resulting outer-loop error dynamics.

8.3.1 The Human Transfer Function

The human transfer function can be described by a simple proportional-derivative (PD) controller with a first-order lag

$$H(s) = \frac{D_h s + K_h}{Ts + 1} e^{-Ls} \tag{8.1}$$

where $K_h, D_h \in \mathbb{R}^{6\times 6}$ are the gains of the controller in the cerebral cortex of the human brain, T is the time constant of the neuromuscular system and L is a delay factor [129]. The parameters will be different for different human operators, and here they are not explicitly needed for the controller formulation, but will be used in the controller stability proof. Both the proportional K_h and derivative D_h gain matrices are assumed to be diagonal, i.e.

$$K_h = \text{diag}\{k_1, k_2, \ldots, k_6\} \equiv \text{diag}\{\vec{k}\}$$
$$D_h = \text{diag}\{d_1, d_2, \ldots, d_6\} \equiv \text{diag}\{\vec{d}\}$$

where $\vec{k}, \vec{d}$ are vectors of coefficients with 3 degrees of freedom (DoF) in position and 3 DoF in rotation. Assuming the human attempts to reach a desired pose x_d and velocity $\dot{x}_d$, ignoring the time constant and delay, the human dynamics become

$$D_h \dot{e}_d + K_h e_d = f_h \tag{8.2}$$

where

$$e_d = x_d - x$$
$$\dot{e}_d = \dot{x}_d - \dot{x}$$

corresponding to the position, and velocity tracking errors in $\mathbb{R}^6$. Once the desired pose has been reached, the human will no longer exert a force on the robot. The human acts as an impedance controller, applying a force f_h based on current motion measurements x and $\dot{x}$. As $e_d, \dot{e}_d \to 0$ the human control effort $f_h \to 0$.

The following principles are used subsequently in our controller design:

Principle 1 (Effort minimization): pHRI can be improved if the robot assists the user by minimizing the applied force onto the robot f_h. This also adds a safety factor in that the robot moves so as to reduce human effort.

Principle 2 (Crossover model): A highly nonlinear machine becomes easier to operate when behaving like a simple admittance model. This makes the interaction more intuitive and requires less effort by the user.

Principle 1 is used in the outer-loop controller in this section. Principle 2 is used in the inner-loop controller in Section 8.4. To summarize, a well performing controller for pHRI is efficient and intuitive. Efficiency can be achieved by minimizing the operator effort with the help of a human intent estimator. Intuitiveness can be achieved by simplifying the robot dynamics with prescribed error dynamics that are linear and low order. Of course the controller should also be inherently safe, which requires guarantees of stability and robustness that we shall provide in Section 8.5.

8.3.2 Human Intent Estimator

The human transfer function (with its unknown PD gains) is part of an outer control loop shown in Fig 8.1. During the human-robot interaction, the human applies a force to move the robot end-effector according to some desired trajectory. This trajectory $x_d(t)$ is unknown to the robot. The human intent estimator approximates this reference trajectory, which then the inner-loop designed in Section 8.4 follows.

It is assumed that the human desired reference trajectory is defined by some nonlinear function, which can be approximated by a single layer neural network (NN)

$$\begin{aligned}\begin{bmatrix} x_d \\ \dot{x}_d \end{bmatrix} &= h(x, \dot{x}, f_h) \\ &= V^{\intercal}\phi(\xi) + \varepsilon_1 \end{aligned} \tag{8.3}$$

with an unknown weight matrix $V^{\intercal} \in \mathbb{R}^{12\times 18}$, activation functions $\phi(\cdot)$, input vector $\xi = [f_h^{\intercal}\ x^{\intercal}\ \dot{x}^{\intercal}]^{\intercal} \in \mathbb{R}^{18}$, and small residual error $\varepsilon_1 \in \mathbb{R}^{12}$. The estimated human intent can be expressed as

$$\begin{bmatrix} \hat{x}_d \\ \dot{\hat{x}}_d \end{bmatrix} = \hat{V}^{\intercal}\phi(\xi) \tag{8.4}$$

where $\hat{V}^{\intercal}$ is an estimated weight matrix. This results in the error estimates

$$\begin{bmatrix} \hat{e}_d \\ \dot{\hat{e}}_d \end{bmatrix} = \begin{bmatrix} \hat{x}_d - x \\ \dot{\hat{x}}_d - \dot{x} \end{bmatrix} \tag{8.5}$$

The difference between the actual and estimated error is defined as

$$\begin{aligned}\bar{\tilde{e}}_d = \begin{bmatrix} \tilde{e}_d \\ \dot{\tilde{e}}_d \end{bmatrix} &= \begin{bmatrix} e_d \\ \dot{e}_d \end{bmatrix} - \begin{bmatrix} \hat{e}_d \\ \dot{\hat{e}}_d \end{bmatrix} \\ &= \begin{bmatrix} x_d \\ \dot{x}_d \end{bmatrix} - \begin{bmatrix} \hat{x}_d \\ \dot{\hat{x}}_d \end{bmatrix} = \tilde{V}^{\intercal}\phi(\xi) + \varepsilon_1 \end{aligned} \tag{8.6}$$

where $\tilde{V} = V - \hat{V}$ is the weight deviation or weight estimation error.

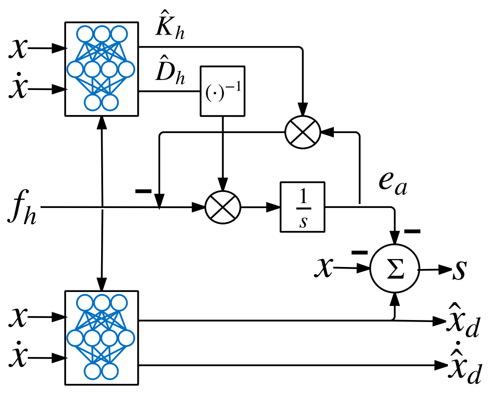


Figure 8.2. The human intent estimator producing trajectories $\hat{x}_d, \dot{\hat{x}}_d$ and human gains K_h, D_h using two neural networks (NN). The sliding mode error s and filtered error e_a are needed in the NN update equations.

Next, auxiliary parameters are introduced to modify the error dynamics such that the Lyapunov proof in Section 8.5 can be used to determine the NN weight update equation. Define the novel sliding mode error

$$s = \hat{e}_d - e_a \tag{8.7}$$

where the filtered error e_a is determined from the approximated human dynamics

$$f_h = \hat{D}_h \dot{e}_a + \hat{K}_h e_a \tag{8.8}$$

$$= J(\hat{P} \odot \bar{e}_a) \tag{8.9}$$

with operator $\odot$ being the element-wise Hadamard product and $J = [I_6 \ I_6]$, where $I_6 \in \mathbb{R}^{6\times 6}$ is the identity matrix. Vector $P \in \mathbb{R}^{12}$ contains the diagonal elements of K_h, D_h and is multiplied element-wise with $\bar{e}_a \in \mathbb{R}^{12}$, which are defined as

$$P = \begin{bmatrix} \vec{k} \\ \vec{d} \end{bmatrix} \tag{8.10}$$

$$\bar{e}_a = \begin{bmatrix} e_a \\ \dot{e}_a \end{bmatrix} \tag{8.11}$$

The structure of the human intent estimator is provided by (8.7) is in Fig. 8.2.

Theorem 2 *If the sliding mode error $s \to 0$ in (8.7), then the human intent estimate approaches the actual value $\hat{x}_d \to x_d$.*

Proof: From (8.7),

$$\begin{aligned} s &= (\hat{x}_d - x) - e_a \\ &= (\hat{x}_d - x_d + x_d - x) - e_a \\ &= -\tilde{e}_d + e_d - e_a \end{aligned} \tag{8.12}$$

As the human reaches the desired goal position, $f_h \to 0$ and $e_d \to 0$ since (8.2) is stable with f_h viewed as the input. Similarly, $e_a \to 0$ from (8.8). Hence, if $s \to 0$, then $\tilde{e}_d \to 0$. This implies that $\tilde{e}_d \to 0$ and therefore $\hat{x}_d \to x_d$. ■

8.3.3 Outer-loop Error Dynamics

Here, we determine the error dynamics of the outer-loop. The following development is required in the proof of our main performance result in Theorem 4, Section 8.5.

The human model in (8.2) can be modified to write

$$D_h(\dot{\hat{e}}_d + \dot{\tilde{e}}_d) + K_h(\hat{e}_d + \tilde{e}_d) = (D_h - \tilde{D}_h)\dot{e}_a + (K_h - \tilde{K}_h)e_a \tag{8.13}$$

$$D_h(\dot{\hat{e}}_d - \dot{e}_a) = -K_h(\hat{e}_d - e_a) - K_h\tilde{e}_d - D_h\dot{\tilde{e}}_d - \tilde{D}_h\dot{e}_a - \tilde{K}_h e_a \tag{8.14}$$

Using the definitions from (8.6), (8.7), (8.10), and (8.11)

$$D_h\dot{s} = -K_h s - J\,(P \odot \bar{\tilde{e}}_d) - J(\tilde{P} \odot \bar{e}_a) \tag{8.15}$$

The human gain matrix can be estimated with a second NN

$$P = U^{\intercal}\phi(\xi) + \varepsilon_2 \tag{8.16}$$

$$\hat{P} = \hat{U}^{\intercal}\phi(\xi) \tag{8.17}$$

$$\tilde{P} = \tilde{U}^{\intercal}\phi(\xi) + \varepsilon_2 \tag{8.18}$$

where $U^{\intercal} \in \mathbb{R}^{12\times 18}$ and $\varepsilon_2 \in \mathbb{R}^{12}$. The weight deviation or estimation error is $\tilde{U} = U - \hat{U}$. This yields the outer-loop error dynamics

$$\begin{aligned} D_h\dot{s} = &- K_h s - J\left((\hat{P} + \tilde{U}^{\intercal}\phi(\xi) + \varepsilon_2) \odot (\tilde{V}^{\intercal}\phi(\xi) + \varepsilon_1)\right) \\ &- J\left((\tilde{U}^{\intercal}\phi(\xi) + \varepsilon_2) \odot \bar{e}_a\right) \end{aligned} \tag{8.19}$$

This is needed in the proof of Theorem 4 in Section 8.5.

8.4 Inner-loop Robot Controller with Prescribed Error Dynamics

In this section we develop an inner-loop neuroadaptive robot controller for tracking the reference trajectory $x_r = \hat{x}_d$. A neural network is used to feedback linearize

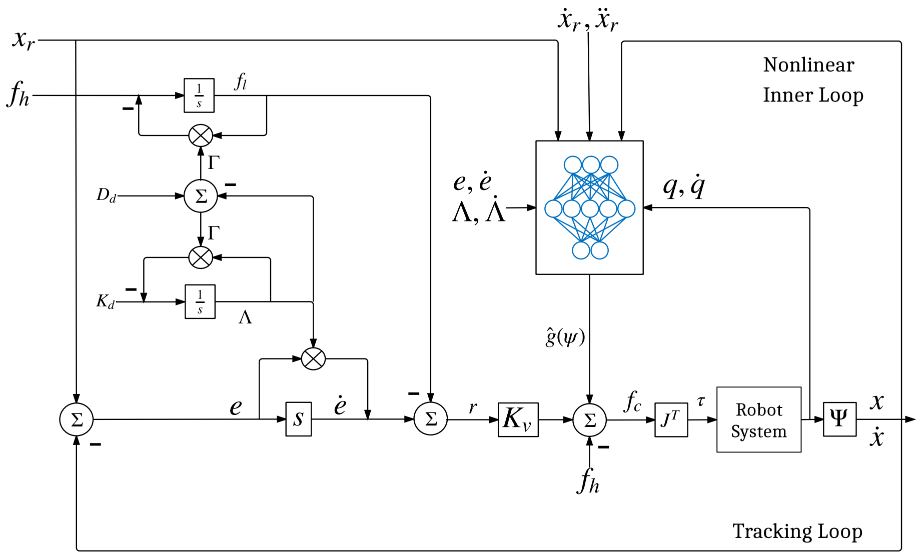


Figure 8.3. The inner-loop control system with prescribed error dynamics (PED). The reference trajectory x_r is provided by the outer-loop.

the controller by estimating the nonlinear robot dynamics. In addition, prescribed error dynamics are imposed by utilizing time-dependent auxiliary parameters. The inner-loop controller is shown in Fig. 8.3.

The robot dynamics equation in Cartesian or task space is given by

$$M(q)\ddot{x} + V(q,\dot{q})\dot{x} + F(\dot{q}) + G(q) = f_c + f_h \tag{8.20}$$

where $q, \dot{q} \in \mathbb{R}^n$ are the joint positions and velocities, n is the degrees of freedom (DoF) of the robot, and $x \in \mathbb{R}^6$ is the Cartesian pose of the end-effector. The operational-space $M(q)$ is the inertia matrix, $V(q,\dot{q})$ contains the Coriolis/centrifugal forces, $G(q)$ is the gravity vector, and $F(\dot{q})$ is the friction term. During pHRI, the operator applies a Cartesian force $f_h \in \mathbb{R}^6$. The neuroadaptive scheme now designed estimates the control force $f_c \in \mathbb{R}^6$, which is converted into joint control torques by multiplying with the robot geometric Jacobian: $\tau = J^\intercal f_c$.

Property 1: Matrix $M(q)$ is positive definite and symmetric. Matrix $V(q, \dot{q})$ is in the Christoffel form, implying that $\dot{M}(q) - 2V(q, \dot{q})$ is skew symmetric.

Assumption 1: During pHRI, the hand of the human is assumed to be in contact with the robot end-effector, i.e. they share the same Cartesian position such that the robot position and human position x in (8.2) are the same. The force measured by the end-effector is equal and opposite to the force applied by the human.

8.4.1 Sliding Mode Formulation for Prescribed Robot Error Dynamics

According to Principle 2, the robot should perform like a simple admittance model. Therefore, suppose the desired error dynamics is

$$\ddot{e} + D_d \dot{e} + K_d e = f_h \tag{8.21}$$

where D_d, K_d can be fixed or adjusted according to some performance metric. Given the tracking error

$$e = x_r - x \tag{8.22}$$

define a novel sliding mode error

$$r = \dot{e} + \Lambda e - f_l \tag{8.23}$$

$$\dot{r} = \ddot{e} + \Lambda \dot{e} + \dot{\Lambda} e - \dot{f}_l \tag{8.24}$$

where matrix $\Lambda(t)$ is a function of time. The filtered force f_l is determined from

$$f_h = \dot{f}_l + \Gamma f_l \tag{8.25}$$

The following theorem is a formalization of the method in [130].

Theorem 3 *Given the filtered force (8.25) and the sliding mode error (8.23), define*

$$D_d = \Lambda + \Gamma \tag{8.26}$$

$$K_d = \dot{\Lambda} + \Gamma\Lambda \tag{8.27}$$

If $r \to 0$, then the system behaves according to the error dynamics in (8.21).

Proof: Define an error signal

$$w = \ddot{e} + D_d\dot{e} + K_d e - f_h \tag{8.28}$$

Substituting the definitions into the error signal gives

$$\begin{aligned} w &= \ddot{e} + (\Lambda + \Gamma)\dot{e} + (\dot{\Lambda} + \Gamma\Lambda)e - (\dot{f}_l + \Gamma f_l) \\ &= \ddot{e} + \Lambda\dot{e} + \dot{\Lambda}e - \dot{f}_l + \Gamma(\dot{e} + \Lambda e - f_l) \\ &= \dot{r} + \Gamma r \end{aligned} \tag{8.29}$$

So if $r \to 0$ then $w \to 0$ and the system behaves according to (8.21). ■

8.4.2 Inner-loop Error Dynamics

Here, we determine the inner-loop error dynamics by combining the robot dynamics with the sliding mode formulation. The following development is required in the proof of our main performance result in Theorem 4, Section 8.5. Rewriting (8.23) as

$$\dot{e} = r - \Lambda e + f_l \tag{8.30}$$

and differentiating (8.22) yields

$$x = x_r - e \tag{8.31}$$

$$\dot{x} = \dot{x}_r - r + \Lambda e - f_l \tag{8.32}$$

$$\ddot{x} = \ddot{x}_r - \dot{r} + \Lambda \dot{e} + \dot{\Lambda} e - \dot{f}_l \tag{8.33}$$

Substituting the above into (8.20) produces the robot error dynamics

$$M(q)\dot{r} = -V(q,\dot{q})r + g(\psi) - f_c - f_h \tag{8.34}$$

and the nonlinear robot dynamics are

$$\begin{aligned} g(\psi) = {} & M(q)(\ddot{x}_r + \Lambda \dot{e} + \dot{\Lambda} e - \dot{f}_l) \\ & + V(q,\dot{q})(\dot{x}_r + \Lambda e - f_l) \\ & + F(\dot{q}) + G(q) + f_d \end{aligned} \tag{8.35}$$

and $\psi = \begin{bmatrix} f_l \ \dot{f}_l \ \text{diag}\{\Lambda\} \ \text{diag}\{\dot{\Lambda}\} \ q^\intercal \ \dot{q}^\intercal \ e^\intercal \ \dot{e}^\intercal \ \dot{x}_r^\intercal \ \ddot{x}_r^\intercal \end{bmatrix}^\intercal$.
In (8.34), define the control input as

$$f_c = \hat{g}(\psi) + K_v r - f_h \tag{8.36}$$

where $K_v = K_v^\intercal > 0$. The nonlinear terms are estimated with a single layer neural network (NN) function approximation

$$\hat{g}(\psi) = \hat{W}^\intercal \sigma(\psi) + \varepsilon \tag{8.37}$$

as in [27]. Substituting (8.36) into (8.34) produces

$$\begin{aligned} M(q)\dot{r} &= -V(q,\dot{q})r + g(\psi) - f_h - \hat{g}(\psi) - K_v r + f_h \\ &= -(V(q,\dot{q}) + K_v)r + \tilde{g}(\psi) \end{aligned} \tag{8.38}$$

where $\tilde{g}(\psi) = g(\psi) - \hat{g}(\psi)$ is the function approximation error. The inner-loop error dynamics are therefore

$$M(q)\dot{r} = -\left(V(q,\dot{q}) + K_v\right)r + \tilde{W}^{\mathsf{T}}\sigma(\psi) + \varepsilon \tag{8.39}$$

This equation is needed in the proof of Theorem 4 in following section.

8.5 Stability Analysis of Human-Robot System

In this section, the combined stability of the two-loop system is investigated. Weight tuning laws for the NNs (8.4), (8.17), and (8.37) are derived from Lyapunov stability analysis that guarantee overall stability and proper performance of the inner and outer-loops. The following standard assumptions are made:

Assumption 2: The reference trajectory is bounded by a scalar, e.g. $\|\bar{x}_r\| \leq x_B \in \mathbb{R}$.

Assumption 3: The ideal NN weights are bounded by a scalar, e.g. $\|U\|_F \leq U_B$ where $||\cdot||_F$ is the Frobenius norm.

Next, it is shown that, using these tuning algorithms, the sliding mode errors (8.7) and (8.23) are uniformly ultimately bounded (UUB). A signal $x(t)$ is UUB if there exists a compact set $S \in \mathbb{R}^n$ such that for all $x(t_0) = x_0 \in S$ there exists an $\epsilon > 0$ and a number $T(\epsilon, x_0)$ such that $\|x(t)\| < \epsilon$ for all $t \geq t_0 + T$ [27,154].

Theorem 4 *Given the human dynamics (8.2) and robot dynamics (8.20), define a control law (8.36) with a feedback linearization term estimated by the NN in (8.37). Furthermore, let the desired trajectory $x_r = \hat{x}_d$ be generated by NN (8.4) and let the gains of the human controller be estimated by NN (8.17). Let the NN weight tuning laws be defined as*

$$\dot{\hat{W}} = F\sigma(\psi)r^{\intercal} \tag{8.40}$$

$$\dot{\hat{V}} = -G\phi(\xi)s^{\intercal}J\,diag\{\hat{P}\} + \kappa\,\|s\|\,G\hat{V} \tag{8.41}$$

$$\dot{\hat{U}} = -H\phi(\xi)s^{\intercal}J\,diag\{\bar{e}_a\} + \kappa\,\|s\|\,H\hat{U} \tag{8.42}$$

with the filtered errors (8.8) and (8.25), constant matrices $F = F^{\intercal} > 0$, $G = G^{\intercal} > 0$, $H = H^{\intercal} > 0$, and scalar design parameter $\kappa > 0$. Then the sliding mode errors in (8.7), (8.23) are UUB and the overall human-robot system is stable.

Proof: Define the Lyapunov function

$$\begin{aligned} L =& \tfrac{1}{2}r^{\intercal}Mr + \tfrac{1}{2}\mathrm{tr}\{\tilde{W}^{\intercal}F^{-1}\tilde{W}\} \\ &+ \tfrac{1}{2}s^{\intercal}D_h s + \tfrac{1}{2}\mathrm{tr}\{\tilde{V}^{\intercal}G^{-1}\tilde{V}\} + \tfrac{1}{2}\mathrm{tr}\{\tilde{U}^{\intercal}H^{-1}\tilde{U}\} \end{aligned} \tag{8.43}$$

with the derivative

$$\dot{L} = r^{\intercal}M\dot{r} + \tfrac{1}{2}r^{\intercal}\dot{M}r + \mathrm{tr}\{\tilde{W}^{\intercal}F^{-1}\dot{\tilde{W}}\} \tag{8.44a}$$

$$+ s^{\intercal}D_h\dot{s} + \mathrm{tr}\{\tilde{V}^{\intercal}G^{-1}\dot{\tilde{V}}\} + \mathrm{tr}\{\tilde{U}^{\intercal}H^{-1}\dot{\tilde{U}}\} \tag{8.44b}$$

which is an equation of the form $\dot{L} = \dot{L}_1 + \dot{L}_2$. Substituting (8.39) with $\varepsilon = 0$ in (8.44a) produces

$$\begin{aligned}\dot{L}_1 =& - r^\intercal K_v r + \tfrac{1}{2} r^\intercal (\dot{M} - 2V) r \\ &+ \operatorname{tr}\{\tilde{W}^\intercal (F^{-1}\dot{\tilde{W}} + \sigma(\psi) r^\intercal)\}\end{aligned}$$

as detailed in [27]. Since

$$\dot{\tilde{W}} = \dot{W} - \dot{\hat{W}} = -\dot{\hat{W}}$$

we can subsitute the NN update law from (8.40) and together with Property 1,

$$\begin{aligned}\dot{L}_1 =& - r^\intercal K_v r \\ \leq& - K_{v_{\min}} \|r\|^2\end{aligned} \tag{8.45}$$

and $K_{v_{\min}}$ is the minimum singular value of K_v. The second part of the Lyapunov derivative (8.44b) can be expanded with (8.19), assuming $\varepsilon_1, \varepsilon_2 = 0$, according to

$$\begin{aligned}
\dot{L}_2 = &- s^\intercal K_h s - s^\intercal J(\hat{P} \odot (\tilde{V}^\intercal \phi(\xi))) \\
&- s^\intercal J((\tilde{U}^\intercal \phi(\xi)) \odot \bar{e}_a) - s^\intercal J((\tilde{U}^\intercal \phi(\xi)) \odot (\tilde{V}^\intercal \phi(\xi))) \\
&+ \mathrm{tr}\{\tilde{V}^\intercal G^{-1} \dot{\tilde{V}}\} + \mathrm{tr}\{\tilde{U}^\intercal H^{-1} \dot{\tilde{U}}\} \\
= &- s^\intercal K_h s - \mathrm{tr}\{s^\intercal J((\tilde{U}^\intercal \phi(\xi)) \odot (\tilde{V}^\intercal \phi(\xi)))\} \\
&- \mathrm{tr}\{\tilde{V}^\intercal \phi(\xi) s^\intercal J \mathrm{diag}\{\hat{P}\}\} - \mathrm{tr}\{\tilde{U}^\intercal \phi(\xi) s^\intercal J \mathrm{diag}\{\bar{e}_a\}\} \\
&+ \mathrm{tr}\{\tilde{V}^\intercal G^{-1} \dot{\tilde{V}}\} + \mathrm{tr}\{\tilde{U}^\intercal H^{-1} \dot{\tilde{U}}\} \\
= &- s^\intercal K_h s - \mathrm{tr}\{s^\intercal J((\tilde{U}^\intercal \phi(\xi)) \odot (\tilde{V}^\intercal \phi(\xi)))\} \\
&+ \mathrm{tr}\{\tilde{V}^\intercal (G^{-1} \dot{\tilde{V}} - \phi(\xi) s^\intercal J \mathrm{diag}\{\hat{P}\})\} \\
&+ \mathrm{tr}\{\tilde{U}^\intercal (H^{-1} \dot{\tilde{U}} - \phi(\xi) s^\intercal J \mathrm{diag}\{\bar{e}_a\})\}
\end{aligned} \tag{8.46}$$

Using the fact that $\dot{\tilde{V}} = -\dot{\hat{V}}$ and $\dot{\tilde{U}} = -\dot{\hat{U}}$ and the NN update equations in (8.41), (8.42)

$$\begin{aligned}
\dot{L}_2 = &- s^\intercal K_h s - \mathrm{tr}\{s^\intercal J((\tilde{U}^\intercal \phi(\xi)) \odot (\tilde{V}^\intercal \phi(\xi)))\} \\
&- \kappa \|s\| \mathrm{tr}\{\tilde{V}^\intercal \hat{V}\} - \kappa \|s\| \mathrm{tr}\{\tilde{U}^\intercal \hat{U}\}
\end{aligned} \tag{8.47}$$

Since

$$\begin{aligned}
\mathrm{tr}\{\tilde{V}^\intercal \hat{V}\} = \mathrm{tr}\{\tilde{V}^\intercal (V - \tilde{V})\} &= < \tilde{V}, V > - ||\tilde{V}||_F^2 \\
&\leq ||\tilde{V}||_F \, \|V\|_F - ||\tilde{V}||_F^2
\end{aligned} \tag{8.48}$$

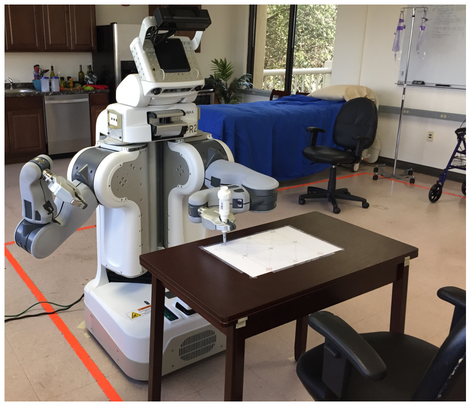


Figure 8.4. Setup for experimental validation testing.

where $||\cdot||_F$ is the Frobenius norm, we can write

$$\begin{aligned}
\dot{L}_2 \leq & - K_{h_{\min}} \|s\|^2 - \|s\| \, \phi_B^2 ||\tilde{V}^\intercal||_F ||\tilde{U}^\intercal||_F \\
& - \kappa \|s\| \cdot ||\tilde{V}||_F (V_B - ||\tilde{V}||_F) \\
& - \kappa \|s\| \cdot ||\tilde{U}||_F (U_B - ||\tilde{U}||_F) \\
= & - \|s\| \, K_{h_{\min}} \|s\| - \|s\| \, Z^\intercal A Z - \|s\| \, Z^\intercal B \\
= & - \|s\| \left\{ K_{h_{\min}} \|s\| + Z^\intercal A Z + Z^\intercal B \right\}
\end{aligned} \tag{8.49}$$

where

$$Z = \begin{bmatrix} ||\tilde{V}^\intercal||_F \\ ||\tilde{U}^\intercal||_F \end{bmatrix}, \quad A = \begin{bmatrix} -\kappa & \frac{1}{2}\phi_B^2 \\ \frac{1}{2}\phi_B^2 & -\kappa \end{bmatrix}, \quad B = \kappa \begin{bmatrix} V_B \\ U_B \end{bmatrix}$$

and $K_{h_{\min}}$ is the minimum singular value of K_h and the bounds ϕ_B, V_B, U_B. Completing the square

$$\begin{aligned} & Z^\intercal A Z + Z^\intercal B \\ & = (Z + \tfrac{1}{2}A^{-1}B)^\intercal A (Z + \tfrac{1}{2}A^{-1}B) - \tfrac{1}{4}B^\intercal A^{-1} B \end{aligned} \tag{8.50}$$

Thus the term in the braces in (8.49) is guaranteed positive if and only if either

$$\|s\| > \frac{\frac{1}{4}B^\intercal A^{-1}B}{K_{h_{\min}}} \equiv b_s \tag{8.51}$$

or

$$Z > -\tfrac{1}{2}A^{-1}B + \sqrt{\tfrac{1}{4}B^\intercal A^{-1}B} \equiv b_Z \tag{8.52}$$

Combining the results from (8.45) and (8.49), $\dot{L}$ is negative outside a compact set for the NN weight update laws (8.40), (8.41), (8.42). Since $L > 0$ and $\dot{L} < 0$, the sliding mode errors s, r and the NN weight errors $\tilde{V}, \tilde{U}, \tilde{W}$ are bounded. ■

Note the UUB bound can be decreased by decreasing the learning rate κ in (8.51), (8.53), and (8.54) which can be shown by expanding the terms

$$\tfrac{1}{4}B^\intercal A^{-1}B = \frac{\kappa^2(\kappa U_B^2 + U_B V_B \phi_B^2 + \kappa V_B^2)}{\phi_B^4 - 4\kappa^2} \tag{8.53}$$

$$\tfrac{1}{2}A^{-1}B = \frac{\kappa}{\phi_B^2 - 4\kappa^2}\begin{bmatrix} U_B\phi_B^2 + 2V_B\kappa \\ V_B\phi_B^2 + 2U_B\kappa \end{bmatrix} \tag{8.54}$$

Since r, s are UUB, Theorem 2 show that the human intent is closely estimated, while Theorem 3 shows that the desired error dynamics (8.21) are closely followed.

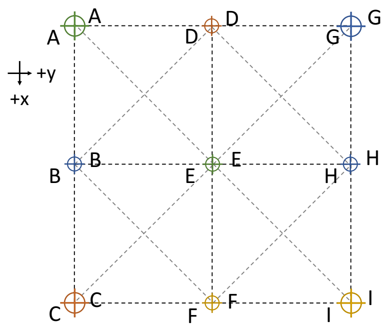


Figure 8.5. Grid layout for visual guidance during trajectory following.

8.6 Experimental Validation

The two-loop controller was experimentally validated on the Personal Robot 2 (PR2) mobile manipulator shown in Fig. 8.4. It has two 7 degrees of freedom (DoF) arms with two parallel grippers and Force/Torque (F/T) sensors in its wrists. A hard real-time control loop is updated at 1 kHz. The robot runs the Robot Operating System (ROS) [102] together with a real-time controller manager framework which can load and unload controllers as plugins. There are ROS packages or libraries with infrastructure to send control torques and access encoder data, as well as kinematic models for computing Jacobians and forward/inverse kinematics. The neural network controller and estimator were implemented with *Eigen*, a C++ library for linear algebra [155], and subsequently incorporated in a new publicly available repository SkinLearn[1]. Due to its computational complexity, the neuroadaptive controller and estimator are updated every 3$^{\text{rd}}$ loop in a parallel thread, resulting in a torque update rate of 333 Hz. This sampling rate is sufficient for pHRI according to known requirements [156].

[1]`https://bitbucket.org/nextgensystems/skinlearn`

8.6.1 Setup

The PR2 robot was positioned in front of a table with the left arm holding a white cylinder, which was able to rotate freely around its vertical axis (Fig. 8.4). Below the gripper, a narrow extension indicated the gripper location on a piece of paper shown in Fig. 8.5. The paper depicted a 3×3 grid with points labeled from A through I, separated at a distance of 12.5 cm. A human operator was asked to sit at the table and grab the cylinder above the gripper. During the experiment, the robot verbally instructed the user to move to certain points. One trial involved moving the gripper in a square and diamond pattern, both counter-clockwise and clockwise. The following 19 points were traversed:

$$A, C, I, G, A, G, I, C, A, D, B, F, H, D, H, F, B, D, A$$

Each point took approximately 5.0 seconds to reach, for a total of 120 seconds per trial including idle time. Data was automatically recorded at 50 Hz, including encoder, sensor, controller, and timing data. The following controller configurations were utilized to evaluate different components of the proposed control scheme,

- Prescribed Error Dynamics
  - (a) Disabled
  - (b) Enabled
- Reference trajectory x_r set to
  - (i) x_d (pose desired by human)
  - (ii) $\hat{x}_d$ (estimated human intent from outer-loop)
  - (iii) x_m (fixed admittance model from [44])

giving a total of 6 permutations. To disable the PED, the inner-loop gain matrix Λ is fixed and the filtered force f_l is set to zero in (8.23). As a result, there is no need to update equations (8.25), (8.26), (8.27) and the parameters Γ, K_d, D_d are irrelevant. This allowed investigating the effect of having PED in the neuroadaptive inner-loop.

For (i), it is assumed that the robot knows the ideal reference trajectory, which is communicated to the human subject with audiovisual and tactile cues. Both the robot and human agree on the reference trajectory, which for example generally is the case during rehabilitation exercises and assisted manufacturing tasks. In the second setup (ii), only the human knows the reference trajectory while the robots tries to assist by estimating the human intent using 8.4 from interaction force, position, and velocity measurements.

For case (iii), the reference trajectory is provided by a fixed autoregressive moving-average (ARMA) model described in [44]. The HIE is replaced by the admittance $M(z)$ of order m_θ in the numerator and n_θ in the denominator. It is implemented as

$$x_m[k] = h^\intercal[k]\theta \tag{8.55}$$

with the measured regression vector

$$\begin{aligned} h^\intercal[k] = [&-x_m[k-1], \ \dots, -x_m[k-n_\theta], \\ &f_h[k], f_h[k-1], \ \dots, f_h[k-m_\theta]] \end{aligned} \tag{8.56}$$

sampled at timestep $t = kT_s$. The ideal ARMA parameter matrix

$$\theta = [a_1, \ \dots, a_{n_\theta}, b_0, b_1, \ \dots, b_{m_\theta}] \tag{8.57}$$

was estimated using recursive least squares (RLS). Compared with the proposed outer-loop HIE, the ARMA model requires a training phase during which the desired human trajectory x_d is being provided.

In the following two sections, the controller settings and implementation details are described.

8.6.1.1 Inner-loop

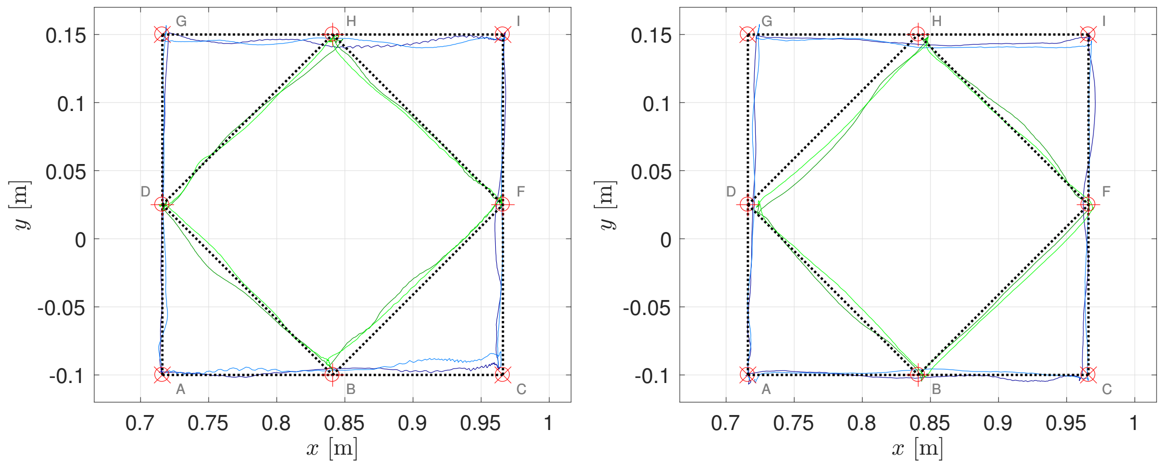
The performance tuned inner-loop neuroadaptive controller parameters are $\Lambda = \text{diag}\{2, 2, 2, 1.2, 1.2, 1.2\}$ and $K_v = \text{diag}\{5, 5, 5, 3, 3, 3\}$, for $K_z = 0.01$, $Z_B = 100$, $F = 10I_{63}$, $G = 10I_{19}$, and $\kappa = 0.1$, where I_n is the $n \times n$ identity matrix. A two-layer NN with 63 inputs (including the bias input), 18 hidden layer neurons, and 6 outputs was implemented. The RBF activation function for the hidden layer neuron i was defined as

$$\sigma_i(\psi) = e^{-\beta(\psi-\mu_i)^\intercal(\psi-\mu_i)} \tag{8.58}$$

with $\beta_i = 1$ and vector μ_i initialized randomly from the discrete set $\{-1, 1\}$. The weights $\hat{W}$ of the neural network were initialized from the uniform random distribution $[-0.1, 0.1]$. The prescribed task model gain matrices were set to $K_d = 20I_6$, $D_d = 10I_6$ and Γ was initialized to $10I_6$.

8.6.1.2 Outer-loop

Since the desired motion was constrained to a planar surface, only the x and y positions were estimated. The outer-loop human intent estimator was initialized with $G = 1.0I_2$, $H = 0.1I_2$, and $\kappa = 0.01$. Similar to the inner-loop controller, it used RBF activation functions and the NN weights $\hat{V}, \hat{U}$ were initialized with a uniform random distribution between $[-0.1, 0.1]$. Since random weights could produce



(a) PED disabled, x_r provided (b) PED enabled, HIE ($k_f = 0.4$)

Figure 8.6. Cartesian trajectory tracking using two different controller setups. One trial consisted of moving the gripper around the square and diamond shape, both clockwise and counter-clockwise.

negative definite matrices for K_h and D_h, a lower limit of 0.01 was imposed on the elements of output vector $\hat{P}$. The tuned ARMA parameter weight matrix θ in (8.57) for the x and y-dimensions was

$$\begin{bmatrix} -0.7639 & -0.0554 & 0.0017 & -0.1824 & -0.0698 & 0.2235 & -0.2590 & 0.1051 \\ -0.8250 & -0.0920 & -0.0065 & -0.0776 & -0.0001 & 0.0091 & -0.0233 & 0.0141 \end{bmatrix}$$

after conducting two training trials, during which the human intent x_d was provided.

8.6.2 Results

The control structure for physical HRI is designed to reduce the human effort and simplify the robot's dynamic behavior as discussed in Section 8.3.1. During the interaction, the inner and outer-loop NN weights are continuously being updated to more accurately follow the robot reference trajectory and to better predict the human intent trajectory, respectively. This is confirmed in Fig. 8.7, depicting the NN weight norm versus time for the entire duration of one trial. At the beginning, the weights were initialized randomly and the first movement occurred close to the 10 s mark. This resulted in a rapid change of the NN weights, especially in the outer-loop human intent estimator. During the first half of the trial, several "spikes" appeared at the start of the point-to-point motions, indicating this was where the majority of the NN tuning took place. As the trial continued, the rate of change of the outer-loop weights decreased implying that the NN approximation was relatively accurate and needed less tuning. The inner-loop weight norm shows no discernible patter, hence the NN weights have to be constantly adjusted for new joint angles and Cartesian trajectories in order to learn the nonlinear dynamics utilized for feedback linearization. Since several systems learned at the same time, i.e. the inner-loop neuroadaptive controller, the outer-loop human intent estimator, and the operator, it is important to consider several aspects of the interaction, such as the resulting position error, smoothness of motion, and interaction forces.

Fig. 8.8 and 8.9 show the position and force value along the x and y-axis during one trial, during which the operator first followed two square and then two diamond patterns defined by the 19 points listed in Section 8.6.1. For this trial, the inner-loop prescribed error dynamics (PED) were enabled and the matrices Λ, Γ converged to $2.8I_6$ and $7.2I_6$ respectively within 1 s. For all three trajectory cases discussed in Section 8.6.1, the end-effector and human hand position x was able to reach the desired points within the 1 cm range. The estimated $\hat{x}_d$ approximately followed x_d, indicating that the neural network was able to approximate the intent trajectory within some error bound ε_1 as defined in (8.4).

To quantify the tracking performance, the position error at each time step kT was computed as the mean error distance over the entire trial,

$$\overline{\|e\|}_2 = \frac{1}{N} \sum_{k=1}^{N} \|x_{\text{des}}(kT) - x(kT)\|_2 \tag{8.59}$$

where N is the number of data points and $T = 0.02$ seconds (the data was recorded at 50 Hz). The error bar plot in Fig. 8.10a shows that the mean position error was less than 9 mm. The high accuracy was expected, since the pattern below the gripper provided the operator with instant visual feedback. The largest mean position error occurred when $x_r = \hat{x}_d$ and the prescribed error dynamics were disabled. When the PED are turned off and the robot no longer behaves like a simple admittance model, the operator has to learn the machine dynamics and tune their own neuromuscular controller. The HIE introduces further uncertainty in form of a neural network estimation error, which made the robot harder to control. With the PED enabled, the NN estimator performed similar to when the actual desired reference trajectory was provided to the robot, indicating good performance. Further user studies with more subjects are needed to determine the precise accuracy range of the different control setups.

The smoothness of the physical interaction was measured in form of the dimensionless squared jerk

$$J_\alpha = \int_{t_s}^{t_f} \dddot{p}(t)^2 dt \; \frac{(t_f - t_s)^5}{A^2} \tag{8.60}$$

where parameter A is the total length of path $p(t)$ taken by the robot gripper and human hand from start time t_s to final time t_f. Experimental evidence suggests that human motion involves the minimization of jerk and is therefore an an effective measurement of the quality of pHRI [146]. Fig. 8.10b shows that the dimensionless squared jerk was decreased with the PED enabled, indicating better HRI performance. Compared with the ARMA reference trajectory x_m, the NA controller benefited more from enabling PED. This can be explained by the nature of the ARMA model − the moving average acts as a lowpass filter on the robot motion. The smallest jerk was produced when $x_r = x_d$, since the robot for the most part is guiding the user who only performs minor corrections. Also, the PED impose a simple admittance model on the robot dynamics making it easy for the human to learn and follow along. If the robot receives the wrong desired human intent, the controller will fight against the operator's movements.

Fig. 8.10c shows the mean 2-norm of the robot control force f_c and human force f_h, i.e.

$$\overline{\|f\|}_2 = \frac{1}{N} \sum_{k=1}^{N} \|f(kT)\|_2 \tag{8.61}$$

With PED disabled, the user had to exert a larger force to follow the desired trajectory. With PED enabled, less effort was required by the user implying that the robot was assisting with the motion. As expected, the robot controller force had to compensate and increased in magnitude, however the amount of increase was nonlinear. The ARMA trajectory performed better than the NN estimation, since it is taking previous measurements into account, which are stored in the parameter vector (8.56). An estimator with a larger neural network (i.e. more hidden layers) could possibly increase the accuracy of the HIE, however the NN update would require more computations and take longer too complete. One of the challenges was to run the framework at a high update rate (e.g. faster than 100 Hz), which is essential during pHRI. The PR2 real-time loop was pushed to its limits with the implemented two-loop structure and three NNs, however it is still very capable considering it was introduced in 2009.

Table 8.1. Performance measurements for different reference trajectories with PED disabled and enabled.

PED	Meas.	Units	(i) x_d		(ii) $\hat{x}_d$		(iii) x_m	
			mean	*std.*	*mean*	*std.*	*mean*	*std.*
(a) Disabled	$\|\|e\|\|_2$	mm	3.979	0.433	7.479	0.738	5.750	0.156
	J_α	10^7	6.586	1.132	8.168	1.175	6.035	1.042
	$\|\|f_h\|\|_2$	N	3.854	0.140	7.461	1.011	3.637	0.226
	$\|\|f_c\|\|_2$	N	3.510	0.087	7.527	0.661	5.371	0.148
(b) Enabled	$\|\|e\|\|_2$	mm	4.580	0.243	3.566	0.704	5.121	0.399
	J_α	10^7	4.128	1.456	5.066	0.917	4.512	0.955
	$\|\|f_h\|\|_2$	N	1.833	0.056	5.425	0.058	2.276	0.072
	$\|\|f_c\|\|_2$	N	6.122	0.182	8.268	0.059	6.372	0.360

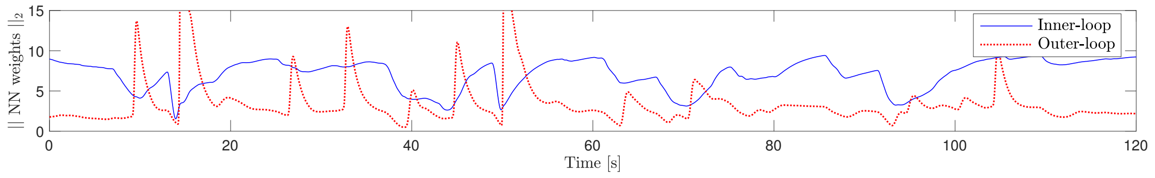


Figure 8.7. NN weight norms during one trial with reference trajectory $x_r = \hat{x}_d$ being provided by the outer-loop human intent estimator (PED is enabled).

Overall, the experiments show that the inner/outer-loop control scheme with human intent estimation has comparable tracking performance when compared to the ARMA admittance predictor. Our approach has the advantage of being online and not require any training. Imposing a simple admittance model by utilizing PED in the inner-loop, simplifies the interaction according to the crossover model and and results in smooth pHRI. In addition, the PED allows tailoring the interaction to individual pHRI preferences by adjusting the two gains K_d and D_d. The performance results plotted in Fig. 8.10 are summarized in Table 8.1.

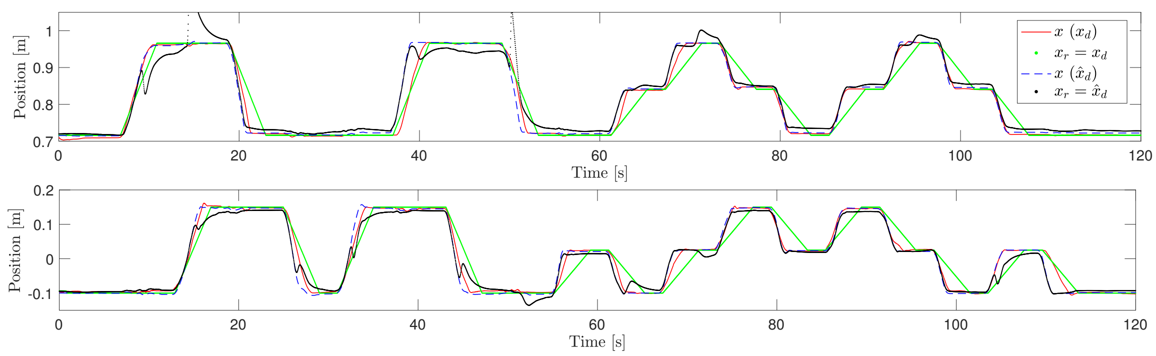


Figure 8.8. Trajectories in x and y-directions during one trial using the desired reference trajectory (x_d) and human intent estimator $(\hat{x}_d)$.

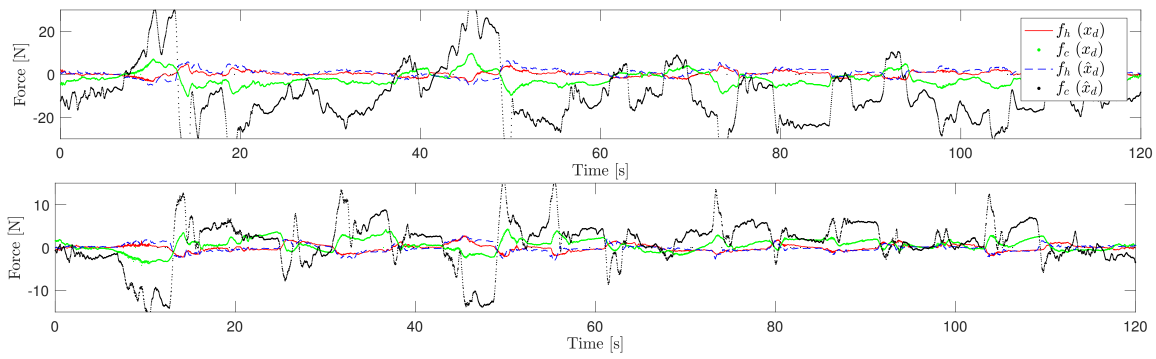
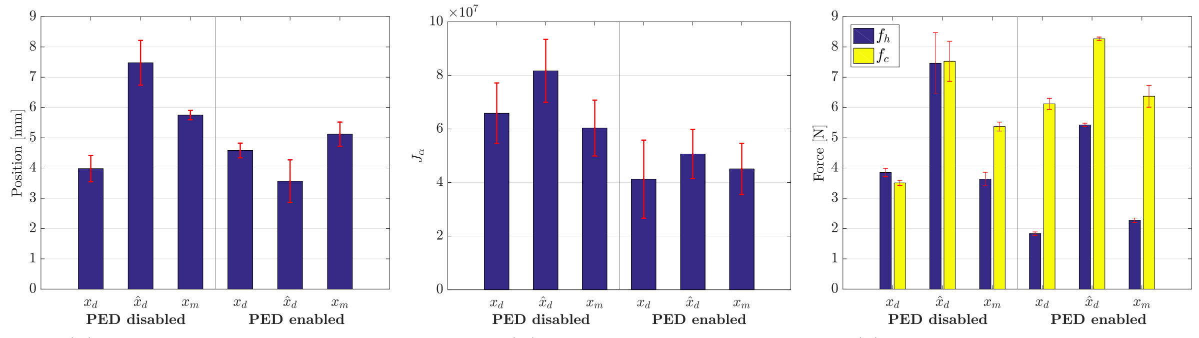


Figure 8.9. Human force measurements and NA control force in x and y-directions during one trial using the desired reference trajectory (x_d) and human intent estimator $(\hat{x}_d)$.



(a) Mean position error (b) Squared jerk (c) Mean force magnitude

Figure 8.10. Performance measures for different reference trajectories with the inner-loop prescribed error dynamics (PED) disabled (i.e. $\dot{\Lambda} = 0$) and enabled.

8.7 Conclusions and Future Work

In this paper we describe a two-loop pHRI controller which relies on human force and pose measurements and can adapt to varying robot dynamics. It can also adapt to different user preferences and simplifies the interaction by making the robot behave according to a prescribed error dynamics. In our controller formulation, two neural networks in the "outer-loop" predicts human motion intent and estimate a reference trajectory for the robot that the "inner-loop" controller follows. The inner-loop imposes prescribed error dynamics with the help of a model-free neuroadaptive controller, which feedback linearizes the robot dynamics with a third neural network. Lyapunov stability analysis gives weight tuning laws that guarantee that the error signals and NN weight errors converge and the desired reference trajectory is achieved. Our control scheme was implemented on a PR2 robot and experimentally validated. Results confirmed that the position error and motion jerk was reduced compared to a standard admittance controller, indicating that the two-loop controller achieves efficient and intuitive human-robot collaboration. The inner and outer control loops for the PR2 robot are available in the SkinLearn public repository at `https://bitbucket.org/nextgensystems/skinlearn`.

Future work will involve testing the proposed control scheme with more human subjects and different types of motions. Instead of simple point-to-point task motions, we envision complex rehabilitation exercises and co-manipulation of objects. More powerful robot real-time hardware would allow for larger neural networks and better performance.

ACKNOWLEDGMENT

This research was supported by US National Science Foundation Grants 1208623 and 1643989.

APPENDIX

To compute a suitable Cartesian pose error [82], the actual and desired trajectories were expressed in terms of 3D transformation matrices of the form

$$T = \begin{bmatrix} n(t) & o(t) & a(t) & p(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.62}$$

The overall Cartesian error

$$e = \begin{bmatrix} e_p \\ e_o \end{bmatrix} \tag{8.63}$$

is given by the linear position error

$$e_p = p_d - p \tag{8.64}$$

and orientation error

$$e_o = \tfrac{1}{2}(n \times n_d + o \times o_d + a \times a_d) \tag{8.65}$$

CHAPTER 9

Calibration of Tactile Sensors for Robot Skin

NEUROADAPTIVE CALIBRATION OF TACTILE SENSORS FOR ROBOT SKIN

S. Cremer, I. Ranatunga, S. K. Das, I. B. Wijayasinghe, and D. O. Popa, "Neuroadaptive Calibration of Tactile Sensors for Robot Skin," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1079–1085.

9.1 Abstract

In this paper we present a novel automated neuroadaptive approach that can characterize pressure sensitive “skin” deployed on a robot. Both the safety and performance of future co-robots can be greatly enhanced by such sensorized skin, by measuring multiple contact forces with humans and the environment. A challenge that arises with robot skin is the task of calibration to achieve reliable measurements necessary for safe human-robot interaction. To this end, the traditional method of calibrating each sensor prior to its use is a tedious task, especially with inexpensive, miniaturized hardware that can experience material degradation with time. Therefore, we propose an adaptive strategy that learns the sensor array characteristics together with the unknown dynamics of both the robot and human during physical interaction. Convincing experimental results with deployed pressure skin sensors on a PR2 robot are presented to validate our approach.

9.2 Introduction

Physical Human-Robot Interaction (pHRI) is an essential ability of co-robots of the future. For safe interaction, robots that work cooperatively with people must detect and limit contact forces [15]. This feature can be facilitated by so-called robot “skin”, which equips the robot with touch [16]. For the last three decades, there have been numerous examples of robot skin that embed tactile sensors and electronic interfaces into compliant or modular substrates. For instance, Dahiya [157] outlines the need for stretchable electronic systems to realize multi-functional electronic skin consisting of ultra-thin, flexible polyimide. Further work on ultra-thin substrates was demonstrated in [158] where hybrid flexible tactile sensors based on low temperature polycrystalline silicon thin film transistors were fabricated.

CellulARSkin is a multi-modal, self-organizing electronic skin [159]. The modular units of the skin perform temperature, acceleration, proximity, and normal force sensing. Each cell has the ability to explore connections and self-organize which, when combined with the accelerometers, allow the units to determine topology and their position on the robot. RoboSKIN is another advanced skin with the ability to communicate between adjacent units. In [160], RoboSKIN is integrated with the iCub robot and a method is proposed to compensate for temperature drift of the capacitive tactile sensors.

Tactile information, such as force magnitude and location during interactions and collisions, can be used in reactive controllers to enhance safety. Furthermore, tactile feedback can be utilized to predict human intent and execute behaviors such as guiding in a collaborative task. Grice et al. [161] show that a robotic care-giver is considerably more useful if physical contact is allowed with the help of tactile feedback.

Despite the need for robot skin, tactile sensing technology has made limited progress and full-fledged, whole-body human-like skin is yet to be realized [162]. There are several technological difficulties, especially the issues related to scaling and reliability.

Recent research has also described strategies for controlling the interaction behavior between humans, the environment, and a robot [81, 147]. Adaptive control schemes such as [148, 149, 163] adjust impedance parameters for safe and stable contact. Our recent work [44, 83] has proposed a neuroadaptive control framework to manage the pHRI problem. The approach involves two control loops that have guaranteed stability. In the controller "inner-loop", the unknown robot dynamics is being identified on-line and linearized through dynamic compensation. A second "outer-loop" determines the user's intent and adjusts an admittance based on task require-

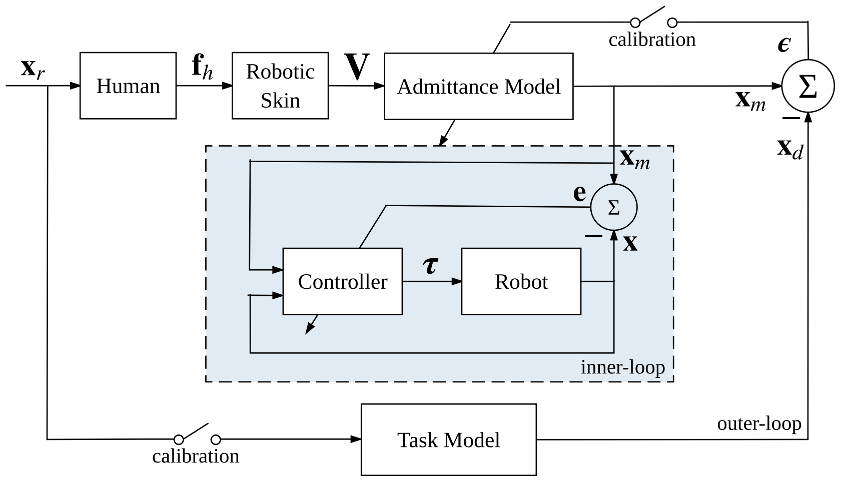


Figure 9.1. The inner and outer-loop control system for physical HRI with robot skin. During calibration the switch is on and both the human and robot follow the same reference trajectory.

ments and user skill. The effectiveness of the neuroadaptive controllers in pHRI tasks are shown in [164, 165] where the controller learns a combined human-robot model for trajectory following tasks. However, in our prior work, the force applied by the human is measured with conventional end-effector force-torque sensors.

In this paper, tactile data from robot skin in the form of conditioned but uncalibrated electronic signals is utilized instead. Recognizing that robot skin sensors may be manufactured using inexpensive and degradable materials, our contribution is to propose and validate an efficient sensor calibration scheme after sensors are deployed.

There has been considerable effort to calibrate tactile sensors to generate correct pressure or force values. Calibration procedures are usually time-consuming, require accurate instruments, and have to be repeated regularly due to changes in sensor response. The review papers [166, 167] discuss the importance of automatic calibration techniques and algorithms to interpret the sensing data. In the case of robot skin, any solution to the calibration problem should be scalable and automated due to the possibility of large number of sensors and their possible irregular spatial distributions. Although some work [168, 169] has been done to address the problem of sensor localization, these do not address the difficulty of having to calibrate each sensor individually to produce typical output values. For example, in [170] an operator has to manually apply different forces along each sensor axis in a controlled environment in order to calibrate a single skin patch.

In contrast, the approach presented in this paper uses a neuroadaptive calibration framework and learns appropriate skin sensor admittance models for each sensor "patch" using interaction cues from a human. Physically, such patches consist of pressure sensitive piezo-resistive "taxels" printed onto flexible Kapton substrates and encapsulated in compliant silicone substrates [84]. Consequently, such sensors experience "hard nonlinearities" such as hysteresis and drift, and can easily degrade with prolonged use. Experimental results with taxels mounted onto PR2's end-effector demonstrate that our approach can account for unknown robot and human dynamics, and, at the same time compensate for poorly known skin sensor characteristics.

The paper is structured as follows: Section 9.3 summarizes the neuroadaptive controller framework as proposed during sensor calibration. Then Section 9.4 describes the experimental setup and procedures. The results and discussion are presented in Section 9.5. Finally, Section 9.6 concludes the paper and proposes future work.

9.3 Neuroadaptive Calibration Approach

The proposed control system for the calibration process is shown in Fig. 9.1. It is composed of: (a) an inner-loop controller employing a neural network to continuously estimate and cancel the nonlinear robot dynamics; (b) an adaptive admittance model that learns to estimate desired Cartesian trajectories from sensor voltage values. In this section we briefly summarize the neuroadaptive control formulation. For full details, including stability proofs of the controller, the reader is directed to [44, 165]. In the later part of this section, our novel calibration procedure is introduced.

9.3.1 Inner-loop Controller

The general dynamics for a robot in contact with a human in Cartesian space can be written as follows [142]

$$\Lambda(q)\ddot{x} + \mu(q,\dot{q})\dot{x} + J^{\dagger T}F(\dot{q}) + g_x(q) = f_c + f_h \tag{9.1}$$

where $q, \dot{q} \in \mathbb{R}^n$ are the joint positions and velocities, n is the degrees of freedom (DOF) of the robot, and $x \in \mathbb{R}^6$ is the Cartesian space pose of the end-effector. The operational-space $\Lambda(q)$ is the inertia matrix, $\mu(q,\dot{q})$ contains the Coriolis/centrifugal forces, $g_x(q)$ is the gravity vector, and $F(\dot{q})$ is the friction term. The robot geometric Jacobian is J and the damped least-squares pseudoinverse is given by $J^\dagger = J^\intercal(JJ^\intercal + \zeta^2 I)^{-1}$, where ζ is a damping factor. During pHRI, the operator applies a Cartesian force $f_h \in \mathbb{R}^6$. A neuroadaptive scheme estimates the control force $f_c \in \mathbb{R}^6$, which is converted into joint control torques with $\tau = J^\intercal f_c \in \mathbb{R}^n$.

The inner-loop neuroadaptive controller defines a sliding mode error $r = \dot{e} + \Gamma e$, where $\Gamma = \Gamma^\intercal > 0$ is a positive definite matrix and $e = x_m - x$. Trajectory $x_m(t)$ is the model trajectory generated by the outer-loop which the robot is supposed to follow. Using (9.1), the dynamics of the robot model following error can be expressed as

$$\Lambda(q)\dot{r} = -\mu(q,\dot{q})r + f(\varphi) - f_c - f_h \tag{9.2}$$

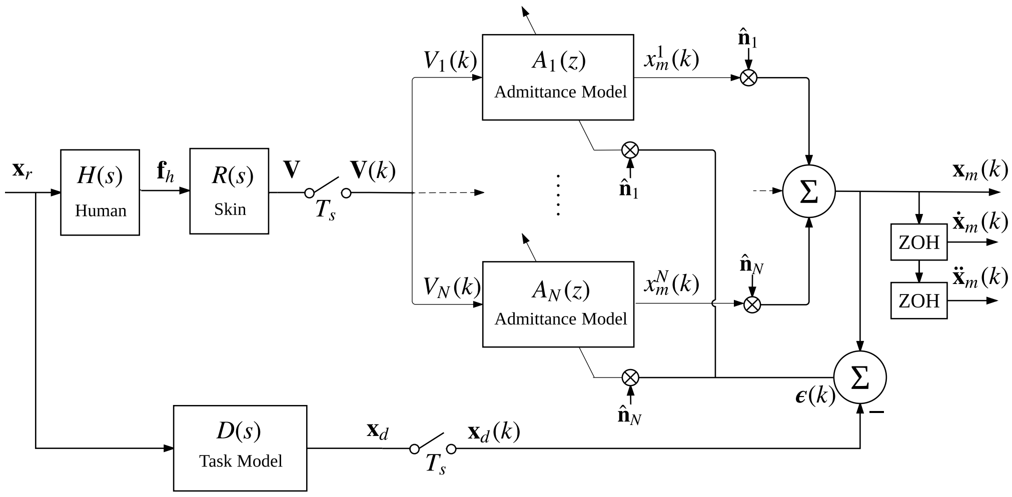


Figure 9.2. Robot admittance model tuning using adaptive inverse filtering during the sensor calibration process.

where

$$\begin{aligned} f(\varphi) =& \Lambda(q)(\ddot{x}_m + \Gamma\dot{e}) + \mu(q,\dot{q})(\dot{x}_m + \Gamma e) \\ &+ J^{\dagger T}F(\dot{q}) + g_x(q) \end{aligned} \tag{9.3}$$

is a nonlinear function of unknown robot parameters and $\varphi = [\ e^\intercal\ \dot{e}^\intercal\ x_m^\intercal\ \dot{x}_m^\intercal\ \ddot{x}_m^\intercal\ q^\intercal\ \dot{q}^\intercal\ ]^\intercal$. The idea of the inner-loop controller is to learn the nonlinear function (9.3) on-line using a neural network, thereby eliminating the need to model the robot dynamics. This approach has the advantage of decoupling the robot dynamics from the task specific outer-loop. The neuroadaptive control signal is given by

$$f_c = \hat{f}(\varphi) + K_v r - v(t) - f_h \tag{9.4}$$

where $\hat{f}(\varphi)$ is the neural network approximation of $f(\varphi)$, $K_v = K_v^\intercal > 0$ is a diagonal gain matrix, and $v(t)$ is a “robustifying” term as detailed in [164].

9.3.2 Outer-loop Controller

During pHRI, the human exerts force f_h on the robot manipulator in order to follow a task reference trajectory x_r. To enhance the performance of the human-robot system, the objective of the outer-loop is to introduce a virtual admittance between f_h and the model trajectory x_m, which the inner-loop controller tracks. For instance,

$$M_m\ddot{x}_m + D_m\dot{x}_m = f_h \tag{9.5}$$

is a mass-damper system in Cartesian space with mass matrix M_m and damping matrix D_m. This scheme is motivated by the so-called crossover model, which states that a human adjusts their control characteristics to the dynamics of the controlled element such that the total system remains unchanged [163]. Hence, a highly non-linear machine becomes easier to operate when behaving like a simple admittance model. In addition, as operators become more skilled, it has been established that "the closed-loop transfer function of the whole human-machine system becomes a first-order system at a wide frequency band" [163].

Therefore, in the absence of calibrated force information, the outer-loop controller will adapt the admittance model to make the human-skin-robot system behave according to a first-order task model

$$D(s) = \frac{a_d}{s + b_d} \tag{9.6}$$

where the parameters a_d and b_d are specific to the physical task to be completed.

In our past work [44] the admittance model was implemented with an Auto Regression Moving Average (ARMA) filter whose weights were updated using a Recursive Least Squares (RLS) algorithm [143]. Results showed that this scheme reduces

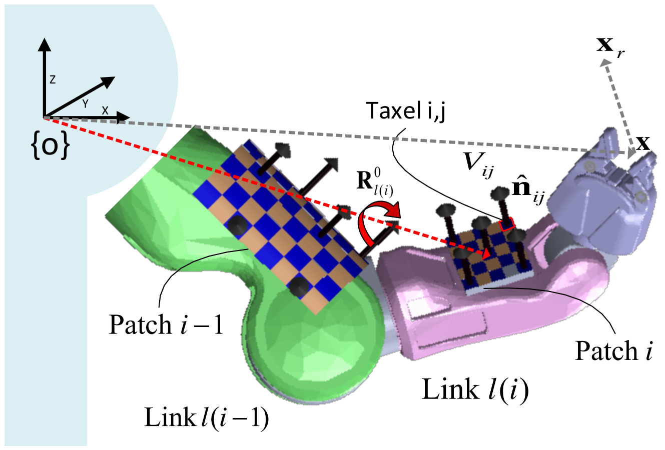


Figure 9.3. Taxel (i, j) inside skin patch i attached to link $l(i)$. The sensor measures a voltage V_{ij} related to the force applied opposite to the sensor normal $\hat{n}_{ij}$. The force has been applied to move the gripper pose x along the reference trajectory x_r.

the force and squared jerk during the combined human-robot interaction, when compared to a fixed model.

In this paper, instead of a known interaction force vector f_h, the robot skin sensors measure uncalibrated voltages V_{ij}, where $i = 1 \dots N$ is the patch index and $j = 1 \dots M$ is the taxel index inside each patch. The taxel orientation is determined by $\hat{n}_{ij}$, which is the outward normal as illustrated in Fig. 10.1. To simplify, let us assume that the robot receives only one resultant measurement V_i along $\hat{n}_i$ per patch, for example $V_i = \sum_{j=1}^{M} V_{ij}$. Then all skin patch measurements can be combined into a vector $V = [V_1, V_2, \dots, V_N]^\intercal$.

As shown in Fig. 9.2, the voltage measurements generate the model trajectory for the inner-loop controller. The outer-loop discretizes V and desired model output $x_d(t)$ by sampling the signals with sampling period T_s. The error $\epsilon(k) = x_m(k) - x_d(k)$ is utilized to update the discrete admittance model transfer functions $A_i(z)$. They will be tuned such that the overall human-skin-robot transfer characteristic $H(z)R(z)A(z)$ equals the sampled task model $D(z)$.

9.3.3 Calibration of Robot Skin

Since sensors are placed at different locations on the robot, they will exhibit varying responses, which in this paper are linearly approximated about operating points and denoted by $R(s)$. Therefore multiple admittances $A_i(z)$, one per sensor patch, will be identified during calibration. The user is asked to grab the robot by direct contact with individual skin sensor patch i and push it to follow a prescribed reference trajectory x_r. The outer-loop controller tunes an adaptive filter $A_i(z)$ with the raw sensor voltage V_i as an input and the model trajectory x_m^i as an output. Assuming a mapping between the force applied on the patch f_i and the voltage generated by the patch V_i, we can relate them by $f_i = g_i(V_i)$. In general, this mapping may be nonlinear, but here we assume that it can be linearized around the physical interaction point.

If the magnitude of the force is in the direction opposite to $\hat{n}_i$, define the resultant voltage average from each skin sensor patch i on link $l(i)$ to be

$$V = -\sum_i R_{l(i)}^0(q)\hat{n}_i V_i \tag{9.7}$$

where the rotation matrix $R_{l(i)}^0(q)$ for link $l(i)$ is with respect to the Cartesian reference frame (see Fig. 9.3). Then (9.5) and (9.7) become

$$M_m \ddot{x}_m + D_m \dot{x}_m = -\sum_i R_{l(i)}^0(q)\hat{n}_i V_i \tag{9.8}$$

For each sensor patch i this can be approximated by an ARMA model as follows

$$x_m^i(k) = -\sum_{p=1}^{n_1} a_p x_m^i(k-p) + \sum_{q=1}^{n_2} b_q V_i(k-q+1) \tag{9.9}$$

where n_1 and n_2 are the degrees of the denominator and the numerator of the admittance model respectively. Then $A_i(z)$ becomes

$$x_m^i(k) = h^\intercal(k)_i \, \theta_i \tag{9.10}$$

with a measured regression vector given by

$$\begin{aligned} h^\intercal(k)_i = [\, &-x_m^i(k-1), \ \dots \ , -x_m^i(k-n_1), \\ &V_i(k), \ \dots \ , V_i(k-n_2) \,] \end{aligned} \tag{9.11}$$

and the ideal ARMA parameter vector

$$\begin{aligned} \theta_i = [\, &a_1, \ \dots \ , a_{n_1}, \\ &b_1, \ \dots \ , b_{n_2} \,] \end{aligned} \tag{9.12}$$

In the RLS algorithm the estimated parameter vector $\hat{\theta}(k)$ is being updated with newly observed data $x_m^i(k+1)$ and $V_i(k+1)$. As the error $\epsilon(k) = x_m(k) - x_d(k)$ approaches zero, the algorithm converges to the optimal Wiener solution.

After the calibration process is complete, the model trajectory for the inner-loop to follow, $x_m(k)$, is calculated by adding the combined effects of all sensor skin patches in their respective normal directions as in

$$x_m(k) = \sum_i x_m^i(k) \, R_{l(i)}^0 \hat{n}_i \tag{9.13}$$

The velocity $\dot{x}_m$ and acceleration $\ddot{x}_m$ is found by applying an zero-order hold (ZOH) and computing the backward difference as indicated in Fig. 9.2.

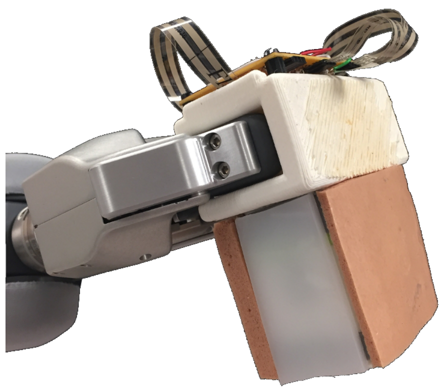


Figure 9.4. PR2 robot holding the tactile box for calibration experiments.

9.4 Experimental Setup

In this section we describe an experimental setup to validate the neuroadaptive calibration scheme. The Personal Robot 2 (PR2) was equipped with a custom made robot skin gripper attachment referred to as a *tactile box* (Fig. 9.4). The PR2 is a human-sized mobile manipulator with two 7 degrees of freedom (DOF) arms and two parallel grippers. It runs the Robot Operating System (ROS) [102] and executes a real-time control loop at 1 kHz. The neural network and adaptive filters were implemented with *Eigen*, a library for linear algebra [155]. Due to its computational complexity, the neuroadaptive controller is updated every 3rd loop in a parallel thread, essentially slowing down the control loop to 333 Hz. The selection of this sampling interval is appropriate for pHRI according to known requirements [156]. In the following section, the physical robot interface and the controller setup is described.

9.4.1 Robot Skin Interface

The neuroadaptive algorithm was used to calibrate unknown robot skin already mounted on the tactile box. Once grabbed by the robot, the box can be incorporated into the PR2 kinematic model. The four sides can each hold a single skin sensor or a

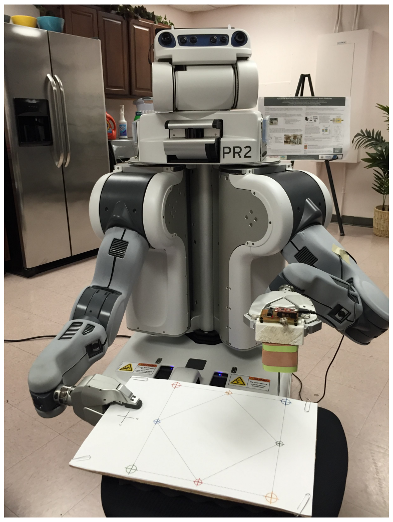


Figure 9.5. Setup for calibration and performance testing.

denser sensor array. In addition, different materials of varying thicknesses have been placed on top to form four skin patches. For electronic signal acquisition, conditioning, and networking, we designed and manufactured a *SkinCell* circuit board which is mounted on top of the tactile box. An Arduino Blend Micro sends the sensor data to the PR2 via USB. The data is normalized and then published at approximately 850 Hz to be used by the real-time controller.

Throughout the experiments described in this chapter, the tactile box accommodated four low-cost, nonlinear piezoresitve Tekscan Flexiforce sensors [171]. Each sensor was coated by an elastomeric material, including Frubber (a compliant life-like skin made by Hanson Robotics) and P10 (a less compliant, silicone elastomer). The thickness of the elastomers and the properties of the Flexiforce transducers were not carefully characterized prior to this experiment, and therefore the sensor response $R_i(s)$ on each side of the box were unknown prior to calibration.

9.4.2 Controller Setup

The performance tuned inner-loop neuroadaptive controller parameters are listed in [44]. For each skin patch, the ARMA filter weights were initialized to zero and then calibrated individually by following a fixed reference trajectory $x_r = -0.25\hat{n}_i$ meters along the sensor axis. Training was stopped once gripper pose x was within 1 cm of the desired reference position. The task model $a_d = b_d = 0.7$ was adjusted to get a fast motion response while limiting the maximum velocity below 0.2 m/s (it takes approximately 4 seconds to move 25 cm). The point-to-point motion was initiated by a human who grabs the tactile box and applies a force perpendicular to the skin patch. Visual guidance was provided in the form of a printed square pattern shown in Fig. 9.5. Two experimental calibration trials where performed, each consisting of three completed loops around the square shape. Hence, each sensor model $A_i(z)$ was updated six times, where each subsequent training episode continued with the previously learned weight vector. Once trained, the filter weights were fixed and the user had to follow the same square outline and an additional diamond pattern. The performance was compared to an "uncalibrated" setup where the average filter weights were used for all sensor patches.

9.5 Results

In the calibration experiments, weights of the four ARMA filters (one for each sensor patch) were adapted a total of six times. The largest updates occured in the first two training episodes, as shown by the filter norms in Fig. 9.6. The models for sensors 2 and 4 (along the y-axis of the gripper frame) changed less than 25% in the remaining episodes. The norm for sensor 1 stabilized in the last calibration step, while the update rate for sensor 3 slowed down. The weights are not expected to converge fully because of small variations in the human model $H(s)$ and skin model $R(s)$. During interaction, the operator continuous to learn and the cheap sensors experience temperature variations, drift, etc. The inner-loop controller also learns and compensates the nonlinear robot dynamics. In fact, several systems in this experiment learn at the same time: the human user, the outer-loop admittance model, as well as the inner-loop feedback linearization compensator. Although it is hard to discern their individual contributions, we can assess the overall performance of our scheme by the resulting trajectories.

The trajectories of the first calibration loop around the square are depicted in Fig. 9.7. The x and y Cartesian positions are shown in the robot torso frame, while z was fixed at $-0.1\,\mathrm{m}$. Note that the inner-loop receives x_m in this global reference frame, while the ARMA filter computes a change in the gripper frame. As previously mentioned, x_r was 0.25 meters in the opposite direction of the sensor normal. The reference trajectory was smoothed by task model $D(s)$, making each episode last approximately 4 seconds when using the $1\,\mathrm{cm}$ threshold for gripper pose x. The filters adapted quickly and model trajectory x_m converged to the task trajectory x_d. When the tactile box was close to the goal position, the desired velocity was drastically reduced since x_d approaches x_r asymptotically. Oscillations can be observed, caused by the robot moving slower than the operator and fighting against the user's movements. Hence, the selection of task model is important and future work could involve testing higher order models. Also, if the robot moves too fast, the operator could lose contact with the sensor patch. This would prevent the ARMA filter from learning the correct mapping between voltage input and model trajectory output.

As illustrated in Fig. 9.8, there were large variations in the signal magnitude from one side of the box to another. For example, sensor 1 saturated at approximately $0.5\,\mathrm{V}$, while sensor 4 was least responsive and produced values below $0.35\,\mathrm{V}$. When

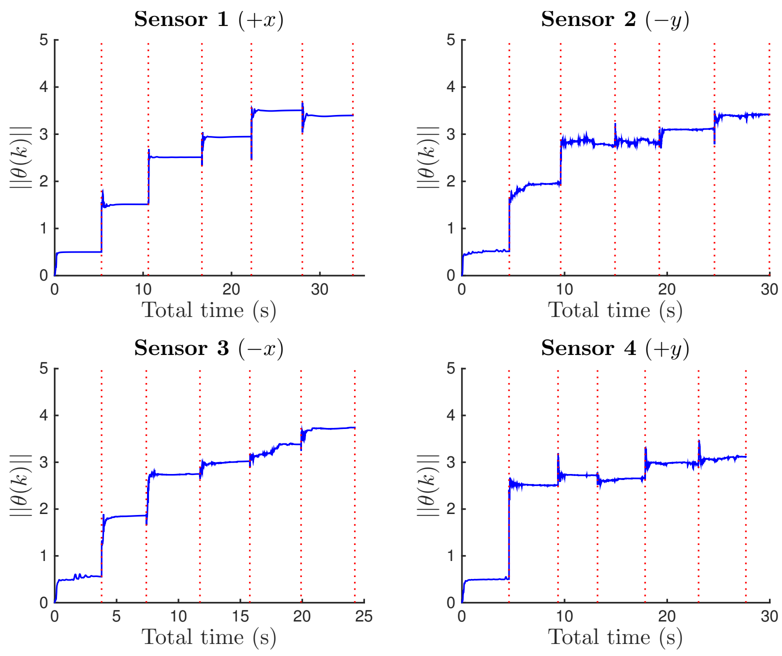


Figure 9.6. ARMA filter weight norms during two calibration trials. The vertical lines indicate the beginning of a training episode.

using a second order ARMA model, the noise from the voltage signals appeared in x_m and resulted in jitter. To improve the performance, the filter size was increased to $n_1 = n_2 = 4$ in (9.9). This also reduced the aforementioned oscillations in x_m. The learned filter weights are listed in Table 9.1. Even though the sensors, electronic transducers, and materials on each side of the box were different, after calibration, they appeared to have a similar response when touched. Thus, each filter learned a unique ARMA model that characterized the tuned admittance for each side of the tactile box.

After two calibration trials, all sensor patch ARMA filters were fixed and the user was asked to follow the square and diamond patterns. The robot no longer knew the reference trajectory x_r and was instead being guided by the human. Fig. 9.9 shows the tracking performance in trial 1 when using calibrated and uncalibrated filter weights. In the latter case, the average ARMA model in Table 9.1 was applied to all

Table 9.1. Learned ARMA filter weights after two trials. The mean value of all weights was used in the uncalibrated setup.

	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4
$\hat{\theta}_1$	-2.20	1.96	-0.85	0.11	0.34	-0.96	0.98	-0.35
$\hat{\theta}_2$	-2.14	2.07	-1.13	0.22	0.33	-0.85	0.79	-0.26
$\hat{\theta}_3$	-2.56	2.29	-0.68	-0.04	0.32	-0.91	0.85	-0.26
$\hat{\theta}_4$	-1.97	1.64	-1.05	0.38	0.42	-0.98	0.84	-0.27
$\bar{\theta}$	-2.22	1.99	-0.93	0.17	0.35	-0.92	0.87	-0.29

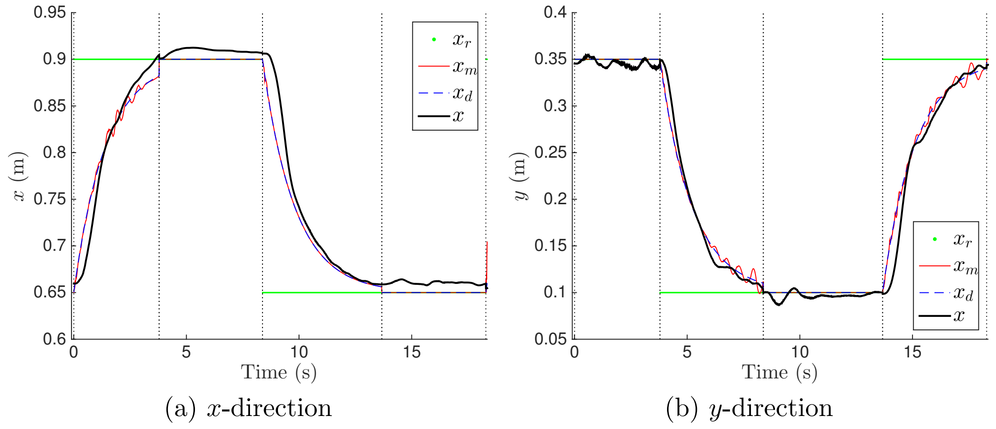


Figure 9.7. Trajectories in x and y-directions of the torso reference frame during one calibration loop around the square pattern.

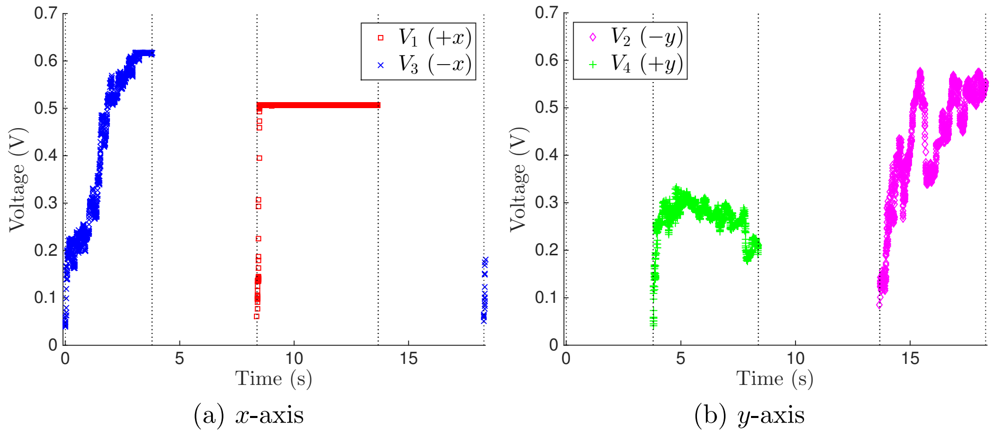


Figure 9.8. Voltage measurements from sensors along the x and y-axis during one calibration loop around the square pattern.

four patches on the box. It is evident that the uncalibrated setup performed worse: the operator often deviated from the desired path and overshot the end-positions. Since each skin patch had a different material and/or thickness, some sensors became too sensitive while others were less responsive.

To quantify the tracking performance, the error at each time step kT was computed as a 2-norm distance and then summed over the entire trial:

$$e_{\text{trial}} = \sum_k \|x_{\text{des}}(kT) - x(kT)\|_2 \tag{9.14}$$

where k is the real-time loop count and $T = 0.003$ seconds. The performance results are summarized in Table 9.2, including the completion time of each trial. A faster time indicates that the user had more control and that the system performed better. The calibrated sensors consistently produced faster times as well as lower error norms. Interestingly, the error is reduced in subsequent trials (with one exception), since the operator is able to learn and adjust his own model $H(s)$. As expected, the square shape was easier to follow than the diamond shape, which required activating two sensors at the same time. However, the results confirm that adding the effects of each side of the box according to (9.13) is valid.

Overall, the experiments show that the calibration scheme greatly improves tracking performance. Our approach has several advantages including a relatively quick calibration time and ease of implementation, i.e. the skin patches can already be mounted on the robot. The behavior can also be tailored to individual pHRI preferences: for example if a new user is not satisfied with performance, the scheme can be re-calibrated according to their own preferences represented by the reference trajectory x_r and the task model $D(s)$. On the other hand, it can also be seen that the user had difficulty reproducing the desired trajectory with high fidelity. This is

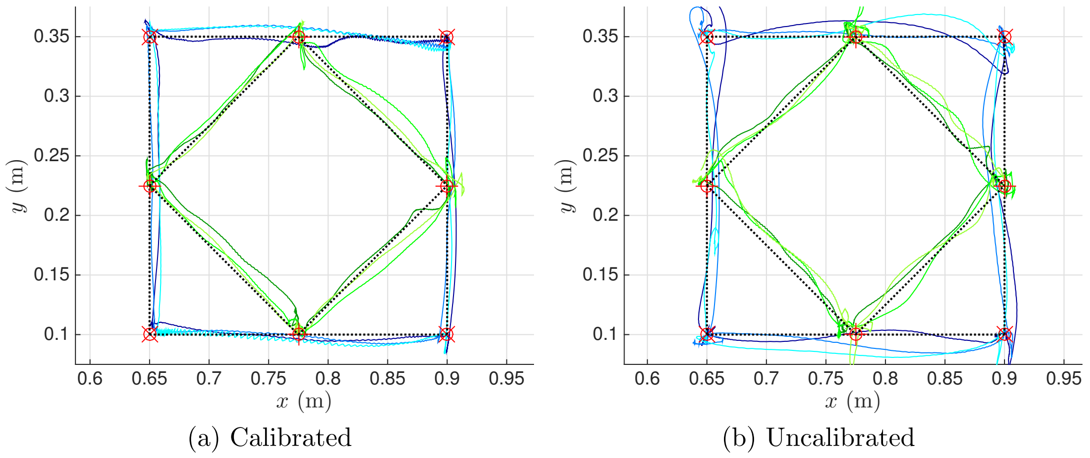


Figure 9.9. Cartesian trajectory tracking using calibrated and uncalibrated filter weights. One trial consisted of moving the tactile box three loops around the square and three loops around the diamond shape.

due to the fact that the sensor density in our experiment was low, represented by a single taxel per skin patch.

9.6 Conclusions and Future Work

In this paper we proposed a novel algorithm for adaptive calibration of robot skins by directly tuning admittance models that map voltages into desired robot motion. An inner-loop neuroadaptive controller then follows the generated model trajectory. The advantages of our approach are the high degree of generality and adaptability to different robots, human preferences, and sensors. The robot skin admittance model of each skin patch $A_i(s)$ are learned using an adaptive inverse algorithm implemented as a linear least square regression in order to satisfy $H(s)R(s)A(s) = D(s)$, where $H(s)$, $R(s)$ are unknown human and sensor transfer functions, and $D(s)$ is a nominal task model.

Experimental results with the PR2 robot holding a sensorized tactile box show that admittance filter weights converge relatively quickly. Once each sensor patch has

Table 9.2. Tracking performance when following square and diamond shapes. Each trial (consisting of three loops) was performed using calibrated (C) and uncalibrated (U) filter weights.

Trial	Square				Diamond			
	Error (m)		Time (s)		Error (m)		Time (s)	
	C	U	C	U	C	U	C	U
1	0.52	1.02	32.9	39.5	0.82	1.09	41.3	50.4
2	0.70	0.89	36.2	52.1	0.76	0.96	39.2	49.5
3	0.53	0.88	34.5	49.7	0.74	0.89	40.2	50.4

been calibrated, the nominal task model can be removed as there are no restrictions on the motion that the human can impose. Results show that the tuned filters perform better at tracking reference trajectories than the uncalibrated ones.

Future work includes validation of our approach with denser arrays of tactile sensors embedded onto larger areas of the robot. More experimental results with non-expert users and more general 3D trajectories will be undertaken. Finally, nonlinear regressors will also be studied in place of a linear ARMA filter to account for nonlinear skin responses.

ACKNOWLEDGMENT

This research was supported by National Science Foundation Grants NRI-1208623 and IIP-1534124. The authors wish to thank Fahad Mirza for his contribution to the skin electronic interface used in our experiments, and Joshua Baptist for the sensor box design and fabrication.

Part IV

Developed Software Tools

CHAPTER 10

SkinSim

SKINSIM 2.0: A DESIGN AND SIMULATION TOOL FOR ROBOT SKIN WITH CLOSED-LOOP PHRI CONTROLLERS

S. Cremer, I. B. Wijayasinghe, S. K. Das, and D. O. Popa, "SkinSim 2.0: A Design and Simulation Tool for Robot Skin with Closed-loop pHRI Controllers," *To Be Submitted: IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2017.

10.1 Abstract

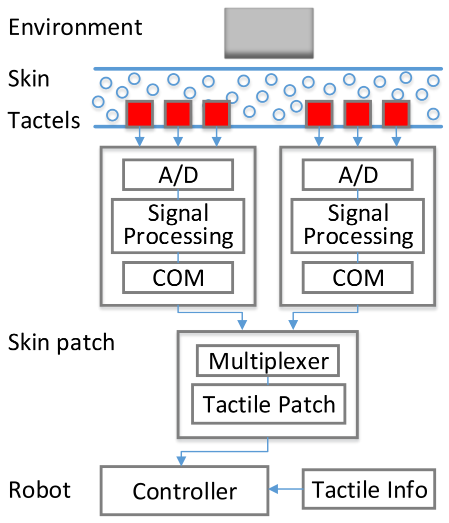
The importance of tactile sensing for physical Human-Robot Interaction (pHRI) and dextrous manipulation is well-known in robotics. SkinSim provides a simulation environment for multimodal robot skin which can address design problems for whole-body tactile sensor arrays. In this paper, scalable modeling approaches are presented for simulating pressure-sensitive skin patches, with consideration of the sensing element geometry and mechanical structure, signal quality, data processing, and force controller. The novel open-source simulation architecture of SkinSim is compatible with Gazebo and ROS environments and supports both robot skin dynamic models, as well as tactile sensing element models. A force dispersion model was introduced for the simulation of sensors embedded in a mechanical damping layer. Simulation examples of robot skin with different tactile resolutions, signal noise, and time delays are presented and design choice impact on simple pHRI controller performance is evaluated. Performance measures include center of pressure estimation errors and control signal rise and settling time, overshoot, and steady state error.

10.2 Introduction

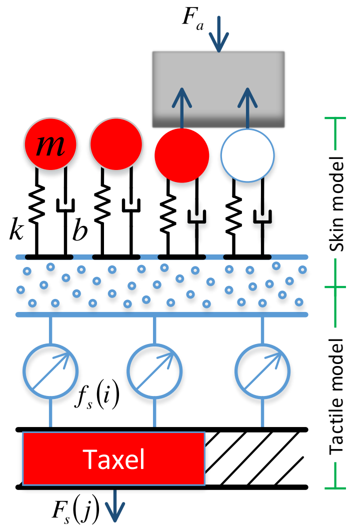
With more and more robots being introduced into human environments, physical Human-Robot Interaction (pHRI) has become essential for high performance collaborative tasks. Assistive and cooperative robots must be able to anticipate and sense interactions with users. This can be enabled by robot skin, which provides the robot with sense of touch [16]. Information from tactile sensors or tactels, such as force location and magnitude during collisions and interactions, can be used with reactive controllers for improvements in safety [172]. Feedback from the tactile sensors can also be used for human intent prediction and execution of behaviors such as robot guiding during a collaborative task. Besides safer and more efficient pHRI, robot skin can help coping with human environments which are highly dynamic and unstructured in nature [173]. For dexterous manipulation, the sensors can be used to determine physical properties through interactions, such as surface roughness and hardness, vibrations, temperature, and moisture [174]. This information can be used in tactile-sensing-based algorithms that are able to deal with uncertainty in object location and shape [175].

Despite the importance of robot skin, several technological challenges related to scaling, calibration, networking, and integration still have to be overcome. Dahiya et al. [176] provide a survey of current transduction methods and challenges in whole-body sensing for robotics. They hypothesize development of tactile sensing has been slow due to the absence of any tactile analog to the CMOS optical array. A system on chip (SoC) or system in package (SiP) approach that would process and send data at the same site could reduce the amount of wires. Since it is generally expensive to develop and test new hardware, realistic simulators are important tools for tackling and solving such issues in an efficient manner. Finite element (FE) modeling has been used for designing robot skin that is suitable for natural human-robot interaction [177]. However, FE methods are inadequate for evaluating the performance of an entire human-skin-robot system, including control performance as a function of sensor placement and tactile density.

Although many robot simulators are available, including Microsoft Robotic Studio [178], MORSE [179], V-REP [180], and Gazebo [181], the simulation of robot skin in these environments is still in its infancy, and mostly interact through rigid contacts and kinematics solvers since they are suitable for open kinematic chains. RobWork-Sim [182] and OpenRAVE [183] both support tactile sensing and utilize ODE (Open Dynamics Engine) to estimate point forces for grasp analysis and generation. ODE



(a) Sensing system



(b) Skin model

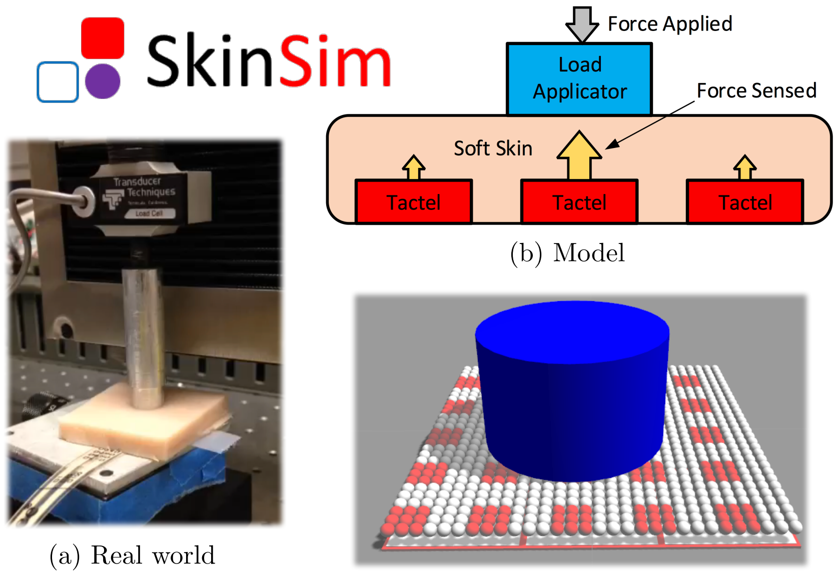
Figure 10.1. Robot skin with skin patches that process and send data. To model tactels embedded in a solid polymer, the skin dynamics are simulated as a layer of mass-spring-damper (MSD) systems while the sensed force is generated from a separate tactile model.

is a popular open source physics engine for solving rigid body dynamics and collision detection [184]. Object interactions are simulated with an impulse method and are constraint-based. The collision detection system utilizes a multi-resolution hash table, where the time complexity of intersection testing of n objects scales with $O(n)$ as opposed to $O(n^2)$. ODE can handle many collisions and is highly scalable. Ashley-Rollman et al. [185] have demonstrated simulations with several million objects and managed to run simulations 108x faster with a multithreaded version of ODE.

In 2014, SkinSim was introduced as an open-source simulation environment for multimodal, compliant robot skin [52]. Its purpose is to study the impact of sensor type, placement, noise, and networking on the performance of robots. The idea of multimodality is combing multiple sensors, thereby providing the robot with more information about the interaction. SkinSim is built on top of Gazebo and ODE, which is one of several compatible physics engines [181]. It is also closely integrated with the Robot Operating System (ROS) [102]. First simulations suffered from solver instabilities whenever the number of elements exceeded 100. Hollis et al. experienced issues with “high-frequency oscillations in the contact force signals” when using SkinSim 1.0 to validate their compressed sensing techniques for large-scale tactile data acquisition [186]. The initial software release included features for auto generating models and batch execution of experiments, however features such as automated data collection and experiment testbed generation were missing.

Two years after its initial release, SkinSim 2.0, the simulator reported in this paper, has seen a major overhaul and several enhancements to make it a truly robot skin design tool with automated performance assessment. Tactile surfaces are modeled in a manner reflecting deployed robot hardware that can be customized in layers (Fig. 10.1a). The robot skin is discretized into skin elements modeled by mass-spring-damper systems (Fig. 10.1b), which was motivated by experimental results from a testbed for robot skin characterization [187]. As depicted in Fig. 10.2a, robot skin was placed on a flat surface and vertical controlled forces were applied with a plunger. This setup provides standardized conditions for force control tests performed on different skin materials and configurations. SkinSim 2.0 replicates the testing process in software for the purpose of skin design prior to prototyping and experimentation.

In this paper, we describe the new SkinSim framework and discuss its improved features such as force dispersion, signal noise, and time delays, as well as an experimental testbed for automated data collection while testing different model and control parameters. SkinSim 2.0 enables stable simulations with a larger number of skin elements (more than 100) by introducing a hierarchal model structure with so-called subplanes. Also, the user can prioritize between simulation accuracy and speed by adjusting configurable parameters such as physics engine solver iterations and step size. In total, the new version provides 39 model, 9 control, and 3 simulation param-



(a) Real world (b) Model (c) Simulation

Figure 10.2. Characterization and modeling of tactile sensors embedded in a superficial damping layer using (a) a controlled force plunger, and the equivalent (b) dispersion model and (c) simulation in SkinSim.

eters that the user can adjust[1]. A case study is presented where the performance of a closed-loop pHRI controller for robot skin is evaluated, considering the influence of different skin parameters such as tactel size and separation.

The paper is organized as follows: Section 10.3 provides an overview of modeling approaches for robot skin, as well as experimental model verification. The SkinSim system architecture and several updates are described in Section 10.4. Sections 10.5 outlines simulation results for several sensor array configurations. Finally, Section 10.6 presents conclusions and future work.

[1]These are listed and described in the configuration files available at `https://bitbucket.org/nextgensystems/skinsim`.

10.3 Modeling of Robot Skin

The purpose of SkinSim is to realistically model robot skins, while being flexible enough to handle several types of sensors and different types of interactions. Dynamic simulations have to be scalable such that simulated robots can be covered in skin patches. In this section we describe different modeling approaches and components of SkinSim 2.0, including sensing elements, geometry and mechanical structures, signal quality, and data processing.

In a survey of tactile human-robot interaction, Argall et al. [16] present several multimodal devices for detecting human touch, e.g. they measure pressure, temperature, or distance between humans and robots. As a simulator for multimodal skin, in its final version, SkinSim will support different types of sensing, including contact (location, area, duration), force magnitude and direction, pressure, temperature, infrared (IR), and acceleration. The Gazebo robot simulator supports a multitude of sensors via so-called plugins [181], which makes it ideal for SkinSim. Version 2.0 includes a major overhaul and improvements to the sensing systems by taking advantage of the Gazebo plugin architecture. However, the primary focus is still on contact sensing, in particular center of pressure, force magnitude, and direction.

In general, there are two approaches for covering a robot in sensorized skin. The 3D-curved surfaces can be divided into planar subsurfaces. This is the most common approach, and includes rigid as well as bendable sensors [188]. Another option is covering the entire surface with elastic, stretchable sensors forming a continuous surface [189]. To be compatible with these different approaches, SkinSim discretizes the skin into small, spherical skin elements, as depicted in Fig. 10.1b. This allows approximation of continuous surfaces as well as planar subsurfaces. In addition, stretchable skin can be modeled by including mass-spring-damper systems between the elements in the 2D manifold (however this feature is not available in this simulator version).

Using spherical skin elements reduces the amount of contact points and simplifies collisions with the simulated environment.

10.3.0.1 Skin Elements

Modeling each skin element as a mass-spring-damper allows for simulation of soft contact interaction without modeling the actual deformation of bodies. If a body comes in contact with the skin array, then the force applied on element i results in displacement $x_i(t)$ determined by the dynamic equation

$$f_{\mathrm{a}}(i) = kx_i + b\dot{x}_i + m\ddot{x}_i \tag{10.1}$$

where k is the spring constant, b the damping coefficient, and m the mass. Since the elements are relatively small, the term $m\ddot{x}_i \approx 0$ and can generally be neglected. The total force exerted on a skin array with N_e elements is simply the summation of the individual forces:

$$F_a = \sum_{i=1}^{N_e} f_{\mathrm{a}}(i) \tag{10.2}$$

By defining each skin element as a prismatic joint, the skin is able to "deform" and the nominal skin boundary can be penetrated. In this way, rigid body solvers such as ODE can still be used to accurately model the contact of skin. This approach is computationally efficient, easily parallelized, and can be experimentally validated.

10.3.0.2 Tactile Elements

To emulate tactile sensors, subgroups of these spherical skin elements are selected. For example, in Fig. 10.2c the tactile size is 3×3 elements (marked red) and the tactels are 3 skin elements apart. This provides flexibility to define a variety of sensor shapes and resolutions. Sensor readings are generated by combining element forces into a sensed force determined by the skin model. The force sensed can be some nonlinear function of the forces applied on the tactel due to force dispersion, noise, etc. If $I(j)$ is a list of element indices that are included in tactel j, then the force sensed by the tactile sensor is

$$F_{\mathrm{s}}(j) = \sum_{i \in I(j)} f_{\mathrm{s}}(i) \tag{10.3}$$

where $f_s(i)$ is determined by the tactile model.

10.3.0.3 Skin Layer

In addition to data acquisition, robot skin serves as a mechanical damping layer that can absorb collisions. If the transduction layer is placed on top of the damping layer, the response time is faster while the underlying layer still provides sufficient damping [190]. However, this requires a robust sensor layer that tolerates impacts and is generally more difficult to manufacture. Another possibility is to embed the tactile sensors within the damping layer (Fig. 10.2b). This protects the sensors but reduces the sensitivity and the spatial resolution. The sensitivity can be improved by reducing the material density and introducing air gaps. This reduces the deformation energy and can also serve as mechanical filtering. For example, geometrically defined voids allow easier compression of the material and prevent the force from spreading throughout the damping layer [190].

For the simulation results presented in this paper, the tactile sensors were assumed to be embedded in a polymer. A dispersion model is proposed that describes how forces spread from the point of force application to adjacent tactels. When a force is applied, the material damping layer tends to spread laterally. As a result, the

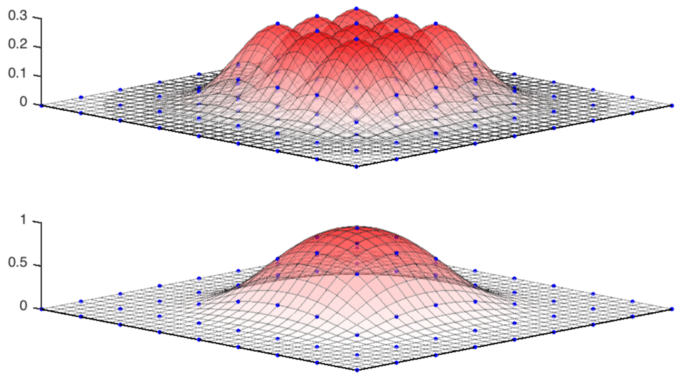


Figure 10.3. Modeling of force dispersion caused by the robot skin dampening layer. Every skin element (blue dot) applies a Gaussian force distribution (top), which are summed to obtain the total force profile (bottom).

sensors lose sensitivity and measure a force smaller than the one applied. At the same time, tactile sensors adjacent to the object in contact are also activated (Fig. 10.2b). The relationship between the tactile reading and the distance between plunger center and tactile sensor follows a Gaussian distribution, which can be characterized for different materials and thicknesses [187].

In SkinSim, this is modeled by multiplying the individual applied forces f_a with Gaussians, which can be summed to obtain the total force measurement as depicted in Fig. 10.3. This is described by

$$f_{\mathrm{s}}(i) = \sum_{k=1}^{N_e} K_d \, \alpha(\, D_{ik} \,) f_{\mathrm{a}}(k) \tag{10.4}$$

where K_d is a constant, matrix $D \in \mathbb{Z}^{N_e \times N_e}$ stores the distance between each element pair, and $\alpha(x)$ is a Gaussian distribution with mean μ_d and standard deviation σ_d,

$$\alpha(x \mid \mu_d, \sigma_d^2) = \frac{1}{\sigma_d \sqrt{2\pi}} \; e^{-(x-\mu_d)^2/2\sigma_d^2} \tag{10.5}$$

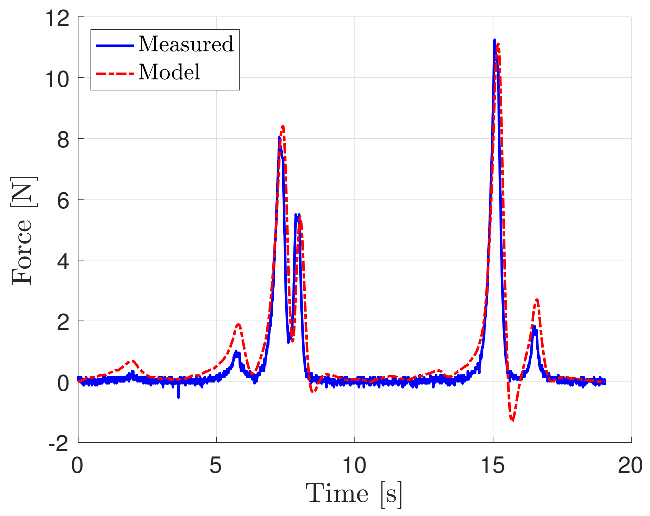


Figure 10.4. Comparison of actual and predicted sensed force through a spring-damper model for 4mm *Frubber* skin [187].

10.3.1 Experimental Validation of Skin Models

Parameters and data fit confirming a mass-spring-damping model were obtained from experiments with robot skins described in [187, 191]. These consisted of piezoresistive *Tekscan FlexiForce* sensors embedded in Silicone elastomers or *Frubber*, a spongy elastic polymer that mimics human skin. The skin samples were loaded with a force controlled plunger that can apply a desired force. The sensors were covered in 4 mm thick *Frubber* and persistently exciting force inputs F_a were applied while recording the sensed force F_s. A second order model identification was carried out using the MATLAB System Identification Toolbox and the model parameters shown in Table 10.1 were extracted. In Fig. 10.4, the measured sensed force from pseudo-random deformation input was compared to the spring-damper model output. The mean-squared error between the measured and predicted sensed force was 0.520 N, or 15.5% [187]. The error is highest at low sensed forces, but is relatively small at the peaks (above 4 N applied).

Experiments for modeling the force spread were also conducted by applying the plunger in varying locations on an embedded array of tactels with diameter 1 cm each, placed 1.4 cm apart. Fig. 10.5 shows the response from a single tactel as the plunger force and distance from the tactel center vary. Gaussian data fit matching the model in (10.5) can be extracted from this data with R^2 values above 0.96 [187].

10.3.2 Signal Quality

The performance of real tactile sensors is negatively impacted by noise. The signal-to-noise ratio (SNR) is crucial for factors such as accuracy, resolution, and the range of measurement. Noise can also have a negative effect on collision detection, the control loop, etc. Therefore, incorporating noise into SkinSim is important to model and to accurately predict robot behavior in a real environment.

In general, transducer noise is a function of material and sensor design (piezoelectric, capacitive, etc.). It can be reduced through different filtering methods, but also by changing structural design parameters such as sensor size, density, and skin thickness. There are different types of noise such as thermal, shot or low frequency, flicker or quantization, burst, and transit-time noise as well as coupled noise such as crosstalk and interference [192]. In the simulation environment, these can be combined into a single white Gaussian noise channel. Gaussian noise $\mathcal{N}$ was added to each tactile sensor, scaled to be inversely proportional to $\sqrt{N_s}$, where N_s is the num-

Table 10.1. Model parameters for 4mm thick Frubber skin [187].

Coefficient	Mean	Std. Dev.
b_1 [Ns/m]	242.6	3.03
k_1 [N/m]	1523	82

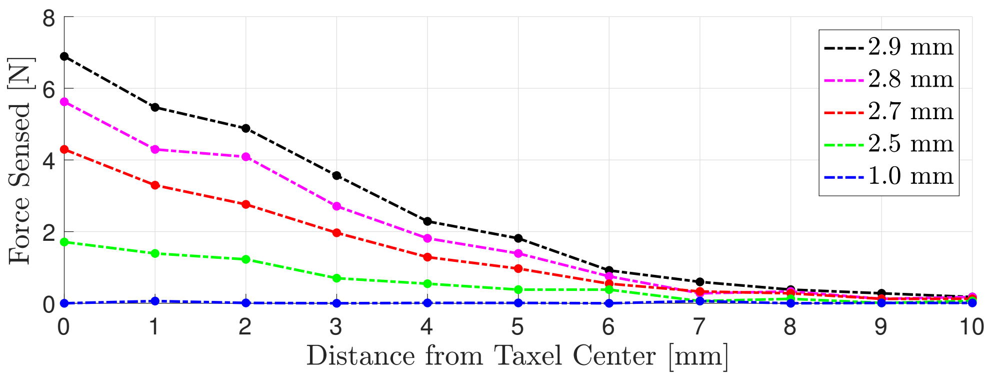


Figure 10.5. Tactel force sensed at varying distances (1 to 10 mm) from tactel center in 4mm *Frubber* skin [187]. Each line represents a different deformation depth proportional to the force applied.

ber of skin elements making up the sensor. If Z_j denotes the noise for tactile sensor j, then

$$Z_j \sim K_n \mathcal{N}(0, \sigma_n^2)/\sqrt{N_s} \tag{10.6}$$

where K_n is a constant, the mean is zero, and σ_n is the standard deviation of the noise. This scaled noise model was inspired by [193] with the following assumptions: The sensors are piezoeresistive, operate at a relatively low frequency, and the input voltage is large with respect to the thermal voltage. Therefore, flicker noise will dominate over thermal and Johnson noise. For this case, Bae et al. [193] demonstrated that the SNR is proportional to the square root of the number of carriers, which in turn is proportional to the side length l for a square sensor. In other words, Z_j is inversely proportional to l. For a constant skin resolution and square sensor, this length is given by $\sqrt{N_s}$.

10.3.3 Data Processing

SkinSim aims to simulate large tactile sensor arrays that cover large surfaces of robots. As a result, it needs to incorporate information such as sampling rate and time-delays from the skin patches to the robot controller. Ideally, an SoC/SiP approach should be used where the individual tactile elements process and send data at the same site (Fig. 10.1a). SkinSim can be used to simulate the throughput by processing data at different levels and using serial or parallel transmission approaches. The sampling speed depends on the sensor quantity and performance characteristics, control circuitry, and the bus communication speed. In this paper, a serial approach was simulated with the time for scanning a sensor array being proportional to the number of sensors

$$T_s = K_{\text{data}} N_t \tag{10.7}$$

where T_s is the maximum controller update rate, N_t is the number of tactels, and K_{data} depends on the number of bits used to represent tactile data and the bus speed [194].

Instead of transmitting raw data, a skin patch could pre-process the data by computing the resultant force and Center of Pressure (COP). For a planar patch, the force sensed in (10.3) is normal to and acting at the center of the tactel j. A force weighted COP can be calculated using the sensed forces aggregated for all tactels

$$\vec{\text{COP}}_s = \frac{\sum_{j=1}^{N_t} F_s(j)\vec{c}_j}{\sum_{j=1}^{N_t} F_s(j)} \tag{10.8}$$

where $\vec{c}_j$ is the vector to the center of tactel j and $F_s(j)$ is the force sensed by tactel j.

10.4 Description of Simulator

This section provides a brief architectural overview of the SkinSim 2.0 simulator, including structural changes and new functionalities. The block diagram in Fig. 10.6 shows the design and testing process of new robot skin. The first step involves mesh manipulation and model generation to create the skin array SDF model files. These are fully autogenerated, however sensor placement still requires external 3D CAD design software tools such as Solidworks and Meshlab. Next, the SDF files are loaded into a Gazebo/ROS environment for automated testing with different control schemes. New user interfaces (UIs) display the data in RVIZ, a 3D visualization tool part of ROS. All these functionalities are part of the following software components:

- *Model, Control Specs*: New parameters specifying the array size, tactile density and model structure, tactile data communication design, experiment setup, etc.
- *Model Builder*: Generates SDF skin and world models from a configuration file. The Gazebo world includes an experiment testbed with a force controlled plunger.
- *Gazebo plugins*: New plunger controllers for testing closed-loop pHRI. Pseudocode for the new robot skin model is shown in Algorithm 1.
- *Auto Generator*: Generates test cases for different combinations of model and control parameters.
- *Auto Calibrator*: Computes calibration constant for the sensed force at steady-state.
- *Auto Experimenter*: Runs the generated test cases.
- *SkinSim ROS*: For 3D visualization and data collection.

10.4.1 Stability and Scalability

The robot skin model is defined in a Gazebo plugin. In version 1.0, there were two separate plugins for skin and tactile elements that were modeled in two separate layers. Since the tactile sensors are usually much stiffer than the mechanical damping layer, the tactile layer was removed and the dynamics were integrated into the skin

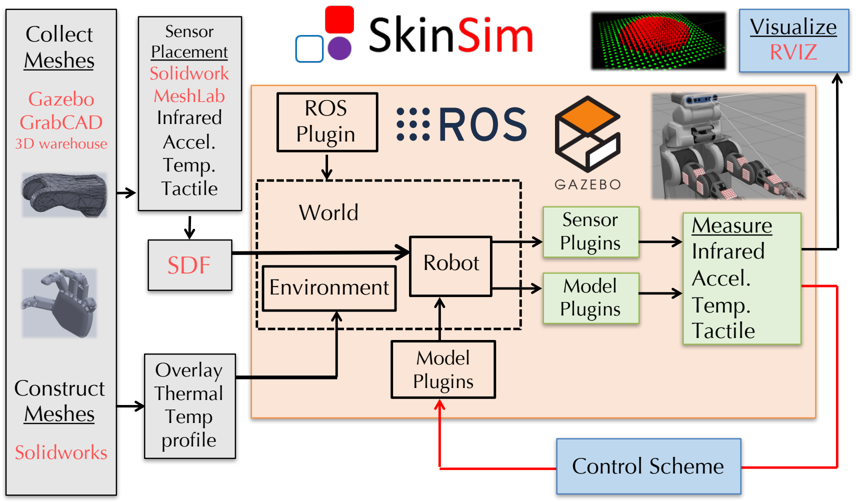


Figure 10.6. The updated data flow diagram for SkinSim 2.0 based on [52].

layer. The new tactile elements only function as sensors and do not require any dynamic modeling, thereby reducing the amount of computations during contact.

The new auto generator creates skin models with a hierarchal model structure. Previously, each skin array model consisted of a single plane (parent) connected to all spheres (children) via prismatic joints. If a parent had too many children, numerical errors in the dynamic solver would introduce instabilities. This is related to how ODE reduces computational complexity by grouping models into "spaces" [195]. In version 2.0, we limit the number of children per parent by introducing so-called subplanes that have no mass and collisions. To make the simulations more stable and numerically robust, the generator also sets parameter such as Constraint Force Mixing (CFM) and Error Reduction Parameter (ERP), described in [195]. In the configuration file, the user can also adjust the physics engine step size and solver iterations to prioritize between accuracy and simulation speed.

The scalability of SkinSim relies on the ODE physics engine. Under the assumption that the number of skin elements in contact with the environment (N_c)

Algorithm 1 Iteration update for Skin plugin

```
1: while simulation is running do
2:     ▷ ODE computes dynamics, contacts
3:     ▷ Reset variables
4:     for each skin element $i$ do
5:         Get $x_i, \dot{x}_i$ from ODE joints
6:         $f_a(i) = kx_i + b\dot{x}_i + m\ddot{x}_i$
7:     for each skin element $i$ do
8:         if $i$ is in collision then
9:             for each $k$ with $D_{ik} <$ threshold do
10:                $f_{\text{dist}} = K_d\,\alpha(D_{ik})\,f_a(k)$
11:                $f_s(i) = f_s(i) + f_{\text{dist}}$
12:    for each tactile element $j$ do
13:        for each $i$ in $I(j)$ do
14:            $F_s(j) = F_s(j) + f_s(i)$
15:        $F_s(j) = F_s(j) + K_n\,\mathcal{N}(0, \sigma_n^2)/\sqrt{N_s}$
16:    ▷ Publish $F_s$
```

is constant, the time complexity of the simulation is $O(N_e)$, where N_e is the total number of skin elements. This is because both the hash table method used for collision detection [195] and the first order Euler integrator are $O(N_e)$. The collision and friction model is based on linear complementarity problem, which is of polynomial time complexity. However, the collision and friction model is only applied on the skin elements in contact with the environment and not on the total number of skin elements. Hence this simulation method is scalable in linear time provided N_c remains the same.

10.4.2 Setup for pHRI Testing

Skin performance testing was conducted for different tactile densities with closed-loop force control. To test influence of different robot skin properties, the performance is evaluated by measuring the response after applying a force with the plunger. The plunger emulates human contact with the robot. Since the skin is attached to a fixed test bed, the feedback from the skin patch is used to control the plunger. This simulates moving the robot limb in response to a human force measured by the skin patch.

Prior to running the experiments, the tactile sensor patches need to be calibrated. To achieve this, a known force F_a is applied to the center of the skin sensor grid using the plunger and a linear calibration constant K_c for the total sensed force can be calculated across all patch tactels according to

$$K_c = \frac{F_{a,ss}}{\sum_{j=1}^{N_t} F_{s,ss}(j)} \tag{10.9}$$

in which the subscript ss represents the steady state measurement. After calibration, an estimated applied force $\hat{F}_a$ is calculated from

$$\hat{F}_a = K_c \sum_{j=1}^{N_t} F_s(j) \tag{10.10}$$

and used in the closed-loop force controller.

SkinSim 2.0 includes different types of closed-loop controllers for pHRI: an impedance [52] and two explicit force controllers (force-based and position-based) [39]. To simulate the influence of measurement time delays, a digital implementation using a zero-order hold was applied to the control signal in-between updates from the sensor patch. In this paper, we describe results based on a digital PI force controller. The plunger control input is given by

$$\begin{aligned} F_c[k] &= 2F_c[k-1] - F_c[k-2] + (K_p + K_i T_s)F_e[k] \\ &\quad - (2K_p + K_i T_s)F_e[k-1] + K_p F_e[k-2] \end{aligned} \tag{10.11}$$

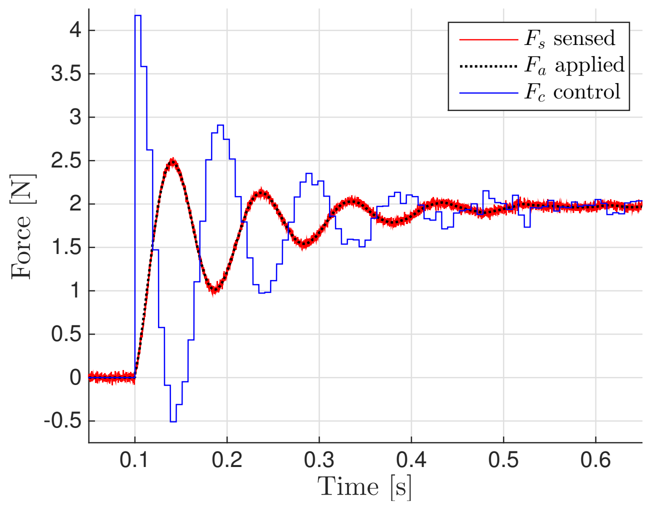


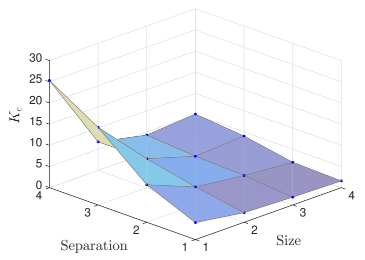
Figure 10.7. Force response from a 24×24 skin patch with an element diameter of 1 mm. Both the tactile size and separation is 3 elements wide.

where K_p is the proportional and K_i the integral gain [39]. The error signal is $F_e = F_d - \hat{F}_a$, given a desired force value F_d. Sampling time T_s is adjusted as in (10.7).

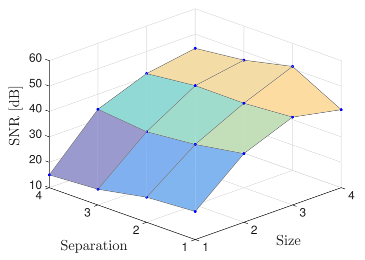
10.5 Simulation Case Study

This section describes simulation results quantifying robot skin performance as a function of sensor resolution. To investigate the influence of scaling, the auto generator was used to perform tests with different tactile sizes and densities. The simulated square skin patches were $5.76\,\mathrm{cm}^2$ in size, with square tactile element sizes between 1 and 4 mm. The skin elements were set to 1 mm diameter size, organized in an array of 24×24 elements. The cylindrical plunger diameter of 1 cm resulted in approximately $N_c = 93$ contact points during testing. The reduced-order dispersion model parameters in Table 10.1 were normalized with respect to the number of skin elements in contact with the plunger,

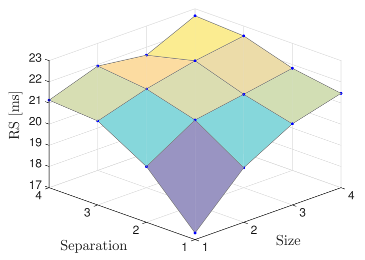
$$k = \frac{k_1}{N_c}, \quad b = \frac{b_1}{N_c} \tag{10.12}$$



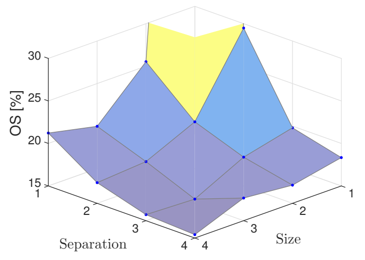
(a) Calibration constant (K_c)



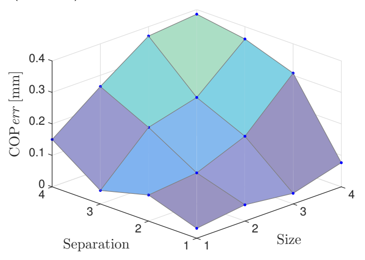
(b) Signal-to-noise ratio (SNR)



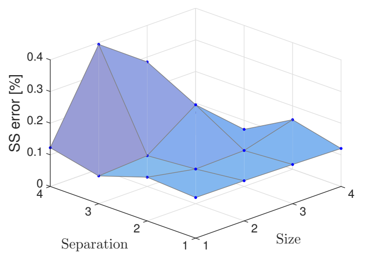
(c) Rise time (RS)



(d) Overshoot (OS)



(e) COP$_{err}$ due to a plunger offset



(f) Steady-state error due to a plunger offset

Figure 10.8. Experiment results for different tactile sizes and separations.

producing a stiffness of 16.37 N/m and damping coefficient of 2.61 N s/m. However, to obtain longer settling times and clearly observe the effects from the parameter variations, the damping coefficient was decreased by a factor of 20 to 0.13 N s/m. The element mass was set to 1 g. A force dispersion scaling factor $K_d = 0.159$, $\mu_d = 0$, and $\sigma_d = 0.01m$ were used in (10.5).

The plunger mass was set to 1 kg with gravity influence turned off and the desired force F_d was set to 2 N. The PI controller gains were fixed at $K_p = 2$ and $K_i = 20$, with a controller update rate given by (10.7) and $K_{\text{data}} = 0.0001$, corresponding to a sampling frequency of 100 Hz for 100 tactels. For high accuracy and stability, the simulation step size was set to 0.1 ms and the physics engine solver iterations between 500 and 1000. Due to the computational complexity, the simulations were not performed in real-time, but at a slower rate of approximately 1:200 (a high-end desktop computer with an Intel Core i7-4790K processor was used). A one-second simulation for the model with 576 skin elements took approximately 220 s depending on the solver iterations.

Both tactile size and separation were varied from 1 to 4 mm (e.g. 1 to 4 skin elements) while keeping the controller settings fixed. The results are summarized in Table 10.2, while a typical force response is shown in Fig. 10.7. The control signal results in the actual force F_a, however the tactile sensors measure F_s multiplied with the calibration constant K_c.

First the auto calibrator was executed, which simulates the experimental testbed and computes K_c according to (10.9). As shown in Fig. 10.8a, the calibration constant increased with tactile separation and decreased with tactile size. Larger K_c values indicate that a smaller tactel area was in contact with the plunger and hence the sensed force was smaller. The consequence of a larger force calibration value is more noise being fed into the controller since the noise is being multiplied with K_c according to (10.10). More importantly, as the tactile size becomes larger, the SNR of each sensor increases due to the scaling by the factor $\sqrt{N_s}$ as explained in Section 10.3.2. This effect can also be seen in the SNR of $\hat{F}_a$, as depicted in Fig. 10.8b, which increases with tactel size. Hence the noise per sensor decreases with increasing tactel size according to both the model in (10.6) and the simulation results. The tactile separation has only a small, indirect effect where large separations increased K_c which in turn influenced the SNR. This behavior can also be seen in Fig. 10.8b. Hence, to minimize measurement noise, the tactel size should be large and the tactel separation plays only a minor role.

The effect of larger numbers of sensors on the controller performance was modeled by increasing the controller sampling time according to (10.7). As the number of tactels increase, and therefore, T_s increases, the controller signal responds slower to changes in the measured force. This results in larger magnitudes of the control signal. Hence, a robot would have to exert larger control efforts to compensate for slow feedback from the skin. If too large, a robot manipulator could start oscillating during pHRI. Hence, larger overshoots are undesirable for force control applications that arise in pHRI. The overshoot values in Fig. 10.8d confirm that as the number of tactels increase, the controller performance degrades. The worst performance (104% overshoot, 17.6ms rise time) occured with tactile size 1 and separation 1, corresponding to 144 tactels. Therefore, if force controller metrics are important design requirements, a better performance is achieved with coarse skin resolutions. It was also observed that more sensors resulted in longer settling times, although the difference was less than 2ms for $N_s \leq 36$ with two exceptions. The configurations with 144 and 64 tactels did not reach the $\pm 0.5\%$ settling threshold within the 1 s simulation time (see Table 10.2). Also, for these two configurations the controller became unstable when adding white Gaussian noise. Hence, no sensor noise was injected into the measurements when evaluating the performance of the controller.

A COP estimation error can be calculated with respect to the applied plunger force

$$\mathrm{COP}_{err} = \mathrm{mean}(||\vec{\mathrm{COP}}_s - \vec{\mathrm{COP}}_a||_2) \tag{10.13}$$

where *err*, *s*, *a* stand for error, sensed, and actual. The maximum COP position error, which was calculated with plunger offsets in x-direction with a size of half the separation, tends to increase with an increase in tactile size or separation, as shown in Fig. 10.8e. Therefore, if COP and resultant force direction is an important design parameter for the robot, the finest resolution skin would yield best COP estimation. From Fig. 10.8f, we can conclude that the steady-state error increases with increasing separation, but there is no identifiable relation to tactel size.

Table 10.2. Experimental results for different skin arrays.*

Sz	Sp	N_s	T_s (ms)	K_c	RT (ms)	ST (ms)	OS (%)	SS (%)	COP$_{err}$ (mm)	SNR (dB)
1	1	144	14.4	4.01	17.6	-	104	0.33	0.31	21.0
1	2	64	6.4	8.85	19.6	-	30	0.46	0.83	19.8
1	3	36	3.6	18.3	21.0	70.7	20	0.28	0.43	16.2
1	4	25	2.5	25.3	21.1	70.9	18	0.12	1.50	15.0
2	1	64	6.4	2.25	19.6	69.7	26	0.13	0.52	37.0
2	2	36	3.6	4.27	21.0	70.9	20	0.18	0.99	33.8
2	3	25	2.5	6.84	21.7	71.3	18	0.46	1.89	32.0
2	4	16	1.6	6.74	21.9	71.3	17	0.85	2.64	34.1
3	1	36	3.6	1.84	20.8	70.7	20	0.15	0.34	44.6
3	2	25	2.5	2.88	21.4	71.0	18	0.15	1.61	43.2
3	3	16	1.6	3.34	22.2	-	15	1.37	2.30	43.4
3	4	16	1.6	4.27	21.6	71.1	18	0.12	3.70	41.3
4	1	25	2.5	1.61	21.4	-	22	0.08	0.77	40.7
4	2	16	1.6	1.94	21.8	71.3	17	0.54	3.07	50.9
4	3	9	0.9	4.04	22.5	71.6	16	0.49	3.61	46.6
4	4	9	0.9	5.14	22.7	71.7	15	0.01	3.85	44.4

*Sz (size), Sp (separation), N_s (number of tactile sensors), T_s (controller update rate), K_c (calibration constant), RT (rise time), ST (settling time), OS (overshoot) SS (steady-state error) COP$_{err}$ (center of pressure position error), and SNR (signal-to-noise ratio).

10.6 Conclusions and Future Work

This paper describes SkinSim 2.0, an open-source simulation environment for multimodal robot skin available at `http://bitbucket.org/nextgensystems/skinsim`. It is a useful tool for designing large tactile sensor arrays, which are necessary for safe pHRI. State-of-the-art robot skin (hardware and software) still face several challenges and development has been slow. New system approaches and a tactile analog to the CMOS optical array could be the breakthrough that robot skin needs. SkinSim is aimed at addressing the design questions in a simulation environment, with considerations of geometric, mechanical, data acquisition, calibration and force control parameters that minimizes time and cost involved with prototyping.

Our approach has the advantage of being able to simulate robot skin in a multi-robot environment as opposed to some other methods such as finite element analysis. Detailed discussion about sensing elements, geometry and mechanical structures, signal quality, and data processing clarify certain requirements for the simulator. SkinSim 2.0 has the flexibility to handle several types of sensors and different types of interactions while maintaining computational efficiency and accuracy. Several updates to the system architecture are presented, in particular the robot skin dynamics and the tactile sensing modeling. The users are also given more control over the model and simulation parameters.

A case study is presented in the paper to demonstrate the usage of SkinSim, which addresses the effects of the skin parameters on the pHRI performance. From these preliminary numerical experiments, we can conclude that on planar surfaces, the highest performance skins from a force control perspective are the coarsest. When the measured center of pressure (COP) and direction of applied force are important, the highest performance skins require the finest resolution, with force control performance tradeoffs.

Future simulator development plans involve integrating new sensor models such as accelerometers, infrared, and thermal sensors. Higher order solver for ODE to improve stability and simulation speed will also be explored. The simulator will provide the modeled noise signal so that noise from the model and numerical errors can be clearly distinguished. Although SkinSim exclusively uses ODE physical engine for the moment, use of other physics engines is a possibility. However, this requires modifications that will be included in a future version of SkinSim. Additional improvements will include generalizing features such as the calculation of COP to non-planar skin patches. Finally, we will deploy whole-body skin patches on simulated 3D robots such as the PR2.

CHAPTER 11

SkinLearn

The SkinLearn software library was created by Ranatunga for "multi-modal skin based Human-Robot Interaction learning, estimation, and control"[1]. Its purpose is to provide base and pHRI functionality as C++ classes that can be used by other applications. My contribution include several ROS based packages for general robot functionality, originally developed in connection with the 2015 Amazon Picking Challenge (APC) [73]. The overall system architecture is shown in Fig. 11.1 and the following are some of the initial software packages:

- *apc_cortex*: Contains state machine nodes for advanced behaviors.
- *apc_msgs*: Defines custom ROS messages and services.
- *apc_robot*: Classes with ROS API for moving the head, base, torso, grippers, and arms.
- *apc_json*: Reads json files and makes the data available via ROS API.
- *apc_marker*: Detects and publishes the pose of QR markers.
- *apc_object_detection*: Interacts and handles data from the Object Recognition Kitchen (ORK).
- *apc_pcl*: Manipulates and filters raw point cloud data.
- *apc_shelf*: Tools for detecting and localizing a shelf and individual bins.
- *apc_vacuum*: Controls vacuum suction cups and reads pressure data via a serial communications channel.
- *apc_simulator*: Launches Gazebo simulation environments for testing.
- *apc_workspace*: Scripts and configuration files for setting up the APC workspace.

[1] `https://bitbucket.org/nextgensystems/skinlearn`

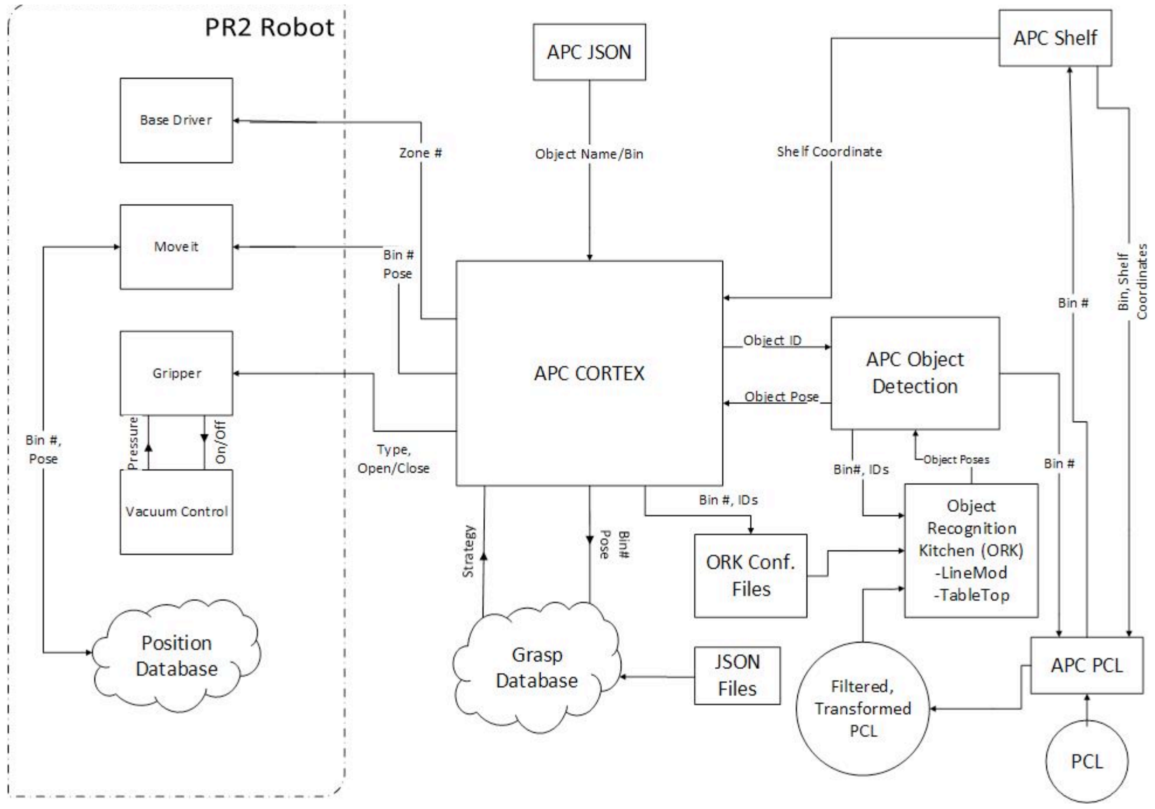


Figure 11.1. The APC architecture for general robot functionalities, which now is part of the SkinLearn library.

Over time, the APC code has evolved naturally into more generalized packages and support several robotic platforms such as Baxter, youBot, and PR2. There is also support for several HMIs, such as tablet teleoperation and physical guidance. The simulation launch files are useful for safe and easy testing of algorithms inside virtual environments.

Another important contribution are the Intelligent Control and Estimation Library (ICE) packages, which contain the controllers and human intent estimators presented in previous chapters. They also include programs for reading tactile sensor data and robot skin calibration. Most of the code was developed for the PR2 real-time controller, but header files, for example containing the neuroadaptive controller, can be imported and used by new applications. The following are some of the packages I have created:

- *ice_robot_controllers*: Contains the neuroadaptive controller as well as inner/outer loop control structures.
- *ice_msgs*: Defines custom ROS messages and services, for example for data collection.
- *ice_experimenter*: Provides a keyboard interface for tuning controllers and conducting pHRI experiments. Communicates with the robot controllers for automatic data collection.
- *ice_sensors*: Tools for reading and visualizing tactile sensor data.

The SkinLearn library also includes my MATLAB scripts for processing ROS topics and bags. To record experiment data from the PR2 real-time loop, it first has to be saved in a memory buffer, then published to a ROS topic, and finally saved to a BAG or CSV file.

SkinLearn will always be in a development stage and continue to evolve over time as NGS members and other researchers add their contributions.

Figure 11.2. The SkinLearn logo.

Part V

Conclusion and Future work

CHAPTER 12

Summary

The research contributions of this thesis encompass the following four topics: (*i*) Investigation and evaluation of state-of-the-art co-robots and their interfaces, (*ii*) validation and expansion of an online, model-free NN controller with stability proof, (*iii*) improvements of pHRI utilizing a two-loop neuroadaptive control architecture, and (*iv*) the development of new open-source software tools for HMIs and pHRI. In the first part, different co-robots and interfaces were tested and the importance of pHRI was demonstrated. Three controller design principles for pHRI led to the main research contributions presented in Parts II and III:

1. *Safe* and stable interaction with neuroadaptive (NA) control.
2. *Intuitive* error dynamics that behave like a simple admittance model using:
    (a) Prescribed Error Dynamics (PED) in the NA controller.
    (b) An inner/outer-loop structure with a prescribed task and admittance model.
3. *Efficient* pHRI by reducing the human effort with a human intent estimator (HIE).

The inner/outer-loop control structure designed in this thesis work allows a high degree of generality and adaptability to different robots, sensors, human preferences, and tasks. In addition, the architecture offers a novel algorithm for adaptive calibration of robot skins by directly tuning admittance models that map voltages into desired robot motion. Several software tools were developed for both simulation and implementation of different HMIs and pHRI, as well as data processing (as described in Part IV). The following sections highlight results and conclusions that emerged from this work, as well as future research directions.

12.1 Co-robots and their Interfaces

In Chapter 2, a Baxter co-robot was compared to a closely priced Denso industrial arm. Their position accuracy and completion times were measured in three different experiments: A point-to-point, writing, and pick-and-place task. Results showed that Baxter is well suited for less precise pick-and-place operations where low position accuracy ($> 2.8\,\mathrm{mm}$) within $1.73\,\mathrm{s}$ is adequate. It can safely handle common household-sized items in semi-structured environments. For higher precision, one still needs industrial robots, but there are tradeoffs. Denso was better suited for applications that require higher position accuracy ($< 1\,\mathrm{mm}$) and velocity ($> 0.3\,\mathrm{m/s}$), but require more complex setup ($> 2.5\,\mathrm{h}$) and safety restrictions. Since co-robots are designed for pHRI, they can simply be programmed by demonstration using physical or kinesthetic teaching.

A youBot was sensorized for increased autonomy and additional HMIs in Chapter 3. A pick-and-place task was used to compare teleoperation and physical interfaces for both expert and novice users. In summary, the physical interaction (mannequin) guidance had similar task completion times as tablet teleoperation, resulted in the most accurate trajectories for both expert and novice users, and was more intuitive (i.e. required less training). The autonomous point-and-click interface had the fastest completion times, but was less accurate and less reliable than user guidance. For future work, testing could be performed with larger user groups and include training time measurements.

It is difficult to make autonomous behavior fully reliable and therefore it is reasonable to include a human operator in the control loop. The motivation for the work presented in Chapter 4 was a task that required a robot pushing a cart through a crowded environment. A co-manipulation system was developed that allows the human to guide the robot and leverages the human's awareness and maneuvering skills. The controller framework consisted of an admittance controller for compliant, stable arm positioning and a velocity controller for smooth base movements. Joystick and physical guidance was compared by asking a user to move a cart together with a PR2 robot in a figure eight pattern. Results showed that physical guidance produced smaller position errors and faster completion times than joystick teleoperation. To summarize, physical HMIs provide a simple, intuitive way for operators to reliably guide a robot. However, the controller gains had to be tuned manually based on user preferences. This demonstrated the need for automatic tuning or adaptive controllers for co-robots, which interact with different types of users in a multitude of tasks.

12.2 Neuroadaptive Control and Inner/Outer-loop Control Structure

Chapter 5 introduces the neuroadaptive (NA) controller and describes its implementation on a PR2 robot, a mobile co-robot with two 6 degrees of freedom and gravity compensated robot arms. NA control was compared against default factory-tuned PID control at different loading conditions and operating speeds, while measuring mean joint tracking errors and control torques, as well as maximum force and impact during collisions. Measurements confirmed that NA control is more accurate at high joint velocities and when lifting objects of unknown mass. These are situations where the robot dynamics becomes highly nonlinear and difficult to model. In addition, the NA controller produced lower control torques and resulted in lower impact forces during collision. Thus, the NA controller is inherently safer for pHRI and applicable to co-robots.

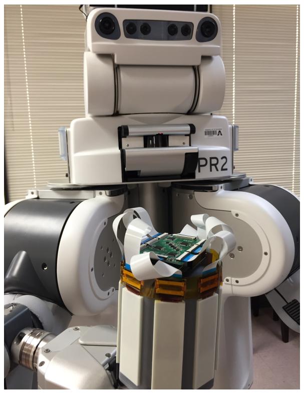
To facilitate intuitive pHRI, the NA controller was augmented with prescribed error dynamics (PED) as presented in Chapter 6. New error terms and time-varying auxiliary matrices were introduced and it was proven that the PED makes the robot behave like a linear, fixed admittance model. This was motivated by the human crossover model: A human adjusts their control characteristics to the dynamics of the controlled element such that the total system remains unchanged. Thus, a highly nonlinear machine like a co-robot becomes easier to interact with, if the dynamic model is simple and constant.

In Chapter 7, intuitive pHRI was achieved utilizing an inner/outer loop control architecture. The robot-specific inner-loop contains the NA controller, which linearizes the robot dynamics. The task-specific outer-loop alters the behavior of the robotic system by modifying the reference trajectory with an admittance model. This model was implemented with an autoregressive moving average (ARMA) filter, which was tuned with recursive least squares based on a prescribed task model and human intent. The approach was validated in several experiments comparing different controller setups. A dimensionless squared jerk measure was used to evaluate the performance and smoothness of the pHRI. The results confirmed that the adaptive admittance controller with human intent estimation reduced jerk in the combined human-robot motion. In addition, the controller framework had the advantage of not needing any offline tuning and being robust to changes in both human and robot dynamics.

The two-loop pHRI controller was further improved with a more advanced human intent estimator (HIE), as presented in Chapter 8. In the previous chapter, the desired human trajectory was estimated using a simple integrator, which does not fully capture human intent. The new approach utilized two NNs in the outer-loop to predict human motion and estimate a reference trajectory that the inner-loop controller follows. The weight tuning laws were derived using Lyapunov stability analysis and it was proven that the error signals and NN weight errors converge. The new controller was implemented on a PR2 robot and its pHRI performance was evaluated by measuring the dimensionless squared jerk, mean tracking error, and mean human and robot effort in terms of force. Three different types of reference trajectories were compared: The actual desired by the human, the estimated from the HIE, and a model trajectory from the ARMA filter presented in Chapter 7. In addition, the inner-loop NA controller was tested with and without prescribed error dynamics. Results confirmed that the novel NA controller combined with the HIE resulted in low motion jerk and was able to reduce the human effort, thereby achieving efficient and intuitive human-robot collaboration, making the controller well suited for co-robots.

In Chapter 9, a novel two-loop architecture was presented for efficient robot skin calibration. The outer-loop contains tuned admittance models that map voltages directly into desired robot motion, which the inner-loop neuroadaptive controller then follows. Despite unknown human and sensor transfer functions, the robot behaves like a linear admittance model and simplifies the pHRI according to the crossover model described in Chapter 6. The methodology was experimentally validated with a PR2 robot holding a sensorized tactile box. Measurements revealed that the admittance filter weights converged relatively quickly and that the overall tracking performance was improved despite the use of cheap, noisy, nonlinear force-sensors.

Future research includes validation with denser arrays of tactile sensors embedded onto larger areas of the robot. My work on tactile skin calibration with second generation robot skin arrays (Fig. 12.1) will be continued by members of the Next Generation Systems (NGS) research group headed by Professor Dan O. Popa. As of this writing, a new Adaptive Robotic Nursing Assistant (ARNA) platform is being developed with more powerful real-time hardware compared to the PR2. This would


(a) The PR2 robot holding the new tactile box.

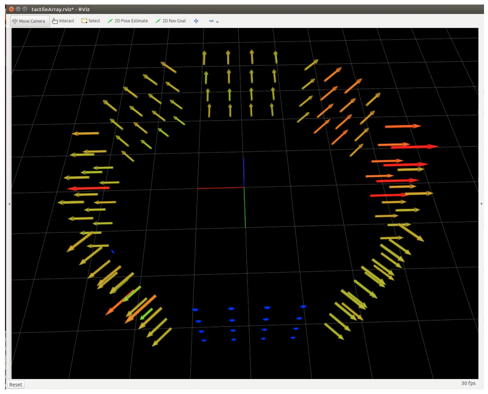

(b) Real-time tactile data visualized in RVIZ.

Figure 12.1. Second generation robot skin mounted on a tactile box. There are 8 skin patches, each consisting of 4×4 individual sensors, which relay force measurements to the robot real-time controller at approximately 150 Hz.

allow pHRI with faster controller update rates and larger neural networks in the inner and outer-loops. The parameters of the NA controller and NN intent estimator could be investigated in more detail, for example by using different weight update laws, number of hidden layers, and activation functions. Future testing could involve more complex tasks (compared to simple point-to-point motions) and include larger human subject groups.

12.3 Developed Software Tools

The robot skin calibrated in Chapter 9 was an early prototype of whole-body sensor arrays that might cover future co-robots. To facilitate the development and characterization of such skins, SkinSim was created as a tool to address the design questions in a simulation environment. It allows adjustments and testing of geometric, mechanical, data acquisition, calibration, and force control parameters, thereby minimizing the time and cost involved with prototyping. Chapter 10 describes a major overhaul of SkinSim, including new modeling approaches and a new simulator architecture. New tactile sensing modeling and tuning of the physics engine has improved stability, allowing simulations of larger skin arrays. The usage of SkinSim was demonstrated in a case study addressing the effects of sensor density (i.e. sensor size and separation) on robot controller performance. Preliminary results show that the highest performance skins from a force control perspective are also the coarsest. If the center of pressure and point of contact are more important, then a finer resolution is needed, but this comes with force control performance tradeoffs.

SkinLearn, briefly described in Chapter 11, implements general robot functionality and advanced controllers for pHRI. The APC packages provide utilities for several HMIs and ROS supported robot platforms. The Intelligent Control and Estimation (ICE) repository includes the NA controller, two-loop control frameworks, and HIE presented in this thesis.

Git version control with public web-based hosting allows for easy sharing of code and coordinated development among multiple people. Therefore, my software has been published in two open-source repositories SkinSim and SkinLearn:

`https://bitbucket.org/nextgensystems/skinsim`

`https://bitbucket.org/nextgensystems/skinlearn`

Development will be continued by members of the NGS research group and potentially by researchers who are interested in the algorithms. The SkinSim simulator roadmap includes new sensor models, higher order solvers for the physics engine to improve stability and simulation speed, and non-planar skin patches. Eventually, whole-body skin patches will be deployed on simulated 3D robots such as the PR2. The NA controller will be continued to be improved and deployed on the Adaptive Robotic Nursing Assistant (ARNA) platform.

REFERENCES

[1] "Through IoT, Japanese Factories Connected Together," *METI Journal*, pp. 4–12, may 2015.

[2] "President Obama Launches Advanced Manufacturing Partnership — whitehouse.gov."

[3] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business and Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[4] D. Gorecky, M. Schmitt, M. Loskyll, and D. Zühlke, "Human-machine-interaction in the industry 4.0 era," in *Proceedings - 2014 12th IEEE International Conference on Industrial Informatics, INDIN 2014*, 2014, pp. 289–294.

[5] **S. Cremer**, K. Doelling, C. L. Lundberg, M. McNair, J. Shin, and D. Popa, "Application requirements for Robotic Nursing Assistants in hospital environments," in *Proc. SPIE*, vol. 9859, 2016, p. 10.

[6] C. Roser, "AllAboutLean.com."

[7] R. Alonzo, **S. Cremer**, F. Mirza, S. Gowda, L. Mastromoro, and D. O. Popa, "Multi-modal sensor and HMI integration with applications in personal robotics," in *Proc. SPIE*, 2015, p. 8.

[8] **S. Cremer**, F. Mirza, Y. Tuladhar, R. Alonzo, A. Hingeley, and D. O. Popa, "Investigation of human-robot interface performance in household environments," in *Proc. SPIE*, vol. 9859, 2016, p. 13.

[9] M. S. Erden and B. Marić, "Assisting manual welding with robot," in *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 4, 2011, pp. 818–828.

[10] A. Safavi and M. H. Zadeh, “Model-Based Haptic Guidance in Surgical Skill Improvement,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, oct 2015, pp. 1104–1109.

[11] A. Cherubini, R. Passama, A. Crosnier, A. Lasnier, and P. Fraisse, “Collaborative manufacturing with physical human-robot interaction,” *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 2016.

[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[13] **S. Cremer**, M. Middleton, and D. O. Popa, “Implementation of advanced manipulation tasks on humanoids through kinesthetic teaching,” in *7th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA ’14*. ACM Press, 2014, pp. 1–6.

[14] H. Yu, S. Huang, G. Chen, Y. Pan, and Z. Guo, “Human-Robot Interaction Control of Rehabilitation Robots with Series Elastic Actuators,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1089–1100, 2015.

[15] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, “An atlas of physical humanrobot interaction,” *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, mar 2008.

[16] B. D. Argall and A. G. Billard, “A survey of Tactile HumanRobot Interactions,” pp. 1159–1176, 2010.

[17] D. Shin, I. Sardellitti, Y.-L. Park, O. Khatib, and M. Cutkosky, “Design and control of a bio-inspired human-friendly robot,” in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, O. Khatib, V. Kumar, and G. Pappas, Eds. Springer Berlin Heidelberg, 2009, vol. 54, pp. 43–52.

[18] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, “Robot Manipulator Control,” *Theory and Practice*, pp. 656–661, 2004.

[19] D. A. White and D. A. Sofge, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptative Approaches*, ser. Vnr Computer Library. Van Nostrand Reinhold, 1992.

[20] R. Zbikowski and A. Dzielinski, *Neural Adaptive Control Technology*, ser. World Scientific series in robotics and intelligent systems, J. Kalkkuhl and K. J. Hunt, Eds. World Scientific, 1996.

[21] Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks*, ser. World Scientific series in robotics and intelligent systems. River Edge, NJ, 1998.

[22] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control Of Robot Manipulators And Non-Linear Systems*, ser. Taylor & Francis systems and control book series. Taylor & Francis, 1999.

[23] J. Si, A. G. Barto, and W. B. Powell, *Handbook of learning and approximate dynamic programming*, ser. IEEE Press Series on Computational Intelligence. IEEE Press, 2004.

[24] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control*, ser. The International Series on Asian Studies in Computer and Information Science. Kluwer, Boston, MA, 2001.

[25] P. J. Werbos, “Neural networks for control and system identification,” in *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*, 1989, pp. 260—-265 vol.1.

[26] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 4–27, 1990.

[27] F. L. Lewis, K. Liu, and A. Yesildirek, “Neural net robot controller with guaranteed tracking performance.” *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 6, no. 3, pp. 703–15, jan 1995.

[28] S. Jagannathan and F. L. Lewis, “Multilayer discrete-time neural-net controller with guaranteed performance,” *Neural Networks, IEEE Transactions on*, vol. 7, no. 1, pp. 107–130, 1996.

[29] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive neural network control of robotic manipulators.* World Scientific.

[30] F.-C. Chen and H. Khalil, “Adaptive control of nonlinear systems using neural networks,” *29th IEEE Conference on Decision and Control*, vol. 55, no. September 2014, pp. 37–41, jun 1990.

[31] M. Polycarpou, “Stable adaptive neural control scheme for nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 41, no. 3, pp. 447–451, mar 1996.

[32] G. Rovithakis and M. Christodoulou, “Adaptive control of unknown plants using dynamical neural networks,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 400–412, mar 1994.

[33] A. Poznyak, Wen Yu, E. Sanchez, and J. Perez, “Nonlinear adaptive trajectory tracking using dynamic neural networks,” *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1402–1411, 1999.

[34] G. Rovithakis, “Performance of a neural adaptive tracking controller for multi-input nonlinear dynamical systems in the presence of additive and multiplicative external disturbances,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 6, pp. 720–730, 2000.

[35] J. E. Colgate and N. Hogan, “Robust control of dynamically interacting systems,” *International Journal of Control*, vol. 48, no. 1, pp. 65–88, 1988.

[36] B. Siciliano and L. Villani, *Robot Force Control.* Boston, MA: Springer US, 1999.

[37] J. J. Craig and M. Raibert, “A systematic method of hybrid position/force control of a manipulator,” in *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International*, 1979, pp. 446–451.

[38] S. Singh and D. Popa, “An analysis of some fundamental problems in adaptive control of force and impedance behavior: theory and experiments,” *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 912–921, 1995.

[39] R. Volpe and P. Khosla, “A theoretical and experimental investigation of explicit force control strategies for manipulators,” *IEEE Transactions on Automatic Control*, vol. 38, no. 11, pp. 1634–1650, 1993.

[40] N. Hogan, “Impedance Control: An Approach to Manipulation: Part ITheory,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, p. 1, 1985.

[41] S. P. Buerger and N. Hogan, “Complementary Stability and Loop Shaping for Improved Human-Robot Interaction,” *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 232–244, 2007.

[42] H. Kazerooni, T. Sheridan, and P. Houpt, “Robust compliant motion for manipulators, part I: The fundamental concepts of compliant motion,” *Robotics and Automation, IEEE Journal of*, vol. 2, no. 2, pp. 83–92, June 1986.

[43] B. C. Kuo and M. F. Golnaraghi, *Automatic control systems.* John Wiley & Sons, 2003.

[44] I. Ranatunga, **S. Cremer**, D. O. Popa, and F. L. Lewis, “Intent aware adaptive admittance control for physical Human-Robot Interaction,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5635–5640.

[45] B. Alqaudi, H. Modares, I. Ranatunga, S. M. Tousif, F. L. Lewis, and D. O. Popa, “Model reference adaptive impedance control for physical human-robot interaction,” *Control Theory and Technology*, vol. 14, no. 1, pp. 68–82, feb 2016.

[46] H. Modares, I. Ranatunga, F. L. Lewis, and D. O. Popa, “Optimized Assistive HumanRobot Interaction Using Reinforcement Learning,” *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 655–667, mar 2016.

[47] **S. Cremer**, I. Ranatunga, and D. O. Popa, “Robotic waiter with physical co-manipulation capabilities,” in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 1153–1158.

[48] **S. Cremer**, L. Mastromoro, and D. O. Popa, “On the performance of the Baxter research robot,” in *IEEE International Symposium on Assembly and Manufacturing (ISAM)*, 2016, pp. 106–111.

[49] I. Ranatunga, **S. Cremer**, F. L. Lewis, and D. O. Popa, “Neuroadaptive control for safe robots in human environments: A case study,” in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 322–327.

[50] **S. Cremer**, D. O. Popa, and F. L. Lewis, “Model-Free Online Neuroadaptive Control with Intent Estimation for Physical Human-Robot Interaction,” *To Be Submitted: IEEE Transactions on Robotics (T-RO)*, 2017.

[51] **S. Cremer**, I. Ranatunga, S. K. Das, I. B. Wijayasinghe, and D. O. Popa, “Neuroadaptive Calibration of Tactile Sensors for Robot Skin,” in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1079–1085.

[52] A. Habib, I. Ranatunga, K. Shook, and D. O. Popa, “SkinSim: A simulation environment for multimodal robot skin,” in *2014 IEEE International Confer-*

ence on Automation Science and Engineering (CASE). IEEE, aug 2014, pp. 1226–1231.

[53] **S. Cremer**, I. B. Wijayasinghe, S. K. Das, and D. O. Popa, "SkinSim 2.0: A Design and Simulation Tool for Robot Skin with Closed-loop pHRI Controllers," *To Be Submitted: IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2017.

[54] I. B. Wijayasinghe, S. Peetha, S. Abubakar, M. N. Saadatzi, **S. Cremer**, and D. O. Popa, "Experimental setup for evaluating an adaptive user interface for teleoperation control," in *Proc. SPIE*, vol. 10216, 2017, p. 10.

[55] **S. Cremer**, D. O. Popa, and N. Bugnariu, "Neuroadaptive pHRI for stroke rehabilitation," in *TBD*, 2017.

[56] J. Sanford, O. Yetkin, **S. Cremer**, and D. Popa, "A novel EMG-free prosthetic interface system using intra-socket force measurement and pinch gestures," in *8th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '15.* ACM Press, 2015, pp. 1–8.

[57] J. Sanford, C. Young, **S. Cremer**, D. Popa, N. Bugnariu, and R. Patterson, "Grip Pressure and Wrist Joint Angle Measurement during Activities of Daily Life," *Procedia Manufacturing*, vol. 3, pp. 1450–1457, 2015.

[58] E. Ackerman, "Heartland Robotics Now Rethink Robotics, Still Developing Mystery Industrial Robot," 2012.

[59] E. Guizzo and E. Ackerman, "The rise of the robot worker," *IEEE Spectrum*, vol. 49, no. 10, pp. 34–41, 2012.

[60] Rethink Robotics, "Baxter Research Robot - Technical Specification Datasheet and Hardware Architecture Overview [Hardware v1.0, SDK v1.0.0]."

[61] E. Guizzo, "Rethink Robotics Opens Up Baxter Robot For Researchers," pp. 2013–05–06, 2013.

[62] “Baxter Research Robot Community - Google Groups.”

[63] M. T. Hoske, “Dual-arm concept robot makes North American debut—Control Engineering,” 2014.

[64] E. Guizzo and T. Deyle, “Robotics Trends for 2012,” 2012.

[65] “Nextage Robot - Technical Data.”

[66] M. Bélanger-Barrette, *ROBOTIQ Collaborative Robot eBook*, 6th ed., 2015.

[67] C. Bernier, “Collaborative Robot Series : Universal Robots,” 2013.

[68] I. 9283:1998, “Manipulating Industrial Robots - Performance Criteria and Related Test Methods,” 1998.

[69] P. I. Corke, “A robotics toolbox for MATLAB,” *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.

[70] Z. Ju, C. Yang, and H. Ma, “Kinematics modeling and experimental verification of baxter robot,” in *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, 2014, pp. 8518–8523.

[71] “Denso VS-6577GM Data Specifications.”

[72] S. Natarajan, “Motion Planning for Manipulator &#8211; MoveIt!, Tech. Rep. 14-03, 2014.

[73] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, “Lessons from the Amazon Picking Challenge,” *Arxiv*, vol. 6, no. 1, pp. 1–14, 2016.

[74] “Amazon Picking Challenge Overview.”

[75] K. Goldberg, “Editorial,” *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, 2012.

[76] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, “Multimodal templates for real-time detection of texture-less ob-

jects in heavily cluttered scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 858–865.

[77] Developers Page, "Object Recognition Kitchen."

[78] P. Kopacek, "Development trends in robotics," *Elektrotechnik und Informationstechnik*, vol. 130, no. 2, pp. 42–47, 2013.

[79] H. Ding, M. Schipper, and B. Matthias, "Optimized task distribution for industrial assembly in mixed human-robot environments - Case study on IO module assembly," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, 2014, pp. 19–24.

[80] K. Dautenhahn and J. Saunders, *New Frontiers in Human-robot Interaction.* Philadelphia: John Benjamins Publishing Company, 2011.

[81] G. Herrmann and C. Melhuish, "Towards Safety in Human Robot Interaction," *International Journal of Social Robotics*, vol. 2, no. 3, pp. 217–219, may 2010.

[82] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, "Robot Manipulator Control," *Theory and Practice*, pp. 656–661, 2004.

[83] I. Ranatunga, **S. Cremer**, F. L. Lewis, and D. O. Popa, "Neuroadaptive Control for Safe Robots in Human Environments: A Case Study," in *Automation Science and Engineering (CASE), 2015 IEEE International Conference on.* IEEE, 2015.

[84] C. Nothnagle, J. R. Baptist, J. Sanford, W. H. Lee, D. O. Popa, and M. B. J. Wijesundara, "EHD printing of PEDOT: PSS inks for fabricating pressure and strain sensor arrays on flexible substrates," in *Next-Generation Robotics II; and Machine Intelligence and Bio-inspired Computation: Theory and Applications IX*, vol. 9494, 2015, p. 949403.

[85] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 2008, vol. 1.

[86] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “RGB-D Mapping : Using Depth Cameras for Dense 3D Modeling of Indoor Environments,” *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[87] C. Fitzgerald, “Developing baxter,” in *IEEE Conference on Technologies for Practical Robot Applications, TePRA*, 2013.

[88] “KUKA youBot User Manual,” 2012.

[89] “NI roboRIO RIO device for Robotics,” 2015.

[90] E. Marder-Eppstein, “Navigation,” 2016.

[91] “Setup and Configuration of the Navigation Stack on a Robot,” 2015.

[92] B. Gerkey, “SLAM GMapping,” 2010.

[93] “Ecto - A C++/Python Computation Graph Framework,” 2011.

[94] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[95] “D6T MEMS Thermal Sensors,” 2012.

[96] A. W. Black, P. Taylor, and R. Caley, “The Festival Speech Synthesis System,” p. 154, 1997.

[97] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky, “Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices,” in *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, vol. 1, 2006, pp. I–185–I–188.

[98] N. Villaroman, D. Rowe, and B. Swan, “Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor,” *Proceedings of the 2011 conference on Information technology education - SIGITE ’11*, p. 227, 2011.

[99] J. Jackson, “Microsoft robotics studio: A technical introduction,” *IEEE Robotics and Automation Magazine*, vol. 14, no. 4, pp. 82–87, 2007.

[100] “MRPT Empowering C++ development in robotics.”

[101] “Rock - the Robot Construction Kit.”

[102] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[103] “rosjava - ROS Wiki.”

[104] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, “Collision detection and reaction: A contribution to safe physical Human-Robot Interaction,” in *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on.* IEEE, Sept. 2008, pp. 3356–3363.

[105] C. Kemp, A. Edsinger, and E. Torres-Jara, “Challenges for robot manipulation in human environments [grand challenges of robotics],” *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 20 –29, march 2007.

[106] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida, “Real-time replanning using 3D environment for humanoid robot,” in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, 2011, pp. 584–589.

[107] R. B. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz, “Perception for mobile manipulation and grasping using active stereo,” in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, 2009, pp. 632–638.

[108] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer, “Towards autonomous robotic butlers: Lessons learned with the PR2,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 5568–5575.

[109] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal, "Robots in Human Environments," in *in Proceedings of the 1st Workshop on Robot Motion and Control*. Ieee, 1999, pp. 213–221.

[110] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control: Theory and Practice*, ser. Control Engineering. Marcel Dekker, 2004.

[111] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[112] S. D. Eppinger and W. P. Seering, "On dynamic models of robot force control," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, 1986, pp. 29–34.

[113] N. Hogan, "Impedance control part1-3," *Transaction of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 1–24, 1985.

[114] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[115] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Transaction of the ASME*, vol. 64, pp. 759–768, 1942.

[116] D. Fox, "KLD-sampling: Adaptive particle filters," in *NIPS*, 2001, pp. 713–720.

[117] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The Office Marathon: Robust Navigation in an Indoor Office Environment," in *International Conference on Robotics and Automation*, 2010.

[118] "Skyball XI," \url{http://www.skyballinfo.com}.

[119] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "The role of the robot mass and velocity in physical human-robot interaction - Part I: Non-constrained blunt impacts," in *Robotics and Automation (ICRA), 2008 IEEE International Conference on*, 2008, pp. 1331–1338.

[120] S. Haddadin, A. Albu-Schäffer, M. Frommberger, and G. Hirzinger, "The role of the robot mass and velocity in physical human-robot interaction - Part II: Constrained blunt impacts," in *Robotics and Automation (ICRA), 2008 IEEE International Conference on*, no. 011838. IEEE, May 2008, pp. 1339–1345.

[121] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Soft-Tissue Injury in Robotics," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 3426–3433.

[122] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, May 2008, pp. 2165–2170.

[123] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[124] O. Sigaud, C. Salaün, and V. Padois, "On-line Regression Algorithms for Learning Mechanical Models of Robots: A Survey," *Robot. Auton. Syst.*, vol. 59, no. 12, pp. 1115–1129, Dec. 2011.

[125] F. L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance," *Neural Networks, IEEE Transactions on*, vol. 6, no. 3, pp. 703–715, 1995.

[126] K. G. Vamvoudakis, F. Lewis, and S. S. Ge, "Neural Networks in Feedback Control Systems," in *Mechanical Engineers' Handbook*. John Wiley & Sons, Inc., 2014, ch. 23.

[127] G. M. Atmeh, I. Ranatunga, D. Popa, K. Subbarao, F. Lewis, and P. Rowe, "Implementation of an Adaptive, Model Free, Learning Controller on the Atlas Robot *," in *American Control Conference (ACC), 2014*, 2014.

[128] D. T. McRuer, “Mathematical Models of Human Pilot Behavior,” *Aerospace*, no. January, p. 67, 1974.

[129] S. Suzuki and K. Furuta, “Adaptive Impedance Control to Enhance Human Skill on a Haptic Interface System,” *Journal of Control Science and Engineering*, vol. 2012, pp. 1–10, 2012.

[130] Y. Li, S. Sam Ge, and C. Yang, “Learning impedance control for physical robotenvironment interaction,” *International Journal of Control*, vol. 85, no. 2, pp. 182–193, feb 2012.

[131] M. S. Erden and A. Billard, “End-point Impedance Measurements at Human Hand during Interactive Manual Welding with Robot,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 126–133.

[132] B. Llorens-Bonilla and H. H. Asada, “A Robot on the Shoulder : Coordinated Human - Wearable Robot Control using Coloured Petri Nets and Partial Least Squares Predictions,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 119–125.

[133] T. Tsuji and Y. Tanaka, “Tracking Control Properties of HumanRobotic Systems Based on Impedance Control,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 4, pp. 523–535, jul 2005.

[134] C. Ott, R. Mukherjee, and Y. Nakamura, “Unified Impedance and Admittance Control,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 554–561.

[135] F. Ficuciello, A. Romano, L. Villani, and B. Siciliano, “Cartesian Impedance Control of Redundant Manipulators for Human-Robot Co-Manipulation,” in *IEEE/RSJ International Conference on Intelligent Robots and System*, 2014, pp. 2120–2125.

[136] E. Gribovskaya, A. Kheddar, and A. Billard, “Motion learning and adaptive impedance for robot control during physical interaction with humans,” in *2011 IEEE International Conference on Robotics and Automation.* IEEE, May 2011, pp. 4326–4332.

[137] Y. Li and S. S. Ge, “HumanRobot Collaboration Based on Motion Intention Estimation,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 1007–1014, 2014.

[138] R. Ikeum, T. Moriguchi, and K. Mizutani, “Optimal Variable Impedance Control for a Robot and Its Application To Lifting an Object with a Human,” in *Proceedings of the 2002 IEEE Int. Workshop an Robot and Human Interactive Communication*, 2002, pp. 500–505.

[139] C. Mitsantisuk, K. Ohishi, and S. Katsura, “Variable mechanical stiffness control based on human stiffness estimation,” in *2011 IEEE International Conference on Mechatronics*, Apr. 2011, pp. 731–736.

[140] F. Dimeas and N. Aspragathos, “Fuzzy Learning Variable Admittance Control for Human-Robot Cooperation,” in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on.* IEEE Press, 2014, pp. 4770–4775.

[141] A. Lecours, B. Mayer-St-Onge, and C. Gosselin, “Variable admittance control of a four-degree-of-freedom intelligent assist device,” in *2012 IEEE International Conference on Robotics and Automation*, no. 2, May 2012, pp. 3903–3908.

[142] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation,” *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43–53, feb 1987.

[143] B. Widrow and E. Walach, *Adaptive Inverse Control, Reissue Edition: A Signal Processing Approach*, ser. IEEE Press series on power engineering. Wiley, 2008.

[144] Matthias Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 464–469.

[145] S. Suzuki and K. Furuta, "Adaptive Impedance Control to Enhance Human Skill on a Haptic Interface System," *Journal of Control Science and Engineering*, vol. 2012, no. 2, 2012.

[146] N. Hogan and D. Sternad, "Sensitivity of smoothness measures to movement duration, amplitude, and arrests." *Journal of Motor Behavior*, vol. 41, no. 6, pp. 529–34, nov 2009.

[147] A. Pervez and J. Ryu, "Safe physical human robot interaction-past, present and future," *Journal of Mechanical Science and Technology*, vol. 22, no. 3, pp. 469–483, may 2008.

[148] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in *2011 IEEE International Conference on Robotics and Automation.* IEEE, may 2011, pp. 4326–4332.

[149] Yanan Li and Shuzhi Sam Ge, "Impedance Learning for Robots Interacting With Unknown Environments," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1422–1432, jul 2014.

[150] Z. Wang, A. Peer, and M. Buss, "An HMM approach to realistic haptic human-robot interaction," in *World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems.* IEEE, mar 2009, pp. 374–379.

[151] J. R. Medina, S. Endo, and S. Hirche, "Impedance-based Gaussian Processes for predicting human behavior during physical interaction," in *2016 IEEE In-*

ternational Conference on Robotics and Automation (ICRA), may 2016, pp. 3055–3061.

[152] D. W. Franklin, E. Burdet, K. P. Tee, R. Osu, C.-M. Chew, T. E. Milner, and M. Kawato, "CNS learns stable, accurate, and efficient movements using a simple algorithm." *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 28, no. 44, pp. 11 165–11 173, 2008.

[153] I. Ranatunga, F. L. Lewis, D. O. Popa, and S. M. Tousif, "Adaptive Admittance Control for Human-Robot Interaction Using Model Reference Design and Adaptive Inverse Filtering," *IEEE Transactions on Control Systems Technology*, pp. 1–8, 2016.

[154] F. L. Lewis, S. Jagannathan, and A. Yesildirak, *Neural Network Control Of Robot Manipulators And Nonlinear Systems.* CRC Press, 1998.

[155] G. Guennebaud and B. Jacob, "Eigen v3," 2010.

[156] C. Breazeal and B. Scassellati, "Challenges in building robots that imitate people," *Imitation in animals and artifacts*, no. 1998, pp. 363–390, 2002.

[157] R. Dahiya, "Electronic skin," *Proceedings of the 2015 18th AISEM Annual Conference, AISEM 2015*, pp. 3–6, 2015.

[158] L. Maiolo, A. Pecora, F. Maita, A. Minotti, E. Zampetti, S. Pantalei, A. Macagnano, A. Bearzotti, D. Ricci, and G. Fortunato, "Flexible sensing systems based on polysilicon thin film transistors technology," *Sensors and Actuators, B: Chemical*, vol. 179, pp. 114–124, 2013.

[159] P. Mittendorfer, E. Dean, and G. Cheng, "3D spatial self-organization of a modular artificial skin," *IEEE International Conference on Intelligent Robots and Systems*, no. Iros, pp. 3969–3974, 2014.

[160] P. Maiolino, M. Maggiali, G. Cannata, G. Metta, and L. Natale, "A flexible and robust large scale capacitive tactile system for robots," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3910–3917, 2013.

[161] P. M. Grice, M. D. Killpack, A. Jain, S. Vaish, J. Hawke, and C. C. Kemp, "Whole-arm tactile sensing for beneficial and acceptable contact during robotic assistance." *IEEE International Conference on Rehabilitation Robotics*, vol. 2013, p. 6650464, jun 2013.

[162] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing-from humans to humanoids," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010.

[163] S. Suzuki and K. Furuta, "Adaptive impedance control to enhance human skill on a haptic interface system," *Journal of Control Science and Engineering*, vol. 2012, 2012.

[164] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural network control of robot manipulators and non-linear systems.* CRC Press, 1999.

[165] F. L. Lewis, A. Yegildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 7, no. 2, pp. 388–99, jan 1996.

[166] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky, "Directions toward effective utilization of tactile skin: A review," *IEEE Sensors Journal*, vol. 13, no. 11, pp. 4121–4138, 2013.

[167] G. Cannata, S. Denei, and F. Mastrogiovanni, "Towards automated self-calibration of robot skin," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4849–4854, 2010.

[168] A. Del Prete, S. Denei, L. Natale, F. Mastrogiovanni, F. Nori, G. Cannata, and G. Metta, "Skin spatial calibration using force/torque measurements," *IEEE*

International Conference on Intelligent Robots and Systems, pp. 3694–3700, 2011.

[169] C. Ciliberto, L. Fiorio, M. Maggiali, L. Natale, L. Rosasco, G. Metta, G. Sandini, and F. Nori, "Exploiting global force torque measurements for local compliance estimation in tactile arrays," *IEEE International Conference on Intelligent Robots and Systems*, no. Iros, pp. 3994–3999, 2014.

[170] A. Cirillo, F. Ficuciello, C. Natale, S. Pirozzi, and L. Villani, "A Conformable Force/Tactile Skin for Physical Human-Robot Interaction," *Robotics and Automation Letters, IEEE*, vol. 1, no. 1, pp. 41–48, 2016.

[171] C. Lebosse, P. Renaud, B. Bayle, and M. de Mathelin, "Modeling and Evaluation of Low-Cost Force Sensors," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 815–822, aug 2011.

[172] J. O'Neill, J. Lu, R. Dockter, and T. Kowalewski, "Practical, stretchable smart skin sensors for contact-aware robots in safe and collaborative interactions," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 624–629, 2015.

[173] D. Kragic, S. Crinier, D. Brunn, and H. Christensen, "Vision and tactile sensing for real world tasks," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1. IEEE, 2003, pp. 1545–1550.

[174] M. Lee and H. Nicholls, "Review Article Tactile sensing for mechatronicsa state of the art survey," *Mechatronics*, vol. 9, no. 1, pp. 1–31, feb 1999.

[175] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*. IEEE, oct 2010, pp. 1228–1235.

[176] R. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing - From Humans to Humanoids," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, feb 2010.

[177] J.-J. Cabibihan, S. Pattofatto, M. Jomâa, A. Benallal, and M. C. Carrozza, "Towards Humanlike Social Touch for Sociable Robotics and Prosthetics: Comparisons on the Compliance, Conformance and Hysteresis of Synthetic and Human Fingertip Skins," *International Journal of Social Robotics*, vol. 1, no. 1, pp. 29–40, 2009.

[178] J. Jackson, "Microsoft robotics studio: A technical introduction," *IEEE Robotics & Automation Magazine*, vol. 14, no. 4, pp. 82–87, dec 2007.

[179] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: MORSE," in *2011 IEEE International Conference on Robotics and Automation.* IEEE, may 2011, pp. 46–51.

[180] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, nov 2013, pp. 1321–1326.

[181] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[182] J. A. Joergensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim - an Open Simulator for Sensor based Grasping," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8, 2010.

[183] R. Diankov, "Automated Construction of Robotic Manipulation Programs," pp. 1–263, 2010.

[184] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proceedings of the 1995 symposium on Interactive 3D graphics - SI3D '95.* New York, New York, USA: ACM Press, apr 1995, pp. 181–ff.

[185] M. P. Ashley-Rollman, P. Pillai, and M. L. Goodstein, "Simulating multi-million-robot ensembles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1006–1013.

[186] B. Hollis, S. Patterson, and J. Trinkle, "Compressed Sensing for Tactile Skins," 2016.

[187] K. Shook, "Experimental testbed for robotic skin characterization and interaction control," Ph.D. dissertation, The University of Texas at Arlington, 2014.

[188] G. Cannata, M. Maggiali, G. Metta, and G. Sandini, "An embedded artificial skin for humanoid robots," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.* IEEE, aug 2008, pp. 434–438.

[189] M.-Y. Cheng, C.-M. Tsao, and Y.-J. Yang, "An anthropomorphic robotic skin using highly twistable tactile sensing array," in *2010 5th IEEE Conference on Industrial Electronics and Applications.* IEEE, jun 2010, pp. 650–655.

[190] M. Strohmayr, H. Worn, and G. Hirzinger, "The DLR artificial skin step I: Uniting sensitivity and collision tolerance," in *2013 IEEE International Conference on Robotics and Automation.* IEEE, may 2013, pp. 1012–1018.

[191] K. R. Shook, A. Habib, W. H. Lee, and D. O. Popa, "Experimental testbed for robot skin characterization and interaction control," vol. 9116, p. 911606, 2014.

[192] F. Bordoni and A. D'Amico, "Noise in sensors," *Sensors and Actuators A: Physical*, vol. 21, no. 1-3, pp. 17–24, feb 1990.

[193] B. Bae, B. R. Flachsbart, K. Park, and M. a. Shannon, "Design optimization of a piezoresistive pressure sensor considering the output signal-to-noise ratio,"

Journal of Micromechanics and Microengineering, vol. 14, no. 12, pp. 1597–1607, 2004.

[194] Y. Tenzer, L. P. Jentoft, and R. D. Howe, "The Feel of MEMS Barometers: Inexpensive and Easily Customized Tactile Array Sensors," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 89–95, sep 2014.

[195] R. Smith, "Open Dynamics Engine v0.5 User Guide," *Retrieved December*, vol. 5, 2006.

BIOGRAPHICAL STATEMENT

Sven Cremer was born near Köln, Germany and grew up in Göteborg, Sweden. He received his B.Sc. in Engineering Physics and Applied Mathematics from the University of the Pacific, California, in 2010. His passion for robotics led him to purse a doctorate degree in Electrical Engineering with a focus on control systems at the University of Texas at Arlington (UTA), which he received in 2017. At UTA, he conducted research in the Next Generation Systems (NGS) group headed by Professor Dan O. Popa and worked on projects involving human-machine interfaces, assistive robotics, neuroadaptive control, and robot skin.