Enabling Close Proximity Human Robot Collaboration via Distributed, Self-Calibrating Robotic Skin

by

Matthew Strong

A thesis submitted to the

Faculty of the Computer Science of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Bachelor's of Science

Department of Computer Science

2021

Committee Members:

Alessandro Roncone, Chair

Bradley Hayes

Christoffer Heckman

Strong, Matthew (B.S., Computer Science)

Enabling Close Proximity Human Robot Collaboration via Distributed, Self-Calibrating Robotic Skin

Thesis directed by Prof. Alessandro Roncone

Physical human-robot interaction is slowly transitioning to more human-centric environments, where robots will need to work with and around humans, instead of the other way around. However, prior methods are limited in that they struggle with effective, close-proximity interaction. Distributed, whole body sensing has recently emerged as a strong solution for this problem; such sensing enables comprehensive coverage of the robot's nearby space. However, current solutions lack a certain degree a flexibility and modularity that is necessary for eventual real-world development. This thesis first addresses the lack of a "plug and play" system with the presentation of a detailed framework for calibrating and setting up the sensor units. The proposed framework is accurate and we demonstrate this in simulation and real-life, followed with an obstacle avoidance example that demonstrates the system's capabilities. Next, this work addresses the lack of a transition between avoidance and contact in close-proximity interactions with a new framework for implicit contact *anticipation*. This framework, which builds on top of the previous one, is also evaluated in a variety of experimental scenarios to showcase its capabilities. Altogether, this work is a strong step forwards to close-proximity human robot interaction, and will hopefully push research closer to true human robot collaboration.

Dedication

To my little (but taller) brother, Andy, who I hope will enjoy his time in CS at CU Boulder as much as I did. Your accomplishments and successes mean more to me than you will ever know.

Acknowledgements

I personally consider myself to be one of the luckiest students at CU because of the remarkable support I've had throughout the years. First, thank you Professor Alessandro Roncone for being an excellent advisor and providing honest and critical feedback in the work I do. Your advising has led to me achieving things I never thought possible. Also good job on convincing me to really enjoy the work in the lab – I was convinced I would love drones, but now, I never want to do drone research. Also, thank you to the committee which read over this thesis. I am proud to tell others that I am a part of CU Robotics. Thank you Chi-Ju Wu for believing in me when I first joined the lab, and telling me that I have a bright future, even when I felt discouraged. Thank you Caleb Escobedo for really "getting" me and my madness. Thank you Yunyang Ye and Yingli Lou, my first research mentors, who inspired me to keep going and to minor in the beautiful language of Chinese. Thank you Wen Cai for being a great friend and mentor in Chinese. Finally, thank you Kandai Watanabe for being one of the greatest friends I've ever had and fully solidifying my goal of eventually pursuing a Ph.D.

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

While robotics is increasingly gaining attention among the general public and media – with people frequently commenting on the soon, incoming "robotic takeover" – the current state-of-the-art research is far behind that current expectation. Even though robotics have been successfully deployed at large scales in industrial and manufacturing environments, the caveat is that the environments must be tailored towards safe robot operation. This means that people currently need to focus on giving the robot ample space to easily operate, as well as ensuring that there will be minimal changes in the robot's operating space. Indeed, while these robots have been commercially successful in the manufacturing setting, many assumptions can be made so that the robot does not need to react to external disturbances in its environment. As a result, the long touted promise that robotics will soon be in close contact and collaboration with humans has still not been met, given the current capabilities. In short, at present, humans have to work around the limitations of robots, instead of the other way around, where robots should work around and with humans.

As the shift from manufacturing and laboratory environments to areas conducive to interaction occurs, it becomes essential that robots have the capabilities for close and extended interactions [1]. By themselves, current collaborative robotic platforms are practically unaware of their surroundings and how to react to unpredictable situations. As robots begin to share areas with humans, whether it be work or living, strong perception capabilities are of the utmost importance.

Recent research on modern robot manipulators has demonstrated a variety of dynamic behaviors when humans enter a robot's workspace; however, the proposed perception methods and algorithms do not achieve the eventual goal of **close proximity interaction**. This additionally goes against another important goal of physical human-robot interaction: to approach the paradigm of *human-human interaction.* In human-human environments, such as cooking, construction, and medical scenarios, close-proximity collaboration is essential for successful completion of complex tasks. Without the ability for these close-proximity interactions, the rate at which robots will be able to become more present in environments with more humans is reduced. In all, research should specifically target pushing human-robot interaction to closely resemble human-human interaction, which is obviously already omnipresent in people's lives.

Accordingly, the first step to achieving close proximity interaction is accurate and fast nearby-space perception of the robot. However, collaborative robots designed to work with humans frequently resort to either external, sparse, and high resolution sensing (such as the work in [2]), which has a low bandwidth and suffers from the occlusion problem, or low-resolution, onboard contact-based sensors (e.g., torque sensing on the joints as in [3] or on the robot's end-effector), which do not take the robot's whole body into account **and** cannot sense objects and humans in its nearby space. From the above, it can be seen that whole body awareness is the most suitable solution; with recent works performing research in the area ([4–9]).

With respect to achieving whole body awareness, *distributed, onboard sensor units* are a strong candidate; they have the potential to enable a information-rich perception of the robot's nearby space with high bandwidth. Currently, existing sensor (skin) units require a considerable amount of time to setup and deploy the hardware on a real platform, are tailored for specific robotic platforms, and require manual, time-consuming calibration. With these three issues, the net result is limiting research in the field, and slowing down real-world deployment.

Overall, the core contribution of this thesis is that **distributed, whole-body sensing** enables rich, complex interactions with humans in the nearby space of a robot manipulator.

The first major contribution of this thesis consists of the creation of a comprehensive framework, calibration, and robot control for close interactions with humans. This unified framework first proposes an accurate and self-contained kinematic calibration algorithm that locates the sixdimensional pose (position and orientation) of the sensor units on the robot's body, only relying on accelerometer data from the novel sensor units that are self-contained, self-powered, and capable of close-proximity sensing. The proposed method is validated in simulation before being deployed successfully in an avoidance controller. The combination of a novel hardware design, a sub-centimeter calibration method, and strong first application demonstrate a plug-and-play that can be easily setup and deployed on real robot platforms.

The second major contribution of this thesis entails building on top of the aforementioned framework to use the calibrated sensor units to enable new collaborative behavior. We first emphasize how in human-robot interaction (HRI), resorting to pure collision avoidance is not completely sufficient, as contact is frequently inevitable and encouraged. Allowing avoidance, contact, and a strong transition in between the two is critical for widening the spectrum of capabilities of truly close physical human robot interaction. Motivated by this, we posit that the previous framework's ability of close-proximity operation allows us to bridge the gap between avoidance and contact. This new ability, which we refer to as *contact anticipation*, reduces the force to trigger a reactive behavior, in turn leading to softer and safer interactions between robots and humans.

To enable this behavior, we first formulate the problem as a quadratic programming optimization problem that allows for collision avoidance with both reduced velocities and relaxed avoidance constraints to better encourage contact. Next, a novel contact detection and reaction scheme is devised that registers anomalies in noisy external force data and leverages the sensor units' environmental information to become sensitive to touch. This new contact anticipation framework is experimentally evaluated on a real robot in a handful of scenarios, demonstrating the multiple capabilities and advantages of our method. Altogether, our results demonstrate that onboard sensing and dynamic contact thresholding allow a smooth transition between avoidance and contact.

Thus, to summarize, the two real-world system/frameworks introduce a novel sensor unit and its applications which will bring us one step closer to ubiquitous and safe human–robot interactions.

1.1 Contributed Work

The work described in this thesis has resulted in the following Under-Review publications. As a result, most of the figures (and corresponding captions) of these works have been reproduced here in their entirety. Also, due to the extensive involvement of the author in writing both papers, the contents and structure of this thesis and the two works are similar, but with extensive elaboration on numerous important points in this thesis from the below works. The thesis is accordingly organized by detailing the work from the first paper, and then detailing the work from the second paper.

Kandai Watanabe, **Matthew Strong**, Mary West, Caleb Escobedo, Ander Aramburu, Krishna Chaitanya, Alessandro Roncone. Self-Contained Kinematic Calibration of a Novel Whole-Body Artificial Skin for Collaborative Robotics, *International Conference on Robotics and Systems 2021.* Under Review.

Caleb Escobedo, **Matthew Strong**, Mary West, Ander Aramburu, Alessandro Roncone. Bridging the Gap Between Collision Avoidance and Contact in Robot Manipulators with Onboard Sensing, *International Conference on Robotics and Systems 2021.* Under Review.

Chapter 2

Plug and Play Robotic Skin: Related Work

This literature review focuses on the relevant subfields pertaining towards the first core contribution of this thesis, which is the plug-and-play robotic skin system.

2.1 Kinematic Calibration

Our novel custom built sensor units, which are introduced in detail in the next chapter, first require accurate calibration of the sensor units along *any* point on a robot's body. The issue of kinematic calibration of a robot's kinematic chain is a well-studied problem for robots, both industrial and collaborative. Accurate and precise modeling of a chain directly impacts the robot's performance and operation. This problem has been very well-studied in previous decades, and open-loop and closed-loop approaches have arisen from this research ([10] provides a more detailed overview of these concepts) as well. However, an important distinction needs to be made from this work in that the purpose of a robot's kinematic calibration is to calibrate its kinematic chain, which improves on an already available kinematic model. This improves convergence of optimization (when the available kinematic model is used as a prior) and the overall accuracy of the results. Moreover, kinematic calibration of a robot is the exact same for the same robot model (e.g., the Franka Panda or Sawyer robot arm). In contrast, this work is concerned with the kinematic calibration of the artificial sensor units that can be placed arbitrarily on the surface of a robot.

Also of relevance to this work is the research performed of robotic kinematic calibration based on IMU data (which is the type of data that this work relies on for calibration) in [11], which slightly overlaps with human pose estimation for motion capture ([12], [13]). However, in addition to addressing completely different problems (the calibration of a robot's or human's already existing kinematic chain vs. the calibration of an arbitrarily placed sensor unit), these prior approaches frequently rely on sensor fusion techniques that bring together information from multiple different sensors, such as magnetometers, GPS, bio-mechanical modeling, etc. Our approach only relies on accelerometer data.

2.2 Artificial Skin/Sensor Units for Robotics

Recently, there has been a concentrated and increased effort to improve technologies in distributed and artificial sensing on robotic manipulators. A great deal of work in this area focuses on pressure and touch onboard sensors (which often provide manipulation and gripping tasks with more fine-grained sensing, see [14–16] for more information). At the same time, the idea of covering a robot's whole body with distributed sensing is gaining traction ([6–8, 17]).

The developed sensor units for this work can be distributed over a whole robot's body and, unlike, the touch-based methods, are focused on the perception of the nearby space of the robot. These sensor units are also self-contained, and their low cost makes them a significantly better option than imu-based or multi-sensor-based capture systems, such as Xsens, which costs thousands of dollars.

2.3 Kinematic Calibration of Artificial Skins

Accurate and precise kinematic calibration of robotic artificial skins is crucial for a robot to accurately detect humans and objects near any part of the robot's body. Thus, calculating the exact pose of a skin unit relative to the robot's kinematic model is of utmost importance.

To reiterate, this section is distinctly different from Kinematic Calibration in that this section focuses on methods for calibrating a sensor unit on a robot, not calibrating the robot itself. Only recently has kinematic calibration of a skin, which was originally manually performed, been started to be automated without the need of a human operator. A fairly recent approach is that of [18], which proposed an automatic calibration method in which the iCub humanoid robot [8] performed “self-touch” actions on a custom robotic skin, which both allowed for the calculation of the 3-D position of the end-effector and the Denavit-Hartenberg (DH) parameters ([19]). While this optimization-based method cleverly lets the robot close its kinematic chain by touching its own body, and was autonomous and not reliant on external measurements, the iCub’s self-touch capabilities are not possible on collaborative robotic manipulators. The reduced Degrees of Freedom (DoF) on such manipulators limit their manipulability.

However, of more importance is the self-calibration method introduced in [20]. This proposed method uses its own custom built skin units for calibration. Their method both estimates the kinematic chain of the robot and each skin units’ DH parameters by 1). collecting IMU data from each skin unit in multiple poses, oscillating certain joints of the robot in the pose to collect dynamic data to better improve parameter estimation, and 2). formulating the calculation of the DH parameters as an optimization problem. Together, [20] is able to achieve a generally ≤ 0.1 radians or meters difference between the ground truth parameters; however, the results were sub-optimal in simulation (over 10 centimeters) and not verified on a real robot platform for its effectiveness in meaningful avoidance or collaboration scenarios. The presented method also only tested one pose per each skin unit, not demonstrating whether their algorithm worked in a variety of poses.

To emphasize, the first core contribution of this thesis is not solely to create a kinematic calibration skin algorithm to improve upon [20]; it is instead to introduce an accurate calibration algorithm for a system that will enable safe and close-contact human-robot interaction. However, the work in [20] serves as the reference work from which we build off of.

Chapter 3

A Framework for a Plug-And-Play Robotic Skin

In this section, we detail the relevant preliminaries and methods pertaining to the first proposed framework.

3.1 Novel Artificial Skin

In this thesis, we define a **skin/sensor unit** (SU) as the most minimal hardware element required to perform the autonomous kinematic calibration and nearby space perception. An example of the SU can be seen in Figure 3.1. It consists of an inertial measurement unit (IMU) and a proximity sensor. Note that future mentions of sensor unit or skin unit are referring to the same unit.

The SU consists of off-the-shelf sensors, with a total overall cost of around \$36, as well as low power consumption (an operational range of around 110 to 160 mA), which allows for a completely self-contained module to ease setup and deployment. The SUs used in this work operate with external but attached 100 mAh batteries, and have been able to operate continuously for up to 10 hours.

3.1.1 Sensor Selection and Specifications

The SU prototype, which measures at 33×36mm (a full picture can be seen in Figure 3.1) is a stand-alone design with a WiFi-enabled ESP8266 microcontroller, a USB-to-UART bridge for programming, the ability for hardware debugging, and system status LEDs. Furthermore, the SU

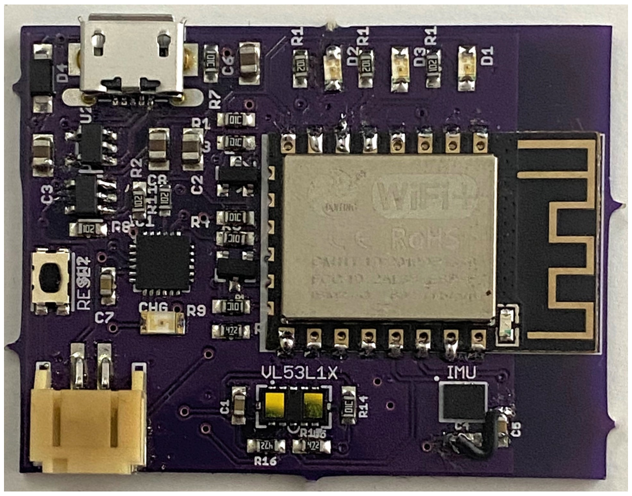


Figure 3.1: The sensor unit used for the work in this thesis.

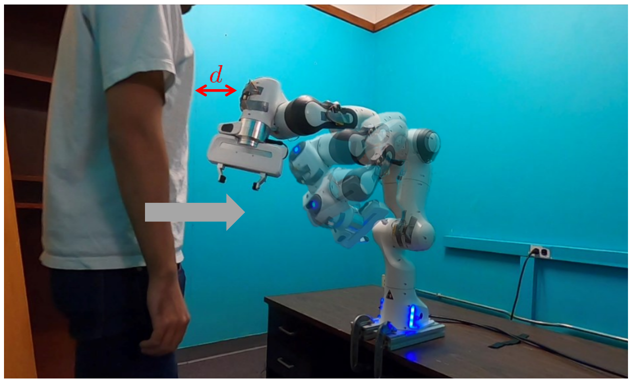


Figure 3.2: The Franka Emika Panda robot autonomously avoiding a human with the wireless skin units.

is powered by a Lithium-Polymer battery, and was fabricated on 0.15mm Kingboard copper clad laminate $175TgFR4$ substrate. The selected IMU was an LSM6DS3 iNEMO, which includes an acceleration range of $\pm 2/\pm 4/\pm 8/\pm 16g$, and an angular rate range of $\pm 125/\pm 245/\pm 500/\pm 1000/\pm 2000$ degrees per second. The VL53L1X time-of-flight (ToF) sensor serves as the proximity sensor, which includes a range of up to 4 meters, a field of view of 27°, ranging frequency of 50 Hz, and the I2C protocol. With the high bandwidth provided with both sensors, we were able to mount the SUs on the robot (as seen in Figure 3.2), and receive IMU data at a rate of 100Hz and proximity sensor data at a rate of 50Hz.

3.2 Accelerometer-Based Kinematic Calibration of Skin Units

Kinematic calibration of the skin units is the process of precisely estimating the kinematic structure of where the **skin units** are in relation to the robot's kinematic chain. The goal of SU kinematic calibration is to compute the 6D pose (in our case, a 3D position and 3D orientation) of each SU mounted on the robot so that its location is precisely known with respect to the robot's kinematic chain and frames of reference. With an accurately calibrated pose, the robot can make better use of proximity data to inform control algorithms. Later, in the Obstacle Avoidance Controller Example section, we utilize accurately-calibrated SU information to demonstrate its effectiveness via an obstacle avoidance example, in which proximity sensor data can detect objects at a distance and alter the robot's trajectory accordingly to prevent unwanted collisions.

The work in this thesis formalizes the problem of estimating an SU's pose along a robot manipulator via the Denavit-Hartenberg convention [19]. The novel kinematic calibration algorithm is solely based on accelerometer data, as initial testing indicated that angular velocity readings (from the gyroscope) were significantly noisy with drift, and the inclusion of such data did not improve calibration performance.

To achieve this, the three-axis linear acceleration data, (a_x, a_y, a_z), is collected from each SU in multiple robot configurations. We then optimize the IMU data to fit the DH parameters of each SU, allowing for the computation of each SU's pose.

Figure 3.3 shows a high-level overview of the proposed artificial SU calibration algorithm. The following sections explain each step of the calibration in extensive detail.

3.2.1 Data Collection and Generation for Calibration

The first step of the algorithm begins with data collection. Multiple sensor readings are used in the optimization algorithm, which is detailed in Section 3.2.3.

Data collection consists of two parts: the first; static pose data collection, and the second, dynamic pose data collection. Static pose collection collects accelerometer readings in multiple

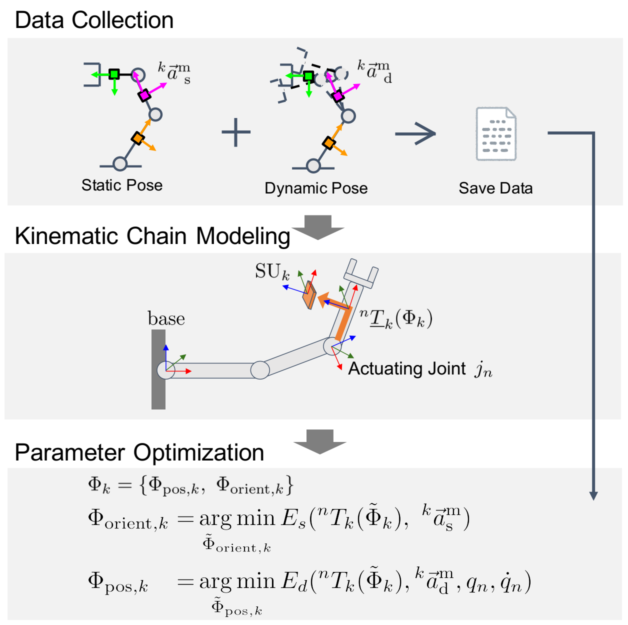

Data Collection
$^{k}\vec{a}^{\mathrm{m}}_{\mathrm{s}}$
$^{k}\vec{a}^{\mathrm{m}}_{\mathrm{d}}$
Static Pose
Dynamic Pose
Save Data
Kinematic Chain Modeling
SU$_k$
base
$^{n}T_k(\Phi_k)$
Actuating Joint j_n
Parameter Optimization
$\Phi_k = \{\Phi_{\mathrm{pos},k},\ \Phi_{\mathrm{orient},k}\}$
$\Phi_{\mathrm{orient},k} = \underset{\tilde{\Phi}_{\mathrm{orient},k}}{\arg\min}\, E_s(^{n}T_k(\tilde{\Phi}_k),\ ^{k}\vec{a}^{\mathrm{m}}_{\mathrm{s}})$
$\Phi_{\mathrm{pos},k} = \underset{\tilde{\Phi}_{\mathrm{pos},k}}{\arg\min}\, E_d(^{n}T_k(\tilde{\Phi}_k), ^{k}\vec{a}^{\mathrm{m}}_{\mathrm{d}}, q_n, \dot{q}_n)$

Figure 3.3: Overview of the kinematic calibration algorithm: 1) collect IMU data $^{k}\vec{a}^{\mathrm{m}}_{\mathrm{s}}$ and $^{k}\vec{a}^{\mathrm{m}}_{\mathrm{d}}$ at static and dynamic poses for all SU$_k$, $\forall k = \{1, \cdots, K\}$, where K is the number of SUs; 2) define a kinematic model from joint j_n to SU$_k$ expressed as $^{n}T_k$ that is parameterized by Φ_k; 3) optimize the parameters Φ_k by minimizing the static and dynamic error functions E_s and E_d for each SU$_k$. q_n and $\dot{q}_n$ represent joint angle and velocity of j_n.

joint configurations while the robot is stationary. This serves as a strong baseline to isolate and compensate for static forces that act on the robot, such as gravity. The static pose data serves as one strong source to optimize the data, and we can collect data in each static pose for a few seconds. The dynamic pose step is tasked with collecting data as each individual joint oscillates in a sinusoidal pattern. More specifically, we achieve the oscillations via actuating each joint p with a low-level joint velocity control scheme, defined by $\dot{q}_p = A\sin(2\pi f t)$, where f is the frequency, t is the time, $\dot{q}_p$ is the joint velocity of a single joint, and A is the amplitude of the pattern.

The selection of this joint velocity pattern allows to easily compute how far, fast, and how much acceleration each joint will undergo; additionally, selecting higher acceleration creates a larger signal-to-noise ratio, which helps the optimizer distinguish the acceleration data from the sensors' noise.

Static and dynamic data collection can be performed as follows: for P poses, the robot first moves to a pose and remains stationary (for a few seconds) in order to collect a sufficient amount

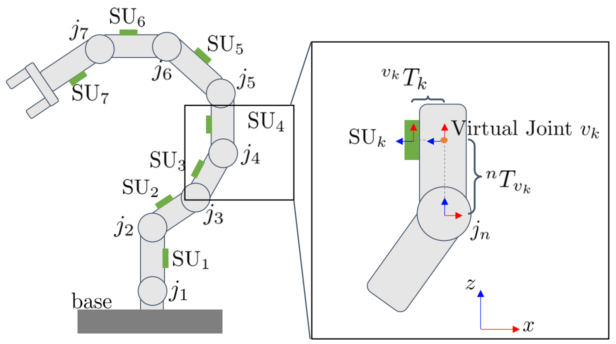


Figure 3.4: Schematic depiction of SUs mounted on a robotic arm (left) with each kinematic element of the robot labeled. The zoomed-in image on the right illustrates how the SU poses are estimated through DH parameters of the joint connecting to the previous link: the transformation from the n-th joint to the k-th SU can be computed via an intermediate "virtual joint" v_k located perpendicular to both the joint's and the SU's reference frames. The axes depict the three coordinate systems: joint n, SU k, and corresponding virtual joint v_k.

of data, then, each of the robot's joints oscillates in the motion described above (also for a few seconds), one by one. Only after this can the robot move onto the next pose. Together, the data from static data pose collection and dynamic data pose collection is incorporated to perform the optimization detailed in Section 3.2.3). The data collection only needs to be performed once for each configuration of SUs, and can be saved for later use.

3.2.2 Kinematic Model Considerations

Before running the next step of optimization, which is detailed in Section 3.2.3, it is crucial to explicitly define how to model the kinematic structure of the robot and how each skin unit is connected to the robot. Thus, we employ the *modified* Denavit-Hartenberg (DH) parameters (detailed in [21]) for modeling the robot's kinematic chain (which we are provided with) and skin unit parameters. In general, the DH notation is convenient for this work because it allows to express the relative poses of the kinematic elements on a robot's kinematic chain with four elements instead of six, further reducing the dimensionality of the optimization problem. This reduction can be achieved with the majority of robot arms that use prismatic and revolute joints, but this is not possible with skin units, which can be placed anywhere on a robot's links.

The transformation matrix that is constructed from modified DH parameters is:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & a \\ \sin(\theta)\cdot\cos(\alpha) & \cos(\theta)\cdot\cos(\alpha) & -\sin(\alpha) & -d\cdot\sin(\alpha) \\ \sin(\theta)\cdot\sin(\alpha) & \cos(\theta)\cdot\sin(\alpha) & \cos(\alpha) & d\cdot\cos(\alpha) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where θ, α, a, and d describe the DH parameters.

To adhere to the DH convention, a “virtual joint” is located between joint n and SU k, which can be seen in Figure 3.4. This then allows to estimate the 6 degree-of-freedom (DoF) transformation matrix, ${}^{n}T_{k}$, between the nth joint’s reference frame and the kth SU’s frame to be broken up into the estimation of two, DH-compatible, 4-DoF matrices:

$${}^{n}T_{k} = \begin{bmatrix} {}^{n}R_{k} & {}^{n}\vec{r}_{k} \\ 0 & 1 \end{bmatrix} = {}^{n}T_{v_k}\left[\Phi\left(v_k\right)\right]\cdot{}^{v_k}T_{k}\left[\Phi\left(SU_k\right)\right], \tag{3.2}$$

where ${}^{n}R_{k}$ and ${}^{n}\vec{r}_{k}$ are the rotational and translational components of ${}^{n}T_{k}$, and $\Phi_k = \left[\Phi\left(v_k\right), \Phi\left(SU_k\right)\right]$ are the DH-compatible parameters to be estimated.

${}^{n}T_{v_k}$, the transformation matrix which describes the transformation from joint n to virtual joint v_k, is parameterized by $\Phi\left(v_k\right) = \left(d_{v_k}, \theta_{v_k}, 0, 0\right)$ for virtual joint v_k, and ${}^{v_k}T_{k}$, the transformation from virtual joint v_k to SU k, is parameterized by $\Phi\left(SU_k\right) = \left(d_{SU_k}, \theta_{SU_k}, a_{SU_k}, \alpha_{SU_k}\right)$ for the corresponding SU_k. The above formulation of the composite transformation matrix, ${}^{n}T_{k}$, still represents a 6-DoF problem, allowing for any SU pose to be represented relative to its corresponding joint n with 6 parameters.

3.2.3 Parameter Optimization

The last step of calibration regards parameter optimization. This is tasked with properly calibrating the kinematic model detailed in Section 3.2.2 along with the data collection section (Section 3.2.1). Initially, the DH parameters for all SUs are randomly initialized within a given range (For specific information of the values of these ranges, please refer to Chapter 4). At the end of each iteration step, the ground truth accelerations are estimated from the kinematic model as detailed in the following section. Next, a global optimization algorithm is used to estimate the DH parameters by minimizing the error between the actual and predicted SU accelerations. The optimization process continues until a stopping criteria; that is, when the change in the estimated DH parameters from one step to another is sufficiently small.

The details are now provided.

3.2.3.1 Acceleration Estimation

The acceleration ${}^{k}\vec{a}_k$ exerted on SU$_k$ in its frame of reference (FoR) k can be estimated via the Newtonian Equation of motion for a rotating coordinate system. To elaborate, the total acceleration can be split up into three components: gravitational acceleration $\vec{g}_k$, centripetal acceleration $\vec{a}_{\text{cen},k}$ and tangential acceleration $\vec{a}_{\tan,k}$. These three components can be formalized as follows:

$$ {}^{k}\vec{a}_k(\Phi_k, q_n, \dot{q}_n, \ddot{q}_n) = {}^{k}R_n(\Phi_k, q) \cdot \left(\vec{g}_k + \vec{a}_{\text{cen},k} + \vec{a}_{\tan,k}\right) \tag{3.3} $$

$$ \vec{a}_{\text{cen},k}(\Phi_k, q_n, \dot{q}_n) = \vec{\dot{q}}_n \times \left(\vec{\dot{q}}_n \times {}^{n}\vec{r}_k(\Phi_k, q)\right) \tag{3.4} $$

$$ \vec{a}_{\tan,k}(\Phi_k, q_n, \ddot{q}_n) = \vec{\ddot{q}}_n \times {}^{n}\vec{r}_k(\Phi_k, q) \tag{3.5} $$

With respect to Eq. (3.4), the centripetal acceleration $\vec{a}_{\text{cen},k}$ consists of the joint angular velocity $\vec{\dot{q}}_n = [0, 0, \dot{q}_n]$ (which can be calculated during the data collection process), and the position vector ${}^{n}\vec{r}_k$, which represents the translational component of ${}^{n}T_k$ in Eq. (3.2). Since these terms are all defined in the nth rotating joint's reference frame, they have to be rotated from reference frame j_n to SU$_k$ through ${}^{k}R_n$. Note that the full acceleration vector calculation is parameterized by Φ_k, which are the DH parameters that will be optimized.

The tangential part of joint motion on acceleration readings of the SUs requires knowing the joint angular acceleration; however, most robot manipulators do not easily provide this information. Thus, computation of the angular acceleration requires a numerical estimation. Conventional methods (such as the method in [20]) resort to using the second derivative of the position vector, or

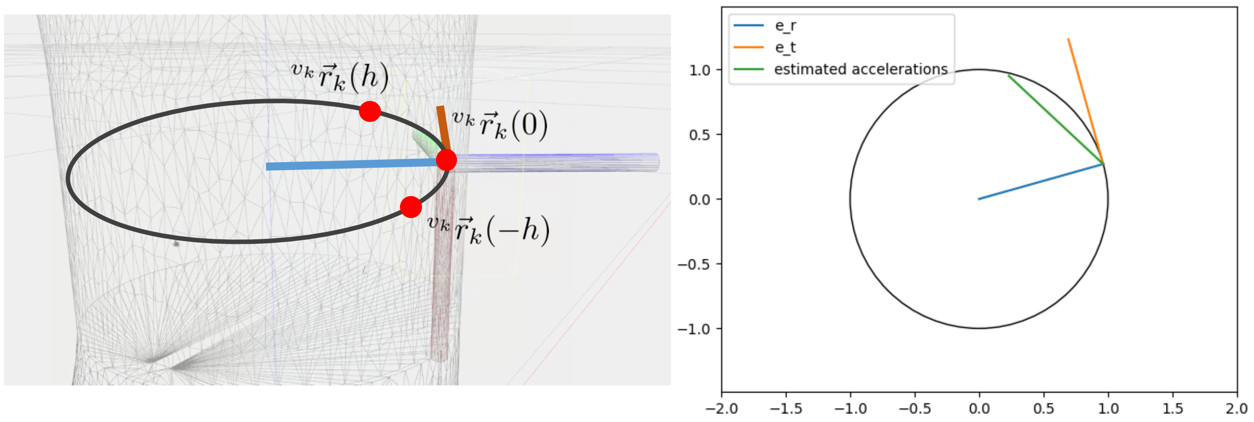


Figure 3.5: Illustration of the numerically computed tangential accelerations for joint n in its respective FoR. The plot on the right shows the accelerations computed by taking the second derivative of the position vectors (the red dots on the left). The accelerations include not only a tangential acceleration but also an unnecessary centripetal acceleration.

${}^{n}\vec{r}_k$, to estimate the SU's acceleration. However, this differentiation does not include gravity and incorrectly includes an extra centripetal force from the SU moving along a rotating joint, which is detailed in Fig. 3.5.

The inclusion of this additional centripetal force is not only redundant, but it also is a detriment on optimization performance – thus, we remove the non-tangential forces by employing the equation in Eq. (3.6), then followed by Eq. (3.7):

$$\vec{a}_{\tan,k} = \vec{e}_{\tan} \cdot \frac{\vec{r}(h) + \vec{r}(-h) - 2 \cdot \vec{r}(0)}{h^2} \tag{3.6}$$

$${}^{k}\vec{a}_{\tan,k} = {}^{k}R_n \cdot \vec{a}_{\tan,k} \tag{3.7}$$

From the equations above, $\vec{r}(h)$ is a parameterization of the position vector ${}^{n}\vec{r}_k$ as a function of discrete time $h = 0.001$, and $\vec{e}_{\tan}$ is an unit vector in the tangential direction, where we only take the tangential direction unit vector in order to avoid the unecessary centripetal acceleration.

3.2.3.2 Error Functions

For each SU$_k$ attached to a joint n, we can optimize its DH parameters, Φ_k, by decomposing the objective function into two separate minimization tasks: namely, we minimize two distinct error functions; first, the static acceleration error E_s and the dynamic acceleration error E_d. For each static pose p (from static pose data collection), the static error function E_s only brings with it relevant rotational information for the DH parameters because the static data collection and data solely depend on the gravitational acceleration (centripetal and tangential acceleration of the SUs is 0); thus, only the rotational DH parameters (included in ${}^{n}R_k(\Phi_k)$) can be estimated from the static data. It is important to note that there is a misalignment in coordinate frames – the measured acceleration of SU$_k$ ${}^{k}\vec{a}^{m}_{k,p}$ is defined in its own reference frame k, but the gravity vector ${}^{b}\vec{g}$ is defined in the base frame. Thus, the acceleration vector must be converted into the robot's base frame using the rotational matrix ${}^{b}R_n$ as computed from forward kinematics. The rotation matrix ${}^{n}R_k$ also rotates the acceleration of SU$_k$ into the joint n's reference frame before rotating it into the base frame.

The static acceleration error is then defined as the L2 norm between the measured acceleration and the gravity vector:

$$E_s(k) = \frac{1}{P}\sum_{p=1}^{P}\left|{}^{b}R_n \cdot {}^{n}R_k(\Phi_k) \cdot {}^{k}\vec{a}^{m}_{k,p} - {}^{b}\vec{g}\right|^2, \tag{3.8}$$

where P represents the total number of static poses collected (user-selected). Conversely, the dynamic acceleration error $E_{d,k}$ is composed of both rotational and translational information. However, the dynamic acceleration error function is not used to optimize the rotational DH parameters because its data is noisier than the static data, and the static data is specifically built for optimization of the rotational DH parameters. Upon optimizing ${}^{n}R_k(\Phi_k)$ from Eq. (3.8) (these rotational parameters are frozen and not optimized later), the translational component of the optimization problem can be estimated via the L2 norm between the measured acceleration ${}^{k}\vec{a}^{m}_{k,d,p}$ and the ground truth estimated from kinematics and current DH parameters ${}^{k}\vec{a}_{k,d,p}(\Phi_k)$, and expressed in the SU's reference frame:

$$E_d(k) = \frac{1}{P} \sum_{p=1}^{P} \sum_{\substack{d>0,\\ d=n-2}}^{n} \left| {}^{k}\vec{a}^{m}_{k,d,p} - {}^{k}\vec{a}_{k,d,p}(\Phi_k) \right|^2, \tag{3.9}$$

where d is the sole joint that is actuated during the dynamic data collection process to exert some acceleration on each SU. At most three previous joints before the nth joint are used to compute the above error function. While the SUs located on earlier links of the robot cannot rely on as much information as ones located on later links, we still notice strong results across the board. Additionally, ${}^{k}\vec{a}_{k,d,p}$ can be computed from Eq. (3.3) for joint d and in pose p. From here, the optimization procedure is repeated for all SUs. After completion, the DH parameters for each SU are saved for later use.

Chapter 4

A Plug-And-Play Robotic Skin: Experimental Design, Results, and Discussion

To effectively validate the proposed method, a comprehensive, quantitative evaluation is first conducted in simulation through the Robot Operating System (ROS [22]) and the Gazebo simulation ([23]). Using ROS and Gazebo makes it easy to create complex robotic algorithms and quick prototyping and testing.

Following this, we deploy our system on a real-world Franka Emika Panda collaborative robot (see Fig. 4.1), in which we show the effectiveness of an obstacle avoidance control example. Our experiments demonstrate an improvement over the state-of-the-art [20] and shows the ability for accurate kinematic calibration on 7-DoF collaborative platforms in simulation and the real-world. Next, the parameters and other specific details of the experiment are described as to improve the reproducibility of the presented work. The corresponding code and data are available at the associated GitHub repository[1] .

Because the previous algorithm proposed in [20] was slightly incorrect, we modified it by including the acceleration due to gravity to adhere to Eq. (3.3). Thus, we instead display the results from Mittendorfer's method with this modification: the following results display this modified version, which is called *modified* Mittendorfer's method (mMM). The original version was orders of magnitude worse than ours, and comparing our method to mMM is more fair.

As with the comparison method in [20], the robot's base frame serves as the starting point for calibration. The robot base frame is set to align with the world reference frame for simplicity. In

[1] The GitHub repository(ies) will be published at the IROS 2021 camera ready stage, or upon inquiry to mast4878@colorado.edu.

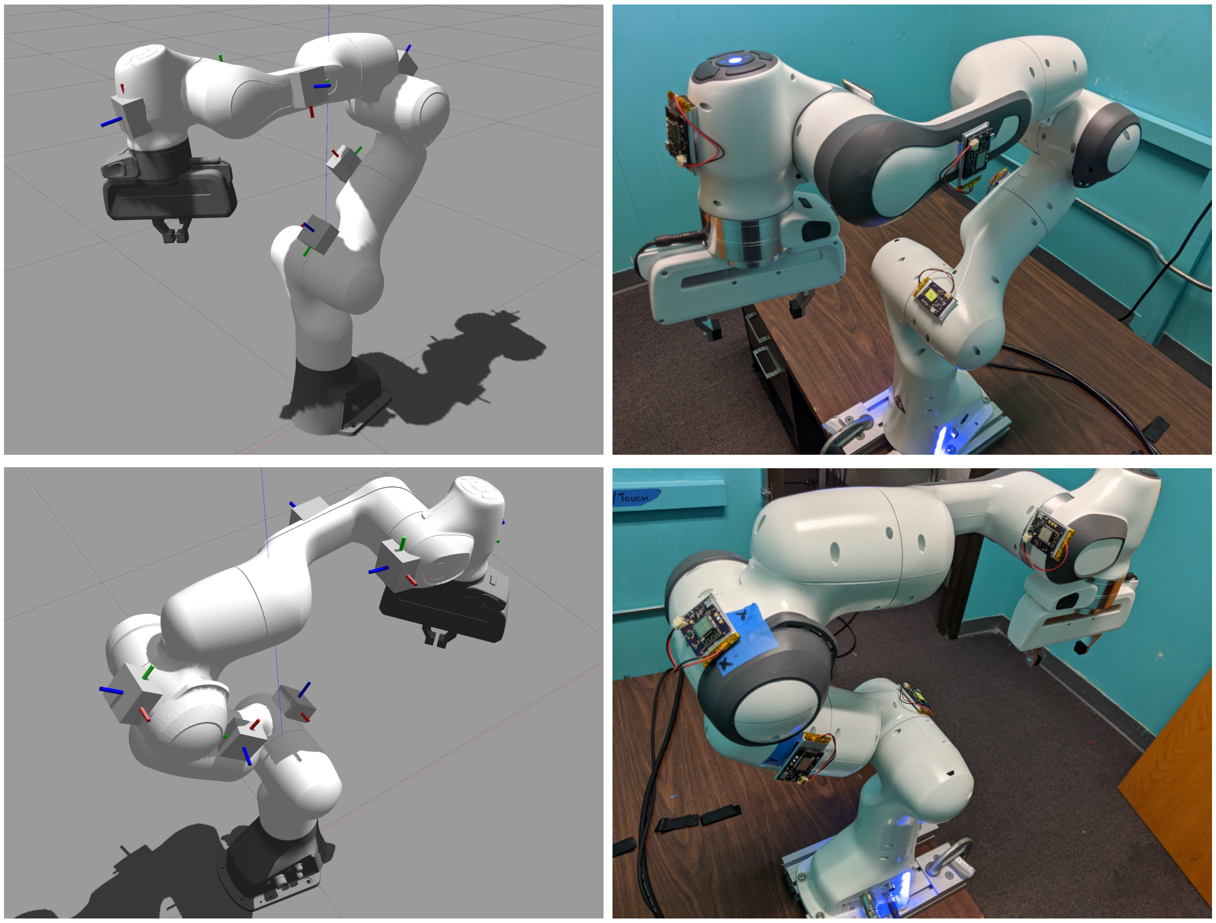


Figure 4.1: Comparison between estimated SU poses (left) and ground truth poses (right) on a Franka Emika Panda robot arm. The grey boxes represent the estimated SU with x (red), y (green), z (blue) axes respectively.

Table 4.1: Comparison of the Average L2 Norm positional error [cm] and absolute distance in quaternion space (with units multiplied by 10^{-1} for reasons of space) of our method (OM) against the modified Mittendorfer method (mMM). 4 different sets of poses are tested and optimization is run 10 times per set; average and standard deviations of these 40 trials are below.

		SU_1	SU_2	SU_3	SU_4	SU_5	SU_6	Average
Positional	OM	0.28 ± 0.15	0.39 ± 0.15	0.78 ± 0.39	1.15 ± 0.88	0.80 ± 0.36	0.25 ± 0.095	0.66 ± 0.60
Error [cm]	mMM	5.3 ± 7.1	2.6 ± 0.23	5.4 ± 4.9	9.5 ± 9.0	1.4 ± 0.91	2.3 ± 2.4	4.4 ± 5.9
Quaternion	OM	0.10 ± 0.058	0.054 ± 0.041	0.023 ± 0.012	0.019 ± 0.013	0.021 ± 0.023	0.045 ± 0.053	0.044 ± 0.059
Distance	mMM	5.5 ± 5.9	0.16 ± 0.11	3.6 ± 6.1	0.057 ± 0.022	0.044 ± 0.015	2.5 ± 4.3	2.0 ± 4.4

Table 4.2: True and Estimated DH Parameters of one of the four SU configuration sets we tested.

	j_1 to SU$_1$		j_2 to SU$_2$		j_3 to SU$_3$		j_4 to SU$_4$		j_5 to SU$_5$		j_6 to SU$_6$	
	true	est	true	est	true	est	true	est	true	est	true	est
θ_{SU}	1.571	1.571	0.000	-0.014	-1.571	-1.571	3.141	3.141	-1.571	-1.571	1.571	1.565
d_{SU}	0.060	0.060	-0.080	-0.076	0.080	0.085	-0.100	-0.007	0.030	0.030	0.000	0.001
a_{SU}	-1.571	-1.545	0.000	-0.001	1.571	1.567	3.141	3.141	1.571	1.541	-1.571	-1.570
α_{SU}	0.060	0.062	0.050	0.053	0.060	0.060	0.100	0.142	0.050	0.048	0.050	0.049
θ_v	0.000	0.001	0.000	0.000	0.000	0.001	0.000	-0.001	0.000	-0.001	0.000	-0.001
d_v	1.571	1.583	1.571	1.568	1.571	1.570	1.571	1.565	1.571	1.570	1.571	1.571

the simulation and real world scenarios, six SUs are mounted perpendicularly to the surface of the each robot link; each link has one SU on it except the first link. A SU cannot be placed on the first link because there are an infinite number of solutions with the same acceleration values. This is because the values $\vec{\dot{q}}$, $\vec{\ddot{q}}$, and $\vec{r}$ are equal along the first link. We were able to use the native Gazebo IMU plugin to simulate IMU behaviors. Before optimization, the DH parameters of every SU is initialized randomly within the following bounds: $\theta_v \in [-\pi; \pi]$, $d_v \in [-1; 1]$, $\theta_{SU} \in [-\pi; \pi]$, $d_{SU} \in [-1; 1]$, $a_{SU} \in [0; 1]$, $\alpha_{SU} \in [-\pi; \pi]$.

The data collection process (including both static pose and dynamic pose collection) is performed only once in both simulation and the real world. For simulation, $P = 16$ poses were collected, with the total time taking under 20 minutes.

We utilized a joint velocity controller to oscillate the robot at 2rad/s, and a joint position controller to move from pose to pose. The IMU messages from the SU are published and received at 100Hz, while the controller operates at a rate of also 100Hz.

As the robot goes through each pose, upon reaching each pose, two seconds of rest were set to prevent outliers and ensure the robot has ample time to reach the pose. We use a randomized global optimization algorithm of MLSL_LDS ([24]), along with a derivative-free local optimizer (LN_NEWUOA, [25]) of the Python NLopt library [26]. Even though we initialize the DH parameters of each SU within a constrained set of boundaries, we select NEWUOA unconstrained optimization to allow for better exploration of the DH parameter search space – in all, there are 36 parameters due to each one of the six SUs being parameterized by six values. Due to the nature of how we set up the parameters, there are actually multiple solutions for each SU's DH parameters, so allowing unconstrained methods enables a more methodical exploration of the search space and avoiding local minima.

The optimization for the static acceleration error is terminated once the error is below a threshold of 0.01, or if the parameter difference in between consecutive iterations is less than 0.001. The optimization for the dynamic acceleration error is also terminated based on the same conditions. However, a distinction needs to be made that the termination of the static acceleration error optimization simply allows the optimizer to optimize the positional related parameters – they are not different estimation problems. Both data collection and optimization were performed on a Lenovo Thinkpad X1 Extreme, 2nd Generation, with 12 Intel i7-9750H CPUs and a GeForce GTX 1650.

4.1 Results and Discussion

4.1.1 Simulation Experiments

In simulation, we evaluate the performance of our algorithm based on the average L2 Euclidean distance and quaternion distance over every SU. Furthermore, to test the robustness of our algorithm, we varied the parameters of our experiment by doing the following: i) We tested 4 different sets of SU configurations along the robot's body. Each SU configuration is different from the others; thus, we have tested 24 different SU poses along the robot. ii) We optimized the SU configurations 10 times per set. Additional tests performed did not seem to alter the averages and variances of our results. The mean and standard deviation of the calibration on the total of 40 trials are shown in Table 4.1, and estimated DH parameter values are displayed in Table 4.2.

As elaborated in the experiment design section, we elected to not compare against the original method in [20] due to the incorrect modeling of acceleration and poor performance – instead, we created the modified version (mMM) for a more fair comparison. Based on the table in Table 4.1, the modified version of Mittendorfer's method (mMM) had an average error of 4.4cm and a significantly high variance of $\pm$5.9cm. On the other hand, our method (OM) has an average error of *under* 1cm at 0.66cm for all of the SU placements, which is a six-fold improvement, demonstrating the correctness of our method. Indeed, similar can be stated for the comparison between the two methods with respect to orientation error. Along with the significant gain in accuracy, separating the optimization problem into two parts (but not two estimation problems) sped up the separated problem (433 seconds) with respect to the original problem (1797 seconds).

4.1.2 Real World Experiments

To test the effectiveness of the algorithm in a real world setting, we validate our method on a Franka Panda collaborative robot arm. The work in [20] did not perform real-world calibration and was exclusively tested in simulation. Additionally, we expect slightly worse calibration performance due to the introduction of additional layers of complexity: for example, possible time delays in the robot network that caused the collection of slightly misaligned data, added sensor noise and bias in the IMUs, and hardware limitations of the platform.

However, IMU data from the SUs can still be retrieved at a rate of 100Hz and recorded for 3 seconds at each pose. The resting time given for a robot to reach a pose is lengthened to 10 seconds. We also modified the oscillation magnitude to be 2.0 and frequencies of each joint to 0.75, 0.25, 0.35, 0.55, 0.75, 0.9, and 0.9Hz. While these reduced frequencies restrained the displacements of the robot, they still allow for reasonably high accelerations. The data collection was conducted on a computer outfitted with a real-time kernel and took less than 30 minutes to allow ample time for the robot to move from one pose to another. For the real world experiments, $P = 12$ different poses were collected.

A qualitative comparison of the estimated vs. actual SU poses can be seen in Fig. 4.1. The grey boxes in the simulation represent the estimated SUs, and the real world pictures are the actual, ground truth SU placements. It is clear that the estimation of the SU poses, for the most part, are highly similar between real life and simulation. After running calibration on the real robot, the average position error for all of the SUs was approximately 5.1cm and the average orientation error

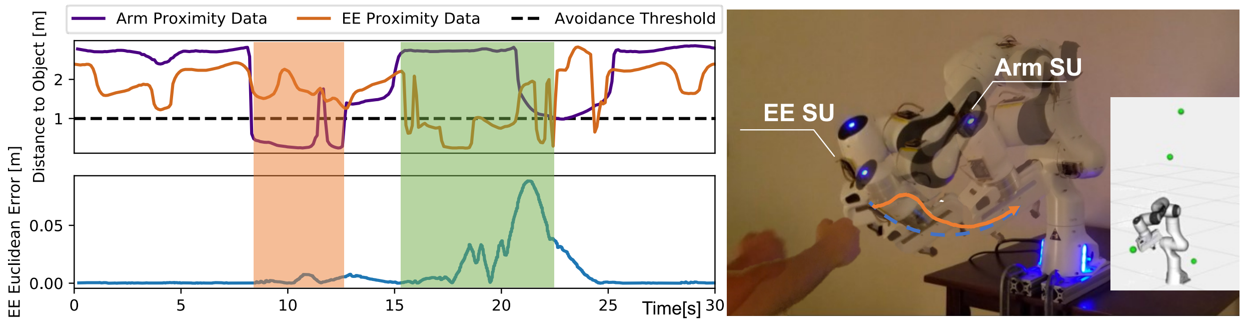


Figure 4.2: Trajectory redirection for obstacle avoidance. Right: a human approaches a robot with calibrated skin units; their accurate pose estimation allows the robot to precisely perceive the person from a distance. Bottom-Right: real-time visualization of the robot and obstacles (shown as green spheres) as sensed by the proximity sensors placed on the calibrated SUs; only below-threshold signals (arm and end-effector SU) affect the robot's motion. Left: combined graph of proximity data over time vs. the robot's end effector error along a circular trajectory. The two shaded areas indicate intervals when the human approaches a skin unit (below the avoidance threshold), causing only a temporary perturbation in the error. The perturbation stays low in the orange colored area (arm proximity activation) compared to the green area (end effector proximity activation) because the robot leverages the redundancy afforded by its kinematic chain to avoid colliding with the person while concurrently satisfying its main task objective.

was 0.12, whereas mMM had an average position error of 9.2cm and 0.43 error for orientation. The results are not shown, but the average positional error for the original Mittendorfer's method was up to 90cm, an order of magnitude worse than our work.

4.1.3 Obstacle Avoidance Controller Example

Immediately after calibration, the estimated SU poses can be used for an obstacle avoidance controller, which can be seen in Fig. 4.2. To showcase the abilities of the calibrated system, we implemented a Cartesian velocity obstacle avoidance controller similar to [27], where proximity data is leveraged to maintain the desired end-effector trajectory and avoid obstacles. In the example in the Fig. 4.2, the robot follows a continuous, circular trajectory in the task space, and proximity data is published (and received) at a rate of 50Hz.

The bottom right image in Fig. 4.2 displays a visualization (using Rviz) of the robot model and obstacles detected in its nearby space; if any one or more obstacles are sensed within 1m of a SU (shown by the dashed line on the graph in Fig. 4.2), then the robot will take corrective actions to avoid the object. The selection of 1m as a threshold for detection is arbitrary as to guarantee safety in the presence of humans; however, this value can be made dynamic to adjust based on a variety of factors. The effectiveness of the control example is demonstrated from the left graph(s) in Fig. 4.2. The orange and green shaded regions indicate a human approaching one of the skin units. In the orange region, an arm SU detects a nearby object, and the change to the end-effector Euclidean error is minimal because the manipulator is able to leverage its redundancy to both stay on the desired trajectory and avoid the person. In the green region, the SU on the EE detects a nearby object, and has to move more to avoid the person. Clearly, this avoidance example demonstrates that the calibration accuracy in real life is sufficient for effective, nearby space, close-proximity human robot interaction.

Chapter 5

Implicit Contact Anticipation via Distributed Whole-Body Sensing: Related Work

This literature review concerns the second framework, which directly builds on top of the calibrated sensor units to present an algorithm that addresses the “gray” area between avoidance and contact.

5.1 Collision Avoidance

Contact is desired in physical Human-Robot Interaction (pHRI), but it has not been receiving as much attention as avoidance algorithms. Even with a variety of sensing capabilities, work in safe HRI has frequently focused on **complete** obstacle avoidance, which leads to zero contact between a robot and human collaborator. The works of [28, 29] utilized designated safety zones around the robot where the robot will slow down, or stop its movement entirely to avoid collisions at all costs. However, these designated safety zones are more closely resembling to the current situation of factory and manufacturing manipulators, where the robot always needs to be operating with an imaginary barrier.

For dynamic environments, there have been dynamic avoidance methods ([2,30,31]) that used an externally-mounted 3D camera to inform avoidance behavior. However, while these methods do not create a virtual safety barrier (that a human or object should not enter), their primary objective is to still avoid an object, even if a human is attempting to make contact. In addition, the external mounted systems of these works do not effectively cover the whole nearby-space of the robot. More approaches in [32,33] do address the idea of working in close proximity, but still rely on external sensing and maintain a safe distance from a human whenever possible. While the presented work in this thesis also has avoidance behaviors, the avoidance is developed in consideration with the transition into contact. [34] did use onboard sensors, but still employed them to solely inform avoidance. More recent works in [27,35] have formulated the avoidance problem as an optimization problem via their onboard sensors. While these methods added more complex behaviors – they allow for avoidance *around* objects instead of direct repulsion – no attempt is made for addressing contact. In fact, none of the avoidance and even contact-based methods in the surveys of [1,36] tackle the transition from avoidance to contact, which is the one of the focal points of this thesis.

5.2 Collision Reaction and Detection

On the other side of the spectrum of human-robot interactions is contact; but is often referred to as (the more negatively coined term) collision. This work has generally not integrated prior-to-contact sensing and perception, and the contribution of collision-based works is at the moment when and after contact is made, and not at any time *before*. Under this view, there is no notion of "proactivity" and preparation for any sort of collision.

Indeed, works such as [37–42] focused on either collision detection, collision reaction, or a combination of the two. While these methods' post-collision behavior was highly varied, none of them address how to perceive the nearby space before a potentially inevitable contact is made. Because none of these works used methods for perception, their abilities are severely limited in that at high speeds, where collision may be unsafe. Our work draws together the best of both collision detection/reaction, as well as collision avoidance methods.

5.3 Collision Avoidance, Reaction and Detection

The work most similar to ours is [43], which proposes a comprehensive framework that combines both collision avoidance with collision detection and reaction. However, the *transition* from any sort of avoidance to collision is not addressed, and no mention is made of what happens right before collision is made. The collision avoidance method in their work is also based on [2], which was developed purely for avoidance. The selected avoidance method of [2] was also limited in its avoidance ability; it only has the ability to move directly away from an obstacle instead of more intelligently maneuvering around it. Our framework enables more expressive avoidance behaviors, but also develops both avoidance and contact with them mutually both in mind, allowing for strong preliminary results.

Chapter 6

Implicit Contact Anticipation

In this section, we detail the relevant preliminaries and methods pertaining to the second proposed framework, which leverages the capabilities of the previous framework to allow for physical human-robot interaction. The framework or system is designed to allow for a spectrum of interactions to safely occur, including avoidance and soft contact.

We also detail each component of our framework, which, at a broad level, constitutes of perception, avoidance, and contact detection and reaction as seen in the four sections of Fig. 6.1.

6.1 Development of Sensor Units

As previously mentioned, this contribution can only be fully realized via distributed, fast, and whole-body sensing. We employ the self-contained, low-cost sensor units from one of the previous sections (Novel Artificial Skin). For future mentions of proximity units in this section, we are referring to the VL53L1X ToF sensor that is a component of the sensor unit. These sensor units are identical to the ones presented in the previous framework.

6.2 Identification of Object Positions

With the capabilities of the aforementioned sensor units, objects can be perceived in the robot's close surroundings. To accomplish this, the sensor units are distributed along the robot body.

Each SU's proximity sensor is positioned so that its distance measurement is parallel to the

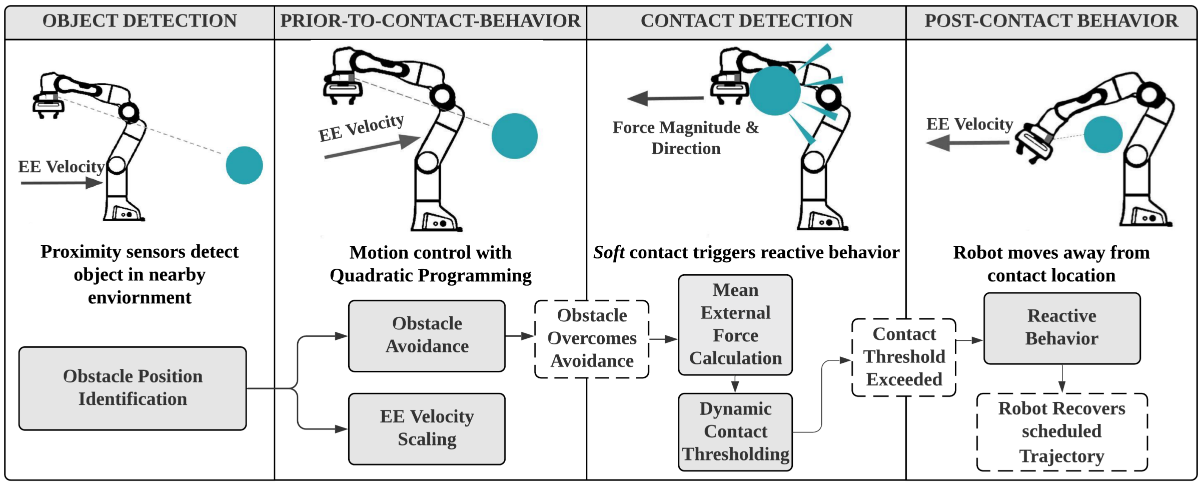
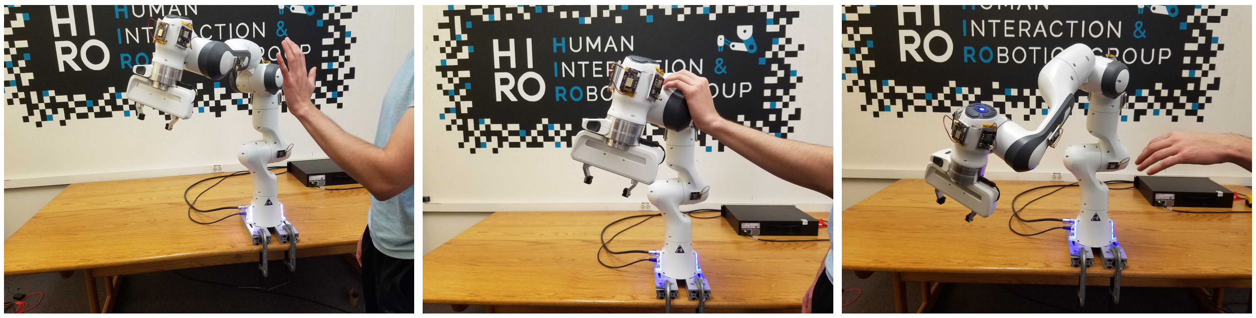

OBJECT DETECTION
PRIOR-TO-CONTACT-BEHAVIOR
CONTACT DETECTION
POST-CONTACT BEHAVIOR
EE Velocity
EE Velocity
Force Magnitude & Direction
EE Velocity
Proximity sensors detect object in nearby enviornment
Motion control with Quadratic Programming
Soft contact triggers reactive behavior
Robot moves away from contact location
Obstacle Position Identification
Obstacle Avoidance
EE Velocity Scaling
Obstacle Overcomes Avoidance
Mean External Force Calculation
Dynamic Contact Thresholding
Contact Threshold Exceeded
Reactive Behavior
Robot Recovers scheduled Trajectory

Figure 6.1: Diagram of the presented framework for contact anticipation, detection, and reaction. Each module represents a distinct interaction, from when an object enters the robot's environment to when the repulsive velocity becomes zero, and the initial trajectory is resumed.



(a) **Prior to Contact:** A human enters the robot's workspace, and is sensed by onboard sensor units, triggering both velocity scaling and avoidance behavior.

(b) **Contact:** When a object makes contact with the robot, we measure the external force and determine if it exceeds the dynamically computed threshold.

(c) **Post Contact:** After contact has been made and the force threshold exceeded, a reactive behavior moves the robot in the direction that the force was applied.

Figure 6.2: A human interacting with a robot to elicit reactive behavior while the robot is in motion. This interaction is outlined in Section 7.1.3.

sensor unit's z-axis. This allows for the computation for an object's position $\mathbf{h} \in \mathbb{R}^3$ as function of the SU proximity reading $d_{obs} \in \mathbb{R}$:

$$\mathbf{h} = {}^{O}\vec{r}_{SU_k} + {}^{O}R_{SU_k} \begin{bmatrix} 0 & 0 & d_{obs} \end{bmatrix}^T, \tag{6.1}$$

where SU_k is the kth sensor unit, ${}^{O}\vec{r}_{SU_k} \in \mathbb{R}^3$ is the position of SU_k in relation to the robot base frame O (which can be computed via kinematics with the calibrated DH parameters), and ${}^{O}R_{SU_k} \in \mathbb{R}^{3\times3}$ is the rotational matrix of SU_k in relation to the robot base frame. This enables the computation of any object position in Cartesian space for each calibrated SU_k. Combined with knowledge of the robot's state, expressive collision avoidance behavior and contact thresholding are now achievable. This concludes the object detection block of Fig. 6.1.

6.3 Motion Control with Quadratic Programming

The main task, as well as the avoidance behaviors of the system, are implemented through a Cartesian velocity controller, which solves for the control joint velocities in a unified quadratic programming (QP) expression. The QP optimization technique is selected due to its ability to incorporate a variety of behavior movement modifications into the optimization problem.

To define the tasks, first, let $\dot{\mathbf{q}} \in \mathbb{R}^n$ represent the joint velocities of a kinematically redundant robot manipulator with n joints, and let the Cartesian velocity of the end-effector (EE) be represented as $\dot{\mathbf{x}} \in \mathbb{R}^m$, where $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$, and $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{m\times n}$ is the Jacobian of the robot. Now, the QP control equation is written as:

$$\begin{aligned} g(\dot{\mathbf{q}}) = &\frac{1}{2}(\dot{\mathbf{x}}_\mathbf{d} - \mathbf{J}\dot{\mathbf{q}})^\top(\dot{\mathbf{x}}_\mathbf{d} - \mathbf{J}\dot{\mathbf{q}}) + \frac{\mu}{2}\dot{\mathbf{q}}^\top\dot{\mathbf{q}} \\ &+ \frac{k}{2}(\dot{\mathbf{q}}_\mathbf{mid} - \dot{\mathbf{q}})^\top(\dot{\mathbf{q}}_\mathbf{mid} - \dot{\mathbf{q}}). \end{aligned} \tag{6.2}$$

The first term in Eq. (6.2) represents the quadratic Cartesian error between the current end-effector velocity $\dot{\mathbf{x}}$ and the desired task Cartesian velocity $\dot{\mathbf{x}}_\mathbf{d}$, whereas the second term is used as a damping term to avoid singularities based on a robot manipulability measure: we refer the reader to [44] for more details.

Furthermore, a third term is added on to the main task which causes the robot to favor joint positions near the middle of its joint limits, which allows a robot to move to a more natural and safer joint configuration, far away from the mechanical limits. In this term, $\dot{\mathbf{q}}_\mathbf{mid}$ consists of the desired joint velocities that will move each joint closer to its middle position (as determined from the joint operational range from the robot's manufacturer), and k is a scaling factor used to weigh the middle joint term in the whole objective function. Eq. (6.2) is then manipulated to conform with the quadratic programming notation and restrictions as noted in [27]. The previous formulation, which did not include the middle joint limit term, allows a manipulator to enter undesired configurations over time, and ultimately error states that prevent further robot operation.

Next, in the following two sections, the end-effector velocity reduction and robot movement restrictions are detailed, which form the constraints of the QP formulation.

6.4 End-Effector Velocity Reduction

When a robot detects an obstacle or human, its end-effector velocity is reduced in order to maximize safety and prepare for contact, independent of avoidance behaviors. As a human collaborator approaches, the robot should slow down execution of its main task, which actually causes the corresponding avoidance motion to be slower; still allowing for contact to easily be made. It is worth noting that this behavior doesn't constitute avoidance behavior, it simply slows down the robot to encourage contact. Without any velocity reduction behavior, purposeful contact is both difficult and unnatural.

The velocity is reduced as follows: for each detected object **h** within a selected range (from a sensor unit) d_{max} (in this work, d_{max} is 0.8m), the norm of the distance is taken $||d||$, between the object and end-effector. The lowest norm is selected to ensure safest behavior. From here, the manipulator's desired velocity is reduced based on the following equation:

$$\dot{\mathbf{x}}_{\mathbf{d}} = \frac{||d||}{d_{max}}\, \dot{\mathbf{x}}_{\mathbf{d}}, \tag{6.3}$$

where d_{max} scales $||d||$ to produce a normalization term that reduces the main task's desired EE velocity, dependent on how close an object is. Note that objects detected beyond d_{max} from the EE are discarded because we desire close-proximity behavior. In addition, to prevent jerky motions induced by vanishing obstacle readings (e.g., a proximity sensor reading goes from 0.8m to 5m instantly in between consecutive timesteps), we apply a linear decay formula to simulate a slowly moving away obstacle:

$$\dot{\mathbf{x}}_{\mathbf{d}} = \dot{\mathbf{x}}_{\mathbf{d}} \cdot \left[\frac{||d||}{d_{max}} + (1 - \frac{||d||}{d_{max}}) \cdot \frac{l_{obs}}{l_{max}} \right], \tag{6.4}$$

where l_{obs}, which starts at 0, is the lifespan of a current obstacle after sensing, that linearly increases over time, and l_{max} is the value of where the original end-effector velocity is completely restored. The value is set to 200 in this work's evaluation.

6.5 Robot Movement Restrictions

The choice to use the QP framework allows for the construction of expressive avoidance behaviors via creating movement restrictions. To further make it easier to make contact, using the QP framework also makes it possible to move *around* objects instead of directly moving *away.*

To obtain such behavior, movement restrictions are placed on the part of the robot closest to each obstacle to permit movement around obstacles. To achieve this, at a high level, a linear inequality constraint is constructed that limits the motion $\dot{\mathbf{x}}_{\mathbf{c},\mathbf{i}} \in \mathbb{R}^3$, which is the Cartesian velocity of the closest point to obstacle $\mathbf{i}$ on the manipulator. This value is computed as $\dot{\mathbf{x}}_{\mathbf{c},\mathbf{i}} = \mathbf{J}_{\mathbf{c},\mathbf{i}}\dot{\mathbf{q}}$, where $\mathbf{J}_{\mathbf{c},\mathbf{i}}$ is the Jacobian of the closest point on the manipulator to the obstacle. Then, $\hat{\mathbf{d}}$ is computed, which describes the direction to the obstacle $\mathbf{h_i} \in \mathbb{R}^3$; thus, the term $\hat{\mathbf{d}}^{\mathbf{T}}\mathbf{J}_{\mathbf{c},\mathbf{i}}\dot{\mathbf{q}}$ describes the approach velocity of the robot towards an object. Last, this value is constrained, for any number of obstacles, to a desired approach velocity value, $\dot{x}_a$, which is parameterized based on the distance to an obstacle.

$\dot{x}_a$, or the desired approach velocity to an obstacle, is computed as a smooth and continuous function. Drawing inspiration from [2], our movement restrictions on the approach velocity are computed as follows:

$$V_a = \frac{V_{max}}{1 + e^{\alpha(2\frac{d}{d_{crit}} - 1)}}; \tag{6.5}$$

$$V_b = \frac{V_{max}}{1 + e^{\alpha(2\frac{d - d_{crit}}{d_{notice} - d_{crit}} - 1)}}; \tag{6.6}$$

$$\dot{x}_a = \begin{cases} V_a - V_{max} : \text{if } d < d_{notice} \text{ and } d < d_{repulse}, \\ V_b : \text{if } d < d_{notice} \text{ and } d \geq d_{repulse}, \\ \text{Drop Restriction : otherwise.} \end{cases} \tag{6.7}$$

Where V_{max} is the maximum repulsive velocity, d is the distance from the object to the closest point on the robot's body, $d_{repulse}$ is the distance where repulsive behavior begins, and d_{notice} is the distance at which we "notice" an obstacle, and start to impose movement restrictions. This behavior ensures smooth approach velocities as well as transitions between repulsive and movement limiting. When an object is close, only a small repulsion will be applied—the robot will only very slowly try to move around the object. In this manner, our framework does not lose the ability for avoidance, but it avoids slowly as to encourage desirable contact behavior. For our experimental scenarios, $V_{max} = 0.04$, $d_{repulse} = 0.1$, and $d_{notice} = 0.6$.

6.6 Dynamic Contact Thresholding

To smoothly transition between prior-to-contact and post-contact behaviors, a robot controller must determine if external contact has been made, along with the direction and magnitude of the contact. Successful achievement of such behavior brings with it three desired qualities: a) perfect or near-perfect external force data should not be required, b) contact is more likely when an object is close to the robot, and c) a robot moving close to an object should move slower than normal and have **increased** sensitivity to interaction. To this end, we propose a dynamic thresholding algorithm that relies on obstacle readings and an estimation of the robot's force; together, both are fused in such a fashion that will capture all three desired qualities.

To detect contact forces, the proposed algorithm does not require a perfect estimation of the force: the signal that we utilize throughout this paper is the estimated external Cartesian contact force, provided through the Franka Control Interface [45]. This estimation is often volatile and can be quite far from zero in all axes, especially when the robot is moving at high velocities. However, our algorithm is robust to noisy and biased data, and can be used to accurately determine if external contact is made on the robot's end-effector.

6.6.1 Average External Force Calculation

In order to work with only recent force data, a sliding window is created to determined the running average.

The manner in which we add new values to the sliding window is detailed below:

$$x_t = \begin{cases} \alpha x_t + (1-\alpha) x_{t-1} : \text{if } |x_t - \mu_{t-1}| > \lambda \sigma_{t-1}, \\ \\ x_t : \text{otherwise}, \end{cases} \tag{6.8}$$

where x_t is the data point appended to the window, α is the influence value of a detected outlier (exponential moving average), λ is a scaling factor which is multiplied by the previous standard deviation σ_{t-1}, and μ_{t-1} is the previous mean value. The term $\lambda\sigma_{t-1}$ determines if a given value is an outlier and should be discounted when added to the signal history. This allows for data points that are outliers to have less influence on the data. Additionally, a window size is selected that only includes a small number of samples because a larger window size would make it challenging for actual contact to be detected. In our experimental scenarios, $\alpha = 0.1$ and $\lambda = 0.75$.

6.6.2 Contact Threshold Calculation

Contact thresholding determines when contact is made, in what directions (x, y, z) contact was made, and the magnitude of the contact. To start, from the constructed external force window, the running average (μ) is used as a base value for the dynamic external contact thresholds, which are expressed as

$$\text{Contact Threshold} = \begin{cases} T_u = \mu + F_b + F_\sigma - F{-}obs, \\ \\ T_l = \mu - F_b - F_\sigma + F{+}obs, \end{cases} \tag{6.9}$$

where T_u is the upper and T_l is the lower limit of the contact threshold, F_b is the base additional force required from the mean to trigger contact behavior (negative and positive), and F_σ increases the external force required to trigger contact behavior, and is based on the standard deviation of the external force signal. This value is computed as

$$F_\sigma = \min(\frac{\sigma}{\sigma_{max}} \cdot F_{std}, F_{std}), \tag{6.10}$$

where σ_{max} is the max (user-defined) standard deviation. σ is the force data's current standard deviation (with the modified sliding window), and F_{std} is the max amount of force (user-defined) that is applied based on the σ value. From the above formulation, when the standard deviation of the current data is low, F_σ will be small, which leads to a threshold that is sensitive to an external force. Indeed, as the robot slows down, the external force data is less noisy, making a slower robot more sensitive to movement. Even when the manipulator has not detected an object, if it is moving with a reduced velocity, it is more sensitive to forces.

Next, $F{-}obs$ and $F{+}obs$ are force reductions of the upper and lower limits, respectively, based on the distance from an obstacle to the robot's EE. A closer object should make the thresholds more constrictive because contact is more likely.

Thus, $F{-}obs$ and $F{+}obs$ are calculated using the following equation:

$$F_{\text{obs}} = (\frac{d_{\text{max}} - d}{d_{\text{max}} - d_{\text{min}}}) \cdot F_d, \tag{6.11}$$

where d_{max} is the distance from the robot's end-effector where F_{obs} begins to be applied to the contact threshold, and d_{min} is the distance at which the maximum force reduction is applied. F_d is the maximum force reduction (user-defined), and d is the minimum distance from the robot EE on the *positive* side of the robot for $F{+}obs$ and the *negative* for $F{-}obs$, relative to each axis (x, y, z). For example, an object with a greater y value than the current end-effector position will lead to a reduction in the required force to trigger contact behavior in the negative y direction.

When contact behavior is triggered, the external force data reading (from the sliding window) must exceed either the upper *or* lower threshold in any one or more of the x, y, z axes. If this condition is met, then the robot will move away accordingly. We calculate the direction and overall force applied to the end-effector as to apply a velocity in the force's direction, as shown below.

$$F_{ext} = F_{reading} - \mu, \qquad \dot{\mathbf{x}}_{des} = C \cdot F_{ext}. \tag{6.12}$$

$F_{reading} \in \mathbb{R}^3$ describes the contact force from the sliding window, and $\mu \in \mathbb{R}^3$ is the mean force of the window, which allows for the computation of $F_{ext} \in \mathbb{R}^3$. C describes the Cartesian compliance matrix that proportionally multiplies each component of the external force to output a desired end-effector velocity, and this desired end-effector velocity temporarily overrides the main task.

The velocity is then linearly decayed to zero over a specified time period, similar to Eq. (6.4). For the experimental scenarios, the user-defined contact thresholding values were set to $F_{std} = 3$, $F_d = 4$, $F_\sigma = 4$, and $F_b = 10$.

Ultimately, as a result of dynamic contact thresholding, the three aforementioned interaction qualities are achieved:

(1) **Robustness to measurement noise:** The method does not rely on perfect external force data; instead, it only requires data that is reasonably noticeable beyond the current mean,

similar to anomaly detection.

(2) **Anticipated contact from obstacles:** As an obstacle approaches the robot end-effector, the contact thresholds decrease due to the F_{obs} value.

(3) **Increased contact sensitivity as velocity decreases:** For noisier data, we are less certain of contact being made, which increases the threshold bounds. When a robot slows down due to obstacles, reduced noise is observed, and, in turn, a smaller F_σ value is added to the thresholds.

One possible concern that may arise with this dynamic contact thresholding method is to use one static threshold; however, unlike static thresholds, the proposed method works on imperfect, noisy estimations, and has the ability to adapt to environmental information, making it clearly more versatile for researchers and robot operators.

Chapter 7

Implicit Contact Anticipation for Physical Human Robot Interaction: Experiment Design and Results

To validate the effectivness of the proposed framework, multiple experiments are conducted on the 7-DoF, real world Franka Panda robot. A computer outfitted with a real time kernel sends joint velocity commands to the robot's control box at a rate of 100Hz. We utilize the calibration algorithm detailed earlier in this thesis and the same SUs as the previous sections for fast object detection. As previously mentioned, the SU contains both an IMU and a proximity sensor, the latter of which informs our avoidance and contact detection algorithms. The experimental pipeline is first developed using C++ and ROS [22]: the avoidance component of our framework was implemented in Gazebo [23] before moving on to the real robot to test out the entire framework. Finally, the framework's multiple capabilities are showcased in multiple scenarios, as detailed below.

7.1 Experimental Scenarios

7.1.1 Static Obstacle Collision

The first experimental scenario consists of a static obstacle collision: the robot is placed near a large, static object and is commanded to move towards the object to cause a collision to occur. The object is placed close enough to the robot so that a collision will occur, as our framework's avoidance behavior is not completely geared towards avoidance. The purpose of this scenario is to highlight the integration of onboard proximity information to slow down and enable soft contact. To achieve this, we position the object in two positions next to the robot to highlight our system's

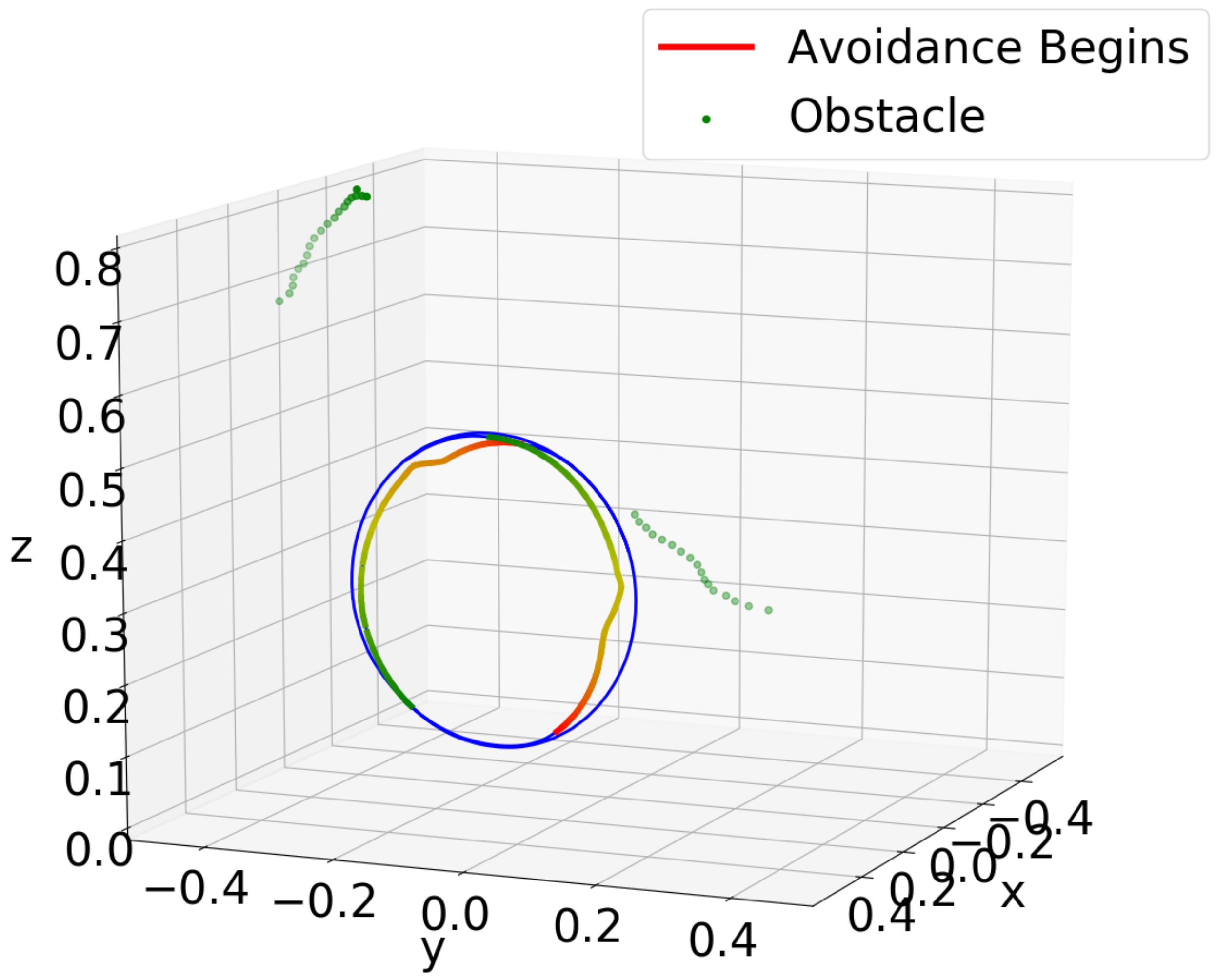


Figure 7.1: Robot end-effector, moving in a circular path (blue circle); when an object is detected (green dots), the end-effector trajectory is modified to avoid collision (red) then gradually recovers (green). We demonstrate this behavior in two interactions on opposite sides of the robot; the objects are noticed by independent proximity sensors.

abilitiy to reduce contact thresholds based on an object's location. In the first position, the static object is placed in the negative y direction with respect to the end-effector; in the second position, it is placed in the positive y direction with respect to the end-effector. We further vary these two scenarios by employing two separate trajectories for each collision: in the first position, the robot moves in a Cartesian circle with a radius of 0.25m, centered at (0.5, 0, 0.25), and moving counter-clockwise only in the y and z directions. For the second position, the robot moves in a line from (0.4, 0.5, 0.5) to (0.4, -0.5, 0.5), with speeds up to 0.3 m/s. The results of our method in these scenarios are compared to that of the same movement when not informed by onboard sensor units to reduce the movement speed and anticipate soft contact.

7.1.2 Obstacle Avoidance

In the obstacle avoidance scenario, the robot's avoidance capabilities are demonstrated by commanding the robot to move in the same circular trajectory as the previous section, then having a human enter the robot's current path. When this happens, a slight deviation is caused in the robot's circular trajectory; however, the robot soon is able to correct it after the human leaves the robot's nearby space. This demonstration is best shown in Fig. 6.2a, in which a human holds their hand up near the robot. The deviation from the desired trajectory is shown, with two separate proximity sensors being activated on opposite sides of the robot.

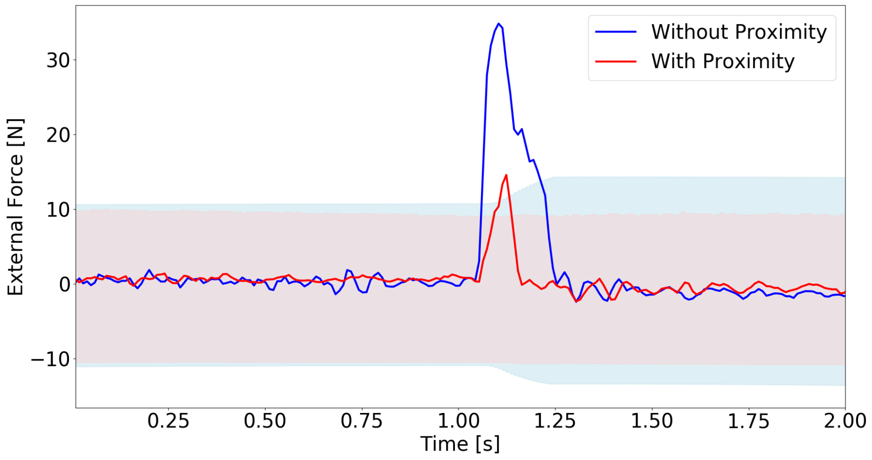
7.1.3 Dynamic Obstacle Collision

In the last experimental scenario, multiple points of contact with a human participant are shown in a continuous interaction. Unlike Section 7.1.1, the person represents a dynamic obstacle. Multiple, different sensors are activated in this interaction, and contact is made from different directions to showcase our framework's capabilities. A visual depiction of this interaction is shown in Fig. 6.2, where each key component of the interaction (prior to contact, contact, and post contact behavior) are pictured.

7.2 Results and Discussion

7.2.1 Static Obstacle Collision

In the static contact scenario, Fig. 7.2 depicts the force reduction caused when proximity data can be used to anticipate collisions. In both of the static collision examples, the difference in the overall detected contact force is approximately two times smaller when using proximity data; the reduced force clearly demonstrates that proximity detection is beneficial for safe pHRI.



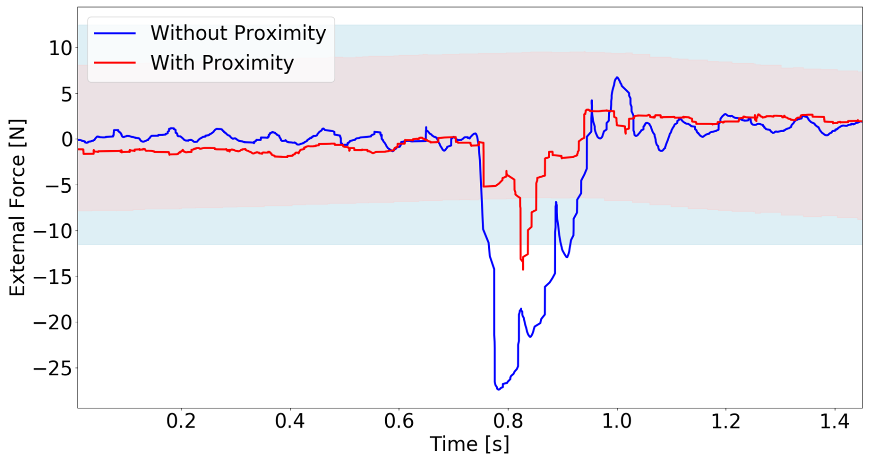


Figure 7.2: Force difference for contact detection, in the y-direction, between our method with proximity sensing (red line) and without proximity sensing (blue line). The shaded regions represent the thresholds for contact. In the top graph, the robot is moving in a circular trajectory, and in the bottom, a horizontal line, before contact is made.

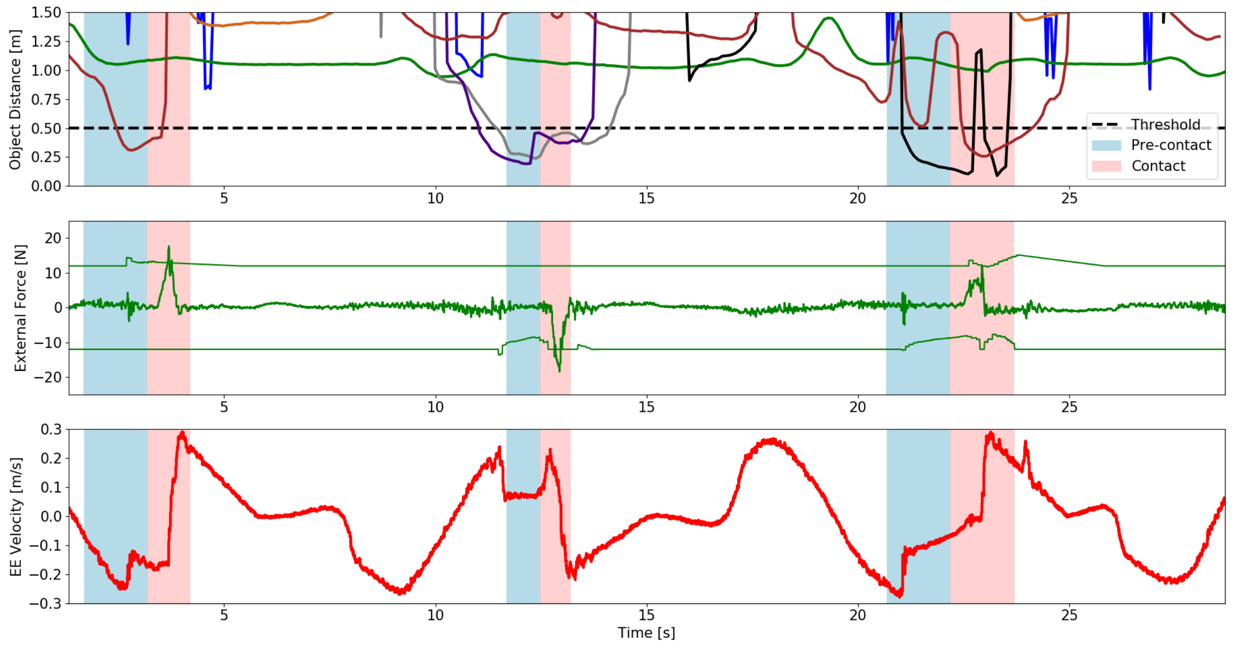


Figure 7.3: Three components of a continuous human–robot interaction over 30 seconds, during this time, three distinct contact are made from multiple directions. Top graph: depth values of seven proximity sensors placed along the robot body and EE, with the activation threshold as a dotted black line. Middle graph: external force exerted on the robot EE in the y-axis with the dynamic thresholds shown above and below. Bottom graph: The manipulator's EE velocity along the y-axis.

Before contact is made, the contact threshold value is reduced, which can be clearly seen in the concave red shaded region of the bottom graph. The change in threshold can be attributed to the object proximity and robot velocity scaling, which reduces the resultant collision force. The two independent contacts in different directions of the y-axis demonstrate that the system works in both directions. Furthermore, it is worth mentioning that while we setup a situation in which collision cannot be prevented, our framework is designed in such a way that allows the altering of avoidance weights to completely avoid the object. Indeed, different use cases can have different settings of avoidance parameters.

7.2.2 Obstacle Avoidance

Fig. 7.1 displays the trajectory of the robot's EE altered due to the avoidance behavior while executing a circular path. As an object is detected by one of the robot's SU's (green dots), the controller decreases the robot's velocity, and the minor avoidance behavior leads to a slight deviation in the circular path. Once the object is sufficiently far away from the robot, or moving in the opposite direction (due to the trajectory), the robot soon returns to its original trajectory. This behavior is demonstrated through multiple sensor units mounted on opposite sides of the manipulator to solidfy the controller's ability to both avoid collisions and prepare for contact. As with the previous section, parameters can be tuned to make our framework have more or less restrictive avoidance behaviors.

7.2.3 Dynamic Obstacle Collision

The final experimental scenario highlights multiple, dynamic contacts with a human, which is outlined in Fig. 6.2. These interactions are graphed in Fig. 7.3. In this figure, three separate contacts are made with the robot. The manipulator only reacts to obstacles with a distance of at most 0.5m away, as seen as a black dotted line in Fig. 7.3. Note that the threshold value of 0.5m can be altered based on different use cases. When an object is “noticed” by the robot; that is, the distance to the nearest SU is at most 0.5m, the robot begins to reduce its velocity, which

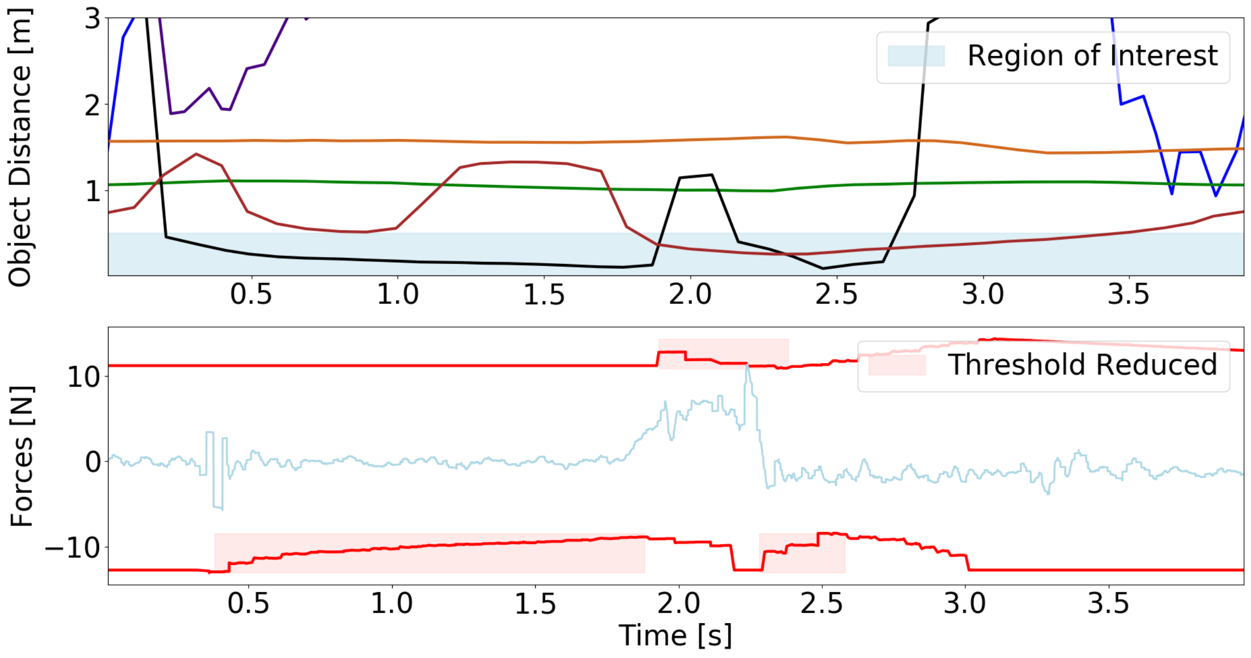


Figure 7.4: Top graph: distance values from six proximity sensors mounted on the robot body and EE during interaction. The region of interest label denotes the region at which sensed distances are meaningful for our algorithm. Bottom graph: external force exerted on the robot EE in the y direction and associated contact thresholds above and below.

can be seen in the blue shaded region of the bottom graph in Fig. 7.3. Along with the slowing down behaviors, the contact thresholds become more constrictive as objects come close, which can be prominently seen in Fig. 7.4, which displays a higher resolution view of the third interaction. During this interaction, two proximity sensors are activated on opposite sides of the robot, leading to a reduction of both the upper *and* lower thresholds, demonstrated in the red shaded regions of Fig. 7.4. After the robot slows down, the actual contact is expressed in the red shaded regions of Fig. 7.3 from multiple directions. The applied force's magnitude and direction directly determines the resulting direction and magnitude of the EE repulsive velocity when such repulsive behavior is triggered. The resulting velocity spikes from contact in the bottom graph of Fig. 7.3 are also shown in the red shaded regions. Altogether, through this comprehensive interaction, we illustrate that the proposed framework is able to both anticipate and react to contact with dynamic objects in its nearby space, from multiple. In fact, without our novel thresholding behavior, such soft contact would not trigger the reactive movement and not allow a human to easily interact with the robot. The last interaction is only noticed through our thresholding. This might bring up the question of whether the threshold could simply be made static and smaller; however, in Fig. 7.4, before the actual contact is made, the force readings are around 5 to 8N, revealing how the force data is far from zero before contact, but our method is still able to accurately pinpoint the one point of contact. The implicit anticipation opens a new realm of behaviors for true human–robot collaboration in close proximity, where a robot can dynamically adapt to its changing environment.

Chapter 8

Conclusion and Future Work

In these thesis, we presented two contributions: the first, which presented a system for a plug-and-play robotic skin, we a) introduced a cheap novel skin unit for nearby space perception; b) an accurate method for kinematically calibrating multiple SUs along any part of a collaborative manipulator; c) the demonstrated effectiveness of the calibration through a real-life avoidance example. Our approach works for any number of SUs and does not need any external systems. With only a few centimeters of error in the real world, we are able to effectively leverage calibrated SU poses for meaningful interaction.

The second contribution introduced a framework for contact anticipation during physical human–robot interaction. Through relaxed avoidance constraints, combined with a novel dynamic thresholding algorithm, our work addresses the unexplored gray area between avoidance and contact. Furthermore, our experiments demonstrate the system's robustness in multiple scenarios with both static and dynamic interactions.

Accordingly, we have open-sourced our code and data for the research community (at the moment, to be available upon inquiry or during the IROS 2021 camera-ready stage), and we are hopeful that this work will continue to push forward plug-and-play distributed, artificial sensing for robotics, and its applications for close-proximity interaction.

With respect to the first contribution, there are a few promising avenues for future work: i) investigate how significant the skin calibration accuracy is for safe robot control by comparing with the sensitivity of a human skin; ii) further improving the skin calibration accuracy by looking deeper into the effectiveness of the dynamic and static poses; iii) improve on our novel skin.

With respect to the second contribution's future work: i) developing a truly collaborative system through both hardware and software – our current sensor units are outfitted with IMUs; while we used the IMUs for our calibration, they were not utilized in this framework. IMU data can be leveraged to further increase sensitivity to external forces acting on the robot; ii) iterating on the next generation of SUs, including tactile sensors, allowing for *localization* of contact from external forces; iii) planning to explore the effectiveness of our perception as we add more sensor units along a robot's exterior and combine this information with external vision to gain a holistic view of the robot's environment.

Ultimately, we will continue to improve our current sensor units and achieve whole–body awareness, inching robotics research closer and closer to natural and extended human-robot collaboration.

Bibliography

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," Mechatronics, vol. 55, 2018.

[2] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[3] E. Magrini, F. Flacco, and A. De Luca, "Control of generalized contact motion and force in physical human-robot interaction," in 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015, pp. 2298–2304.

[4] R. S. Dahiya and M. Valle, Robotic tactile sensing: technologies and system. Springer Science & Business Media, 2012.

[5] A. Roncone, M. Hoffmann, U. Pattacini, and G. Metta, "Learning peripersonal space representation through artificial skin for avoidance and reaching with whole body surface," in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015, pp. 3366–3373.

[6] F. Mastrogiovanni, L. Natale, G. Cannata, and G. Metta, "Special issue on advances in tactile sensing and tactile-based human–robot interaction," Robotics and Autonomous Systems, no. 63, pp. 227–229, 2015.

[7] G. Cheng, E. Dean-Leon, F. Bergner, J. R. G. Olvera, Q. Leboutet, and P. Mittendorfer, "A comprehensive realization of robot skin: Sensors, sensing, control, and applications," Proceedings of the IEEE, vol. 107, no. 10, pp. 2034–2051, 2019.

[8] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor et al., "The icub humanoid robot: An open-systems platform for research in cognitive development," Neural networks, vol. 23, no. 8-9, pp. 1125–1134, 2010.

[9] D. H. P. Nguyen, M. Hoffmann, A. Roncone, U. Pattacini, and G. Metta, "Compact real-time avoidance on a humanoid robot for human-robot interaction," in Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, 2018, pp. 416–424.

[10] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in Springer handbook of robotics. Springer, 2016, pp. 113–138.

[11] G. Du and P. Zhang, "Imu-based online kinematic calibration of robot manipulator," The Scientific World Journal, vol. 2013, 2013.

[12] R. Zhu and Z. Zhou, "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package," IEEE Transactions on Neural systems and rehabilitation engineering, vol. 12, no. 2, pp. 295–302, 2004.

[13] D. Roetenberg, H. Luinge, and P. Slycke, "Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors," Xsens Motion Technologies BV, Tech. Rep, vol. 1, 2009.

[14] A. Yamaguchi and C. G. Atkeson, "Implementing tactile behaviors using fingervision," in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). IEEE, 2017, pp. 241–248.

[15] W. Yuan, S. Dong, and E. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," Sensors, vol. 17, no. 12, p. 2762, 2017.

[16] R. Patel and N. Correll, "Integrated force and distance sensing using elastomer-embedded commodity proximity sensors." in Robotics: Science and systems, 2016.

[17] D. Hughes, J. Lammie, and N. Correll, "A robotic skin for collision avoidance and affective touch recognition," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1386–1393, 2018.

[18] A. Roncone, M. Hoffmann, U. Pattacini, and G. Metta, "Automatic kinematic chain calibration using artificial skin: self-touch in the icub humanoid robot," in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 2305–2312.

[19] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," Journal of applied mechanics, vol. 77, no. 2, pp. 215–221, 1955.

[20] P. Mittendorfer and G. Cheng, "Open-loop self-calibration of articulated robots with artificial skins," in 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 4539–4545.

[21] B. Siciliano and O. Khatib, Springer handbook of robotics. Springer, 2016.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in ICRA workshop on open source software, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[23] 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, pp. 2149–2154.

[24] A. H. G. R. Kan and G. T. Timmer, "Stochastic global optimization methods," Mathematical Programming, vol. 39, no. 12, pp. 27–78, 1987.

[25] M. J. D. Powell, "The newuoa software for unconstrained optimization without derivatives," in Proc. 40th Workshop on Large Scale Nonlinear Optimization, 2004.

[26] S. G. Johnson, "Nlopt." [Online]. Available: https://nlopt.readthedocs.io/en/latest/

[27] Y. Ding and U. Thomas, “Collision avoidance with proximity servoing for redundant serial robot manipulators,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[28] C. Vogel, C. Walter, and N. Elkmann, “A projection-based sensor system for safe physical human-robot collaboration,” in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013.

[29] P. Svarny, M. Tesar, J. K. Behrens, and M. Hoffmann, “Safe physical hri: Toward a unified treatment of speed and separation monitoring together with power and force limiting,” 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov 2019. [Online]. Available: http://dx.doi.org/10.1109/IROS40897.2019.8968463

[30] H. Nascimento, M. Mujica, and M. Benoussaad, “Collision avoidance in human-robot interaction using Kinect vision system combined with robot’s model and data,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[31] A. Tulbure and O. Khatib, “Closing the loop: Real-time perception and control for robust collision avoidance with occluded obstacles,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[32] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, “Safety in human-robot collaborative manufacturing environments: Metrics and control,” IEEE Transactions on Automation Science and Engineering, vol. 13, no. 2, 2015.

[33] P. A. Lasota, G. F. Rossano, and J. A. Shah, “Toward safe close-proximity human-robot interaction with standard industrial robots,” in 2014 IEEE International Conference on Automation Science and Engineering (CASE). IEEE, 2014.

[34] G. B. Avanzini, N. M. Ceriani, A. M. Zanchettin, P. Rocco, and L. Bascetta, “Safety control of industrial robots based on a distributed distance sensor,” IEEE Transactions on Control Systems Technology, vol. 22, no. 6, 2014.

[35] Y. Ding, F. Wilhelm, L. Faulhammer, and U. Thomas, “With proximity servoing towards safe human-robot-interaction,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.

[36] P. A. Lasota, T. Fong, J. A. Shah et al., A survey of methods for safe human-robot interaction. Now Publishers, 2017.

[37] M. Geravand, F. Flacco, and A. De Luca, “Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture,” in 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013.

[38] E. Mariotti, E. Magrini, and A. De Luca, “Admittance control for human-robot interaction using an industrial robot equipped with a f/t sensor,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

[39] E. Magrini and A. De Luca, “Hybrid force/velocity control for physical human-robot collaboration tasks,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016.

[40] E. Magrini, F. Flacco, and A. De Luca, "Estimation of contact forces using a virtual force sensor," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014.

[41] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," IEEE Transactions on Robotics, vol. 33, no. 6, 2017.

[42] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2008.

[43] A. De Luca and F. Flacco, "Integrated control for phri: Collision avoidance, detection, reaction and collaboration," in 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob). IEEE, 2012.

[44] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," 1986.

[45] "Franka control interface documentation." [Online]. Available: https://frankaemika.github.io/docs/

[46] K. Watanabe, M. Strong, M. West, C. Escobedo, A. Aramburu, K. Chaitanya, and A. Roncone, "Self-contained kinematic calibration of a novel whole-body artificial skin for collaborative robotics," 2021, under review.

[47] P. Mittendorfer, E. Yoshida, and G. Cheng, "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot," Advanced Robotics, vol. 29, no. 1, pp. 51–67, 2015.