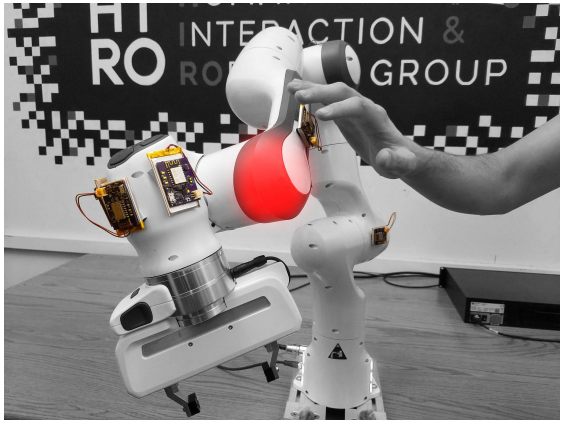
Enabling **Close Proximity** Human Robot Collaboration via Distributed, Self-Calibrating Robotic Skin



Matt **Strong**

Self-Introduction

- Fun facts:
  - Twin
  - Half Indian
  - Decent at **running** and **biking**
  - Learning some **Chinese**

Table of Contents

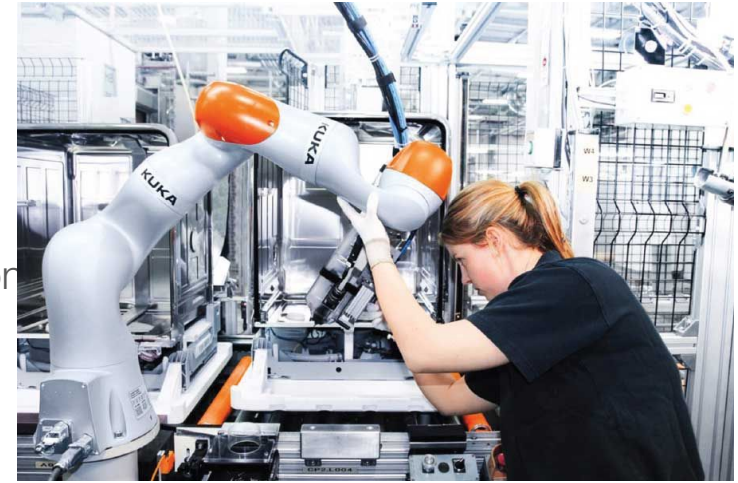
Introduction: The Problem

- Robotics currently exist at a large scale in **industrial/manufacturing** environments
- Humans work around robotics, robots don’t work around humans
- But, need to drive the **transition** from industrial to environments with people

Robots working on a car

Introduction: Nearby Space Perception

- In these environments, **extended**, **close proximity** human robot collaboration is essential
- To achieve this, first step: **Perception.** But there's some problems.
  - External, sparse, high-resolution sensing -- occlusion problem
  - Onboard, contact-based sensors
- Solution (and **Contribution**): Whole Body Distributed Sensing is **key** to such HRC

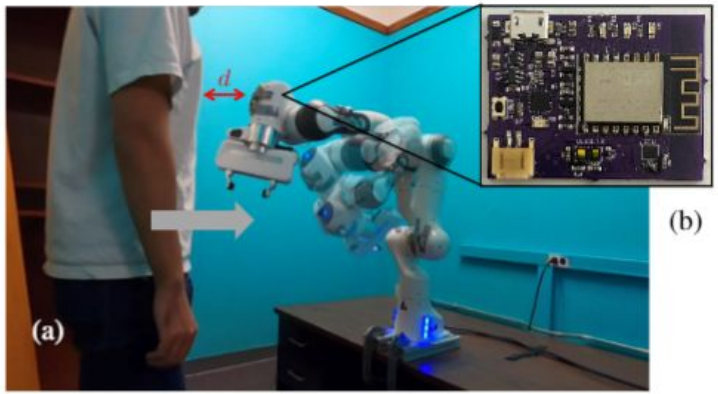


A human and robot collaborate

Contribution 1

- **Problem 1**: Current Whole Body Sensing lacks a certain degree of modularity, accuracy, and ease of use
- **Solution 1**: A new plug-and-play robotic skin system for calibration, demonstrated on a real avoidance example

A robot avoids a human with the calibrated skin units.



Contribution 2

- **Problem 2**: Lack of a smooth transition between avoidance and (desirable) contact
- **Solution 2**: Implicit contact anticipation via those same onboard sensor units

Under our framework, a robot can anticipate contact with the SUs.

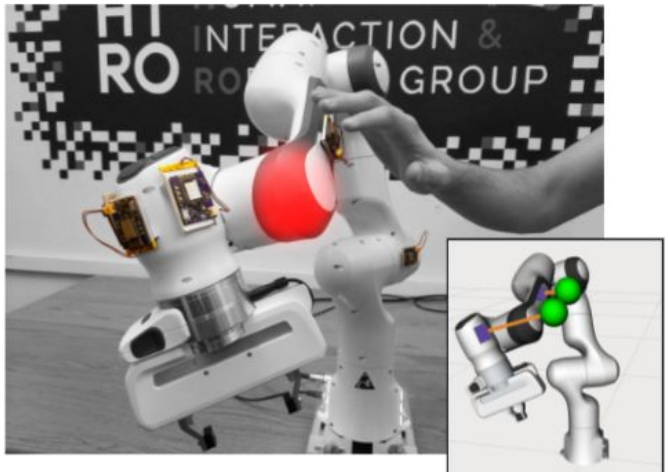
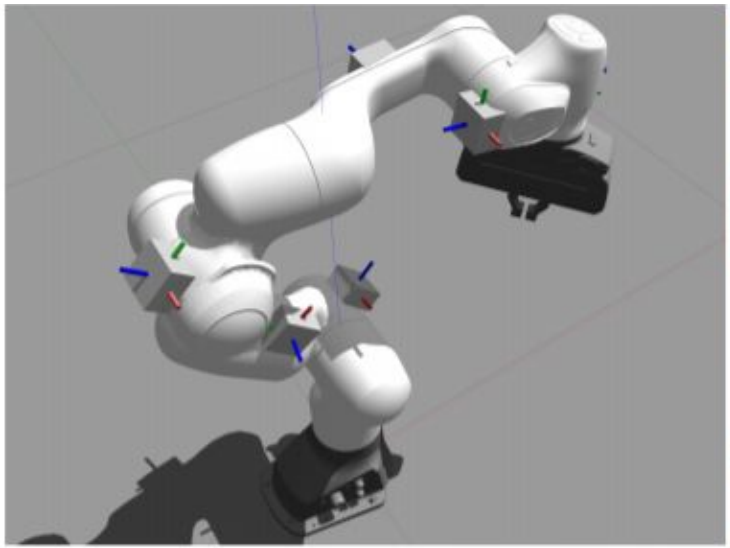


Table of Contents

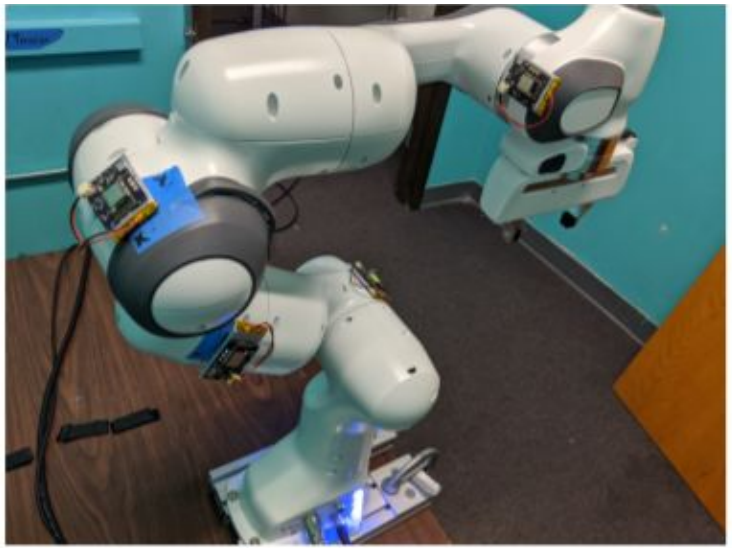
A Plug and Play Robotic Skin

- Goal: Automatically calibrate the skin units along a robot's body

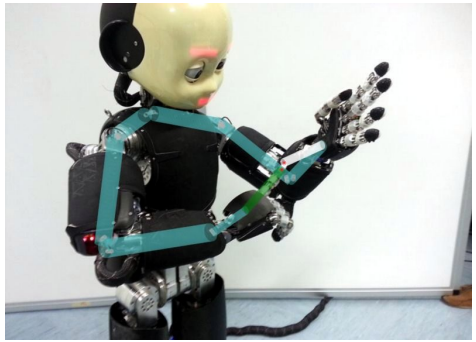
Result of Calibration

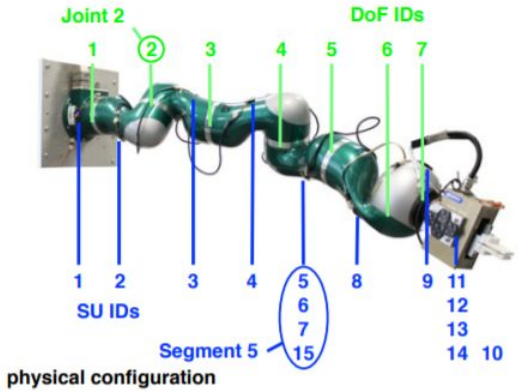


Actual skin unit poses



Related Work

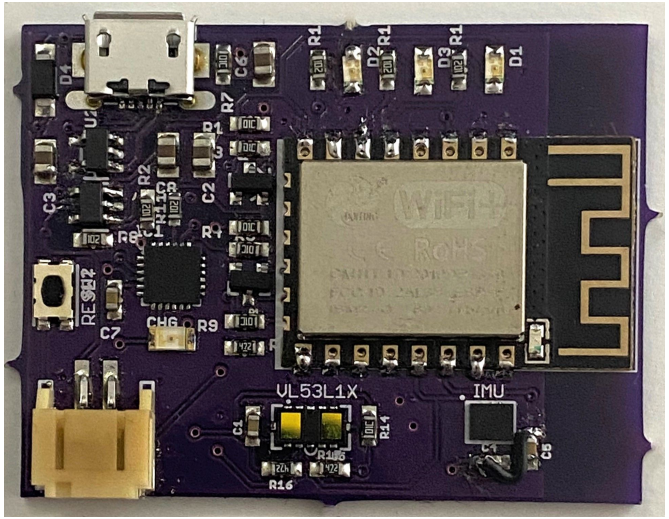




- **Kinematic Calibration**
  - Approach is to calibrate the robot's **kinematic chain**
  - Using IMU data to calibrate the robot
  - Human pose estimation (e.g. XSens), different problem, data fusion, expensive
- **Skin Units**
  - Most are contact-based sensors, and cannot sense the robot's nearby space
- ***Skin* Calibration**
  - I-Cub calibration
  - Mittendorfer calibration
    - Reference work, but performance is not feasible for accurate **nearby space perception**

The Skin

- **$36**
- Low power consumption, up to **10** hours usage
- Small 33x36mm unit and wireless
- Equipped with:
  - LSM6DS3 iNEMO
  - VL53L1X ToF Sensor (4m)
- IMU data: 100Hz, Proximity Data: 50Hz
- Kudos to **Mary**!

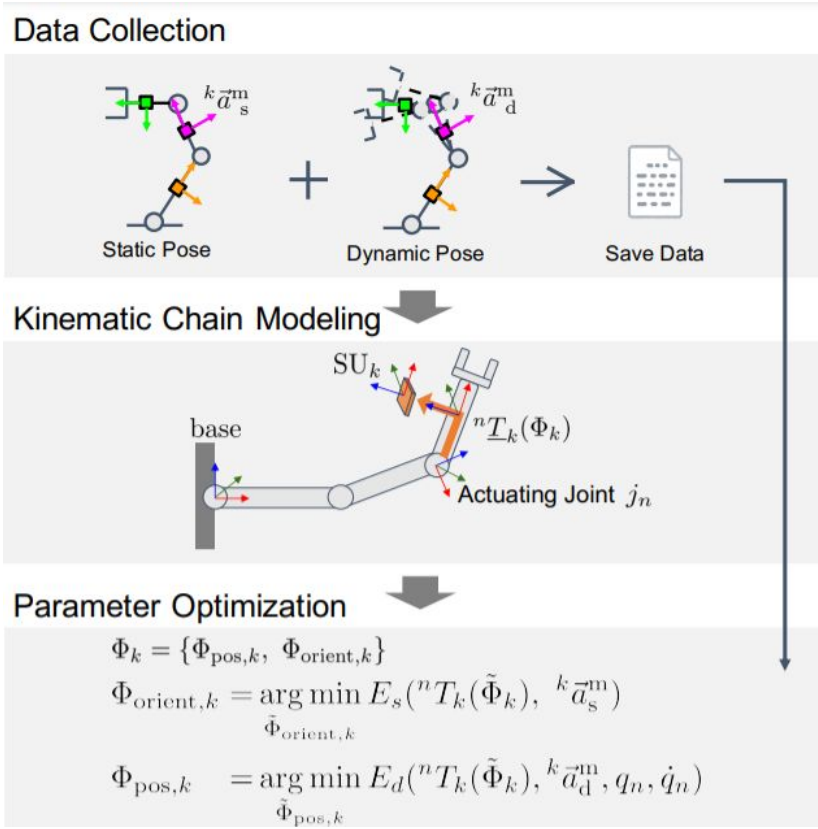


The skin unit

A Plug and Play Robotic Skin: Skin Calibration

- Skin Calibration:
  - Desire the **6D** pose with respect to the nearest link
- Procedure:

1. Data **Collection**
2. Kinematic Chain **Modeling**
3. DH Parameter **Optimization**



Data Collection

- For **P** poses:
- **Static** Data Collection
  - Capture data while robot is stationary
  - Strong baseline for static forces acting on the robot
- **Dynamic** Data Collection
  - At each pose, the robot then moves in an **oscillatory** pattern
  - Joint by joint according to:

$$\dot{q}_p = A \sin(2\pi f t)$$

Joint velocity at pose p

Amplitude

frequency

time

Modified DH Parameters for SU Estimation

- Modified DH parameters:
  - **2** parameters from joint to virtual joint
  - **4** parameters from virtual joint to SU
- Represented by:

Rotation matrix

position

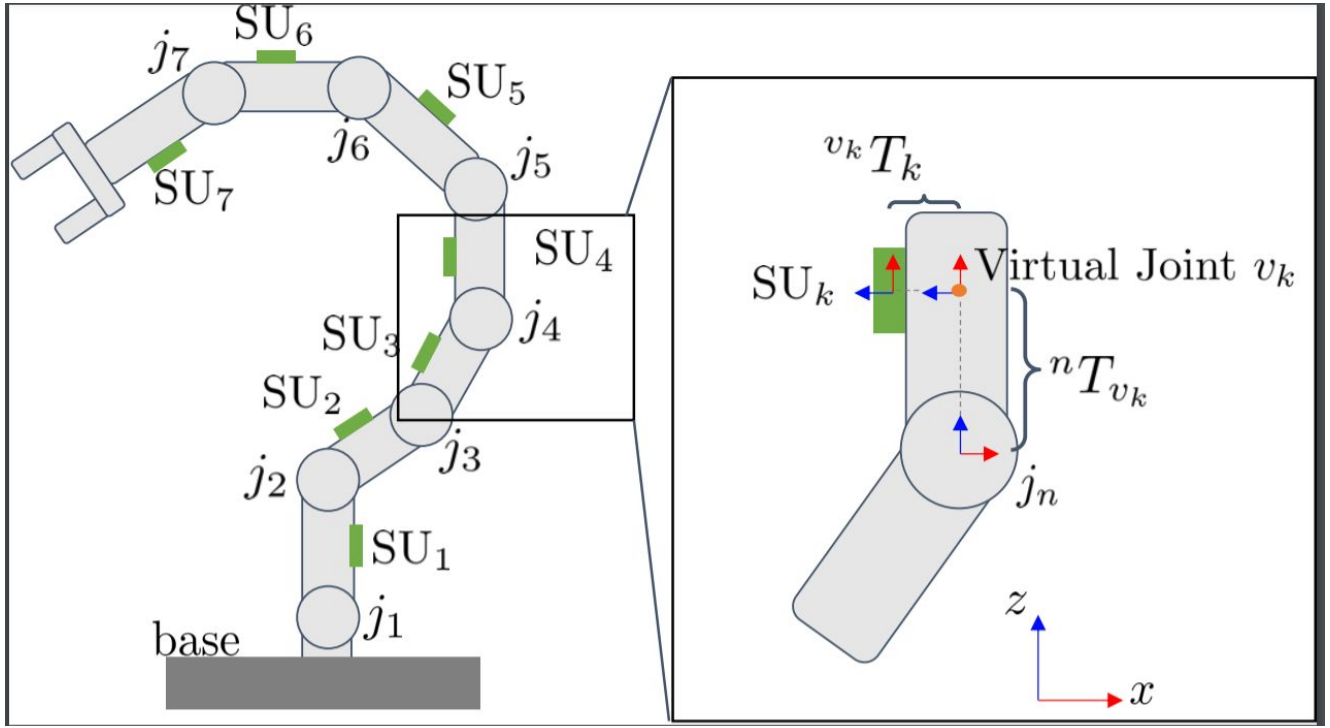
$$ {}^{n}T_{k} = \begin{bmatrix} {}^{n}R_{k} & {}^{n}\vec{r}_{k} \\ 0 & 1 \end{bmatrix} = {}^{n}T_{v_k}\left[\Phi\left(v_k\right)\right] \cdot {}^{v_k}T_{k}\left[\Phi\left(SU_k\right)\right] $$

Transformation to SU k from joint n

Transformation to virtual joint from joint n

Transformation to SU k from virtual joint k

Transformation Matrices from joint **n** to SU **k**



Acceleration Modeling

- Later, optimization will rely on modeling acceleration of SUs from the accelerometer data
- How to correctly model the acceleration of the SU in its frame of reference?

$$\underbrace{{}^{k}\vec{a}_k(\Phi_k, q_n, \dot{q}_n, \ddot{q}_n)}_{\text{SU k acceleration in its frame}} = \underbrace{{}^{k}R_n(\Phi_k, q)}_{\text{Rotates to SU k frame}} \cdot \left[\left(\vec{g}_k + \vec{a}_{\text{cen},k} + \vec{a}_{\text{tan},k} \right) \right]$$

gravity | Centrip. acc | Tangential acc

Acceleration Modeling, continued

- One problem with the joint accelerations...

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

Centrip. acc

Position of SU k in joint n's frame

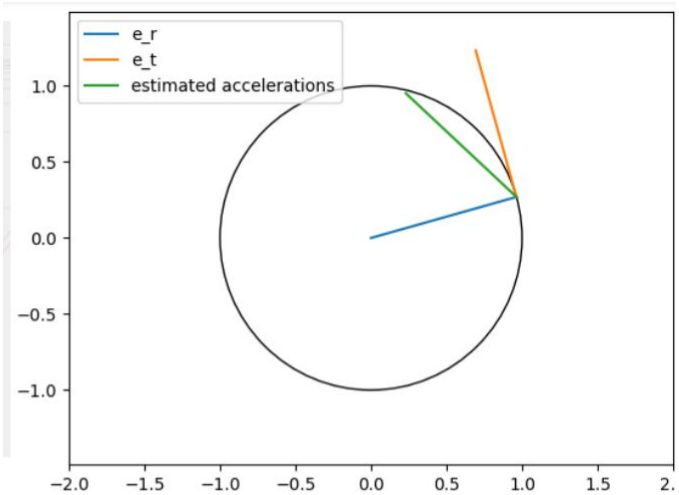
$$\vec{a}_{\text{tan},k}(\Phi_k, q_n, \ddot{q}_n) = \vec{\ddot{q}}_n \times {}^n\vec{r}_k(\Phi_k, q)$$

Tang. acc

Tangential Acceleration Optimization

- Use second numerical differentiation
- But there's an extra centripetal force and gravity!

Graph of centripetal (e_r), tangential (e_t), and estimated acc



Static Error Function

- First, optimize static error function for rotational parameters

$$E_s(k) = \frac{1}{P}\sum_{p=1}^{P}\left|\underbrace{{}^{b}R_n}_{\text{Rotation from joint to base}} \cdot \underbrace{{}^{n}R_k(\Phi_k)}_{\text{Rotation from SU to joint}} \cdot \underbrace{{}^{k}\vec{a}^{m}_{k,p}}_{\text{Measured acc}} - \underbrace{{}^{b}\vec{g}}_{\text{gravity}}\right|^2$$

Then, Freeze Rotational Parameters

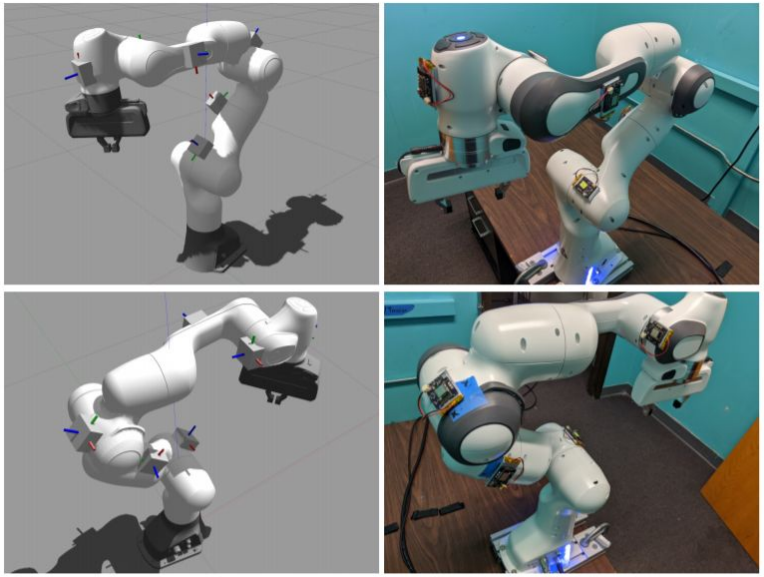
Dynamic Error Function

- Then, optimize dynamic error function for positional parameters

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

A Plug and Play Robotic Skin: Experiment Results

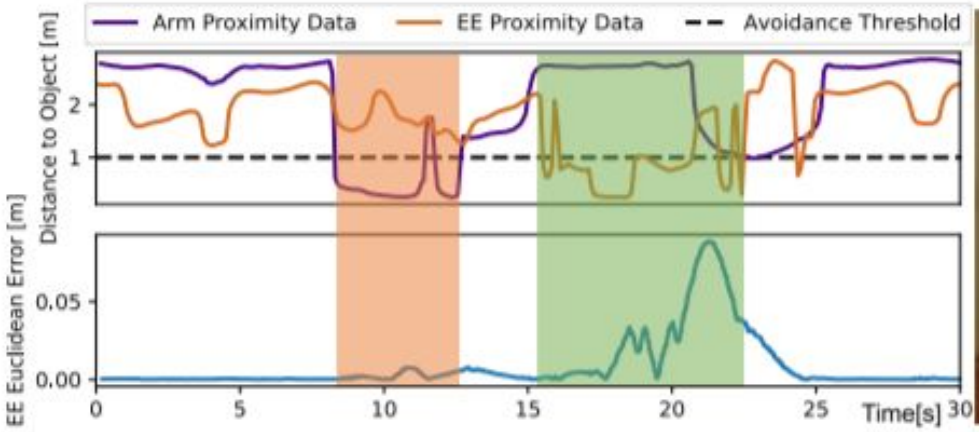
- Tested in ROS+Gazebo on Franka Panda
- Mittendorfer's method: completely unusable
- **Modified** Mittendorfer's method (**mMM**)
- 4 different configurations, 10 times
- Average error: position (cm), orientation
  - **mMM: 4.4, 2.0**
  - **Ours: 0.66, 0.04**
- Real life:
  - **mMM: 9.2, 0.43**
  - **Ours: 5.1, 0.12**

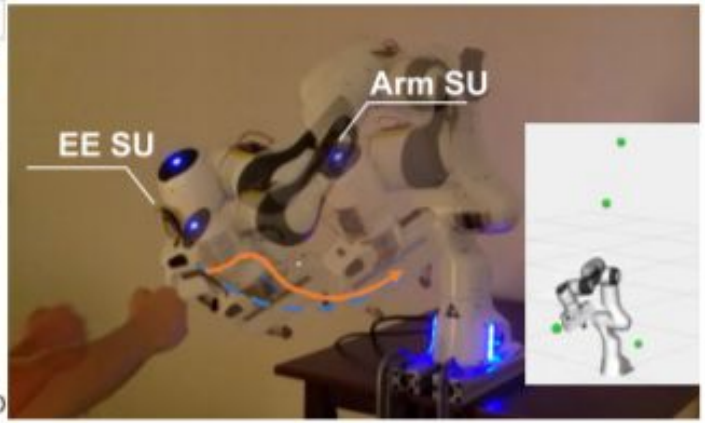


Result of Calibration Actual SU Poses

A Plug and Play Robotic Skin: Controller Example

- Obstacle avoidance is demonstrated with the calibrated SUs to complete the system -- the **application**



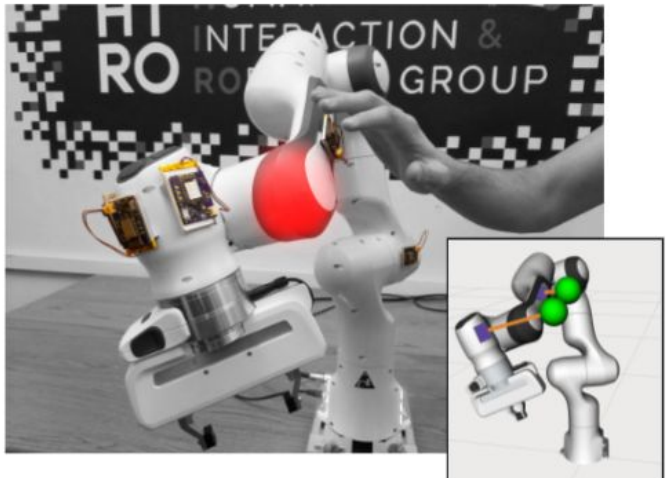


Proximity Data vs EE Error

Table of Contents

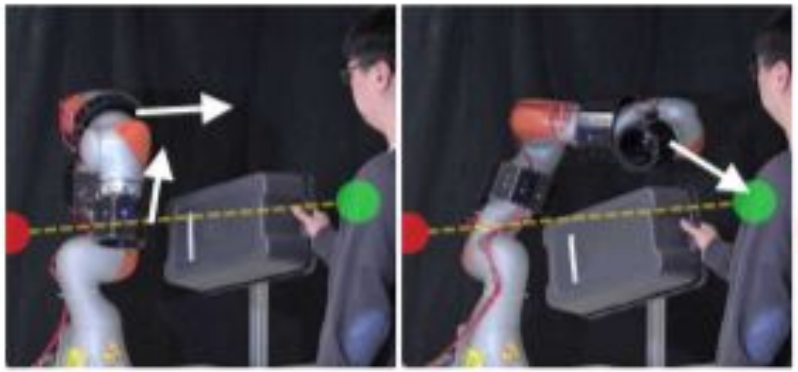
Implicit Contact Anticipation via Distributed Whole-Body Sensing

- Goal:
  - Enable the transition from avoidance to contact using whole body, nearby space perception



Related Work Part 1

- Collision Avoidance:
  - Has focused on complete obstacle avoidance
  - Safety zones
  - External cameras and onboard sensor methods still focus on avoidance
  - “Smart” avoidance is **still** avoidance

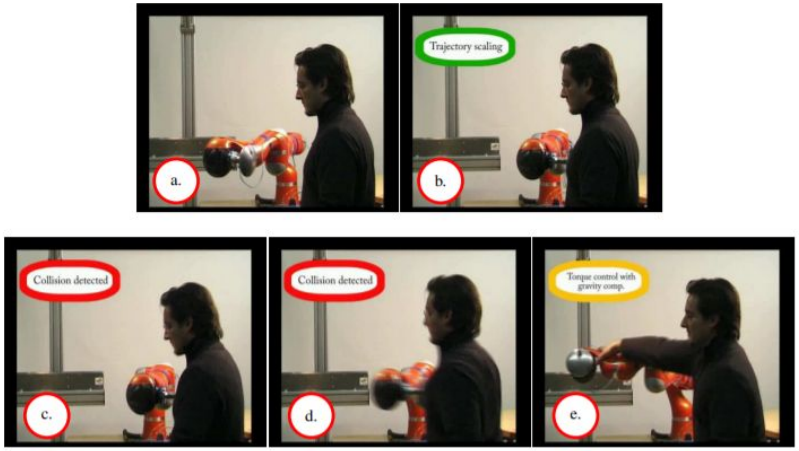


Ding 2020: The robot goes around the object

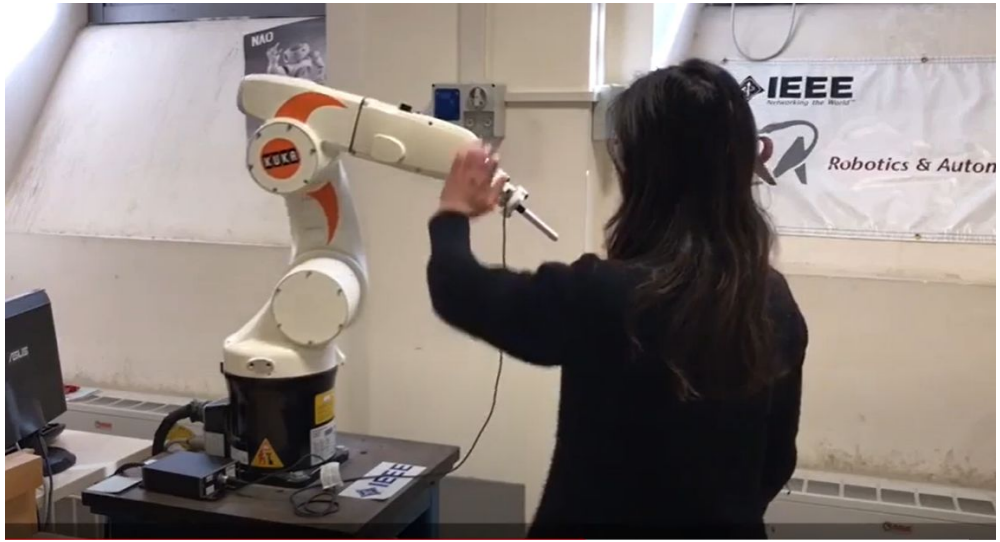
Related Work Part 2

- Contact Detection and Reaction:
  - No perception to prepare and brace for contact
  - **Zero** focus on what happens before

Collision with a human



A human interacts with the robot

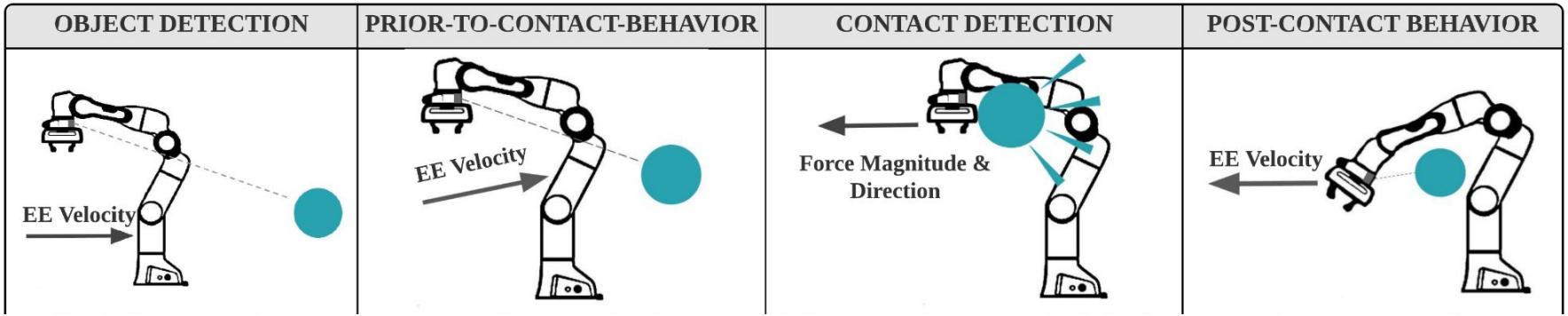


Related Work Part 3

Flacco: Integrated control for pHRI, 2012

- Avoidance + Detection
  - Does not focus on the **transition**
  - Avoidance: completely repulsive, developed ignoring the possibility of contact

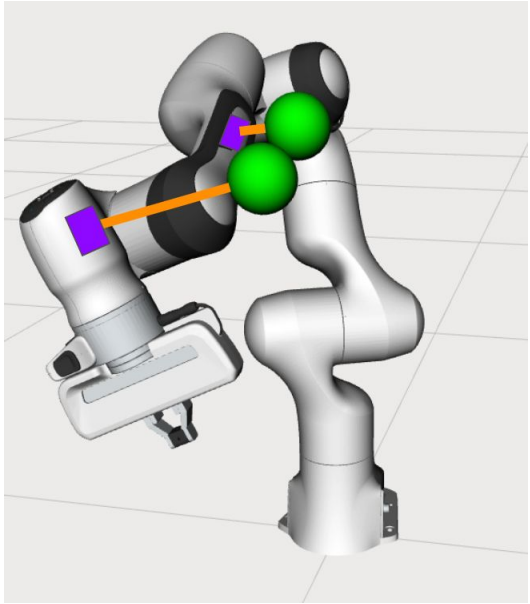
Framework

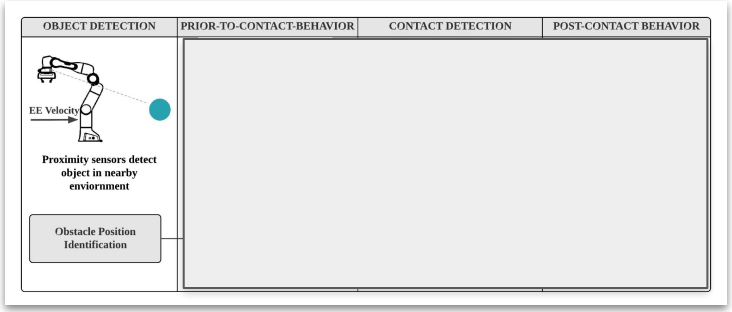

OBJECT DETECTION
PRIOR-TO-CONTACT-BEHAVIOR
CONTACT DETECTION
POST-CONTACT BEHAVIOR
EE Velocity
EE Velocity
Force Magnitude &
Direction
EE Velocity

Object Detection with Proximity Data

- We use the same units from before
- An obstacle position can be computed as with the calibrated DH parameters

Obstacles detected by the robot in RVIZ

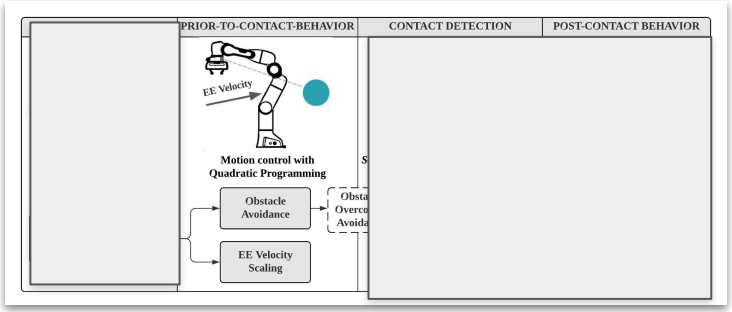




Prior To Contact Behavior

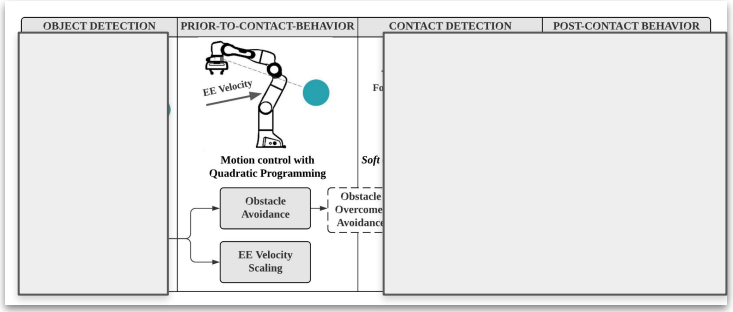
- Formulate the problem as a Quadratic Programming problem
- Include damping term and middle joint limits term

$$g(\dot{\mathbf{q}}) = \underbrace{\frac{1}{2}(\dot{\mathbf{x}}_\mathbf{d} - \mathbf{J}\dot{\mathbf{q}})^\top(\dot{\mathbf{x}}_\mathbf{d} - \mathbf{J}\dot{\mathbf{q}})}_{\text{Main task error}} + \underbrace{\frac{\mu}{2}\dot{\mathbf{q}}^\top\dot{\mathbf{q}}}_{\text{Damping term}} + \underbrace{\frac{k}{2}(\dot{\mathbf{q}}_\mathbf{mid} - \dot{\mathbf{q}})^\top(\dot{\mathbf{q}}_\mathbf{mid} - \dot{\mathbf{q}}).}_{\text{Middle joint term}}$$



Prior To Contact Behavior

- Velocity Scaling -- the desired EE velocity **scales down** as objects come closer
  - Avoidance still possible
- Limit the approach velocity of the robot to the object
  - But allows movement **around** the object



Dynamic Contact Thresholding

- How to detect when **contact** is made?
- Data is external Cartesian force data
- We reduce the impact of outliers on the data (sliding window) we use
  - But, when contact is made, the force will go up a lot

Upper threshold

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

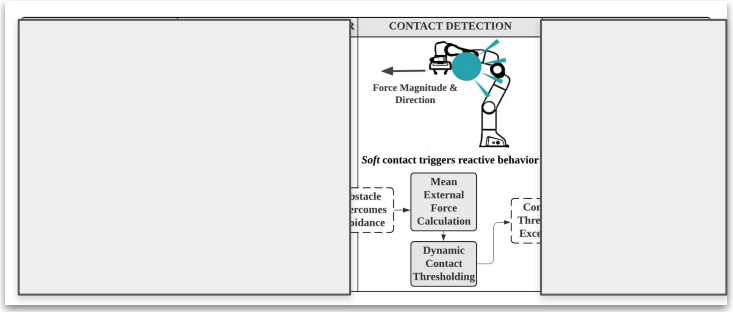
Lower threshold

Mean

Base force

Std of force

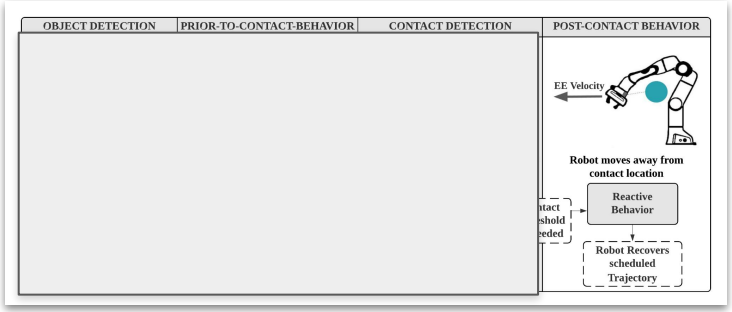
Based on obs



Contact Reaction

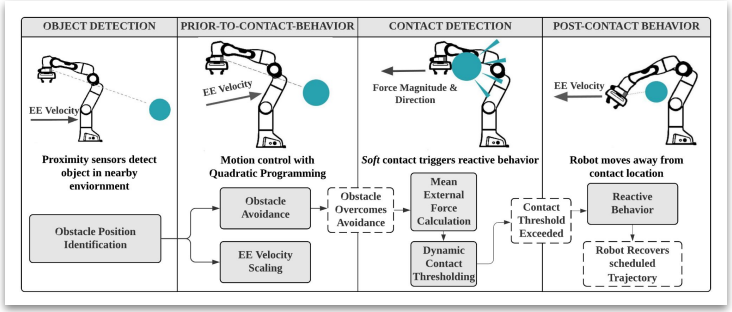
- Upon contact, the robot moves with a velocity based on the force magnitude and direction
- Velocity linearly decays over time, then the robot resumes the main task

$$F_{ext} = F_{reading} - \mu$$



Contact Detection

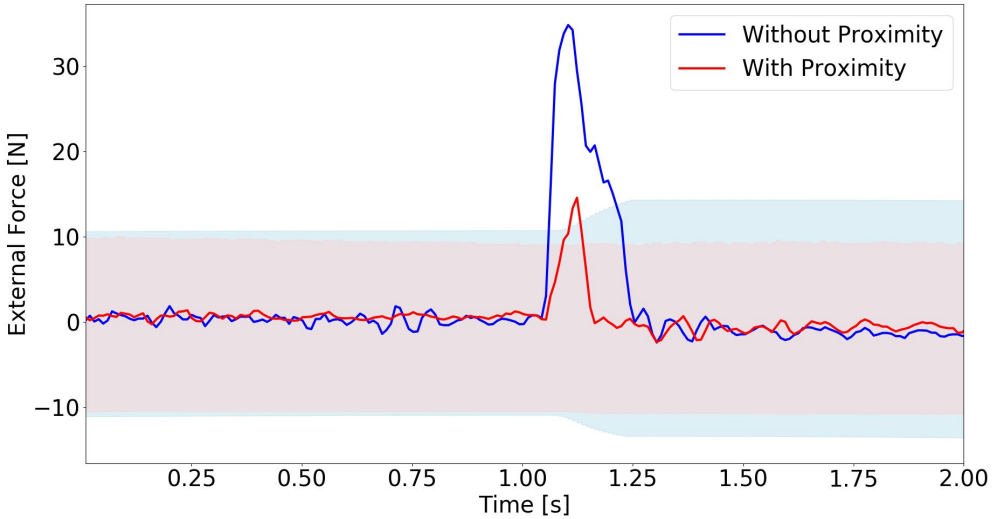
- **Robustness to measurement noise**
  - Only requires noticeable data beyond the mean
- **Anticipated contact from obstacles**
  - As obstacles come close, the thresholds get tighter
- **Increased contact sensitivity as velocity decreases**
  - As the robot slows down, the data is less noisy and the thresholds tighten

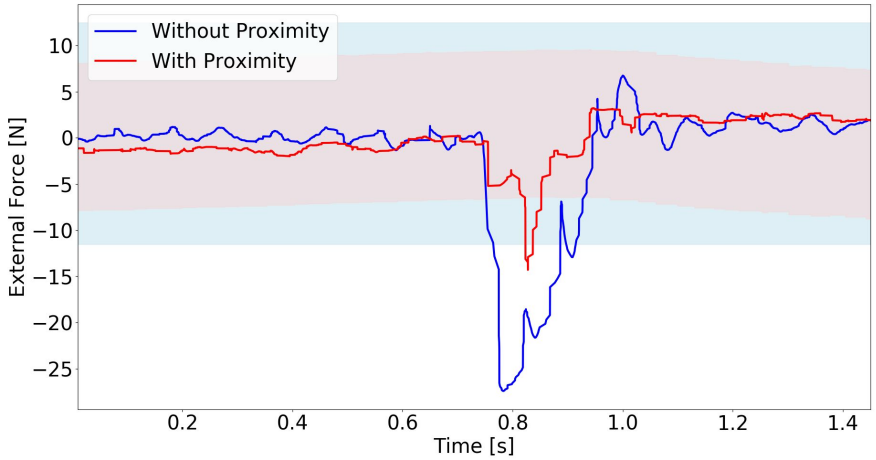


Experiment Scenarios

- **Static** Obstacle Collision
  - Static object is placed on two sides of robot
- Obstacle **Avoidance**
  - The retained ability for avoidance is also shown with a human
- **Dynamic** Obstacle Collision
  - Contact is made by a dynamic object

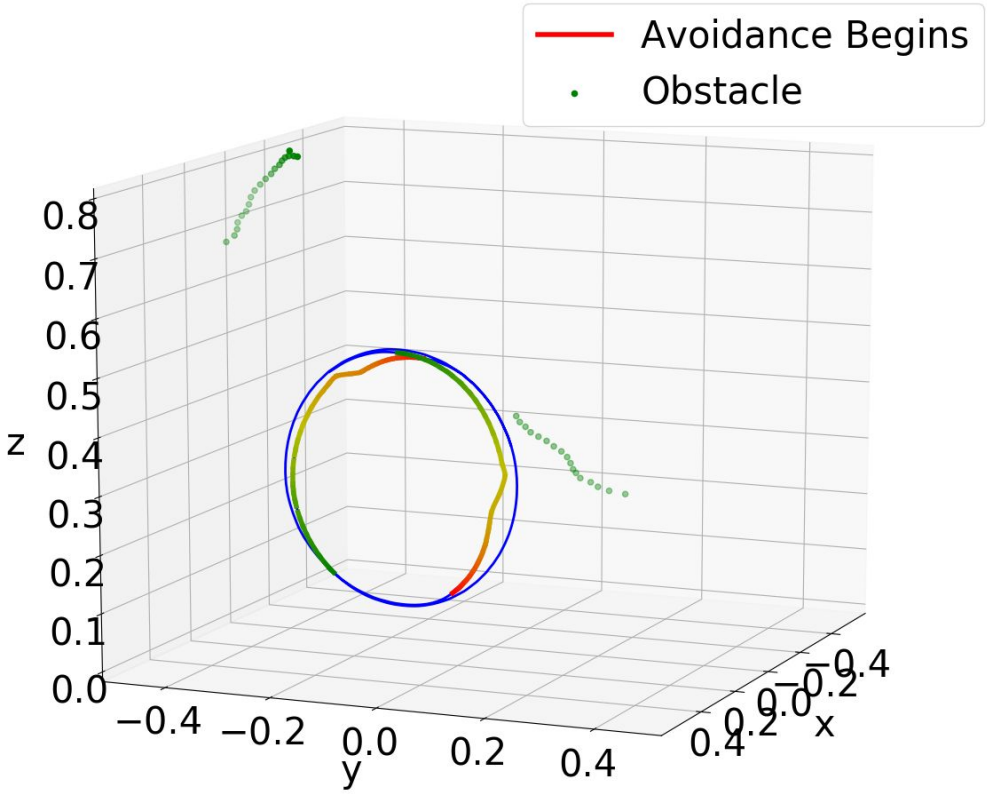
Static Obstacle Collision





Forces (and thresholds) with and without proximity

Obstacle Avoidance



Blue Circle: Nominal trajectory

Dynamic Contact: Proximity Data vs. **Forces**

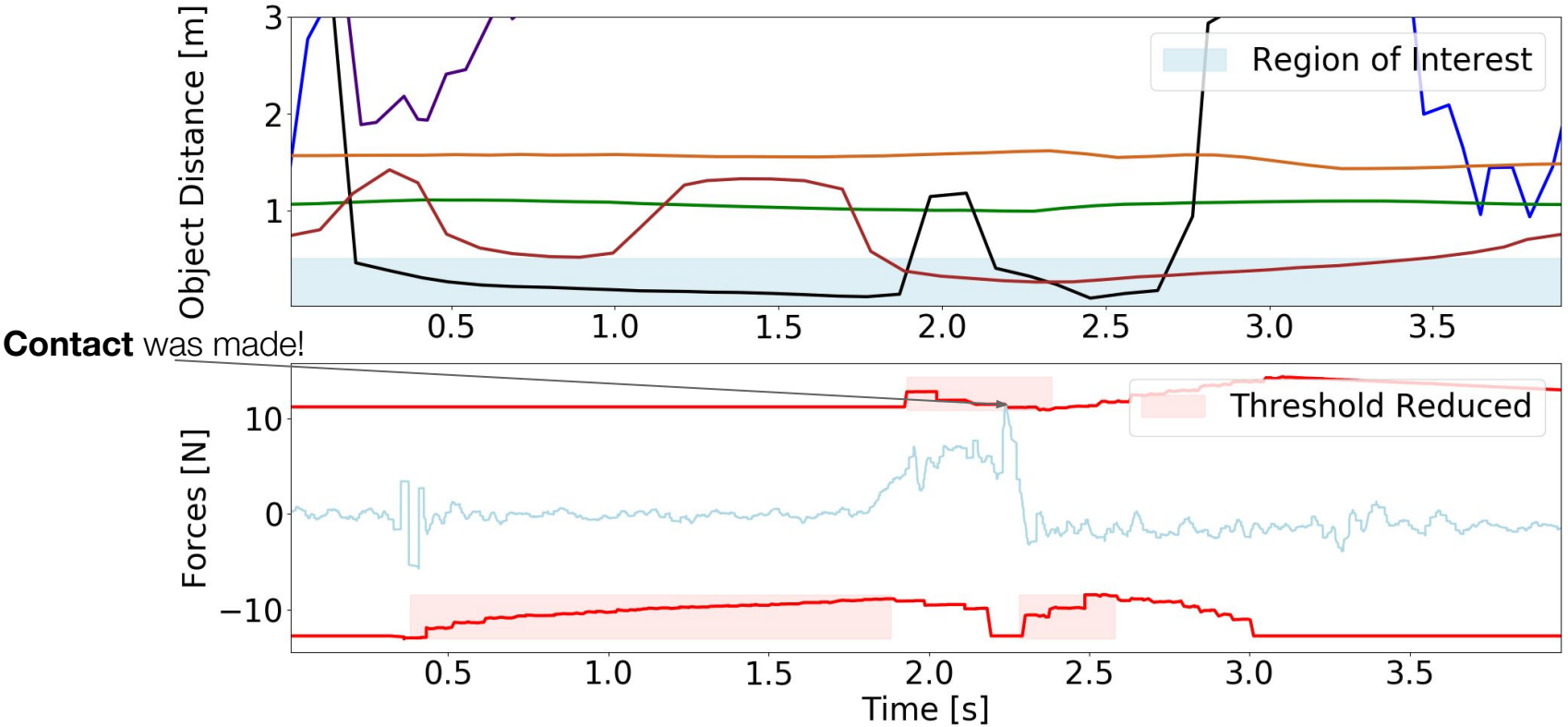


Table of Contents

Future Work

- 1st contribution
  - Analysis of how important accuracy is effective collaboration
- 2nd contribution
  - Incorporate IMU information
  - Force sensors for localization
- Overall:
  - **Dense, whole** body sensing
  - Hard to conceptualize the future direction without the above

Future Plans

- Microsoft Software Engineer
  - ML+RL (but not research)
- Ph.D in Robotics soon after
- 14er **spree**
- Become **fluent** in Chinese
- Iron Man by **30 years old**
  - But there is hope, thanks to **Alessandro**

My mom and I on Torrey's peak

Questions?