



Compliant Teleoperation using Hand Tracking on the Franka Panda manipulator within ROS/Gazebo framework

A semester project in the course of
Project in Advanced Robotics

written by

Martin Androvich
marta16@student.sdu.dk
92411942

Victor Melbye Staven
vista17@student.sdu.dk
454607

Peter Ørholm Nielsen
pniel17@student.sdu.dk
454701

The code for this project is available at
<https://github.com/teleop-grasp>

University of Southern Denmark (SDU)
Technical Faculty (Faculty of Engineering)

May, 2022

Abstract

The teleoperation system uses the ROS/Gazebo framework to interface a 7-degrees of freedom (DOF) Franka collaborative robot (cobot), allowing to establish a platform, both in simulation and reality, in which the cobot with an attached gripper is able to track the pose (position and orientation) as well as mimic the binary gesture state of an operator's right hand.

Compliant robot control is utilized to render a desired end-effector (EE) impedance, allowing for compliant interaction with the operator. The cobot is actuated using a Cartesian Admittance controller implemented using the ROS Control framework via hardware interfaces provided by `franka_ros`, such that the same controller can be used both in simulation and on the real robot.

Hand tracking is realized using the Hands module of the deep neural network (DNN) based model provided by the MediaPipe framework. Given an RGB image, estimated hand landmarks are used to provide both binary gesture estimation as well as exponential moving average (EMA) filtered pose estimation with an intuitive hand representation.

Integration of the teleoperation system is achieved by mapping relative changes in the estimated hand pose to a desired EE pose, which is commanded to the Franka controller together with the desired gripper state. Furthermore, safety constraints are added to limit the workspace of the manipulator, as well as handle any sudden changes in velocity or acceleration of the hand pose.

The integration was tested on the real manipulator, in which an operator is able to leverage the system in order to grasp and move an object held by a participant, whilst the participant is able to resist. A demonstration is available at <https://youtube.com/shorts/IuGVKiLKSjc>.

Contents

Contributions	iii
Acknowledgments	iii
Acronyms and Terms	iv
1 Introduction	1
1.1 Project description	1
1.2 Pipeline	1
1.3 Structure	2
2 Hand tracking	3
2.1 Hand representation	3
2.2 Gesture estimation	3
2.3 Pose estimation	4
2.4 Evaluation	5
3 Robot control	6
3.1 Control framework	6
3.2 Cartesian Admittance Controller	7
3.3 Gain selection	9
3.4 Gripper	10
3.5 Evaluation	10
4 Integration	13
4.1 Safety	13
4.2 Mapping	14
4.3 Motion prediction	14
4.4 Evaluation	15
4.5 Conclusion and Discussion	15

Contributions

The contributions to this project are summarized in Table 1.

Section	Primary assignee
1 Introduction	Martin Androvich
1.1 Project description	Martin Androvich
1.2 Pipeline	Martin Androvich
1.3 Structure	Martin Androvich
2 Hand tracking	Peter Ørholm Nielsen
2.1 Hand representation	Peter Ørholm Nielsen
2.1 Hand representation (Implementation)	Martin Androvich
2.1 Hand representation (Debugging/Review)	Peter Ørholm Nielsen
2.3 Pose estimation	Peter Ørholm Nielsen
2.3 Pose estimation (Implementation)	Peter Ørholm Nielsen
2.2 Gesture estimation	Peter Ørholm Nielsen
2.2 Gesture estimation (Implementation)	Peter Ørholm Nielsen
2.4 Evaluation	Peter Ørholm Nielsen
3 Robot control	Martin Androvich
3.2 Cartesian Admittance Controller	Martin Androvich
3.2 Cartesian Admittance Controller (Implementation)	Martin Androvich
3.2 Cartesian Admittance Controller (Debugging/Review)	Peter Ørholm Nielsen
3.3 Gain selection	Martin Androvich
3.4 Gripper	Victor Melbye Staven
3.4 Gripper (Implementation)	Victor Melbye Staven
3.5 Evaluation	Martin Androvich
4 Integration	Victor Melbye Staven
4.2 Mapping	Victor Melbye Staven
4.2 Mapping (Implementation)	Martin Androvich
4.3 Motion prediction	Victor Melbye Staven
4.4 Evaluation	Victor Melbye Staven
4.5 Conclusion and Discussion	-

Table 1: Overview of contributions to the project.

Acknowledgements

We would like to thank our supervisor Christoffer Sloth for the guidance he provided during the project.

Acronyms

cobot collaborative robot.

DNN deep neural network.

DOF degrees of freedom.

EE end-effector.

EMA exponential moving average.

KDL Orocos Kinematics and Dynamics Library.

MCP metacarpophalangeal.

PIP proximal interphalangeal.

ROS Robot Operating System.

RPY Roll-Pitch-Yaw.

Slerp spherical linear interpolation.

TIP distal phalanx.

URDF Unified Robot Description Format.

Terms

collaborative robot (cobot) Robots intended for direct human robot interaction within a shared space or where humans and robots are in close proximity.

distal phalanx (TIP) Distal phalanx, aka. the tip of the finger.

metacarpophalangeal (MCP) The metacarpophalangeal joint (MCP joint), or knuckle, is where the finger bones meet the hand bones.

pose The position and orientation of a coordinate frame.

proximal interphalangeal (PIP) The proximal interphalangeal (PIP) joints are commonly known as the middle knuckles of the fingers..

robot dynamics variables The variables of the joint space robot dynamics for a given robot state, including the mass matrix, $\mathbf{M}(\mathbf{q})$, Coriolis matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and gravity vector, $\mathbf{g}(\mathbf{q})$.

robot state The state of the robot in terms of joint space position, \mathbf{q} , and velocity, $\dot{\mathbf{q}}$.

Roll-Pitch-Yaw (RPY) Orientation described by Roll-Pitch-Yaw, using the ZYX Tait-Bryan sequence.

rotatum The derivative of torque with respect to time, i.e., the rate of torque.

1 Introduction

1.1 Project description

This project aims to develop a teleoperation system in which a collaborative robot (cobot) with an attached gripper is able to track the pose (position and orientation) as well as mimic the binary gesture state of an operator's right hand, as shown in Fig. 1. The hand is captured using an RGB camera, in which a neural network is used to estimate the pose and gesture of the hand.

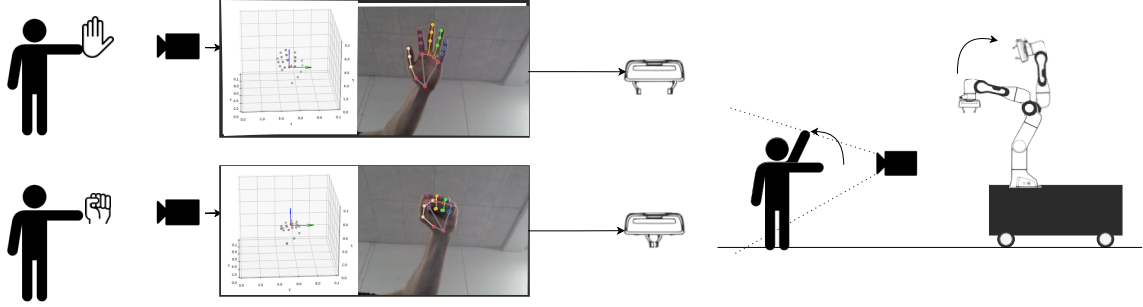


Fig. 1: Overview of the teleoperation system.

The teleoperation system makes use of a Franka 7-degrees of freedom (DOF) cobot, which is interfaced via the ROS/Gazebo framework [1] and the `franka_ros` package [2]. The cobot must be equipped with a compliant controller, such that the cobot can be used to interact with participants with a desired dynamic behavior. An example application would be to lift the wrist of a participant via teleoperation, unless the participant resists.

1.2 Pipeline

The teleoperation system is implemented as a set of decentralized modules using the publisher-subscriber design pattern. The pipeline of the teleoperation system is shown in Fig. 2.

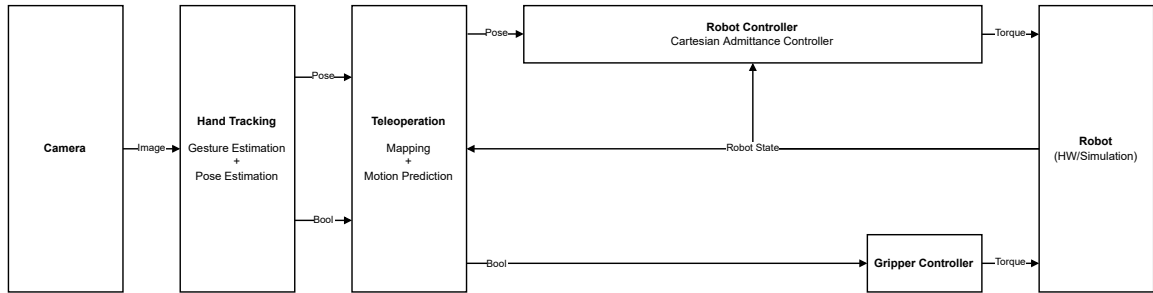


Fig. 2: Pipeline and modules of the teleoperation system.

The **Camera** module continuously captures and publishes RGB images. The **Hand tracking** module uses the published RGB images to estimate a hand pose and gesture state (boolean), which are then published. The **Teleoperation** module maps the tracked hand pose to a desired end-effector (EE) pose, which is commanded to the robot controller, together with the gesture state. At last, the **Robot control** module implements a Cartesian Admittance Controller, which computes and commands the desired joint torques necessary to track the desired pose, both on the real robot and in simulation.

1.3 Structure

Development of the teleoperation system greatly benefits from a modular project structure, as well as access to common tools and libraries within the robotics domain. Robot Operating System (ROS) [3] is a state-of-the-art robotics framework and tends to be a standard for robotics application development [4]. It provides a modular software framework with reusable components and a communication pipeline, while also seamlessly integrating with the Gazebo simulator [5].

The decentralized modules of the teleoperation system project are structured using ROS packages and meta-packages, all of which is built within a `catkin` workspace [6]. A package encapsulates whatever necessary (code, scripts etc.) to constitute a useful module, whereas a meta-package groups several packages specific to some domain.

The packages of the teleoperation system are shown in Fig. 3. The project meta-package `teleop_grasp` provides an entry point to the teleoperation system (launch and configuration files), as well as implements the pipeline integration module. The `franka_ros` meta-package provides a ROS interface to the Franka robot, both in simulation and on the real robot. Finally, the `franka_controllers` package implement robot control module, whereas the `hand_tracking` package implements the hand tracking module.






Package		Description
teleop_grasp		Project package of the teleoperation system.
teleop_grasp		Integration of the teleoperation pipeline.
franka_ros		ROS integration of Franka robot.
franka_gazebo		Simulation of the Franka robot in Gazebo.
...		
franka_controllers		ROS Control controllers for the Franka robot.
hand_tracking		Hand pose- and gesture estimation using MediaPipe.
ros_utils		Utilities for interfacing ROS/Gazebo/MoveIt/Eigen etc.

Fig. 3: Packages of the teleoperation system.

This approach enables a decoupled development process, where packages are either completely decoupled or only loosely coupled, allowing packages to be developed and tested independently. Furthermore, using the ROS/Gazebo allows to implement and test the system in simulation, before running it on the real robot.

2 Hand tracking

The hand tracking module continuously reads RGB images and outputs an estimate of the hand pose and gesture of a right hand, if one is detected. The hand tracking module is a standalone ROS package implemented as a Python module using the MediaPipe [7] framework from Google, enabling fast and robust solutions using pre-trained deep neural network (DNN) models to solve problems such as tracking, segmentation, detection, and pose-detection of human-like features, such as a hand.

Specifically, the Hands module [8] of the MediaPipe framework is utilized, which is a high-fidelity hand and finger tracking DNN based model, which estimates 21 3D landmarks of a hand from a single RGB image. It consists of a pipeline of multiple models, each performing a designated task. The first step of the pipeline is a palm detection model, which operates on the whole image to return an oriented bounding box of the palm. The next step is a hand landmark model, which operates on the cropped region of the image defined by the oriented bounding box produced by the palm to infer 21 high-fidelity 3D hand landmarks. Having the hand landmark model operate on cropped image regions allows for faster throughput when inferring on consecutive images since the palm detection model is only needed once the hand landmark model no longer can detect the hand in the cropped images.

2.1 Hand representation

The inferred landmarks from the hand landmark model are shown in Fig. 4. Each landmark contains two 3D coordinate representations: real-world and normalized image coordinates. The 3D real-world hand-knuckle coordinates are given in meters with the origin of the hand's approximate geometric center. The normalized 3D coordinates of $x, y \in [0, 1]$ are relative to the image width and height, respectively. The z value corresponds to an estimate of the depth, with the wrist being the origin, in which lower z values correspond to the landmark being closer to the camera.

Furthermore, the hand landmark detector detects what type of hand it has predicted, i.e., left, right, or both; however, only the right hand is considered to reduce complexity.

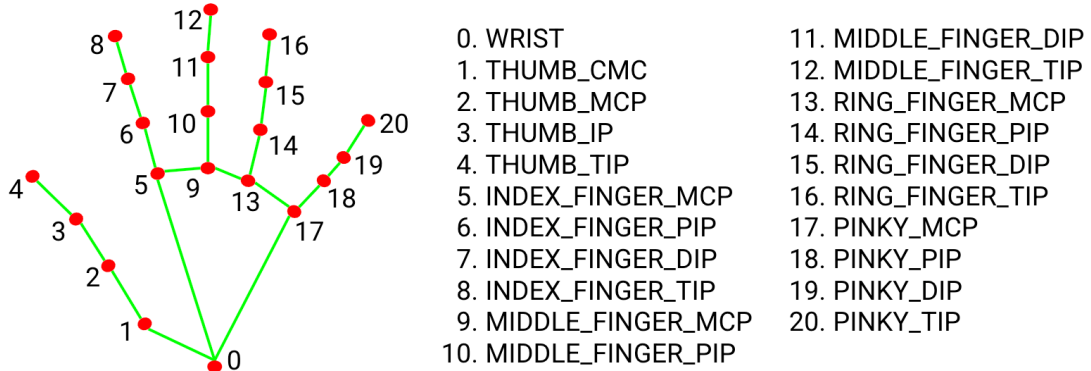


Fig. 4: Hand landmarks [9].

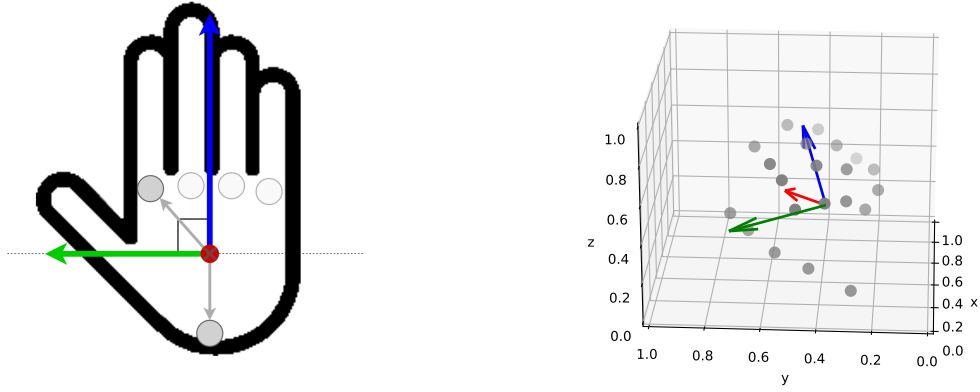
2.2 Gesture estimation

Given the normalized 3D hand landmarks, the gesture estimation is achieved by comparing the y-component of the distal phalanx (TIP) and proximal interphalangeal (PIP) of the fingers. If the TIP is less than the PIP for all fingers but the thumb, then the hand is considered closed.

2.3 Pose estimation

Given the estimated landmarks, an intuitive representation of the hand pose $\mathbf{T}_h = (\mathbf{p}_h, \mathbf{R}_h)$ is given by a frame that is invariant to the position of the fingers. Given the 3D real-world landmarks, a frame is established by making use of the wrist landmark and index finger metacarpophalangeal (MCP) joint landmark, since these constitute a plane that is invariant to the movement of the fingers.

Constructing a coordinate frame from the plane is achieved using Gram Schmidt, to get two orthogonal vectors describing the y -axis and z -axis, with y and z pointing in the direction of the thumb and fingers, respectively, in order to mimic the pose of the hand seen from the camera. The x -axis is given by a cross product between the inferred y and z axes, allowing to represent the orientation \mathbf{R}_h of the hand, as shown in Fig. 5.



(a) Construction of frame using Gram Schmidt.

(b) 3D real-world landmarks and the constructed frame.

Fig. 5: Constructed frame from the inferred real-world 3D landmarks.

The hand position \mathbf{p}_h is inferred from the 3D normalized image coordinates. However, evaluation has shown that the accuracy of depth component, corresponding to the x -axis of the hand frame, was inadequate for teleoperation due to noisy estimations. Attempts to infer the depth using Body Pose module of MediaPipe also yielded poor accuracy and stability; hence only the y and z components will be estimated, in which the x component is kept constant.

The hand pose is filtered using an exponential moving average (EMA) filter to provide smoother transitions between consecutive hand poses. For the translational part, the EMA filter is given by

$$\mathbf{p}_t = \begin{cases} \mathbf{p}_0, & t = 0 \\ \alpha_p \cdot \mathbf{p}_t + (1 - \alpha) \cdot \mathbf{p}_{t-1}, & t > 0 \end{cases}, \quad (1)$$

where \mathbf{p}_t is the position, y and z , and α_p is the discount factor.

Likewise, an EMA filter for the orientation is provided using spherical linear interpolation (Slerp) [10]. The orientation \mathbf{R}_h is transformed into a unit quaternion q_h , for which Slerp is performed as

$$\text{Slerp}(q_0, q_1, \alpha_o) = q_0(q_0^{-1}q_1)^{\alpha_o}, \quad (2)$$

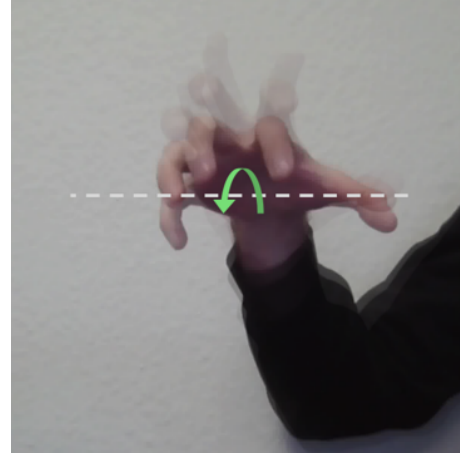
where $\alpha_o \in [0, 1]$ is a scalar expressing the discount factor of the rotation from q_0 to q_1 .

2.4 Evaluation

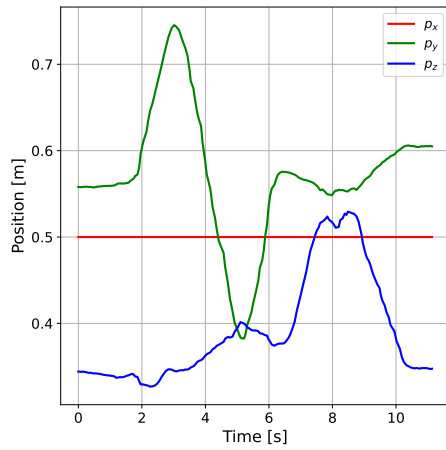
The hand tracking module is evaluated by capturing hand movement along the y-axis followed by a rotation around the y-axis, as shown in Fig. 6(a) and Fig. 6(b), respectively. The resulting estimated hand pose (position and orientation) is filtered with EMA parameters $\alpha_p = 0.4$, $\alpha_o = 0.1$, and is shown in Fig. 6(c) and Fig. 6(d).



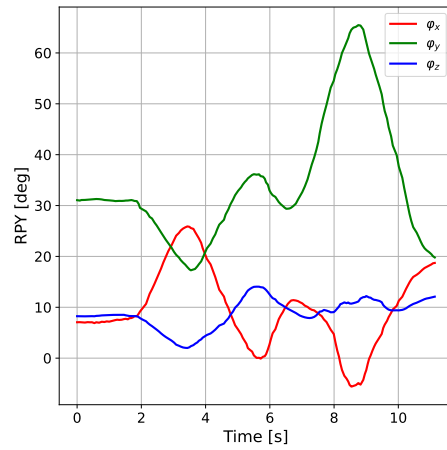
(a) Movement along the y-axis at $t \in [0, 3]$.



(b) Rotation around the y-axis at $t \in [6, 9]$.



(c) Estimated hand position.



(d) Estimated hand Roll-Pitch-Yaw (RPY) orientation.

Fig. 6: Evaluation of the hand pose estimation.

By inspection of the estimated poses, it is seen that the hand tracking module achieves desired performance.

3 Robot control

For the teleoperation system in which a manipulator is interacting with a human, it is desirable to ensure a suitable compliant behavior via robot control. This can be achieved using a decentralized control method such as **Cartesian Impedance Control**, given the differential equations of motion that describe the joint-space dynamics of a serial manipulator consisting of n rigid bodies [11], excluding the attached gripper, are governed by

$$\mathbf{M}(\mathbf{q}) \cdot \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q}) \cdot \mathbf{h}_e, \quad (3)$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the symmetric inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ represents the centrifugal and Coriolis effects, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the configuration dependent gravity vector, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the torque commanded by the motors, $\mathbf{J} \in \mathbb{R}^{6 \times n}$ is the geometric Jacobian, and the wrench $\mathbf{h}_e = [\mathbf{f}_e^T \ \boldsymbol{\mu}_e^T]^T \in \mathbb{R}^6$ is the vector of contact forces and torques acting on the end-effector.

Impedance control allows to control the force of resistance to external motions that are imposed by the environment. The end-effector renders a desired mass-spring-damper system, i.e., controlling how the robot behaves during an interaction with the environment by defining its stiffness and damping in each of the six axis of motion.

3.1 Control framework

The Franka manipulator is actuated using ROS Control [12], which is a framework for implementing hardware agnostic controllers in the ROS ecosystem. This allows to define a custom controller by inheriting the classes provided by the framework [13, 14] using the standard hardware interfaces. The `gazebo_ros_control` plugin embedded in the robot description provides a simulated hardware interface [15], which allows to control robots simulated in Gazebo using any ROS Control controller that is based one of the standard hardware interfaces.

The `franka_ros` package provides a custom hardware interface `FrankaHWSim` [16], which allows to implement a single ROS Control controller to be used in both simulation and on the real hardware. The `EffortJointInterface` is used to interface the manipulator, allowing to command desired joint torques $\boldsymbol{\tau}_d$. The robot state $[\mathbf{q}, \dot{\mathbf{q}}]$, as well as the current pose $\mathbf{T}_e = (\mathbf{p}_e, \mathbf{R}_e)$ and its derivative $\dot{\mathbf{T}}_e = (\dot{\mathbf{p}}_e, \boldsymbol{\omega}_e)$ of the end-effector, Jacobian $\mathbf{J} \in \mathbb{R}^{6 \times 7}$ and estimated EE wrench $\mathbf{h}_e = [\mathbf{f}_e^T \ \boldsymbol{\mu}_e^T]^T \in \mathbb{R}^6$, are read using the `FrankaStateInterface`. Furthermore, the `FrankaModelInterface` exposes the variables of the joint-space robot dynamics $[\mathbf{M}(\mathbf{q}), \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}), \mathbf{g}(\mathbf{q})]$ of the Franka manipulator.

The dynamic parameters used for simulating the Franka manipulator in Gazebo, specified in the Unified Robot Description Format (URDF) of the robot description, are not directly provided by Franka. Instead, these values are estimated as given by [17], where Orocos Kinematics and Dynamics Library (KDL) [18] is used to compute the robot dynamics variables.

When implementing a controller, there are several necessary conditions to uphold [19], referred to as joint limits; including limits on joint position, velocity, torque and rotatum (rate of torque). Furthermore, any discontinuities in both the joint space and Cartesian space trajectory must also be avoided. If the necessary conditions are violated, an error will abort the motion.

3.2 Cartesian Admittance Controller

Impedance control relies on using the same gains (impedance parameters) for tracking as for rendering a compliant EE. When tuning the parameters it is necessary to ensure high enough values to reject the disturbances due to model uncertainties and the approximations of the inverse dynamics computation [20], thus leading to an indirect coupling of the tracking and compliance gains.

The issue of coupled gains can be resolved by using a **Cartesian Admittance Control** in which the motion control problem is separated from the impedance control problem, as shown in Fig. 7, based on [21].

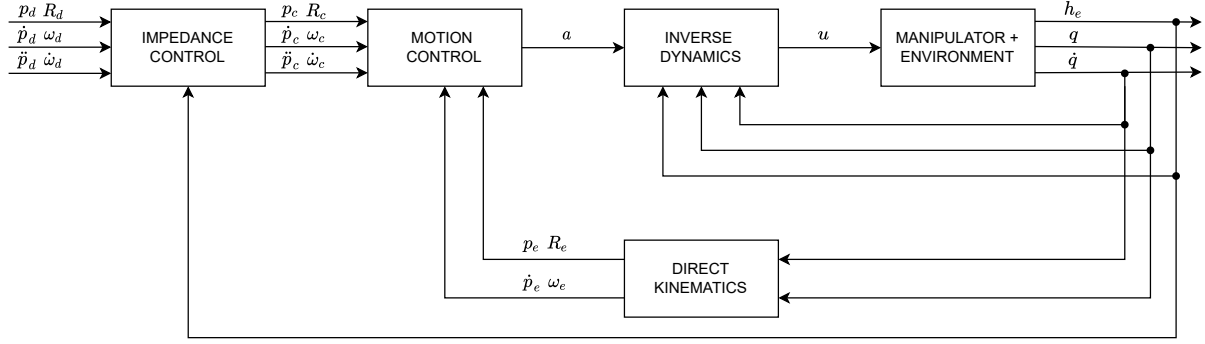


Fig. 7: Block scheme of Cartesian Admittance Control.

For impedance control, in the view of (3), given the concept of inverse dynamics, the driving torques are chosen as

$$\tau_d = \mathbf{M}(\mathbf{q}) \cdot \mathbf{J}^{-1}(\mathbf{q}) \cdot (\mathbf{a} - \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}}) + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{J}^T(\mathbf{q}) \cdot \mathbf{h}_e, \quad (4)$$

where $\mathbf{a} = [\mathbf{a}_p^T \ \mathbf{a}_o^T]^T \in \mathbb{R}^6$ is a new control input (under the assumption of perfect dynamic compensation), representing the resolved acceleration composed of a linear component \mathbf{a}_p and an angular component \mathbf{a}_o . In inverse dynamics control, the control input \mathbf{a} is designed to track a desired frame, whereas in admittance control, the objective is to track a compliant frame. Thus, \mathbf{a} is designed as to guarantee tracking of the compliant frame.

Given a desired frame $(\mathbf{p}_d, \mathbf{R}_d, \dot{\mathbf{p}}_d, \omega_d, \ddot{\mathbf{p}}_d, \dot{\omega}_d)$ and EE wrench \mathbf{h}_e , the **impedance control** block computes the compliant frame $(\mathbf{p}_c, \mathbf{R}_c, \dot{\mathbf{p}}_c, \omega_c, \ddot{\mathbf{p}}_c, \dot{\omega}_c)$ using integration. With the compliant frame and current EE frame, the **motion control** block computes a resolved acceleration \mathbf{a} , which is used in the **inverse dynamics** block to compute the desired joint torques τ_d , at last commanded to the manipulator.

The **resolved linear acceleration** \mathbf{a}_p is chosen as

$$\mathbf{a}_p = \ddot{\mathbf{p}}_c + k_{V_p} \cdot \dot{\mathbf{p}}_{ce} + k_{P_p} \cdot \mathbf{p}_{ce}, \quad (5)$$

where $\mathbf{p}_{ce} = \mathbf{p}_c - \mathbf{p}_e$, and k_{V_p} and k_{P_p} are the translational damping and stiffness gains, respectively, chosen to ensure tracking of the compliant frame. Then, the mechanical impedance for the translation part is given by

$$\mathbf{M}_p \ddot{\mathbf{p}}_{cd} + \mathbf{D}_p \dot{\mathbf{p}}_{cd} + \mathbf{K}_p \mathbf{p}_{cd} = \mathbf{f}_e, \quad (6)$$

where $\mathbf{p}_{cd} = \mathbf{p}_c - \mathbf{p}_d$, and $\mathbf{M}_p, \mathbf{D}_p, \mathbf{K}_p \in \mathbb{R}^{3 \times 3}$ are positive definite matrices specifying the translational compliance, specifically the mass, damping and stiffness, respectively. Given the translational part \mathbf{p}_d of a desired frame and contact force \mathbf{f}_e , the compliant frame can be computed using numerical integration given a time step dt at time t , as

$$\begin{aligned} \ddot{\mathbf{p}}_{cd}(t + dt) &= \mathbf{M}_p^{-1} \cdot (\mathbf{f}_e - \mathbf{D}_p \dot{\mathbf{p}}_{cd}(t) - \mathbf{K}_p \mathbf{p}_{cd}(t)), \\ \dot{\mathbf{p}}_{cd}(t + dt) &= \dot{\mathbf{p}}_{cd}(t) + dt \cdot \ddot{\mathbf{p}}_{cd}(t + dt), \\ \mathbf{p}_{cd}(t + dt) &= \mathbf{p}_{cd}(t) + dt \cdot \dot{\mathbf{p}}_{cd}(t + dt). \end{aligned} \quad (7)$$

Then, allowing to compute the translational compliance as $\mathbf{p}_c = \mathbf{p}_{cd} + \mathbf{p}_d$.

Computations relating to orientation are represented using a unit quaternion $q = (\eta, \boldsymbol{\varepsilon})$ defined in terms of a rotation angle ϑ about an axis in space described by a unit vector $\mathbf{r} \in \mathbb{R}^3$, as

$$\eta = \cos \frac{\vartheta}{2}, \quad \boldsymbol{\varepsilon} = \sin \frac{\vartheta}{2} \mathbf{r}, \quad (8)$$

in which the unit quaternion $q = (\eta, \boldsymbol{\varepsilon})$ is related to a rotation matrix \mathbf{R} as

$$\mathbf{R}(\eta, \boldsymbol{\varepsilon}) = (\eta^2 - \boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon}) \mathbf{I} + 2\boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon} - 2\eta \mathbf{S}(\boldsymbol{\varepsilon}), \quad (9)$$

where $\mathbf{S}(\cdot)$ is the skew-symmetric matrix operator [22].

The **resolved angular acceleration** \mathbf{a}_o is chosen as

$$\mathbf{a}_o = \dot{\boldsymbol{\omega}}_c + k_{V_o} \cdot \boldsymbol{\omega}_{ce} + k_{P_o} \cdot \boldsymbol{\varepsilon}_{ce}, \quad (10)$$

where k_{V_o} and k_{P_o} are the orientational damping and stiffness gains, respectively, and $\boldsymbol{\varepsilon}_{ce}$ denotes the vector part, given in base frame, of the unit quaternion representing the orientation error between the compliant frame \mathbf{R}_c and EE frame \mathbf{R}_e , as $\mathbf{R}_{ec} = \mathbf{R}_e^\top \cdot \mathbf{R}_c$, given by the relation

$$\mathbf{R}_2^1 = (\mathbf{R}_1)^\top \cdot \mathbf{R}_2, \quad \boldsymbol{\varepsilon}_{21}^1 = \eta_1 \boldsymbol{\varepsilon}_2 - \eta_2 \boldsymbol{\varepsilon}_1 - \mathbf{S}(\boldsymbol{\varepsilon}_1) \boldsymbol{\varepsilon}_2. \quad (11)$$

such that \mathbf{R}_2^1 is related to $q = (\eta_{21}, \boldsymbol{\varepsilon}_{21}^1)$ via (9). In practice, several efficient algorithms exist to convert unit quaternions from-and-to rotation matrices; we make use of the Eigen linear algebra library [23].

The mechanical impedance for the orientational part is given by

$$\mathbf{M}_o \dot{\boldsymbol{\omega}}_{cd}^d + \mathbf{D}_o \boldsymbol{\omega}_{cd}^d + \mathbf{K}_o' \boldsymbol{\varepsilon}_{cd}^d = \boldsymbol{\mu}_e^d, \quad (12)$$

where the equivalent rotational stiffness matrix is

$$\mathbf{K}_o' = 2 \cdot \mathbf{E}^\top(\eta_{cd}, \boldsymbol{\varepsilon}_{cd}^d) \cdot \mathbf{K}_o \quad \text{where} \quad \mathbf{E}(\eta, \boldsymbol{\varepsilon}) = \eta \mathbf{I} - \mathbf{S}(\boldsymbol{\varepsilon}), \quad (13)$$

in which $\mathbf{M}_o, \mathbf{D}_o, \mathbf{K}_o \in \mathbb{R}^{3 \times 3}$ are positive definite matrices specifying the orientational compliance. Here, the measured EE torque must be transformed in to the desired frame $\boldsymbol{\mu}_e^d$. Using the wrench transform relation [24]

$$\mathbf{h}_e^d = \mathbf{Ad}(\mathbf{T}_0^d) \cdot \mathbf{h}_e^0 = \begin{bmatrix} \mathbf{f}_e^d \\ \boldsymbol{\mu}_e^d \end{bmatrix}, \quad \mathbf{Ad}(\mathbf{T}_B^A) = \begin{bmatrix} \mathbf{R}_B^A & \mathbf{0} \\ \mathbf{S}(\mathbf{t}_B^A) \mathbf{R}_B^A & \mathbf{R}_B^A \end{bmatrix}, \quad (14)$$

where $\mathbf{Ad}(\mathbf{T}_B^A)$ is the adjoint matrix for a given transformation \mathbf{T}_B^A , the measured end-effector torque is given by

$$\boldsymbol{\mu}_e^d = \begin{bmatrix} \mathbf{S}(\mathbf{t}_B^A) \mathbf{R}_B^A & \mathbf{R}_B^A \end{bmatrix} \cdot \mathbf{h}_e^0. \quad (15)$$

Then, the compliant orientation can be computed using numerical integration, given a time step dt and time t , as

$$\begin{aligned} \dot{\boldsymbol{\omega}}_{cd}^d(t+dt) &= \mathbf{M}_o^{-1} \cdot \left(\boldsymbol{\mu}_e^d - \mathbf{D}_o \boldsymbol{\omega}_{cd}^d(t) - \mathbf{K}_o' \boldsymbol{\varepsilon}_{cd}^d(t) \right), \\ \boldsymbol{\omega}_{cd}^d(t+dt) &= \boldsymbol{\omega}_{cd}^d(t) + dt \cdot \dot{\boldsymbol{\omega}}_{cd}^d(t+dt). \end{aligned} \quad (16)$$

where $\boldsymbol{\varepsilon}_{cd}^d$, represented by the quaternion $\mathbf{q}_c^d = (\eta_{cd}, \boldsymbol{\varepsilon}_{cd}^d)$, is computed using quaternion integration, as

$$\mathbf{q}_c^d(t+dt) = \exp\left(\frac{dt}{2} \cdot \boldsymbol{\omega}_{cd}^d(t+dt)\right) \otimes \mathbf{q}_c^d(t) \quad \text{where} \quad \exp(\mathbf{r}) = (\eta, \boldsymbol{\varepsilon}) = \left(\cos(\|\mathbf{r}\|), \frac{\mathbf{r}}{\|\mathbf{r}\|} \sin(\|\mathbf{r}\|) \right), \quad (17)$$

in which \otimes refers to the quaternion product operator. Given the desired orientation as a quaternion \mathbf{q}_d , the compliant orientation \mathbf{q}_c is given (in the base frame) as

$$\mathbf{q}_c = \mathbf{q}_d \otimes \mathbf{q}_c^d, \quad (18)$$

which can be transformed to a rotation matrix \mathbf{R}_c , where quantities such as $\boldsymbol{\omega}_c$ are computed as

$$\boldsymbol{\omega}_c = \mathbf{R}_d \cdot \boldsymbol{\omega}_{cd}^d + \boldsymbol{\omega}_d. \quad (19)$$

The driving torques in (4) require the computation of the Jacobian inverse, which does not exist for a redundant manipulator such as the Franka. The inverse Jacobian is approximated with a dynamically consistent pseudoinverse of the Jacobian [25], given as

$$\mathbf{J}^{-1}(\mathbf{q}) \approx \mathbf{J}^\dagger(\mathbf{q}) = \mathbf{M}^{-1}(\mathbf{q})\mathbf{J}^\top(\mathbf{q}) \cdot \left(\mathbf{J}(\mathbf{q})\mathbf{M}^{-1}(\mathbf{q})\mathbf{J}^\top(\mathbf{q})\right)^{-1}. \quad (20)$$

Due to the redundancy of the manipulator, nullspace drift may occur, leading singular joint configurations with lessened manipulability. A method for stabilized nullspace velocities is proposed in [21], based on which, the driving torque is chosen as

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q}) \cdot \mathbf{J}^\dagger(\mathbf{q}) \cdot \left(\mathbf{a} - \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}}\right) + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \cdot \dot{\mathbf{q}} + \mathbf{J}^\top(\mathbf{q}) \cdot \mathbf{h}_e + \boldsymbol{\phi}_n, \quad (21)$$

where the nullspace joint torques $\boldsymbol{\phi}_n$ are chosen as

$$\boldsymbol{\phi}_n = \left(\mathbf{I} - \mathbf{J}^\top(\mathbf{q}) \cdot (\mathbf{J}^\top(\mathbf{q}))^\dagger\right) \cdot \left(k_n \cdot (\mathbf{q}_{N_d} - \mathbf{q}) - k_c \cdot \dot{\mathbf{q}}\right), \quad (22)$$

where \mathbf{q}_{N_d} is the desired nullspace configuration and k_n, k_c are nullspace stiffness and damping gains, respectively.

3.3 Gain selection

Given a second order system on the form

$$m \frac{d^2x}{dt^2} + b \frac{dx}{dt} + kx = 0, \quad (23)$$

the poles that characterize the system's behavior are parameterized in terms of the damping ratio ζ , and natural frequency ω_n , where

$$\zeta = \frac{b}{2\sqrt{km}}, \quad \omega_n = \sqrt{\frac{k}{m}}, \quad (24)$$

For a critically damped system, the poles must be on the real axis, such that

$$b^2 - 4mk = 0 \quad \therefore \quad \zeta = 1. \quad (25)$$

Tracking

The tracking gains $(k_{V_p}, k_{P_p}, k_{V_o}, k_{P_o})$ are tuned manually to guarantee tracking of a desired frame, with a damping ratio of $\zeta = 1.0$ (critically damped), such that $k_{V_p} = 2 \cdot \sqrt{k_{P_p}}$. Likewise, the nullspace stiffness is manually tuned to $k_n = 0.5$ with nullspace damping $k_c = 2 \cdot \sqrt{k_n}$.

Compliance

To select the impedance parameters, some choices must be made; for example, the stiffness k can be defined in terms of the maximum desired displacement \tilde{x}_d in steady state for a given force f , as

$$k = \frac{f}{\tilde{x}_d}. \quad (26)$$

Then, given some desired mass $m = m_d$ the damping b can be computed as

$$b = \sqrt{4m_d k}. \quad (27)$$

Dynamic reconfigure

Using the `dynamic_reconfigure` package [26], it is possible to alter the gains live. This allows to manually tune both the tracking and compliance gains while the manipulator is in operation, both on the real hardware or in simulation.

3.4 Gripper

The attached gripper, as shown in Fig. 8, is interfaced using the `franka_gripper` ROS interface, which offers a set of action servers, such as `franka_gripper::HomingAction` and `franka_gripper::MoveAction`, allowing to command a binary open/close state to the gripper, respectively.

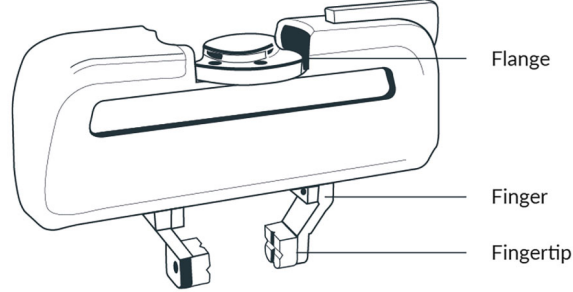


Fig. 8: The attached Franka hand gripper.

The homing message opens the gripper at a width of 2.2 cm away from the center point for each finger at a velocity of 0.1 m/s. The move action message closes the gripper to a centered width of 0.01 m at a velocity of 0.2 m/s.

3.5 Evaluation

The Cartesian Admittance Controller is evaluated by comparing the behavior of translational and orientational compliance while a wrench is applied at the EE of the manipulator for a given set $Z = \{\mathbf{M}_p, \mathbf{M}_o, \mathbf{D}_p, \mathbf{D}_o, \mathbf{K}_p, \mathbf{K}_o\}$ of impedance parameters. Specifically, given an initial joint configuration of

$$\mathbf{q} = \left[0 \quad -\frac{\pi}{4} \quad 0 \quad -\frac{3\pi}{4} \quad 0 \quad \frac{\pi}{2} \quad 0 \right]^T, \quad (28)$$

at $t \in [0, 5]$ a force of 5 N m is applied along the x -axis, and at $t \in [10, 15]$ a torque of 1 N m is applied around the x -axis, as shown in Fig. 9.

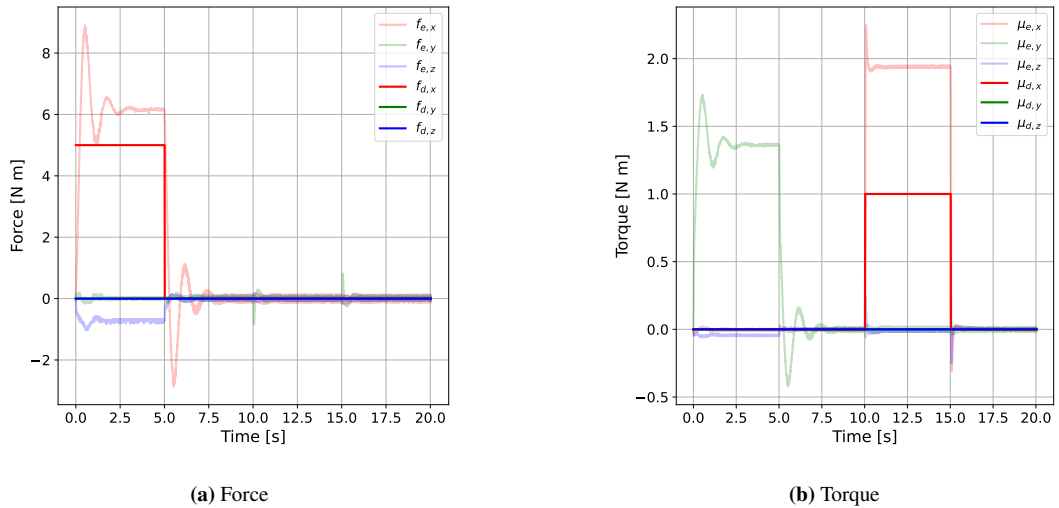


Fig. 9: Applied (desired) wrench \mathbf{h}_d and measured estimated wrench \mathbf{h}_e at the end-effector.

For brevity, the impedance parameters are represented by a set $\mathbf{z} = \{m_p, m_o, d_p, d_o, k_p, k_o\}$ of scalar values, where each element represents the diagonal values of the corresponding gain matrix, e.g., $\mathbf{M}_p = m_p \cdot \mathbf{I} \in \mathbb{R}^{3 \times 3}$.

Translational compliance

The translational compliance is evaluated with different impedance parameters z in order to demonstrate how these affect the compliant frame. The results are shown in Fig. 10.

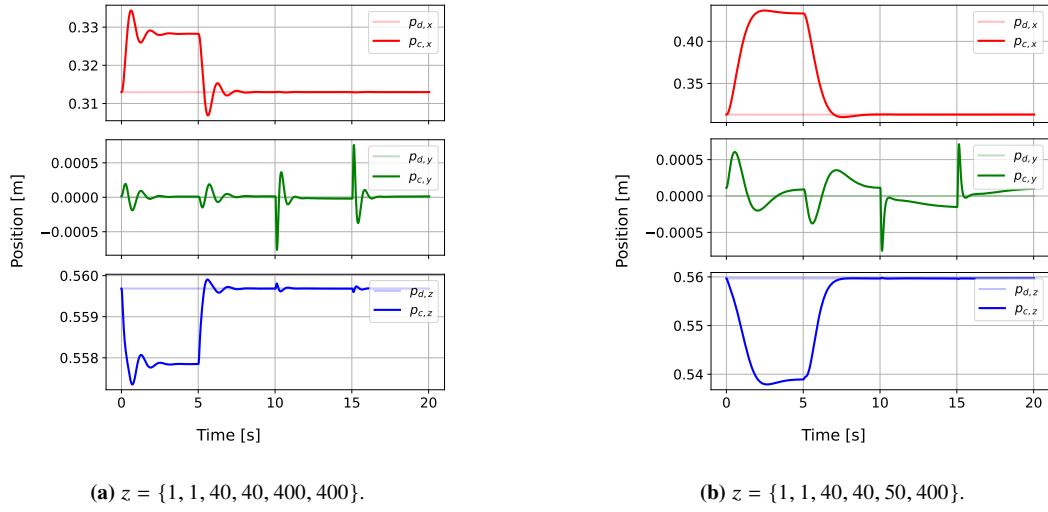


Fig. 10: Position vector \mathbf{p}_c of the compliant frame with different impedance parameters.

Orientalional compliance

The orinetational compliance is evaluated with different impedance parameters z in order to demonstrate how these affect the compliant frame. The results are shown in Fig. 11.

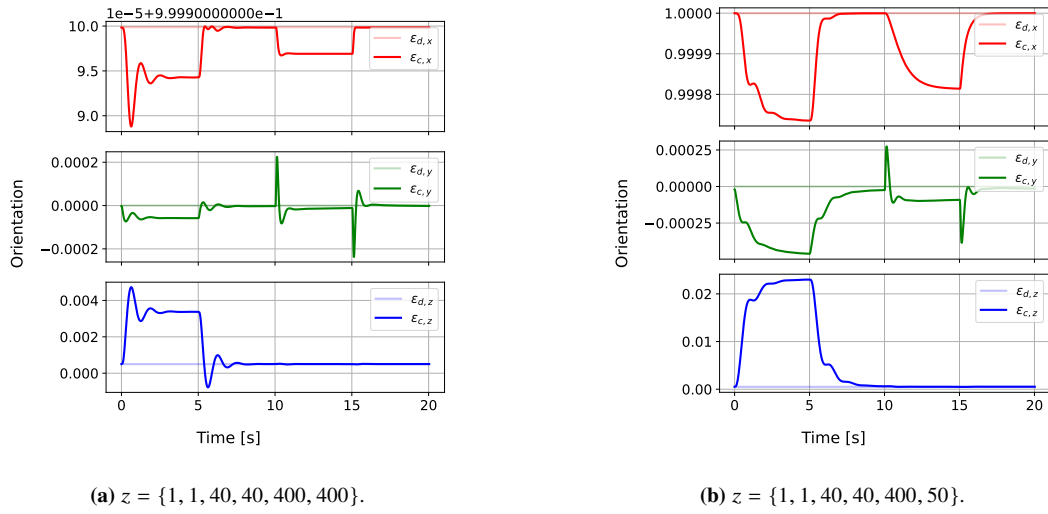


Fig. 11: Orientation vector \mathbf{e}_c of compliant frame with different stiffness parameters.

Real manipulator

A controlled evaluation of an applied wrench at the EE of the real robot was not possible. Instead, the manipulator was initialized in a similar joint configuration to (28), where a force was manually applied to the 6th link of the manipulator in the negative direction of the x -axis. The controller was evaluated with two different translational stiffness parameters, soft stiffness $z = \{2, 5, 20, 28.28, 200, 200\}$, and hard stiffness $z = \{2, 5, 40, 28.28, 1000, 200\}$.

For the two experiments, the measured force at the EE is shown in Fig. 12, and the resulting compliant frame is shown in Fig. 13. Given approximately similar force applications, the position of the compliant frame deviates more in case of the lower stiffness value.

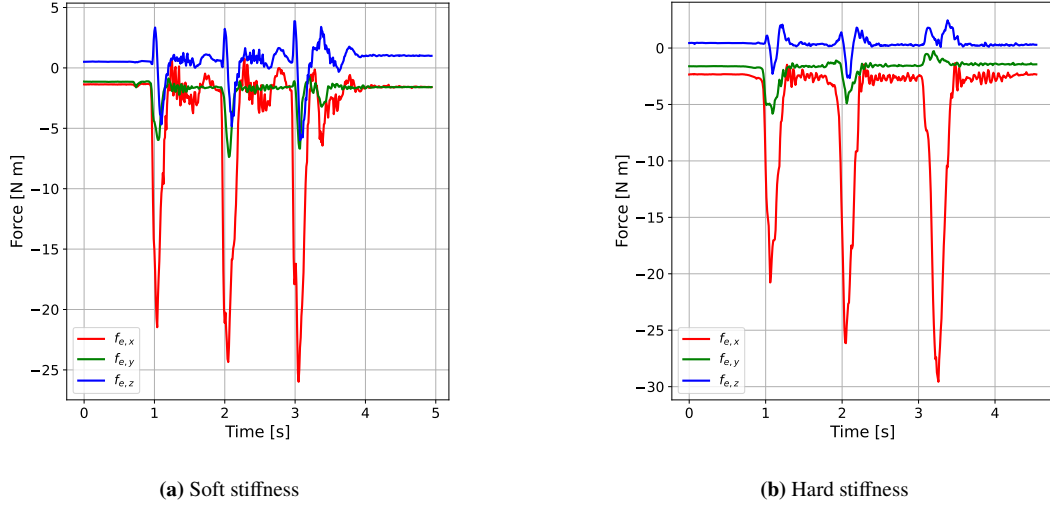


Fig. 12: Measured force at the end-effector for the two experiments.

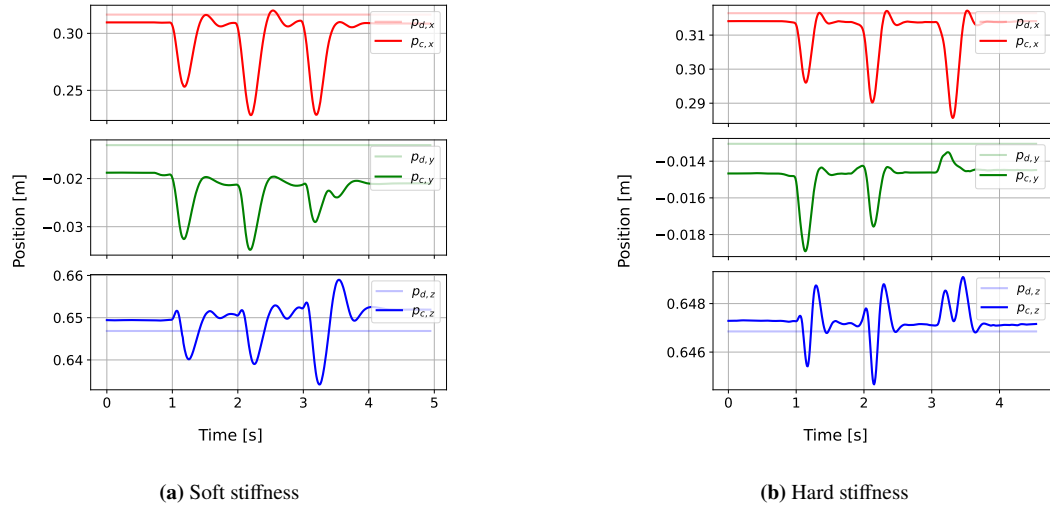


Fig. 13: Position vector \mathbf{p}_c of compliant frame for the two experiments.

The evaluation of the robot control on the real manipulator indicates that a Cartesian Admittance Controller is able to provide compliant behavior, for which the impedance of the EE can be set as desired. It also shows that the same controller, albeit with different gains, can run both in simulation as well as on the real manipulator.

4 Integration

The integration of the teleoperation system is achieved by mapping the estimated hand poses to desired EE poses. This process has several considerations: calibration, safety, mapping and motion prediction.

A calibration is required since the hand pose is relative to a different frame than the EE of the Franka manipulator. The calibration process links the pose of the hand to the initial pose of the manipulator, such that relative changes in the hand pose are mapped to a desired EE pose in order to achieve tracking.

4.1 Safety

Two safety constraints are considered: position and velocity. Position constraints are applied in the form of a bounding plane, as shown in Fig. 14, encapsulating a subspace of the manipulator's workspace, such that collisions with the environment and singularities, when fully extending multiple joints, are avoided. The height is bounded by $z \in [0.3, z_{\max}]$ with respect to the base frame, where z_{\max} is the maximum height reachable by the manipulator. The y -axis is constrained to ± 0.4 m with respect to the origin of the base frame.

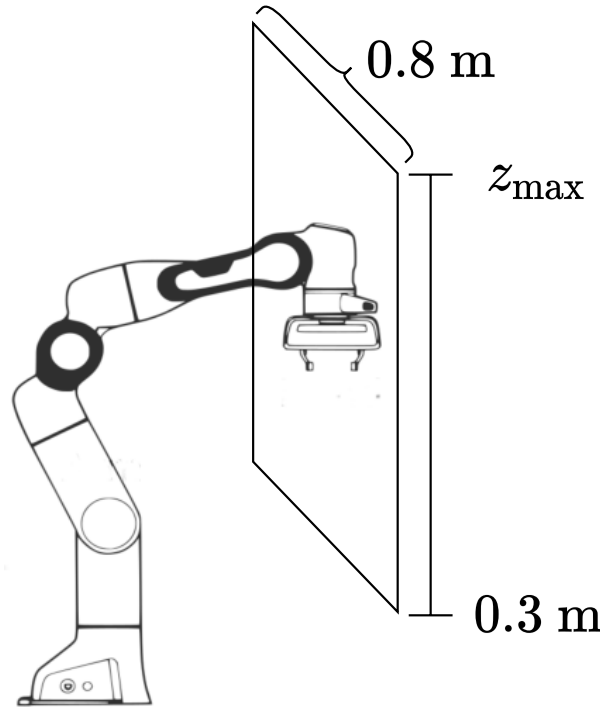


Fig. 14: A bounding plane constraining the workspace of the Franka manipulator.

When considering the velocity constraints, sudden changes in linear and angular velocities due to changes in the hand pose must be avoided. Two scenarios are potentially problematic: (1) when the user suddenly moves the hand outside the camera frame, and (2) when the hand plane is perpendicular to the image plane of the camera. Sudden changes in either linear or angular velocity are restricted by commanding the previous desired EE pose, if any the velocities are above a certain threshold.

4.2 Mapping

Mapping is performed by computing a desired end-effector pose $\mathbf{T}_{ee,d} = (\mathbf{p}_{ee,d}, \mathbf{R}_{ee,d})$ based on the difference from initial hand pose \mathbf{T}_{hi} to the current hand pose \mathbf{T}_{hc} . This difference is given by a difference in orientation between the initial and current hand orientation

$$\mathbf{R}_{hc}^{hi} = \mathbf{R}_{hi}^T \mathbf{R}_{hc}, \quad (29)$$

and a difference in position

$$\mathbf{p}_{hihc} = \mathbf{p}_{hc} - \mathbf{p}_{hi}. \quad (30)$$

This allows to define the desired orientation and position as

$$\mathbf{R}_{ee,d} = \mathbf{R}_{hc}^{hi} \mathbf{R}_{ee,i}, \quad \mathbf{p}_{ee,d} = \mathbf{p}_{ee,i} + \mathbf{p}_{hihc}, \quad (31)$$

allowing to construct the desired pose $\mathbf{T}_{ee,d} = (\mathbf{p}_{ee,d}, \mathbf{R}_{ee,d})$.

In order impose the safety constraints, the desired pose is limited as

$$\mathbf{T}_{ee,d,z} = \begin{cases} \mathbf{T}_{ee,d,z} & \text{if } \mathbf{T}_{ee,d,z} > 0.3 \\ 0.3 & \text{if } \mathbf{T}_{ee,d,z} < 0.3 \end{cases}, \quad \mathbf{T}_{ee,d,y} = \begin{cases} \mathbf{T}_{ee,d,y} & \text{if } 0.4 > \mathbf{T}_{ee,d,z} > -0.4 \\ -0.4 & \text{if } \mathbf{T}_{ee,d,z} < -0.4 \\ 0.4 & \text{if } \mathbf{T}_{ee,d,z} > 0.4 \end{cases} \quad (32)$$

4.3 Motion prediction

Due to expected delay through the pipeline, a linear motion prediction method is implemented assuming a linear path of motion, as shown in Fig. 15, where $\mathbf{p}_{h,t+1}$ is the real next position, $\hat{\mathbf{p}}_{h,t+1}$ is the estimated next position, $\mathbf{p}_{h,t}$ is the current position, and $\mathbf{p}_{h,t-1}$ is the previous.

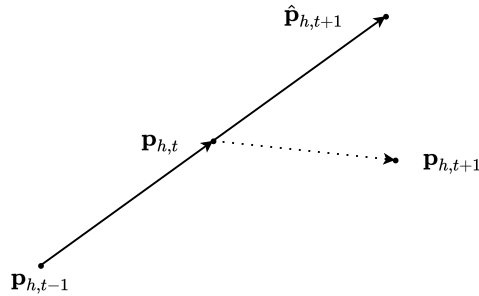


Fig. 15: Linear motion prediction method.

The difference in position between the time steps t and $t - 1$ is computed as

$$\mathbf{p}_{h,t,t-1} = \mathbf{p}_{h,t} - \mathbf{p}_{h,t-1} \quad (33)$$

A scaling of this difference is applied with $\alpha \in \mathbb{R}$, which indicates the expected covered distance between time steps. Where $\alpha = 1$ is the hand moving with constant velocity, as

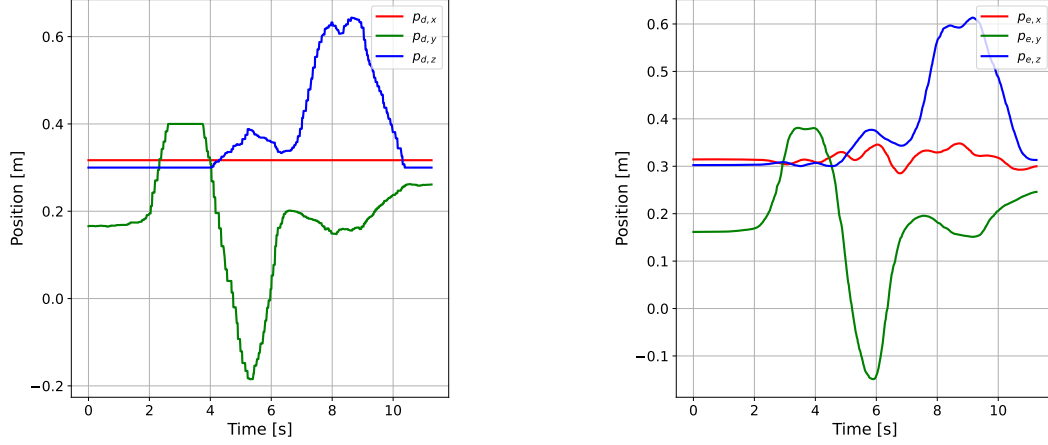
$$\hat{\mathbf{p}}_{h,t+1} = \alpha \cdot \mathbf{p}_{h,t,t-1} \quad (34)$$

$$\hat{\mathbf{T}}_{ee,t+1} = \mathbf{T}_{ee,t} \hat{\mathbf{T}}_{ee,t+1}, \quad \hat{\mathbf{T}}_{ee,t+1} = (\hat{\mathbf{p}}_{h,t+1}, \mathbf{I}) \quad (35)$$

The expected next pose can thus be found as shown in equation (35). As it here can be seen, this model does not consider orientation which could be added in the form of Slerp [10].

4.4 Evaluation

Evaluation of the integration is based on the evaluation of hand tracking in 2.4 Evaluation, in which the hand is moved along the y -axis followed by a rotation around the y -axis, as shown in Fig. 6. The integration module commands a desired EE pose, as shown in Fig. 16(a). The resulting EE pose is then shown in Fig. 16(b).



(a) The constrained desired positions send to the Franka from the integration module.

(b) The positions executed by the Franka after being received from the integration module.

Fig. 16: Positions produced by integration module and said positions executed by Franka.

When comparing to the estimated hand pose in Fig. 6(c), the mapping of the desired EE pose is evidenced by the y -axis being centered around 0, where a position constraint can be seen for the y -axis at $t \in [2, 4]$. Due to the pipeline having a satisfactory throughput it was decided not to use the presented linear motion prediction method.

The integration was tested on the real manipulator, in which an operator is able to leverage the system in order to grasp and move an object held by a participant, whilst the participant is able to resist. A demonstration of the teleoperation system can be seen at <https://youtube.com/shorts/IuGVKiLKSjc>.

4.5 Conclusion and Discussion

As evidenced by the live demonstration, the teleoperation system is able to use an RGB camera input to track the pose of a hand, which is then mapped to a desired EE pose, then tracked by a Cartesian Admittance controller. The demonstration also shows a participant resisting the motion of the manipulator, in which a compliant behavior is observed. However, there are improvements to be made.

A steady-state error in tracking was observed during evaluation, especially in orientation, likely due to errors in the dynamic model. A more optimal controller performance can be achieved using an automatic tuning algorithm. Moreover, an integrator term could be added to obtain zero steady-state error, resulting in better tracking of the compliant frame when the robot interacts with the environment.

A significant improvement of the hand tracking module could be achieved by getting accurate depth estimations, as this allows for teleoperation in 6 DOF. Accurate depth estimation using a single RGB image could be realized by using designated pre-trained DNN models.

Since the hand estimation provides detailed information of the hand state, a proportional grasp state could be implemented in order to provide a more dexterous grasping, instead of the binary grasp. Furthermore, a more dexterous teleoperation could be realized by implementing a sophisticated motion prediction, as this would imply a more responsive feedback for the operator of the system.

References

- [1] Qian, Wei et al. “Manipulation Task Simulation using ROS and Gazebo”. In: *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBOT 2014* (Dec. 2014). DOI: 10.1109/ROBOT.2014.7090732.
- [2] Franka Emika GmbH. *Franka ROS Documentation*. URL: https://frankaemika.github.io/docs/franka_ros.
- [3] Quigley, Morgan et al. “ROS: an open-source Robot Operating System”. In: *ICRA Workshop on Open Source Software 3* (Jan. 2009).
- [4] Ivaldi, Serena et al. “Tools for simulating humanoid robot dynamics: A survey based on user feedback”. In: 2015 (Nov. 2014). DOI: 10.1109/HUMANOIDS.2014.7041462.
- [5] Koenig, N. and Howard, A. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: 3 (2004), 2149–2154 vol.3. DOI: 10.1109/IRROS.2004.1389727.
- [6] Strasheim, Troy et al. *Conceptual overview of ROS catkin*. URL: http://wiki.ros.org/catkin/conceptual_overview.
- [7] LLC, GOOGLE. *MediaPipe*. URL: <https://google.github.io/mediapipe/>.
- [8] GOOGLE LLC. *MediaPipeHands*. URL: <https://google.github.io/mediapipe/solutions/hands>.
- [9] GOOGLE LLC. *Hand landmarks*. URL: https://google.github.io/mediapipe/images/mobile/hand_landmarks.png.
- [10] Erik B. Dam Martin Koch, Martin Lillholm. “Quaternions, Interpolation and Animation”. In: (1998), pp. 42–48.
- [11] Spong, M.W., Hutchinson, S., and Vidyasagar, M. *Robot Modeling and Control*. Wiley, 2005. ISBN: 9780471649908. URL: <https://books.google.dk/books?id%20=%20jyD3xQEACAAJ>.
- [12] Chitta, Sachin et al. “ros_control: A generic and simple control framework for ROS”. In: *The Journal of Open Source Software* (2017). DOI: 10.21105/joss.00456. URL: <http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf>.
- [13] *ros_control wiki on GitHub*. URL: https://github.com/ros-controls/ros_control/wiki.
- [14] Slate Robotics. *How to implement ros_control on a custom robot*. URL: <https://medium.com/@slaterobotics/how-to-implement-ros-control-on-a-custom-robot-748b52751f2e>.
- [15] Open Source Robotics Foundation. *ros_control in Gazebo*. URL: http://gazebo.org/tutorials/?tut=ros_control.
- [16] Franka Emika GmbH. *Franka simulated hardware interface*. URL: https://frankaemika.github.io/docs/franka_ros.html#frankahwsim.
- [17] Gaz, Claudio et al. “Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization”. In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 4147–4154. DOI: 10.1109/LRA.2019.2931248. URL: <https://hal.inria.fr/hal-02265293>.
- [18] Smits, R. *KDL: Kinematics and Dynamics Library*. URL: <http://www.orocos.org/kdl>.
- [19] Franka Emika GmbH. *Franka Panda Joint Limits (Necessary Conditions)*. URL: https://frankaemika.github.io/docs/control_parameters.html.
- [20] Caccavale, Fabrizio, Siciliano, Bruno, and Villani, Luigi. “The role of Euler parameters in robot control”. In: *Asian journal of control* 1.1 (1999), pp. 25–34.
- [21] Natale, Ciro, Siciliano, Bruno, and Villani, Luigi. “Spatial impedance control of redundant manipulators”. In: 3 (1999), pp. 1788–1793.

- [22] Rowland, Todd and Weisstein, Eric W. *Antisymmetric Matrix (Skew-symmetric)*. URL: <https://mathworld.wolfram.com/AntisymmetricMatrix.html>.
- [23] Guennebaud, Gaël, Jacob, Benoît, et al. *Eigen v3*. 2010.
- [24] Lynch, Kevin M. and Park, Frank C. *Modern Robotics: Mechanics, Planning, and Control*. 1st. Cambridge University Press, 2017. ISBN: 1107156300.
- [25] Khatib, Oussama. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1 (1987), pp. 43–53.
- [26] Gassend, Blaise. *dynamic_reconfigure*. URL: http://wiki.ros.org/dynamic_reconfigure.