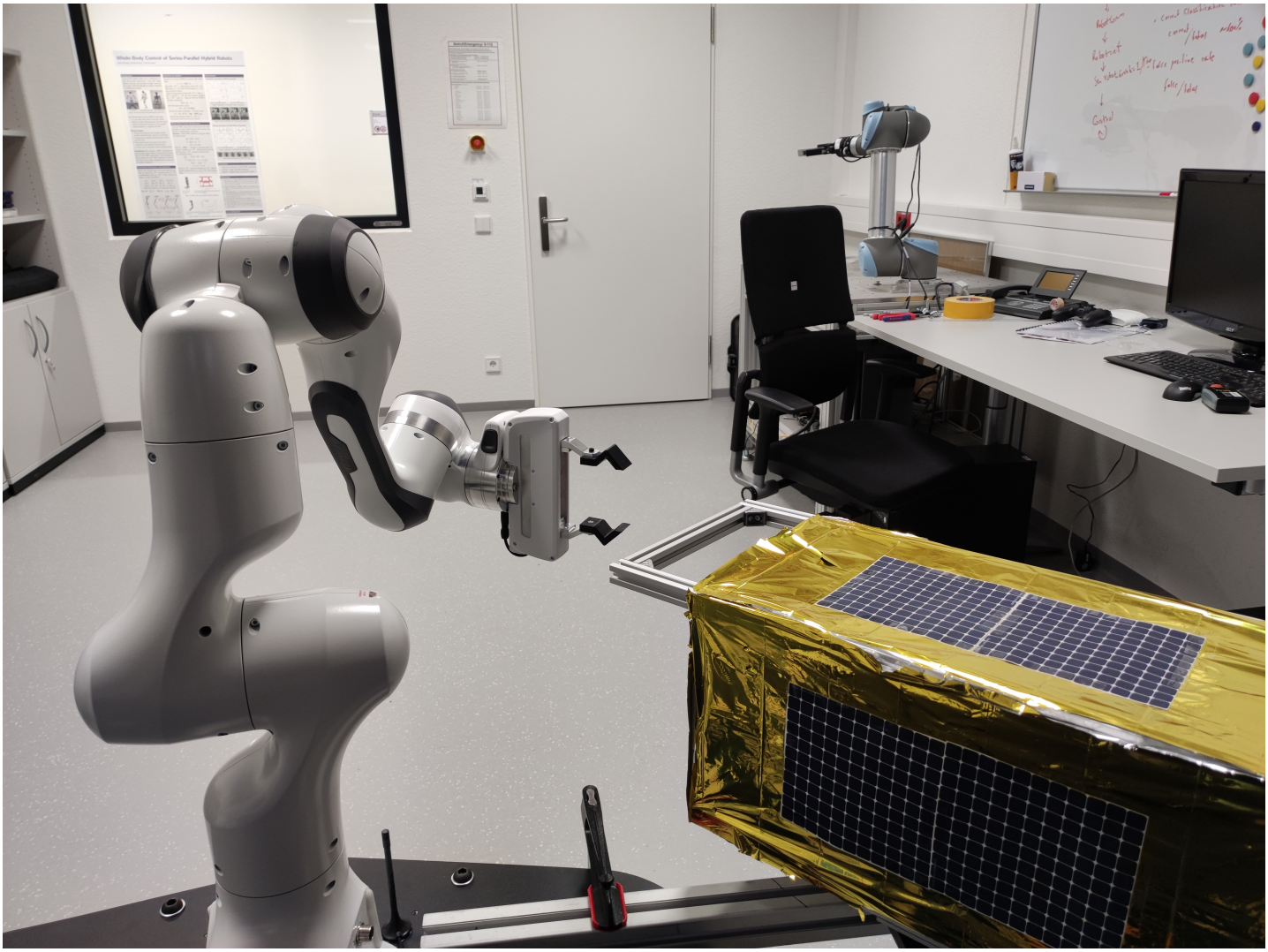
Floating Contact Dynamics Testbed

Robotics Reasearch for Active Debris Removal

AE5050: Internship Report

Francesco Branca



TU Delft

Floating Contact Dynamics Testbed

Robotics Reasearch for Active Debris Removal

by

Francesco Branca

Student Number: 4884981

Project Duration: September, 2022 - February, 2023

Preface

It is my pleasure to present this internship report which documents my experience at DFKI from September 2022 to February 2023. The internship was an invaluable opportunity to gain practical experience in the field of robotics and to work with a team of experts in the field.

During my time at DFKI, I had the privilege of working on a challenging project involving the development of a contact dynamics testbed for active debris removal research. This involved developing and implementing software algorithms, testing and troubleshooting the system, and providing solutions to overcome any challenges that arose during the project.

I was fortunate to be mentored by highly experienced professionals who provided me with the guidance, knowledge, and support. Their mentorship not only helped me develop my technical skills, but also provided me with valuable insight into the field of robotics.

I would like to express my sincere appreciation and gratitude to my supervisors Shubham Vyas and Shivesh Kumar for providing me with this opportunity. Their support and encouragement throughout the project have allowed me to grow both personally and professionally. In conclusion, the internship has been an enriching experience, and I am grateful for the opportunity to have worked at DFKI.

Francesco Branca

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

Abbreviation	Definition
ADR	Active Debris Removal
FCI	Franka Control Interface
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
LCM	Lightweight Communications and Marshalling

Symbols

Symbol	Definition	Unit
q	joint angle	[rad]
$\dot{q}$ joint angular rate	[rad/s]	
τ	joint torque	[Nm]
t	time [s]	

Abstract

This report documents a 6-month internship at DFKI where the goal was to create a testbed that simulates the interaction between a robot arm and space debris for ADR research. A literature study was conducted to gather knowledge on ADR methods and testbeds, and a preliminary design of the testbed was developed. The testbed used a Franka Emika robot arm controlled by a PC, a motor to spin a satellite mock-up, and a 3D printed extension attached to the gripper. The experiments provided insights on contact dynamics characteristics and on how to improve the testbed in the future, such as improving the free-floating behavior of the robot.

1

Introduction

The following internship report details the work undertaken during a six-month internship period on the development of a contact dynamics testbed for Active Debris Removal (ADR) research. The internship took place at DFKI, a research center for artificial intelligence located in Germany, from September 2022 to February 2023.

Space debris is an increasing concern for the international community, with over 34,000 pieces of debris currently in orbit, ranging from tiny fragments to large, defunct satellites. This debris poses a significant threat to operational spacecraft and satellites, as even small pieces of debris can cause serious damage upon impact due to their high velocities.

ADR is a proposed solution to this problem, involving the use of robotic systems to capture and remove debris from orbit. To ensure the success of ADR missions, it is necessary to test and validate the contact dynamics between the debris and the robotic system.

The goal of this internship was to develop a system to test different control algorithms for investigating on contact dynamics in space. The testbed was designed to simulate the interaction between a debris moving in 1 degree of freedom and a robot arm. This was meant to provide a better understanding of the forces and torques that are present during this interaction. The testbed was designed with a multidisciplinary approach and it includes elements of robotics software as well as structural components.

The following report outlines the processes and methods used to develop the testbed, as well as the results obtained during testing. The report also discusses the potential applications of the testbed in the field of ADR research and the future work that could be done to improve its performance.

In Chapter 2, some background information on DFKI are given. Chapter 3 presents a literature study on ADR. Chapter 4 gives a short overview of all the testbed components. Chapter 5 and 6 explain the driver structure of the robot arm and the motor respectively.

2

DFKI Background Information

The German Research Institute of Artificial Intelligence (DFKI) was founded in 1988 and is one of the world's largest non-profit research centers dedicated to artificial intelligence (AI). The institute is headquartered in Kaiserslautern, Germany, and has additional locations in Saarbrücken, Bremen, and Berlin.

DFKI has established itself as a leading research institution in the field of AI, with a focus on the development of intelligent software systems and robotics. The institute's research activities are conducted in collaboration with industry partners, government agencies, and other academic institutions.

DFKI's research areas cover a wide range of topics in AI, including machine learning, natural language processing, computer vision, robotics, and knowledge management. The institute's research is driven by a desire to address some of the world's most pressing challenges, such as sustainability, mobility, and healthcare.

One of the primary research areas at DFKI is intelligent software systems. Researchers at the institute are working to develop software systems that can interact with users and the environment in a more intelligent and intuitive way. This includes the development of natural language processing algorithms, machine learning algorithms, and cognitive computing techniques.

DFKI is also heavily involved in the development of robotics technologies. Researchers at the institute are working to develop robots that are capable of performing complex tasks in a variety of environments, such as space exploration, healthcare, and manufacturing. This includes the development of advanced control algorithms, sensing and perception technologies, and intelligent human-robot interaction.

Other research areas at DFKI include cyber-physical systems, data science and big data analytics, and human-centered computing. Through its innovative research activities, DFKI has established itself as a leading research institution in the field of AI, with a strong reputation for excellence and innovation.

3

Active Debris Removal Literature Study

This chapter provides a literature study on ADR methods, as well as examples of other contact dynamics models and testbeds.

3.1. Active Debris Removal Phases

When considering the case of an active manipulator having to capture and detumble an object in space, the operation can be divided into 4 main phases [6]:

- **Observing and planning phase**
  During this phase the information about the target satellite's motion, especially rotational, are obtained. This is meant to identify the appropriate time and location for capturing. Research on observation of the target satellite's motion has been done in the field of computer vision and sensing technologies.
- **Approaching phase**
  This is the phase when the robot moves its end-effector to the determined grasping point. The challenging aspect is having the manipulator's trajectory intersecting the moving target object.
- **Capturing phase**
  This is when the interception occurs and it is the most hazardous phase. If the contact is almost instantaneous, the robot and the satellite become one underactuated body, with the same motion characteristics.
- **Post-capture stabilization phase**
  The final phase is when the manipulator stabilizes and detumbles the target satellite.

Knowing all the ADR phases is crucial to understand the challenges of this problem and develop a testbed that resembles a real life scenario.

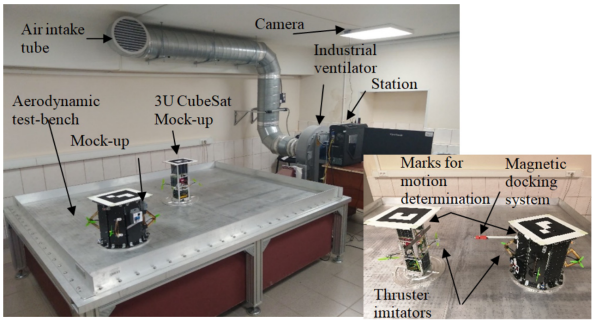
3.2. Air-Bearing Systems

To simulate space frictionless conditions and reproduce a free-floating motion of the objects, air-bearing systems are often used. In the setups built by James et al., 2015 [4] and Kozin et al., 2022 [5] the systems consist in a table surface with holes, through which air flows. This results in a air cushion below the mock-ups base, which reduces the friction between the surfaces. In this way, the motion of the object can be observed in 3 degrees-of-freedom (2D translation and 1D rotation).

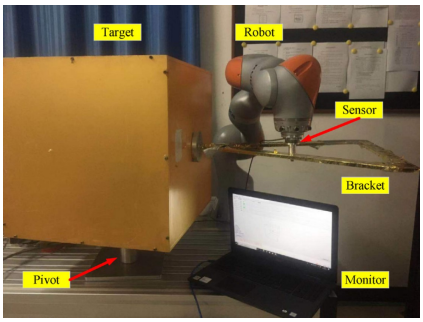
Figure 3.1a shows another example of a similar setup [3], where a target rotating in 1 degree-of-freedom is captured and detumbled using a 7 degree-of-freedom robotic arm LBR iiwa made by KUKA. To stabilize the system after contact, the reaction wheels of the robot are controlled to absorb the impact.

3.3. Debris Grasping Methods

Different techniques were developed and tested to capture a target debris. Shan et al., 2016 [9] provides an overview and comparison of the capturing methods (stiff connection capturing and flexible

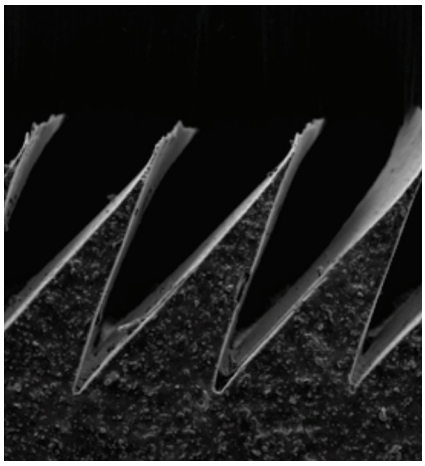


(a) Experimental setup by Kozin et al., 2022 [5]

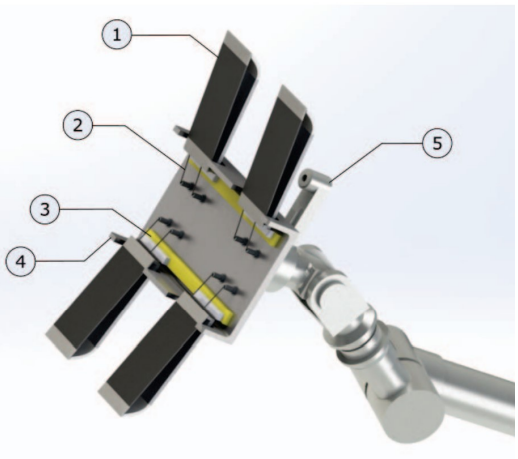


(b) Experimental setup by Dong Han et al. 2020 [3]

connection capturing). Regarding the detumbling phase, Matthew A. Estrada et al. [2] used a gecko-inspired adhesive material attached to a gripper. This type of dry adhesion can be very helpful, because it is activated only when forces are applied in a specific preferred direction between the gripper and the surface. This means that the adhesive can be easily controlled because it ”activates” and ”deactivates” quickly. The microscopic pattern of the adhesive can be visualized in Figure 3.2a. Another example of this technology applied to debris removal is given by Andrew Bylard et al. [1]. The gripper design that was developped is depicted in Figure 3.2b.



(a) Dry adhesive microscopic look.



(b) Gripper with dry adhesive.

Another capturing method is magnetic docking, which means using a magnetic gripper to detumble the debris. In Kozin et al. [7], the experimental setup with the aerodynamcis bed built in the previous paper is used to test the contact between a magnetic gripper and a satellite mock-up. Laser range finder measurements are used to estimate the motion of the debris and a collision avoidance system is implemented during the observation phase. The magnetic docking approach can be a possible choice, however it introduces additional loads in the experiment, which will have to be taken into account when developing a controller for the manipulator.

3.4. Problem Definition

With the knowledge gained in section 3.1, 3.2 and 3.3, the outlines of the testbed design can be better defined. The goal of the experimental setup is to test the interaction between a satellite mock-up and a robot arm and the resulting characteristics of the motion and loads.

The contact should be close to instantaneous and the connection should be stiff, as if after the collision the two bodies combined into one. A new gripper will be designed and manufactured and different rubber materials will be tested to increase the friction at the contact point. No special material or magnetic extension will be installed on the gripper for the moment.

To reduce the complexity of the problem, the target object will be moving only in one degree of freedom (rotating on its own axis), thus no air bearing bed will be used, also for lack of resources. Regarding the rotating speed of the mock-up, according to Nishida et al. [8], objects tumbling at 3 deg/s can be captured easily. Objects rotating at rates above 30 deg/s are not considered as capturable targets. For these specific cases different methods are being researched.

Moreover, the design of the setup will be focused on the approaching phase, capturing phase and post-capture stabilization phase. The observing and planning phase will be simplified, because the whole testbed will be considered as one system only, which means that a central controller will communicate with both the target object and the robot arm. In this way, the central controller will know the state of both the systems at all times and coordinate the actions to make the contact occur.

4

Testbed Design Overview

In this chapter, an overview of the testbed design is given. The testbed components are divided in active (such as the robot arm and the motor) or inactive (such as the 3D printed gripper and the satellite). This overview includes a description of the methods to control the active components form the workstation PC and the manufacturing details of the inactive components.

4.1. Robot Arm

The robot arm model that was chosen is the Franka Emika Panda, a 7 degrees of freedom manipulator with a linearly closing gripper. The arm is connected to the Franka Control Interface (FCI), which converts the executed C++ files to commands for the robot.

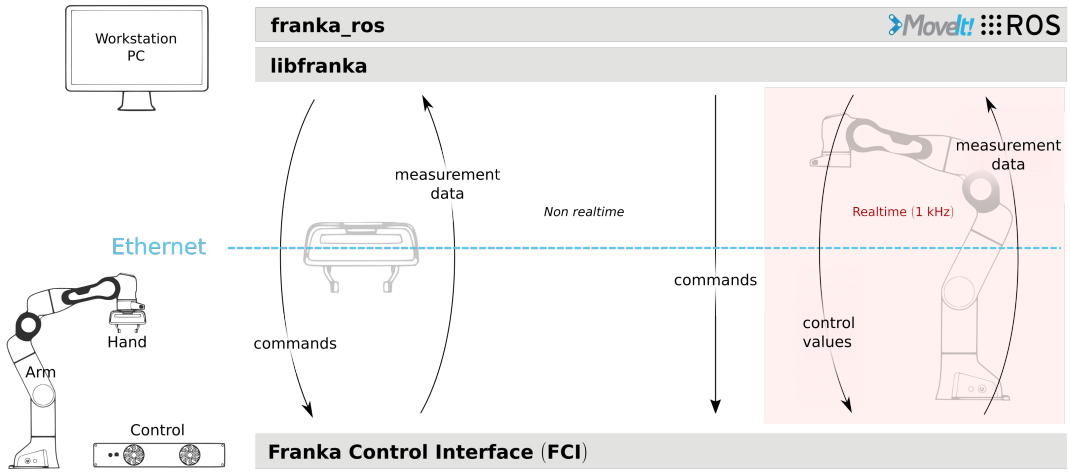


Figure 4.1: FCI architecture

Figure 4.1 [1] shows how the FCI is connected to the robot and the PC and how both real-time and non-real-time commands are exchanged. Note that the ROS extension `franka_ros` was not used in the driver, instead the workstation PC was directly connected to the FCI and the libfranka C++ library was used. The PC used was rebooted to install Ubuntu 20, which is compatible with the libfranka. For controlling the robot it was required to have a PC with high computational power to allow real-time communication at 1 kHz with the robot.

[1] `https://frankaemika.github.io/docs/overview.html`

4.1.1. Franka Emika Desk

The Franka Emika robot arm can also be controlled using the desk [2] (Figure 4.2), which is a user interface tool. The desk can be accessed by browsing the address link (192.168.131.40). From here, the joints have to be unlocked before executing any programs, by clicking on the lock button. The robot arm has a LED light which changes color depending on different conditions.

- blue: the robot is ready and programs can be executed.
- yellow: the joints are locked and/or the kill switch is pressed.
- red: when one of the joints exceeds the allowed limit.
- white: there is a button placed near the control box. When it is turned on and the user keeps the two buttons on the gripper pressed, the robot can be moved by hand.
- pink: when the robot receives conflicting information. For instance, the control box is waiting for a program to be executed, but the user tries to move the robot by pressing the buttons on top of the gripper.

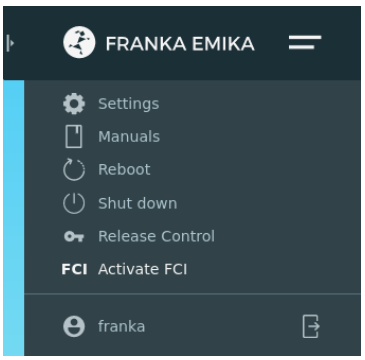


Figure 4.2: Franka Emika desk.

In the desk, some simple actions can be defined to move the robot from a certain waypoint to another, as shown in Figure 4.3 [3]. This is helpful to get familiar with how the robot works. However, in order to make the robot do more complicated actions, the libfranka library has to be used.

Figure 4.3: Franka Emika desk actions.

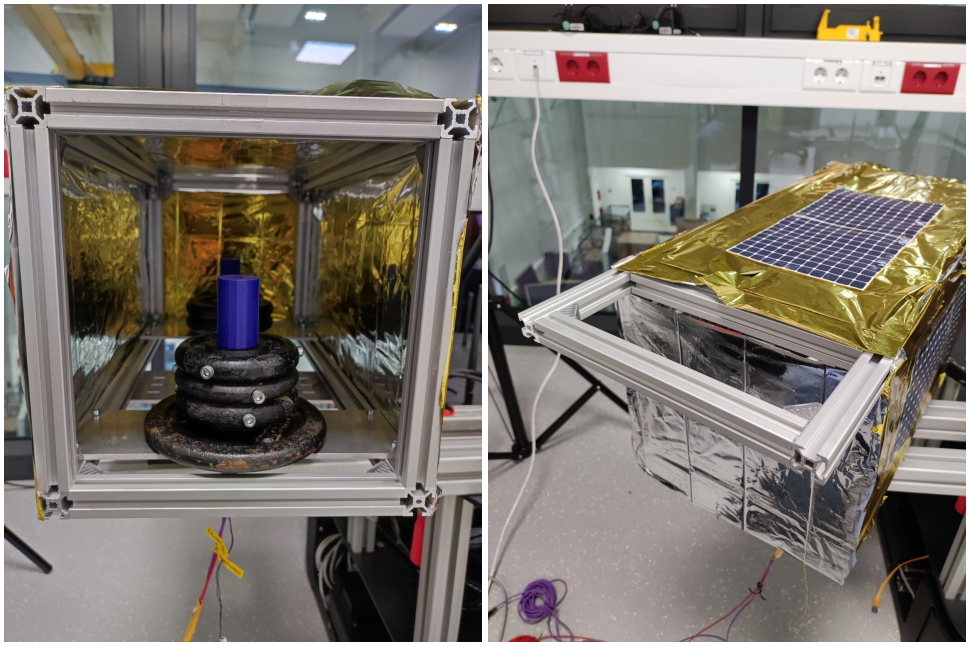
[2] https://frankaemika.github.io/docs/getting_started.html
[3] http://donar.messe.de/exhibitor/hannovermesse/2017/X376456/product-brochure-eng-481544.pdf

4.2. Motor

The motor that was used is a T-Motor Cubemars AK80-6 [4], which is connected to a power supply and to a CAN-to-USB cable. The position, velocity, torque and PD gains can be controlled through a python driver [5] developed at the Underactuated Lab in Robotics Innovation Center at DFKI GmbH, Bremen. To simulate zero gravity conditions, a friction compensation algorithm will be implemented on the motor. Furthermore a mechanical interface was manufactured to attach the satellite mock-up to the motor. The interface consisted in two parts, one extension fastened on the motor and one aluminum sheet fixed at the bottom of the mock-up.

4.3. Space Debris Mock-up

The target satellite to be mounted on the motor will be a 50x25x25 mm block, made with Bosch profiles with 20x20 mm cross section. Moreover, one of the beams will be kept longer to serve as a grasping point as shown in Figure 4.4b. A 3D printed column is mounted on the inside of the satellite and it is used to stack weights and increase or decrease the moment of inertia of the object. A total of 2.5 kg was put on each of the 2 columns. The motor is mounted on two 40x40 Bosch profiles that are clamped on the base of the Franka robot arm.



(a) Mock-up weights **(b)** Mock-up grasping point

4.4. 3D printed fingertip

A new fingertip was designed to be able to grasp the satellite better, since a simple linearly closing gripper could easily let the satellite slip away. The new fingertip was sketched on SolidWorks and manufactured using a 3D printed.

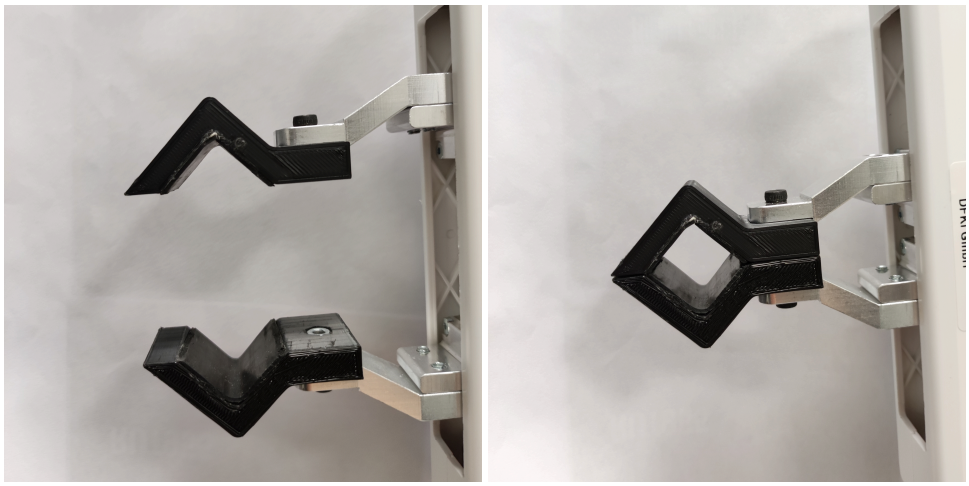
The first iteration of the design is a 3D printed fingertip with a 20x20 mm square enclosing shape. However, the gripper is not guaranteed to perfectly enclose around the Bosch profile, which is why a rubber material was glued on the fingertips to increase friction and avoid the satellite to slip away. In this iteration the 3D printed piece was not threaded, thus the screw attaching it to the robot was cutting through it. This was not optimal and when forces where applied repeatedly, the plastic was slowly consuming and eventually broke.

For this reason, in the second iteration, a metal nut was embedded in the 3D printed piece, so that the point of contact for the thread was metal and not plastic. This provided better stiffness and a more solid connection to the gripper. Regarding the rubber material, other options were also considered, such as

[4] https://store.tmotor.com/goods.php?id=981

[5] https://pypi.org/project/mini-cheetah-motor-driver-socketcan/

sponge-like materials that could adapt to the shape of the grasping point, however this could result in a non completely stiff contact, which is not desired for the experiment. The final gripper designed is showed in Figure 4.5a and 4.5b.



(a) Gripper in open position. **(b)** Gripper in closed position.

4.5. Final Setup

A preliminary schematic of the experimental setup is depicted in Figure 5.1. The robot arm is connected to the FCI, which works as an interface between the robot and the control PC. The PC is connected via Ethernet cable to the control box and it runs different files in C++ and Python. On the other side, the motor is connected to the PC via CAN port and it is connected to a generator and a kill switch.

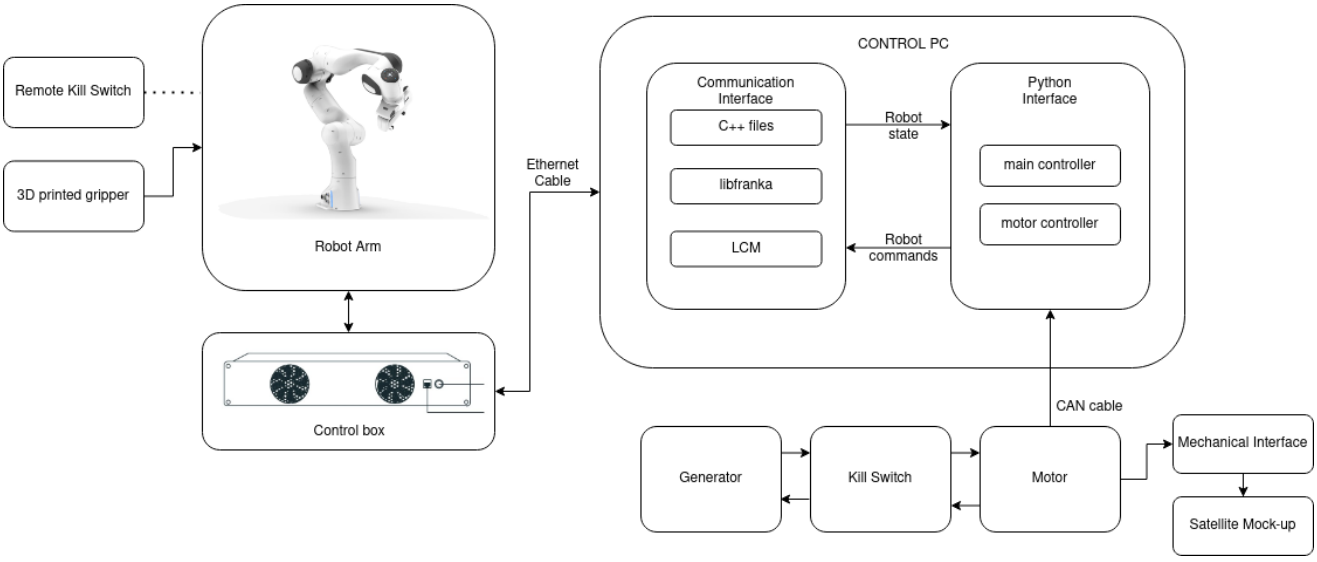


Figure 4.6: Preliminary diagram of the testbed

Figure 4.7 shows how the final setup looks like. In the following chapters, further details will be given on how the contact occurs and how the active components are controlled.

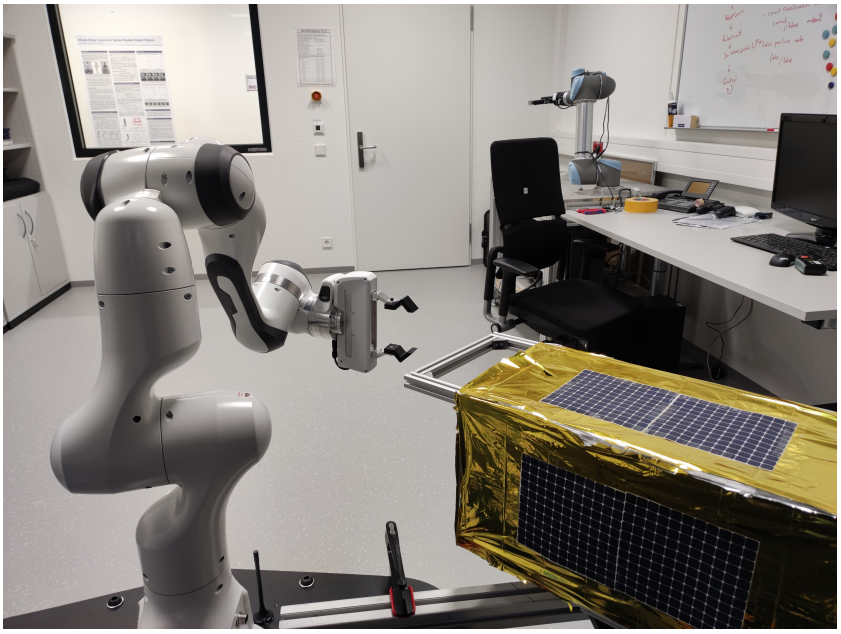


Figure 4.7: Experimental Setup

5

Franka Robot Arm Driver

In this chapter, the installation process to control the Franka robot arm is explained. Following, an overview of the libfranka library and LCM is given. Finally, the driver architecture is explained.

5.1. Real-Time Kernel

The Franka robot arm requires communication at 1 kHz. For the workstation PC to communicate with the robot at such frequency, it must run with real-time priority under a `PREEMPT_RT` kernel. After installing and compiling, the kernel can be found in the advanced options for Ubuntu. Further details for the installation can be found on the Franka Emika website [1].

5.2. Franka Library

To build the franka robot driver, the C++ library [2]. The libfranka documentation provides common examples for usage of the robot, including joint torque and gripper control. To develop the driver, the common functions given in the codes `examples_common.h` and `examples_common.cpp` were used, thus these two codes are included in `utils` folder. Gravity compensation is already implemented in libfranka, which means that when zero torque commands are sent the robot behaves as if it was free-floating.

5.2.1. Real-Time Commands

Real-time commands require a 1 kHz connection to control. There are two types of real-time interfaces:

- *Motion generators*, which define a robot motion by sending joint or Cartesian poses.
- *Controllers*, which define the torques to be sent to the robot joints.

To control the robot there are 3 different methods:

- defining only a motion generator and using the internal controllers to follow the commanded motion.
- defining only an external controller to send torque commands uniquely.
- using both the motion generator and the external controller.

5.2.2. Non Real-Time Commands

Non-realtime commands are always executed outside of any real-time control loop and include the hand commands and some configuration-related commands for the arm. To communicate with the gripper, a different C++ script was made.

[1] `https://frankaemika.github.io/docs/installation_linux.html`
[2] `https://frankaemika.github.io/libfranka/`

5.2.3. Franka Main Library Functions

The most important functions used in the driver are:

- *robot.control:* works like a real-time loop. Inside this function commands can be sent at every time step.
- *setCollisionBehavior:* it is used to set the contact and collision detection thresholds.
- *MotionGenerator:* real-time data cannot be accessed. The joint angles or Cartesian pose are given as input and the robot follows the most optimal route to reach the specified point in space.
- *gripper.grasp:* this function takes as input the closing width, speed and the force of the gripper.
- *gripper.homing:* this function is used to open the gripper.

For the development of this driver, only torque commands are sent to the robot.

5.3. LCM

LCM [3](Lightweight Communications and Marshalling) is a set of libraries and tools for communication between different programming languages. To exchange information the data has to be translated with the help of automatically generated files, which encode and decode from one language to the other. A simple LCM example involves a messenger and a listener file. The first one publishes the data on a channel with `lcm.publish`.The latter one subscribes to the same channel using the `lcm.subscribe` command and uses a message handling function to print out or store the data. The function `lcm.handle` is used to wait for the messages to be published.

An example of LCM between 2 python files is shown below. For the messenger file:

```
import lcm
from exlcm import example_t
msg = example_t()
msg.position = (1, 2, 3)
lc = lcm.LCM()
lc.publish("EXAMPLE", msg.encode())
```

While for the listener file:

```
import lcm
from exlcm import example_t
def my_handler(channel, data):
    msg = example_t.decode(data)
    print("Received message on channel \"%s\"" % channel)
    print("   position    = %s" % str(msg.position))
lc = lcm.LCM()
subscription = lc.subscribe("EXAMPLE", my_handler)
try:
    while True:
        lc.handle()
except KeyboardInterrupt:
    pass
```

For the robot driver, a two-way communication between the python controller and the C++ codes is established, meaning that both the codes are listener and messenger. This is meant to have a closed loop controller, where the python file sends commands to the C++ file and receives the feedback.

5.4. Driver Architecture

The driver structure is depicted in Figure 5.1. The `controller.py` file is the centre of the driver, where all the data flows. The `torque_control.cpp` executable communicates in a closed loop with the main file. The other two files (`gripper_control.cpp` and `motor_controller.py`) only have open loop control, which means that once the enabling command is sent, there is no data being fed back to the main controller.

[3] `https://lcm-proj.github.io/index.html`

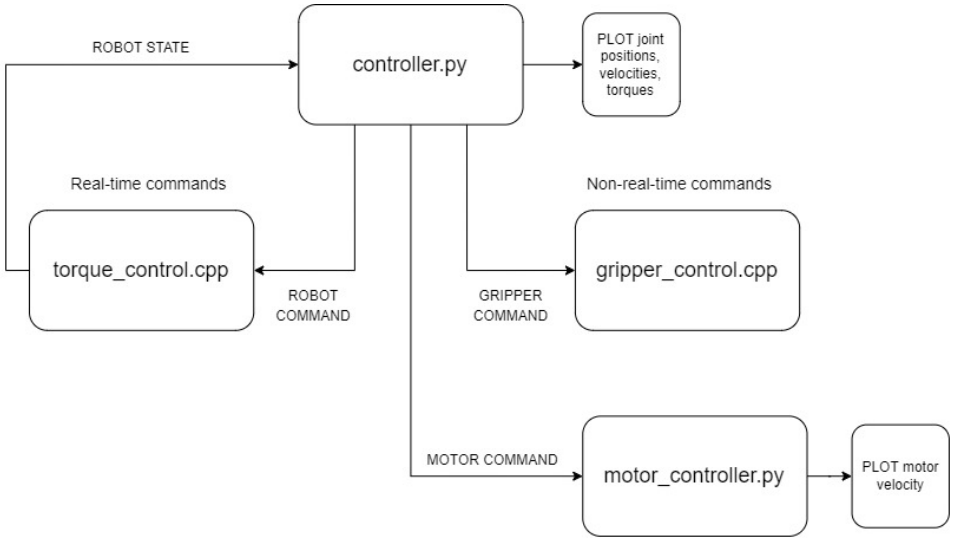


Figure 5.1: Driver Diagram

5.4.1. LCM Channels

Four main channels are defined. Two of them exchange real-time information to send torque commands to the robot joints, while the other two are used to send non real-time commands to the gripper.

Table 5.1: Driver LCM channels

Channel	Type	Name	Variable
ROBOT COMMAND	double array	`tau`	torque commands
ROBOT STATE	boolean	`robot_enable`	enabling command to start communication with robot
	double array	`q`	joint positions
	double array	`dq`	joint velocities
	double array	`tau_J`	joint torques
GRIPPER COMMAND	double	`width`	closing width of the gripper
	double	`speed`	closing speed of the gripper
	double	`force`	closing force of the gripper
MOTOR COMMAND	boolean	`motor_enable`	enabling command to start motor

5.4.2. controller.py

The main controller was structured as a class and includes functions to read the incoming messages from the other files and send the commands. To avoid errors and ensure a smooth communication at 1 kHz with the robot, additional actions such as writing the data on text files were included outside of the main loop.
The commands sent to the robot are only torque commands, thus no motion generators are used in the C++ file. The torque to be sent is computed based on a predefined velocity and position trajectory (subsection 5.6.2).

5.4.3. torque_control.cpp

This file serves as a communication interface with the robot arm joints and it can be divided in 3 phases. In the first phase the robot returns to a predefined initial position using a *MotionGenerator*. Following, the collision behaviour is defined using the *setCollisionBehavior* function. Finally, the file sets the `robot_enable` boolean to true and initiates the *robot.control* function, where the torque commands are received. The received commands are stored in a struct defined as:

```
1 struct command_received{
2     std::array<double, 7> tau;
```

```
    bool robot_moving;
};
command_received rcm_struct;
```

For handling the LCM messages, a class is defined as:

```
// define message handler
class Handler
{
    public:
        ~Handler() {}
        void handleMessage(const lcm::ReceiveBuffer* rbuf,
                const std::string& chan,
                const frankalcm::robot_command* msg_received){
              int i;
              rcm_struct.robot_moving = msg_received->robot_moving;
              for (i=0; i<7; i++){
                rcm_struct.tau[i] = msg_received->tau[i];}
              }
};
```

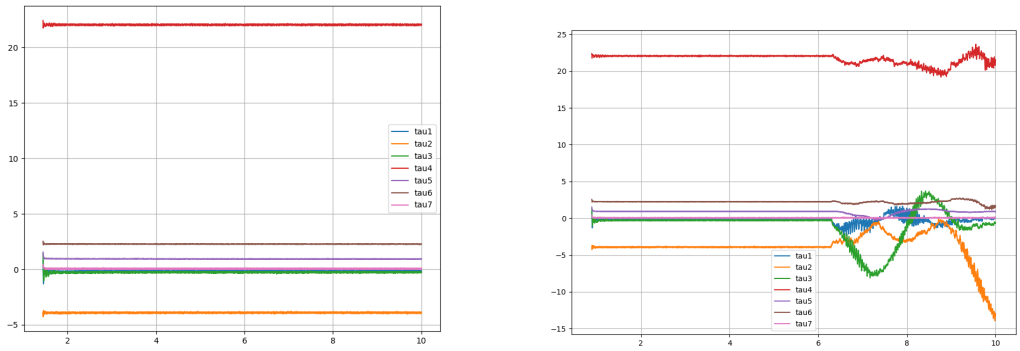
5.4.4. gripper_control.cpp

This file has two different phases. Firstly, the *gripper.homing()* function is called and the gripper returns to the initial position. Secondly, the file waits for the width, speed and force commands coming from `controller.py` with the *lcm.handle()* function. All the commands executed are not in real-time. Similarly to `torque_control.cpp`, the struct and message handler are defined as follows:

```
struct command_received{
    double width;
    double speed;
    double force;
};

command_received gcm_struct;

// define message handler
class Handler
{
    public:
        ~Handler() {}
        void handleMessage(const lcm::ReceiveBuffer* rbuf,
                const std::string& chan,
                const frankalcm::gripper_command* msg_received){
              gcm_struct.width = msg_received->width;
              gcm_struct.speed = msg_received->speed;
              gcm_struct.force = msg_received->force;
}
};
```

5.5. Torque Measurements

Figure 5.2a and 5.2b depict the torque measurements from the robot. It is important to notice that the torque command sent is zero but the effective measured torque is nonzero because of gravity compensation. Two different scenarios are presented. In the first one (Figure 5.2a), the arm is left in the initial position and zero torque is sent. In the second one (Figure 5.2b), the arm is moved by hand. Being able to retrieve torque measurements and stream them from C++ to Python is important for the development of the controller, which will use a torque feedback loop.



(a) (b)

Figure 5.2: Joint torque measurements when (a) arm is not moved (b) arm is moved.

For better visualization, the plotted torque measurements will be normalized by subtracting the average of the initial torque measurements, such that the deviation from the steady state can be observed. This will be shown later in chapter 7.

5.6. Time Matching

The first method implemented to make the testbed work is time matching. After recording the timings for each of the components, the starting time for the robot and the gripper can be tweaked based on the satellite time, so that the robot can reach the contact point in time.

5.6.1. Time Measurements

To test the contact dynamics, the robot has to be able to grasp the satellite while it is still spinning. However the satellite cannot received torque commands while it is grabbed, because that could result in collisions and damage to both the motor and the robot. For this reason, a specific position of the satellite is chosen and, considering the inertia of the structure, the torque commands are sent a certain amount of time before. In this way, the satellite would reach the chosen point with some velocity but with zero torque commands. On the other side, the robot will have to reach the point of contact starting from an initial position. To reach the point and enclose the rod, three time measurements have to be considered:

- Gripper closing time.
- Robot moving time.
- Satellite decelerating time.

The robot joints and the gripper commands have to be executed in parallel, so that by the time the robot reaches the contact point, the gripper can close around the rod and stop the structure from tumbling. Note that this setup is not ideal, because the satellite does not spin by inertia for long. This means that the commanded torque has to be much higher for the satellite to still have some speed when it is grasped. Moreover, the experiment is way less repeatable and there is also more risk of collision when the grasping is not successful. In the following chapter, it is explained how this issue is solved using the friction compensation control method on the motor.

5.6.2. Franka Robot Joints Closed Loop Control

The main control loop is divided in sections based on the time stamps of each component. At different time stamps, a different controller is implemented.

- *waiting phase*: the robot is controlled with a simple PD controller to maintain it's initial position. The desired rotational rate is zero for all joints.

```
1 if (t <= robot_time):
2     rcm.robot_moving = False
```

```
    q_des  = np.array([0., 0., 0., 0., 0., 0., 0.])
    dq_des = np.array([0., 0., 0., 0., 0., 0., 0.])

    q_error  = q_des - self.q
    dq_error = dq_des - self.dq

    # robot acts as a damper hold the position before the trajectory is followed
    rcm.tau = Kd_wait * dq_error
    self.lc.publish(self.rcm_channel, rcm.encode())
```

- *trajectory phase*: the robot is controlled with a PD controller to follow a certain position and velocity trajectory. The desired position and velocity are defined as a function of time.

```
if (t >= robot_time) and (t < robot_time + 2.0):
    q1_des  = 0.5 - 0.5*t_robot + 0.5 / np.pi *np.sin(np.pi * t_robot) + q_fix
    dq1_des = -0.5 + 0.5*np.cos(np.pi * t_robot)

    q_des = np.array([q1_des, 0., 0., 0., 0., 0., 0.])
    dq_des = np.array([dq1_des, 0., 0., 0., 0., 0., 0.])

    q_error = q_des - self.q
    dq_error = dq_des - self.dq

    rcm.tau = Kp_traj * q_error + Kd_traj * dq_error

    rcm.robot_moving = True
    self.lc.publish(self.rcm_channel, rcm.encode())
```

- *damping phase*: the robot is controlled with a D controller, to damp the satellite motion. Alternatively a zero torque command can be sent to the robot joints.

```
else:
    rcm.robot_moving = False
    dq_des = np.array([0., 0., 0., 0., 0., 0., 0.])

    dq_error = dq_des - self.dq

    # robot acts as a damper to detumble satellite
    rcm.tau = Kd_damp * dq_error
    # rcm.tau = np.array([0., 0., 0., 0., 0., 0., 0.])
    self.lc.publish(self.rcm_channel, rcm.encode())
```

The torque command sent from the `controller.py` file is:

$$\tau = K_p \cdot (q_{des} - q) + K_d \cdot (\dot{q}_{des} - \dot{q}) \tag{5.1}$$

Depending on the different phases, the control gains were tweaked to obtain a desirable behaviour.
In the waiting phase it is important to keep each of the joint angles into place, because when the end effector follows the trajectory the starting and ending points are important for the contact to occur. If the robot slightly drifts away from the intended position, the grasping may not occur or the robot may overshoot the contact point.
In the second phase, the velocity trajectory is described by Equation 5.2. By integrating the velocity, the commanded position is found (5.3). To bring the end effector to the grasping position, only the first joint is required to move. The other 6 are kept at constant position and zero speed. Note that the PD values for the first joints are indeed higher.

$$\dot{q}_{1,des} = -\frac{1}{2} + \frac{1}{2} \cdot \cos(\pi \cdot t_r) \tag{5.2}$$

$$q_{1,des} = \frac{1}{2} - \frac{1}{2} \cdot t_r + \frac{1}{2}\pi \cdot \sin(\pi \cdot t_r) \tag{5.3}$$

Finally, in the damping phase the gains are kept low. It is also an option to command zero torque and see how the robot behaves after grasping with only gravity compensation.

Table 5.2: Gain values for every phase of the experiments

		phase		
		waiting	trajectory	damping
gain	K_p		$(2.7, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$	
	K_d	$(1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$	$(0.3, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$	$(0.1, 0.1, 1.0, 0.1, 0.1, 0.1, 0.1)$

5.7. Git Repository

The project was documented in a git repository [4]. All the files required to communicate with the robot arm, such as LCM bindings and the C++ executable files, are included in the `franka_interface` folder. A CMake file was made to compile the C++ files and link the libfranka and the LCM libraries. To run multiple files at once, different bash scripts were made. Firstly, the LCM communication is initiated with the `start_lcm.sh` file. Secondly, the CAN communication with the robot is set up using `start_motor.sh`. Finally, the experiment is run with `tm_driver.sh` (time matching driver). All the data recorded during the tests is included in the experiments folder and all the different parts of the repository have been documented and explained on README.md files.

[4] `https://github.com/frabranca/contact-dynamics-testbed`

6

Motor Driver

This chapter explains how the motor is controlled and how it is connected to the main controller.

6.1. Motor Implementation in the Testbed

The motor will be given a torque input to make the satellite spin. Before the grasping occurs, the motor commands have to be set to zero, otherwise the Franka robot might collide into it or try to counteract the torque for detumbling. This can result in damage of both the motor and the robot arm. After all the commands are set to zero, the satellite is going to keep spinning for some time due to inertia, before eventually being stopped by friction. During such time, the robot arm will have to intercept the grasping point. The `motor_controller.py` file contains a loop function where torque commands are sent to the motor for a certain amount of time. The position, velocity and torque of the motor are also communicated through a LCM channel to the `controller.py` file. Due to noisiness of the velocity and torque measurements, a filter was applied for better estimation. The code averages out every 100 measurements and saves the value obtained.

The biggest problem with this method is that, after the torque command is set to zero, the mock-up does not spin for long. This not only means that the robot arm has a very short time window to perform the grasping, but also that the speed decreases quite quickly. To capture the satellite at a certain speed, for instance 20 deg/s, the motor has to spin almost twice as fast to be intercepted at the desired speed. Figure 6.1 shows how quickly the satellite brakes after the torque control is stopped. After approximately 8.3 seconds the satellite is captured and the velocity drops to zero.

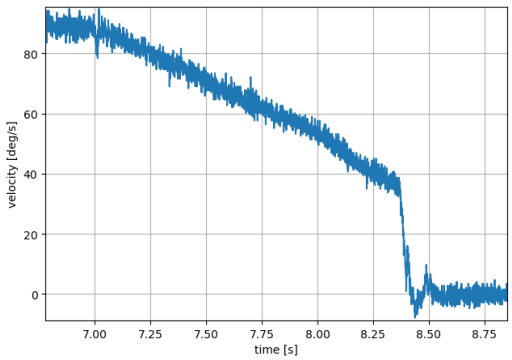


Figure 6.1: Satellite velocity

6.2. Motor Friction Compensation

To solve the issue a friction compensation algorithm is implemented. Friction can be categorized into static friction and dynamic friction. In the case presented, only the dynamic friction is relevant, since the motor is already spinning when the controller switches from normal torque and velocity commands to friction compensated commands. For dynamic friction, the Coulomb friction model is considered. Equation 6.1 gives the relation between the friction force and the rotating speed.

$$F_c(\omega) = C_0 \cdot sgn(\omega) \tag{6.1}$$

ω is the angular velocity and C_0 is the Coulomb friction coefficient, which has to be derived empirically. To avoid singularities in the code, the friction force is implemented as in Equation 6.2, so that the function is continuous.

$$F_c(\omega) = C_0 \cdot \arctan(100 \cdot \omega) \tag{6.2}$$

Figure 6.2 shows the control loop for the motor. After a certain time the controller switches from simple PD to friction compensation. The second controller takes as input the measured velocity and derives the corresponding force required to counteract the friction force. By tweaking the C_0 value, the forces can be balanced so that the total force is cancelled out and the satellite keeps spinning without accelerating or decelerating.

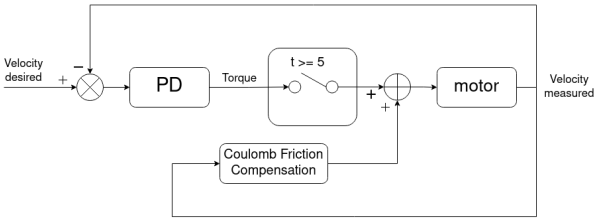


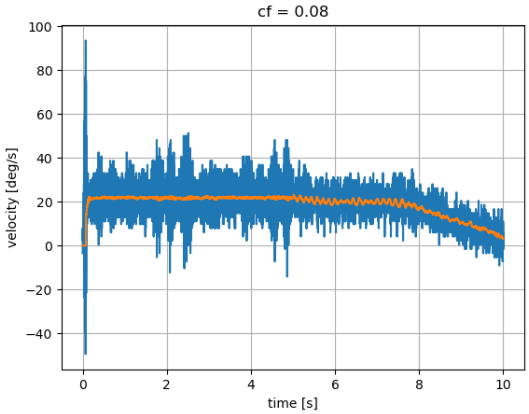
Figure 6.2: Friction compensation control loop

The implementation of the algorithm in the code and the filter is as follows:

```
def friction_compensation(self, v):
    return self.cf*np.arctan(100*v)
```

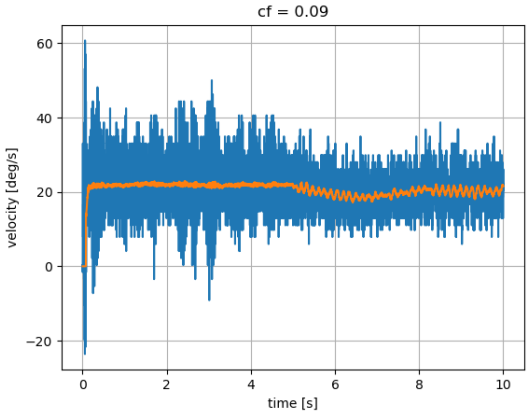
```
while (time.time() - start) < 20:
    if (time.time() - start) >= 5.:
        torque = self.friction_compensation(vel_meas)
        pos_meas, vel_meas, tau_meas = self.motor.send_deg_command(0, 0, 0, 0, torque)
    else:
        torque = self.friction_compensation(vel_meas) + tau_des
        pos_meas, vel_meas, tau_meas = self.motor.send_deg_command(pos_des, vel_des, Kp, Kd,
            torque)

    filter = 100
    filter_size = filter-1
    if i > filter_size+1:
        vel_filtered = sum(self.vel_save[i-filter_size:i+1]) / (filter_size+1)
    else:
        vel_filtered = 0

    self.vel_filter_save.append(vel_filtered)
    i+=1

    self.t_save.append(time.time()-start)
    self.pos_save.append(pos_meas)
    self.vel_save.append(vel_meas)
    self.tau_save.append(tau_meas)
```

Figure 6.3, 6.4 and 6.5 show how changing the coefficient influences the motor behaviour. The motor is initially controlled with a torque input for 5 seconds. After that, the friction compensation is implemented. A higher value will make the satellite slightly accelerate because the commanded torque will be more than required to counteract the friction. The opposite happens when the coefficient is too low. With a C_0 of 0.09 a satisfying result is achieved.



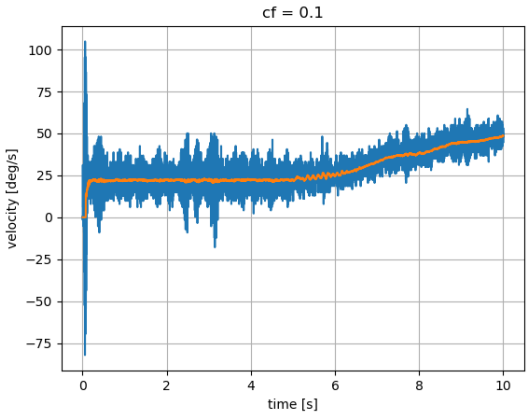
H

Figure 6.3: Satellite velocity for coulomb friction coefficient of 0.08



H

Figure 6.4: Satellite velocity for coulomb friction coefficient of 0.09



H

Figure 6.5: Satellite velocity for coulomb friction coefficient of 0.10

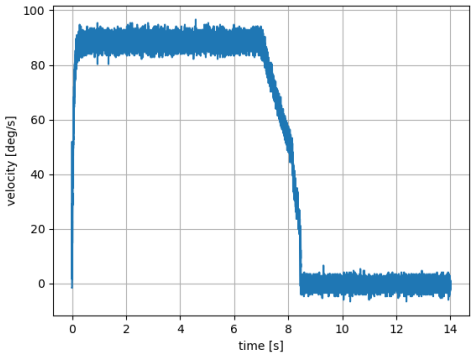
7

Experimental Results

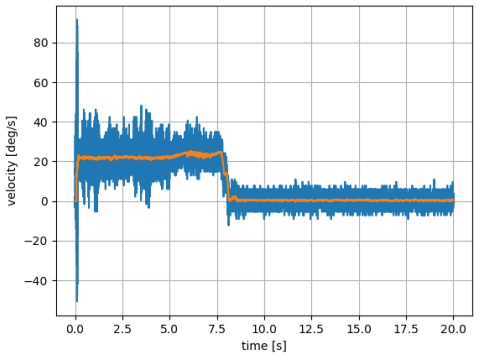
In this chapter the experimental results obtained from the testbed are presented and discussed. Firstly the effect of friction compensation is shown. Secondly, the data obtained by the robot are analysed. Finally, the validity of the measurements is assessed. Note that the data shown in this chapter all belong to the same experiment. The data for the other experiments can be found on the [1].

7.1. Effect of Friction Compensation

The effect of friction compensation can be seen in Figure 7.1a and 7.1b. To better estimate and visualize the velocity, a smoothing filter is applied to the noisy data. The filter averages out every 50 measurements and the result is shown by the orange line in Figure 7.1b.



(a) Satellite velocity without friction compensation



(b) Satellite velocity with friction compensation

The friction compensation resulted to be more handy for the experiments and also it reduced the torque applied from the satellite to the robot. Moreover it allows to simulate more accurately the floating behaviour of an object in space and it makes it easier to determine precisely at what speed the satellite was captured.

7.2. Position and Velocity Measurements

The first joint of the robot follows a trajectory with a velocity defined by Equation 5.2. This equation was defined so that the trajectory started and ended with zero velocity. This is meant to avoid joint motion generator velocity discontinuity errors, which occur when one or more of the joints is commanded a velocity that exceeds the safety limits. By defining a cosine function, it is ensured that the joint will have a gradual velocity increment. Figure 7.2 shows the desired joint velocity versus the measured one. Note that despite the discrepancy, the behaviour of the robot arm is satisfying enough for the

[1] `https://github.com/frabranca/contact-dynamics-testbed`

experiment to be successful. The end effector follows the path and arrives at the contact point location with zero velocity and without any errors in between. It is not strictly required to follow the velocity trajectory closely, which is also why the damping gains where kept much lower. Regarding the position trajectory, the proportional gains where set higher to ensure that the end effector would arrive at the exact point of contact.

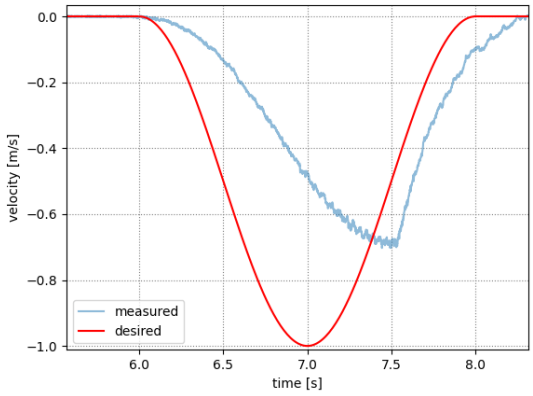
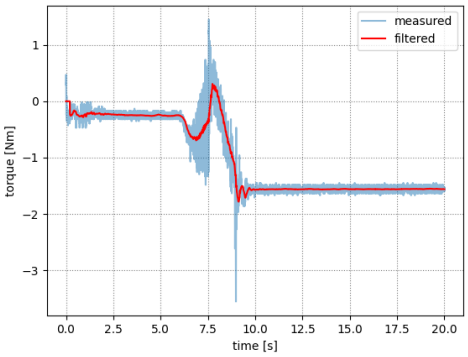


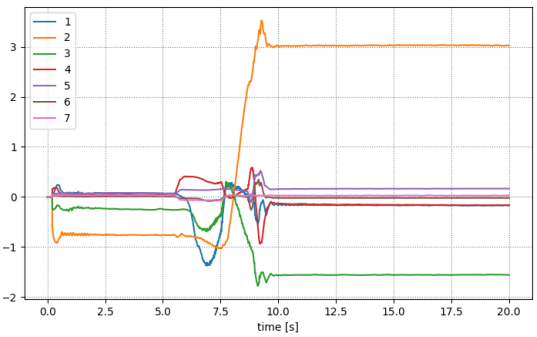
Figure 7.2: Desired versus measured velocity for joint 1.

7.3. Joint Torque Measurements

Because the recorded data is subject to noise, some of the unfiltered measurements are not to be trusted completely and could be considered outlier data points. To better visualize and estimate the effective torque, an averaging filter was applied. Moreover the measurements were normalized with respect to the initial value, so that the variation from the steady state could be observed. Figure 7.3a shows the torque filtered every 200 measurements for the third joint of the robot. The different phases of the experiments can be clearly distinguished from the graph. There is the initial waiting phase and after approximately 6 seconds, the robot starts following the trajectory. Finally, after 8 seconds the contact occurs and there is a spike in the torque measurements. After the contact the robot stabilizes itself and converges to a new steady state torque value of roughly -1.5 Nm with respect to the initial one. By observing the all joint torque measurements (Figure 7.3b) it can be noticed that the second joint is the one that experiences the highest torque change at the moment of contact.



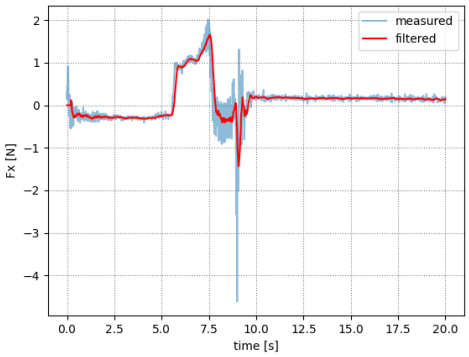
(a) Filtered torque measurements on joint 3.



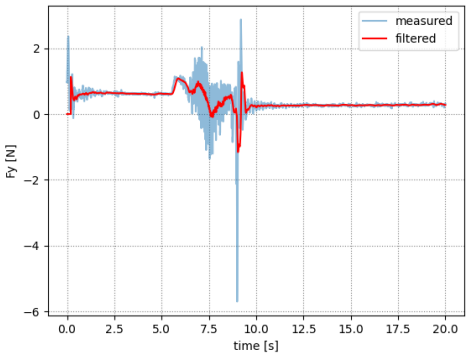
(b) Joint torque measurements with satellite speed of 20 m/s.

7.4. End-Effector Forces Measurements

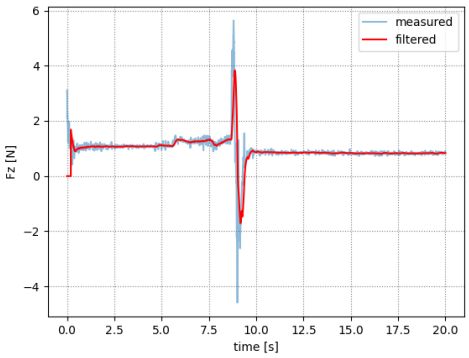
Another interesting measurement is the forces and moments exerted on the end effector. From Figure 7.4a, 7.4b and 7.5a it can be seen that the load exerted on the end effector does not exceed 4 N in all directions, while for the torques the maximum value is approximately 1 Nm (Figure 7.5b, 7.6a, 7.6b).



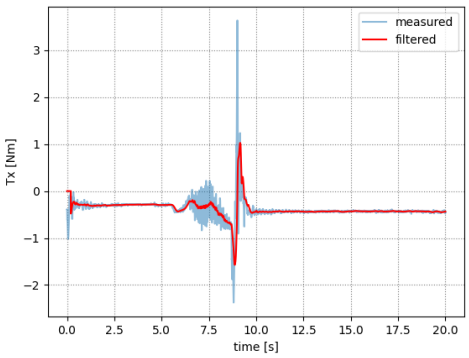
(a) Force in X direction.



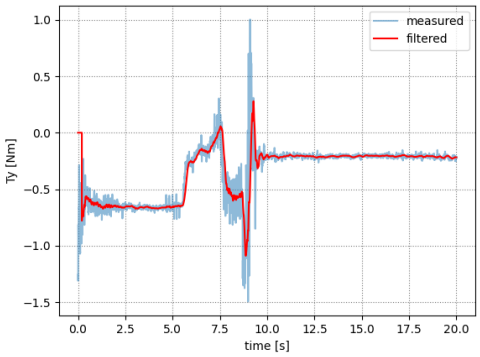
(b) Force in Y direction.



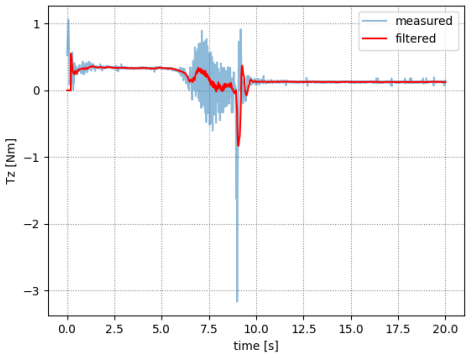
(a) Force in Z direction.



(b) Torque in X direction.



(a) Torque in Y direction.



(b) Torque in Z direction.

7.5. Validity of the Measurements

Since this project is still on an early stage of development, the validity of the obtained measurements is not expected to be valuable yet for research purposes. However, it did give some meaningful insights for the future development of the testbed. It was initially expected to observe a more significant motion of the robot arm after capturing the satellite. In an ideal testbed, the two objects would start moving together and eventually detumble. However the motion of the robot arm was still extremely damped, even when zero torque commands were sent. Although the robot is supposed to be gravity compensated, it is possible that there is some internal damper that attenuates the motion for safety reason.

8

Conclusions

This report documented the work achievements of a 6 months internship at DFKI. The aim of this internship assignment was to build a contact dynamics testbed, to simulate the interaction between a robot arm and a space debris. The purpose of this experimental setup is to be used to test different control algorithms for ADR research.

Firstly, to gather the necessary knowledge for the project and collect insights on ADR methods and testbeds, a literature study was conducted. It was found that most of testbeds included an air bearing system to also simulate the translational and rotational motion of the debris. Furthermore, different gripper mechanisms, such as dry adhesive, were used to ensure a stiff connection once the object is captured.

Using the results of the literature study, the problem was defined in more details and a preliminary design of the testbed was developed. For the purpose of this project, the target object was constrained to only rotational motion in one direction only. An motor was used to spin the satellite mock-up to be captured. The robot arm was a Franka Emika robot arm, which was controlled using a PC connected via Ethernet cable to the FCI. On the PC, C++ and Python files were coded to control the robot. Regarding the gripper, a 3D printed extension was made to be attached to the robot arm. To increase friction, a rubber material was glued on the pieces.

To communicate with the robot and the motor, a C++ to Python driver was constructed, using the C++ libraries LCM (for communication between the different programming languages) and libfranka (to send and receive commands to the robot). The actual control algorithms were implemented in Python and, using the LCM, the data was sent and received from the C++ executable files. To simulate the free-floating behaviour of the space debris, a friction compensation was implemented on the motor, which resulted useful for the experiments repeatability.

In conclusion, the results obtained from the experiments were useful, as they provided insights on the the contact dynamics characteristics and on how to develop a more representative testbed in the future. In particular, the free-floating behaviour of the robot will need to be improved, because the motion is still damped even when the torque being sent is zero.

9

Future Recommendations

In this chapter, some useful insights on the future of this project are given and recommendations on possible new challenges are explained.

9.1. Gravity Compensation of the Franka Emika Robot Arm

In the previous section section 7.5, we noted that there could be some internal damping present in the robot joints. This damping serves to restrict the motion of the robot arm for safety reasons. The result is that the validity of any measurements is reduced. This is because the motion of the robot is not completely gravity compensated, and is therefore not entirely representative of how the arm would behave in space.

In a situation where the two objects are connected, it would be expected that the robot arm and the motor would move together. However, because the robot arm is too damped, the moment of inertia of the mock-up satellite is not sufficient to induce significant forces on the arm at the moment of contact. It would be valuable to explore potential solutions to this issue. For instance, it may be possible to adjust the internal damping settings of the robot joints to reduce the level of restriction on the arm's motion. Alternatively, other strategies for minimizing the impact of damping on the robot arm's behavior could be explored.

Overall, by taking steps to address the issue of internal damping in the robot joints, we could significantly improve the accuracy and validity of the measurements taken by the arm, and thus enhance the overall effectiveness of the project.

9.2. Contact Point Detection Methods

As for the moment, the detection of the contact point is done with the time matching method, which is a very simplified solution. However, in the future it would be interesting to apply a closed loop method to the testbed. The position and velocity data coming from the motor can be streamed to the main controller, which will process this information and decide accordingly when and how to approach the object. Instead of making a predefined path to follow, a trajectory could be computed based on how the satellite moves. Other more advanced detection method could also be applied, such as computer vision algorithms or using different types of sensor to estimate the satellite motion characteristics.

Currently, the contact between the satellite and the robot arm occurs only under very specific conditions. This means that the system is somewhat limited in terms of its overall versatility. To address this limitation, it would be valuable to explore different detection methods that could be implemented to enhance the system's flexibility and adaptability. By doing so, we could ensure that the system is capable of detecting and responding appropriately to a wide range of scenarios.

9.3. Capturing Methods

In chapter 3, we explored various examples of capturing methods that could be used to capture the satellite using the robot arm. However, there is still a wide variety of other technologies that have been developed and could be tested.

For example, one possibility would be to install grippers on the robot arm that have magnetic properties or are coated with a dry adhesive. This would allow the arm to attach more securely to the satellite, potentially enabling it to manipulate the object in more precise and controlled ways.
Of course, before testing different types of manipulators or capture methods, it is important to ensure that the overall system is sufficiently versatile and adaptable. This is because different capturing methods would require different control methods, which would need to be implemented on the robot.

10

Assignment on Engineering Profession

This chapter covers certain aspects of DFKI and how it is meeting professional standards with respect to the management of projects, knowledge, value and safety.

10.1. Project Management

I had the opportunity to work as an intern with DFKI, and I must say that their policy towards interns is commendable. It is inclusive, proactive, and designed to ensure that interns gain valuable experience while working with professionals in the field. I was thrilled to be part of a team of experts who were willing to offer guidance and support whenever needed. My supervisor was always available to answer my questions and provide valuable insights into the project.

One of the things that stood out during my internship was the balance between formality and informality. Although the work relationship was not overly formal, my supervisor maintained a professional demeanor that instilled a sense of respect and responsibility. This balance allowed for a comfortable working environment where interns could feel confident in their abilities while also being open to learning from experienced professionals.

In terms of project time management, DFKI's approach was unique. Unlike some internships where interns are given strict deadlines, I was provided with a final goal to achieve by the end of the internship. My supervisor gave me intermediate milestones to reach before the final objective, which made the process more manageable and less stressful. This approach not only allowed me to work at my own pace, but it also motivated me to give my best and strive for excellence.

I had the chance to discuss my experience with my fellow interns, and I realized that we all had a similar positive experience. This approach not only contributes to a positive experience for interns, but it also increases the chances of retaining them as possible future workers.

To ensure efficient progress, DFKI requires all interns to fill in a time sheet at the end of each day. I found this to be extremely helpful as it allowed me to keep track of my progress and have an overview of the direction the project was taking. It also gave me a sense of how I was managing my time, and I could easily identify areas that needed improvement.

Overall, the project management policy of DFKI contributed to making my internship experience exceptional. Their inclusive and proactive approach towards interns not only allows interns to gain valuable skills, but also motivates them to give their best. The combination of guidance, support, and autonomy makes it possible for the interns to succeed and enjoy the experience to the fullest.

10.2. Knowledge Management

The management of previously done work at DFKI is conducted in a highly effective manner. To ensure that the information about all the past and present projects is documented, every new project is stored in its own git repository and extensive explanation is given. My supervisor was extremely serious about this aspect, as he wanted to make sure that all my work was properly documented, so that he could continue on the project after the end of the internship.

I also had to explore other employee's git repositories and look into their work, in order to apply a friction compensation control technique on my motor. I found the documentation very helpful and clearly

explained. I believe this to be an important factor, because it is crucial for new employees to be able to access information about previously done work and understand them quickly.
Another peculiar aspect of DFKI is that every employee has its own area of specialization and everyone is always available and willing to clear any doubts. I often had to ask advice to other employees regarding the manufacturing and mechanical engineering parts of my project. I also talked with other coworkers about their project and they were happy to answer all my questions and explain in details all their work.

10.3. Value Management

Besides focusing on industrial applications, DFKI's main core is research and innovation in the area of AI and robotics. This is reflected in the working environment, which I found highly conducive to learning and growth. I was surrounded by knowledgeable and experienced professionals who were always ready to assist and guide me whenever I had questions or doubts. They provided me with valuable insights, feedback, and advice, which helped me improve my technical skills and become a better problem solver.
Despite working on a project related to the Space Department of DFKI, I had the chance to work and interact with many colleagues from the Underactuated Robotics Lab [1]. The focus of this lab is on modern nature-inspired systems, such as quadruped and humanoid robots, and the aim is to thoroughly understand the systems dynamics in order to develop more agile and nimble robots. To keep on track with the current robotics trends, the Underactuated Lab organizes a study group with sessions every 2 weeks. The participants follow lectures on a certain course (for instance the last one was about *Optimal Control and Reinforcement Learning*) and discuss the homework together.
There is also a chance for all the employees to participate to presentations and seminars, mostly held by interns and master students working on their thesis. The presenters introduce their project and discuss how the approached it and the main challenges that they tackled. I found this aspect particularly interesting, as it gives the opportunity to have a better understanding on the ongoing projects of the institute.
Regarding the more promotional aspects of the company, there is a yearly open day [2] event held twice a year, where external visitors have the chance to visit the institute and see robot demonstrations. For instance the humanoid robot developed from the Underactuated Lab performed a dance choreography, in the Submarine Robotics department the visitors could use a joystick to remotely control a submarine robot in a pool, the robot rovers were driving around obstacles. I believe this experience to be very enriching both for the visitors and the employees, because it is an opportunity to show your work and possibly motivate other people to get involved in the field of robotics.
Another important event is the Christmas Lecture, which is held around mid December every year. During this event the chief gives an overview of the performance of the institute, mainly in terms of number of scientific publications. Moreover, the some future trends and areas of focus are highlighted and discussed, in order to motivate the employees and push them to give their best. This year the chief gave a speech about improving in the AI research to push the boundaries of what is possible to achieve with it.
DFKI is not only an active participant in cutting-edge research and development, but also places a strong emphasis on social media outreach. In addition to maintaining a presence on popular platforms such as LinkedIn, Instagram, and YouTube, the center has a dedicated department for video production and editing. On these platforms, they regularly upload engaging photos, videos, and posts, providing insights into their work and projects. I was fortunate enough to have the opportunity to work with this department and participate in the creation of a video showcasing my internship project. Through this experience, I gained a valuable understanding of the importance of social media in the scientific community, and the role it plays in fostering collaboration and communication between researchers and the wider public.

[1] https://robotik.dfki-bremen.de/en/research/research-facilities-labs/underactuated-lab/
[2] https://robotik.dfki-bremen.de/en/startpage/news/entry/open-day-at-the-dfki/

10.4. Safety Management

Regarding the safety management, every intern is required to participate to the security briefing. During this lecture, the security chief of DFKI explains the safety measures to take in case of fire hazards and other similar accidents. The lecturer gave examples of actual accidents that occurred at DFKI and how the people handled the situation.

I had to comply with safety measures on a daily basis when operating in the lab robot. For instance, all the employees had to leave all liquids outside of the room and when using the robots. Moreover, when working with a certain robot, the kill switch always had to be installed in the system, so that in case of malfunctioning the robot could be stopped. This policy ensures that the employees carry their experiments in a safe and responsible manner.

11

Self-Reflection

During my internship experience at DFKI, I had the opportunity to work under the supervision of more experienced colleagues on an engineering problem involving not only programming but also robotics hardware and manufacturing aspects. I believe that this experience has helped to grow both professionally and personally.
Regarding the more technical aspects, I gained a deeper understanding of Python and C++ programming languages and how they can be used to develop robotics software. I learned the importance of organizing my code in a precise way, to make it understandable also to other users and to be able to debug it more efficiently. I also learned how to be more systematic when approaching a problem. Whenever I had to implement something new in the code, I applied it to a simpler case first to get acquainted with it.
Besides improving my technical skills, I also became more organized and efficient in my work. I learned how to frame problems and approach them systematically, which helped me to solve them more effectively. This skill has proven to be highly valuable, not just in my professional life but also in my personal life.
In many occasions I had to interact with other colleagues to ask them for advice regarding their field of expertise. I realized how important it is to learn from other people and try to increase my knowledge every day by just listening and asking questions to more experienced colleagues. Before starting the internship, I was afraid of not being qualified enough to work on such a project, but then I understood that knowledge and expertise can be gained by interacting with co-workers and trying to learn as much as possible from them. It was very satisfying to see that, by the end of the internship, I was also helping other interns with similar problems that I had to tackle myself before.
Outside of the working environment, I had the opportunity to hear from other colleagues not only about their career, but also about the hobbies and passions. This taught me that building a certain bond with co-workers can also help to improve work dynamics and communication.
Despite these accomplishments, I realize that there is still room for improvement in my organizational skills and literature study skills. I believe that I need to be more proactive in my approach to research and study, to be more thorough in my analysis of the topics. Furthermore, I feel like sometimes I could try to be more curious about topics that do not seem so appealing to be at first. I realized this especially when dealing with the manufacturing part of the project, which was not the main focus but was still interesting to explore.
In conclusion, my internship experience at DFKI has been invaluable to me. I have gained a wealth of knowledge and experience in various areas, which I believe will be highly beneficial to me in my future endeavors. The working environment has been highly supportive and conducive to learning and growth. I am grateful for this experience and look forward to applying the skills and knowledge that I have gained to make a positive impact in my future work.

12

Tips & tricks for future interns

I did my internship at DFKI in Bremen (German Research Center for Artificial Intelligence). The project that was assigned to me was building an experimental setup to test contact dynamics in space for research on Active Debris Removal methods.

The internship exceeded my expectations. I was welcomed very well and I immediately got along with the other employees. Having a friendly and cheerful environment helped me to get used to the working life sooner than expected. Furthermore, I believe this experience to be extremely precious for my professional growth, as I learned and familiarized with robotics engineering concepts that I did not know before.

I chose to work at DFKI, because I found the internship vacancy on the brightspace page and I liked the project description. I also had a friend who worked there as an intern and I got the chance to know some insights about the working environment. I liked the idea of working in the field of research and I was happy to get accepted for a project that is very research-based. I believe that working in a research institute, instead of a company, can be an enriching experience and I recommend it.

Regarding the more practical tips about the life in Bremen, finding a house is way easier than in Delft and also cheaper deals can be found. The institute is located in the University of Bremen campus, so I would recommend to look for a house in that area. However, in the center also has cheap accommodations and it is very well connected to the campus with the tram system.

To apply for DFKI, the potential interns are required to solve an assignment. This includes both theoretical and practical questions about robotics systems. Whether the given answers are correct or not is not so relevant for the people processing the application. It is more important to show your motivation through the approach that you take to solve the problems.

References

[1] Andrew Bylard et al. “Robust capture and deorbit of rocket body debris using controllable dry adhesion”. In: (2017), pp. 1–9. DOI: 10.1109/AERO.2017.7943844.

[2] Matthew A. Estrada et al. “Free-flyer acquisition of spinning objects with gecko-inspired adhesives”. In: (2016), pp. 4907–4913. DOI: 10.1109/ICRA.2016.7487696.

[3] Dong Han et al. “Combined spacecraft stabilization control after multiple impacts during the capture of a tumbling target by a space robot”. In: *Acta Astronautica* 176 (2020), pp. 24–32. URL: https://www.sciencedirect.com/science/article/pii/S009457652030318.

[4] Francis James et al. “Design and Development of an Earth Based Experimental Setup for Testing Algorithms on Space Robots”. In: AIR ’15 (2015). URL: https://dl.acm.org/doi/10.1145/2783449.2783487.

[5] Filipp Kozin, Mahdi Akhloumadi, and Danil Ivanov. “Laboratory Study of Microsatellite Control Algorithms Performance for Active Space Debris Removal Using UAV Mock-Ups on a Planar Air-Bearing Test Bed”. In: (Dec. 2022).

[6] Ou Ma, Angel Flores-Abad, and Khanh Pham. “Control of a Space Robot for Capturing a Tumbling Object”. In: (2012). DOI: 10.1117/12.918523.

[7] Akhloumadi Mahdi, Kozin Filipp, and Danil Ivanov. “Laboratory Study of the Active Debris Removal Algorithms on Air-Bearing Test Bed”. In: *IOP Conference Series: Materials Science and Engineering* 984.1 (Nov. 2020), p. 012026. DOI: 10.1088/1757-899x/984/1/012026. URL: https://doi.org/10.1088/1757-899x/984/1/012026.

[8] Shin-Ichiro Nishida and Satomi Kawamoto. “Strategy for capturing of a tumbling space debris”. In: *Acta Astronautica* 68.1 (2011), pp. 113–120. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2010.06.045. URL: https://www.sciencedirect.com/science/article/pii/S0094576510002365.

[9] Minghe Shan, Jian Guo, and Eberhard Gill. “Review and comparison of active space debris capturing and removal methods”. In: *Progress in Aerospace Sciences* 80 (2016), pp. 18–32. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2015.11.001. URL: https://www.sciencedirect.com/science/article/pii/S0376042115300221.

A

Final Evaluation Form

Evaluation Form Internship MSc Aerospace Engineering

Intern's Name : **Francesco Branca**

Organization : **Robotics Innovation Center, DFKI GmbH**

Supervisor Name : **Shubham Vyas**

Telephone Number : **+49 421 17845 4157**

Scores:				
Weak	Moderate	Sufficient	Good	Very Good
=1	=2	=3	=4	=5

Work Result **Score: 4=Good**

Overall performance during the internship

What is particularly good?
Francesco was able to perform most of the initially given tasks within the given time and in a good manner. Some of the initially planned tasks were not completed but these were due to hardware issues or time issues.
What can be improved?
I can recommend improvement in time management and planning during a project as this saves time during the duration of the project. Also, documentation and version early on in the project would be another area for improvement.

Competent in doing research/ Competent in designing **Score: 4=Good**

For this purpose, research means: the development of new knowledge and new insights in a purposeful and methodical way.

What is particularly good?
He was able to quickly develop solutions to problems that came forth and found a working solution every time. This showcases his ability to research, learn, and quickly apply new methods.
What can be improved?
While he usually found solutions to the problems faced, I would recommend a deeper understanding of the topics to develop a better intuition as that sometimes helps in debugging and can save time. However, it is difficult to maintain a balance.

Problem Analysis/Analytical Ability **Score: 5=Very Good**

Identify problems and important information; connect data. Find out possible problem causes; Searching for relevant data.

What is particularly good?
Solved countless problems encountered with good attitude and determination.
What can be improved?

Initiative and willingness to learn **Score: 4=Good**

This is about being keen and enthusiastic, taking responsibility for own learning and development and being hungry to succeed and learn.

What is particularly good? Motivation to learn new concepts in a new field he has no experience in.
What can be improved? Going deeper in the research sometimes to get better intuition even though it costs more time.

Interpersonal Skills **Score: 4=Good**

Works harmoniously and effectively with subordinates, peers, supervisors? Team player? Shares information with others? Resolves conflicts? Welcomes and seeks constructive feedback on own performance? Cooperative?

What is particularly good? Well integrated with colleagues and other interns.
What can be improved?

Diligence and resilience **Score: 5=Very Good**

This is about being determined, hardworking, and results focused and being able to work effectively under pressure. Target driven.

What is particularly good? Worked through multiple problems that came up during the internship with good attitude and determination. Was nice to see.
What can be improved?

Flexibility **Score: 5=Very Good**

Ability to change behavior if problems or chances arise in order to reach the specified aim. Competent in reasoning, reflecting, and forming a judgment.

What is particularly good? Was able to pivot effectively and quickly when certain approaches were not giving results to new approaches that gave good results.
What can be improved?

General Remarks

To what extent did the student meet the expectations?

Francesco met the expected we had from him regarding his performance in the internship.

Do you have recommendations for the student concerning his/her personal and professional development?

I would recommend more documentation, systems engineering, and going a bit furhter than the assigned tasks (attending guest lectures, etc.) as these usually help in exploring new areas of engineering and research.

Can you imagine the student as a future colleague?

Yes. We can imagine having Francesco as a future colleague here at DFKI.

Overall performance of the student; Please indicate the category:

☐ outstanding performance (top 5%)

☒ -excellent student

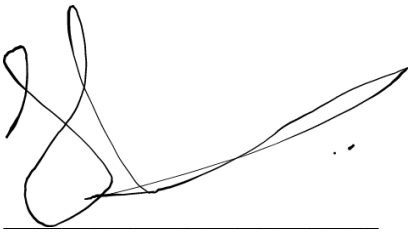
☐ -good student

☐ -average student

☐ -this student does not fulfill the minimum requirements

Additional comments

Signature supervisor:



Date: 22/02/2023

Stamp (optional):

On behalf of TU Delft, Faculty of Aerospace Engineering we would very much like to thank you for all your effort and time. If there are any questions, do not hesitate to contact us.
E-mail address: Internship-AE@tudelft.nl.